PERGAMON

Neural Networks 11 (1998) 739–747

Neural Networks

Contributed article

Robust maximum likelihood training of heteroscedastic probabilistic neural networks

Zheng Rong Yang[a], Sheng Chen[b,*]

[a]Department of Electronics and Computer Science, University of Southampton, Southampton PO17 5BJ, UK
[b]Department of Electrical and Electronic Engineering, University of Portsmouth, Portsmouth PO1 3DJ, UK

Received 1 September 1997; accepted 2 February 1998

Abstract

We consider the probabilistic neural network (PNN) that is a mixture of Gaussian basis functions having different variances. Such a Gaussian heteroscedastic PNN is more economic, in terms of the number of kernel functions required, than the Gaussian mixture PNN of a common variance. The expectation-maximisation (EM) algorithm, although a powerful technique for constructing maximum likelihood (ML) homoscedastic PNNs, often encounters numerical difficulties when training heteroscedastic PNNs. We combine a robust statistical technique known as the Jack-knife with the EM algorithm to provide a robust ML training algorithm. An artificial-data case, the two-dimensional XOR problem, and a real-data case, success or failure prediction of UK private construction companies, are used to evaluate the performance of this robust learning algorithm. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords: Probabilistic neural networks; EM algorithm; Jack-knife; Mixture distributions; Bayes classifier; Pattern classification; Parzen windows; Gaussian basis functions

1. Introduction

The key to applying the optimal Bayes strategy for pattern classification is the estimation of the class conditional probability density functions (PDFs), and a classical PDF estimator is the Parzen window estimator (Parzen, 1962). PNNs implement the Parzen window estimator using mixtures of Gaussian kernels. As in a Parzen window estimator, the original PNN proposed by Specht (1988, 1990) uses all the training pattern samples as centres or mean vectors of kernel functions, leaving only mixing coefficients and a common variance or covariance matrix to be estimated from training data. In practice, it is desirable to employ a smaller set of kernel functions without sacrificing classification accuracy. A careful learning of kernel mean vectors is then required. Streit and Luginbuhl (1994) applied the EM algorithm (Dempster et al., 1977) to derive an optimal ML training algorithm for Gaussian mixture PNNs of common covariance matrices (termed homoscedastic PNNs). In this study, we will assume that a Gaussian kernel has a variance parameter, instead of a covariance matrix. That is, we use uncorrelated Gaussian kernel functions.

If each Gaussian basis function is allowed to have a different variance, a much more parsimonious PNN can be used to adequately approximate the underlying conditional PDFs of training data. When extending the ML training algorithm (Streit and Luginbuhl, 1994) to this class of heteroscedastic PNNs, however, numerical difficulties frequently occurs. We analyse the root of this numerical problem, and propose to use a robust statistical method called the Jack-knife (Efron and Gong, 1983; Miller, 1974) to overcome this problem. The Jack-knife is a statistical technique which is capable of removing the effect of outliers to a statistical estimate, and it has widely been used for robust estimation and modelling in econometric problems (Chatfield, 1987; Efron and Tibshirani, 1993). We incorporate the Jack-knife technique with the EM algorithm to derive a robust ML training method for heteroscedastic PNNs.

To demonstrate the effectiveness of this robust learning algorithm, we use it to train heteroscedastic PNNs for the XOR problem and a real-data problem. The latter is to classify successful and failed companies in the UK private construction industry. This is a complex real-world problem, which requires nonlinear decision boundaries to achieve adequate performance (Yang et al., 1997). Our study confirms that the ML training algorithm (Streit and

* Corresponding author. Tel: +44 (0)1705 842599; Fax: +44 (0)1705 842561; E-mail: schen@ee.port.ac.uk.

0893-6080/98/$19.00 © 1998 Elsevier Science Ltd. All rights reserved.
PII S0893-6080(98)00024-0

Luginbuhl, 1994) may fall to work for heteroscedastic PNNs owing to numerical difficulties and our robust ML training algorithm does not suffer the same problem. The results also confirm that a much smaller heteroscedastic PNN is required to achieve the same level of accuracy, compared with using a homoscedastic PNN.

2. The heteroscedastic PNN

Let $\mathbf{x} \in \mathcal{R}^d$ be a d-dimensional pattern vector and its associated class be $j \in \{1,\ldots,K\}$, where K is the number of possible classes. A classifier maps a given pattern $\mathbf{x}$ to its class index $g(\mathbf{x})$, where $g:\mathcal{R}^d \to \{1,\ldots,K\}$. If the a priori probability of class j is α_j and the conditional PDF of class j is f_j, for $1 \le j \le K$, the Bayes classifier that minimises the misclassification error (Devijer and Kittler, 1982) is given by

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

The PNN is a four-layer feedforward neural network that is capable of realizing or approximating this optimal classifier. In order to realize the Bayes decision (Eq. (1)), the class conditional PDFs must be estimated, and the PNN do so using mixtures of Gaussian kernel functions or Parzen windows.

The first layer of the PNN is the input layer which accepts input patterns. The nodes in the second layer are divided into K groups, one for each class. The generic second-layer node, the ith kernel in the jth group, is defined as a Gaussian basis function

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

where $\mathbf{c}_{i,j} \in \mathcal{R}^d$ is the centre or mean vector and $\sigma_{i,j}^2$ a positive variance parameter. A node in the second layer is also called a pattern unit. Let the number of pattern units for class j be M_j. Then the total number of the second-layer nodes is

$$M = \sum_{j=1}^{K} M_j. \tag{3}$$

The third layer has K nodes, and each node estimates a class conditional PDF f_j using a mixture of Gaussian kernels

$$f_j(\mathbf{x}) = \sum_{i=1}^{M_j} \beta_{i,j} p_{i,j}(\mathbf{x}),\ 1 \le j \le K, \tag{4}$$

where the positive mixing coefficients $\beta_{i,j}$ satisfy

$$\sum_{i=1}^{M_j} \beta_{i,j} = 1,\ 1 \le j \le K. \tag{5}$$

The fourth layer of the PNN makes the decision according to Eq. (1).

The PNN defined above is heteroscedastic as each Gaussian kernel has its own different variance. Such a PNN is more difficult to train than the homoscedastic PNN. The latter, however, may require a larger set of basis functions to adequately approximate PDFs because the same variance is used in every basis function. For the heteroscedastic PNN, the centres $\mathbf{c}_{i,j}$ and variances $\sigma_{i,j}^2$ as well as the a priori probabilities α_j and mixing coefficients $\beta_{i,j}$ need to be estimated from the training data set. The class a priori probabilities α_j are problem dependent, and it may not always be feasible to estimate α_j purely from the training data since the training set may contain little meaningful information regarding the class a priori probabilities (Streit and Luginbuhl, 1994). We, therefore, assume that every class is equiprobable

$$\alpha_j = \frac{1}{K},\ 1 \le j \le K. \tag{6}$$

3. The EM algorithm for training PNNs

The EM algorithm (Dempster et al., 1977) is a powerful iterative procedure for solving diverse estimation problems. Each iteration of the algorithm consists of an expectation process (the E-step) followed by a maximisation process (the M-step). Each E-step computes the expected value of some ‘unobserved’ data using the current parameter estimate and the observed data. Each M-step uses the data from the E-step as though they were actually measured data to form a likelihood function and determines an ML estimate of the parameter. The EM algorithm is guaranteed to converge to an ML estimate (Wu, 1983; Xu and Jordan, 1996), and the convergence rate of the EM algorithm is usually quite fast. Another advantage of the EM algorithm is that it is not necessary to compute gradients or Hessians. Streit and Luginbuhl (1994) applied the EM algorithm to train the homoscedastic PNN. Extension of their ML learning algorithm to the heteroscedastic PNN is straightforward.

Let the training data set be partitioned into the K labelled subsets

$$\{\mathbf{x}_n\}_{n=1}^{N} = \{\{\mathbf{x}_{n,j}\}_{n=1}^{N_j}\}_{j=1}^{K}, \tag{7}$$

where

$$\sum_{j=1}^{K} N_j = N \tag{8}$$

is the total number of training samples and N_j the number of training samples for class j. Under Eq. (6), the log posterior likelihood function of the training set is

$$\log L_f = \sum_{j=1}^{K}\sum_{n=1}^{N_j} \log f_j(\mathbf{x}_{n,j}), \tag{9}$$

subject to Eq. (5), where f_j is defined in Eq. (4). The appropriate Lagrangian is

$$\mathcal{L} = \log L_f + \sum_{j=1}^{K} \lambda_j \Big(\sum_{i=1}^{M_j} \beta_{i,j} - 1 \Big), \tag{10}$$

where λ_j are the Lagrange multipliers. Setting

$$\left. \begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{c}_{m,i}} &= 0 \\ \frac{\partial \mathcal{L}}{\partial \sigma^2_{m,i}} &= 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda_i} &= 0 \\ \frac{\partial \mathcal{L}}{\partial \beta_{m,i}} &= 0 \end{aligned} \right\} \tag{11}$$

gives rise to

$$\mathbf{c}_{m,i} = \frac{\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})}, \tag{12}$$

$$\sigma^2_{m,i} = \frac{\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \mathbf{c}_{m,i}\|^2}{d\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})}, \tag{13}$$

and

$$\beta_{m,i} = \frac{1}{N_i} \sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i}), \tag{14}$$

where

$$w_{m,i}(\mathbf{x}_{n,i}) = \frac{\beta_{m,i} p_{m,i}(\mathbf{x}_{n,i})}{\sum_{l=1}^{M_i} \beta_{l,i} p_{l,i}(\mathbf{x}_{n,i})}. \tag{15}$$

Let $\beta_{m,i}|^{(k)}$, $\sigma^2_{m,i}|^{(k)}$, and $\mathbf{c}_{m,i}|^{(k)}$ be the previous values of $\beta_{m,i}$, $\sigma^2_{m,i}$ and $\mathbf{c}_{m,i}$, respectively. Each iteration of the EM algorithm can then be summarised as follows:

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

$$w^{(k)}_{m,i}(\mathbf{x}_{n,i}) = \frac{\beta_{m,i}|^{(k)} p^{(k)}_{m,i}(\mathbf{x}_{n,i})}{\sum_{l=1}^{M_i} \beta_{l,i}|^{(k)} p^{(k)}_{l,i}(\mathbf{x}_{n,i})}, \tag{16}$$

where

$$p^{(k)}_{l,i}(\mathbf{x}_{n,i}) = \frac{1}{(2\pi\sigma^2_{l,i}|^{(k)})^{d/2}} \exp\left(-\frac{\|\mathbf{x}_{n,i} - \mathbf{c}_{l,i}|^{(k)}\|^2}{2\sigma^2_{l,i}|^{(k)}} \right). \tag{17}$$

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

$$\mathbf{c}_{m,i}|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w^{(k)}_{m,i}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1}^{N_i} w^{(k)}_{m,i}(\mathbf{x}_{n,i})}, \tag{18}$$

$$\sigma^2_{m,i}|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w^{(k)}_{m,i}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \mathbf{c}_{m,i}|^{(k)}\|^2}{d\sum_{n=1}^{N_i} w^{(k)}_{m,i}(\mathbf{x}_{n,i})}, \tag{19}$$

$$\beta_{m,i}|^{(k+1)} = \frac{1}{N_i} \sum_{n=1}^{N_i} w^{(k)}_{m,i}(\mathbf{x}_{n,i}). \tag{20}$$

4. Analysis of numerical difficulties

In theory, the above EM algorithm should achieve an ML estimate of the parameters $\{\beta_{m,i}, \mathbf{c}_{m,i}, \sigma^2_{m,i}\}$. In practice, however, the algorithm frequently fails owing to numerical difficulties. The root of this numerical problem can be traced. From Eqs. (12) and (13), it can be seen that the EM algorithm places a kernel on the weighted expectation of a group of training samples and calculates the variance as the weighted Euclidean distance between the mean vector and the samples. Consider the case where a sparsely distributed area exists with just one sample. We have found out that a kernel will approach this sample with a variance approaching approximately zero. This inevitably causes numerical difficulties and is the root of the problem. Notice that this problem does not exist for the homoscedastic PNN as a single variance is computed based on all the samples.

To explain the problem more clearly, rewrite Eq. (13) in the following expanded form

$$\sigma^2_{m,i} = \frac{1}{d\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})} \left(\sum_{n=1}^{N_i} \frac{\beta_{m,i} \frac{1}{(2\pi\sigma^2_{m,i})^{d/2}} \exp\left(-\frac{\|\mathbf{x}_{n,i} - \mathbf{c}_{m,i}\|^2}{2\sigma^2_{m,i}} \right) \|\mathbf{x}_{n,i} - \mathbf{c}_{m,i}\|^2}{\sum_{l=1}^{M_i} \beta_{l,i} p_{l,i}(\mathbf{x}_{n,i})} \right). \tag{21}$$

Clearly $d\sum_{n=1}^{N_i} w_{m,i}(\mathbf{x}_{n,i})$ and $\sum_{l=1}^{M_i} \beta_{l,i} p_{l,i}(\mathbf{x}_{n,i})$ are positive and finite. Define

$$y = \frac{\|\mathbf{x}_{n,i} - \mathbf{c}_{m,i}\|^2}{2\sigma^2_{m,i}}. \tag{22}$$

If a sample $\mathbf{x}_{n,i}$ is far away from $\mathbf{c}_{m,i}$, its contribution to the variance in the limit case is approximately

$$\lim_{y \to \infty} \eta \exp(-y) y = 0, \tag{23}$$

where η is some positive scalar. Thus, if the mean vector $\mathbf{c}_{m,i}$ approaches an isolated sample $\mathbf{x}_{q,i}$, the variance $\sigma^2_{m,i}$ will approximately be

$$\sigma^2_{m,i} \propto \|\mathbf{x}_{q,i} - \mathbf{c}_{m,i}\|^2 \to 0 \text{ as } \mathbf{c}_{m,i} \to \mathbf{x}_{q,i}. \tag{24}$$

This situation occurs not just when the data space has a few outliers. Typically, when a sufficient number of kernel functions are used, some mean vectors will approach edges or sparsely distributed areas of the data space with variances getting very small, which eventually causes the algorithm to collapse. In Section 7, examples will be used to illustrate this situation.

5. The Jack-knife method

The Jack-knife is a robust statistical method that is widely used in statistical analysis for bias reduction and interval estimation (Efron and Gong, 1983; Miller, 1974). Typical applications of the Jack-knife include point and interval estimations in econometric modelling (Chatfield, 1987; Efron and Tibshirani, 1993), and unbiased estimation of the misclassification rate in discriminant analysis (Lachenbruch, 1967). More recently, the Jack-knife has also been applied to compare the outputs of neural network models with those of discriminant analysis models (Tam and Kiang, 1993). In this study, we apply the Jack-knife to improve the robustness of the EM algorithm.

Basically, the Jack-knife procedure partitions a sample space into Q subsets and observes the influence of each subset on the estimating process. Let $\hat{\theta}_{\text{all}}$ be an estimator based on all the N samples and $\hat{\theta}_{-j}$ be an estimator derived with the jth subset deleted from the sample pool, where $1 \le j \le Q$. The Q pseudo-values are first computed as

$$\tilde{\theta}_j = Q\hat{\theta}_{\text{all}} - (Q-1)\hat{\theta}_{-j},\ 1 \le j \le Q. \tag{25}$$

The Jack-knife estimate of θ is derived by averaging all of these Q pseudo-values

$$\tilde{\theta} = \frac{1}{Q}\sum_{j=1}^{Q}\tilde{\theta}_j. \tag{26}$$

If the estimator $\hat{\theta}_{\text{all}}$ has a bias of the form

$$E[\hat{\theta}_{\text{all}}] - \theta = \frac{a}{N} + O\left(\frac{1}{N^2}\right), \tag{27}$$

the Jack-knife estimator (Eq. (26)) has the property of eliminating the order $1/N$ term from the bias (Miller, 1974). Another desired property of the Jack-knife technique is that it can remove the effect of a few outliers within a data space, giving rise to a robust estimate (Chatfield, 1987; Efron and Gong, 1983; Efron and Tibshirani, 1993).

6. The robust ML training algorithm

We incorporate the Jack-knife technique with the EM algorithm to provide an unbiased and robust ML training algorithm for heteroscedastic PNNs. The modification of the algorithm is straightforward. The step 1 in the EM algorithm remains unchanged, except that $\beta_{l,i}|^{(k)}$, $\sigma_{l,i}^2|^{(k)}$ and $\mathbf{c}_{l,i}|^{(k)}$ are replaced by their Jack-knife estimates $\tilde{\beta}_{l,i}|^{(k)}$, $\tilde{\sigma}_{l,i}^2|^{(k)}$ and $\tilde{\mathbf{c}}_{l,i}|^{(k)}$. The updating Eqs. (18)–(20) in the step 2 are modified as

$$\mathbf{c}_{m,i}|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})}, \tag{28}$$

$$\mathbf{c}_{m,i}|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})},\ 1 \le j \le N_i, \tag{29}$$

$$\tilde{\mathbf{c}}_{m,i}|^{(k+1)} = N_i\mathbf{c}_{m,i}|^{(k+1)} - \frac{N_i-1}{N_i}\sum_{j=1}^{N_i}\mathbf{c}_{m,i}|_{-j}^{(k+1)}; \tag{30}$$

$$\sigma_{m,i}^2|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}|^{(k)}\|^2}{d\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})}, \tag{31}$$

$$\sigma_{m,i}^2|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}|^{(k)}\|^2}{d\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})},\ 1 \le j \le N_i, \tag{32}$$

$$\tilde{\sigma}_{m,i}|^{(k+1)} = N_i\sigma_{m,i}^2|^{(k+1)} - \frac{N_i-1}{N_i}\sum_{j=1}^{N_i}\sigma_{m,i}^2|_{-j}^{(k+1)}, \tag{33}$$

$$\beta_{m,i}|^{(k+1)} = \frac{1}{N_i}\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i}), \tag{34}$$

$$\beta_{m,i}|_{-j}^{(k+1)} = \frac{1}{N_i-1}\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i}),\ 1 \le j \le N_i, \tag{35}$$

$$\tilde{\beta}_{m,i}|^{(k+1)} = N_i\beta_{m,i}|^{(k+1)} - \frac{N_i-1}{N_i}\sum_{j=1}^{N_i}\beta_{m,i}|_{-j}^{(k+1)}. \tag{36}$$

If the sample set $\{\mathbf{x}_{n,i}\}_{n=1}^{N_i}$ is large, deleting one sample at a time may requires too much computation and, instead, a subset can be removed at a time.

Since the root of the numerical problem appears to lie in the computation of variances, the Jack-knife technique could be applied only to the variance updating, and this would limit the increase in computational complexity to a minimum. However, we have noticed that using the Jack-knife technique only on the variance updating, although alleviating numerical difficulties to some degree, is not as robust as applying the Jack-knife technique to update all the parameters. The computations of mixing coefficients, variances and mean vectors are apparently interlinked.

7. The experiments

An artificial-data case, the XOR problem, and a real-data case, the prediction of the UK private construction industry, were used in our experiments. Four PNNs were used in the comparative study:

1. the original PNN;
2. the homoscedastic PNN trained by the ML learning algorithm;
3. the heteroscedastic PNN trained by the ML learning algorithm;
4. the robust heteroscedastic PNN trained by the robust ML learning algorithm.

The original PNN adopted in this study was homoscedastic and was trained by splitting the training data set into two parts, one part used as the mean vectors of pattern units and the other, called the validation set, used for searching an optimal value of the common variance. The variance started from a small value and was gradually increased. The classification performance on the validation set was observed for each trial value of the variance, and learning was terminated when a minimum misclassification rate was achieved. The mixing coefficients in the original PNN were fixed to $\beta_{i,j} = 1/M_j$ for $1 \le i \le M_j$ and $1 \le j \le K$.

7.1. The two-dimensional XOR problem

The two classes in this two-dimensional XOR problem were defined as:

$$\text{Class one} \begin{cases} (0.0 < x < 0.5 \text{ and } 0.5 < y < 1.0) \\ \text{or} \\ (0.5 < x < 1.0 \text{ and } 0.0 < y < 0.5) \end{cases} \tag{37}$$

$$\text{Class two} \begin{cases} (0.0 < x < 0.5 \text{ and } 0.0 < y < 0.5) \\ \text{or} \\ (0.5 < x < 1.0 \text{ and } 0.5 < y < 1.0) \end{cases} \tag{38}$$

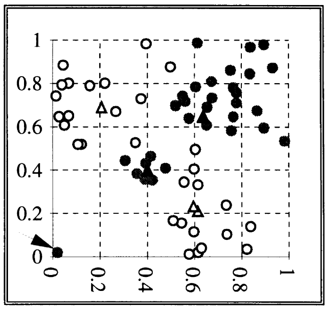
7.1.1. Case A

The training data set, depicted in Fig. 1, contained 61 patterns. Among the training data, 30 class-one patterns were randomly generated inside the set defined by Eq. (37), 30 class-two patterns were randomly generated inside a subset of class two, defined by

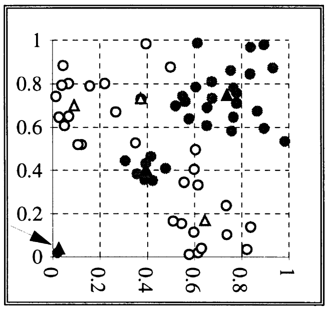
$$(0.3 < x < 0.5 \text{ and } 0.3 < y < 0.5) \text{ or } (0.5 < x < 1.0 \text{ and } 0.5 < y < 1.0),$$

and an extra class-two pattern was inserted at (0.03, 0.03) to serve as an outlier (pointed by the arrow in Fig. 1). The open and filled circles in Fig. 1 denote class-one and class-two patterns, respectively. The open and filled triangles are the positions of the pattern units after training for the two class PDFs, respectively. A separate testing data set of 60 patterns, 30 for each class, was randomly generated using Eqs. (37) and (38). The percentage of correct classification on the testing set using the original PNN was 96.77%, and the classification accuracies of the other three PNNs with different sizes on the testing set are listed in Table 1.

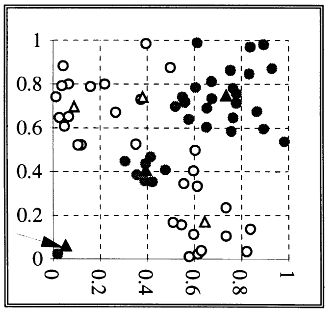
The homoscedastic PNN in this case never achieved the accuracy of the original PNN. From Fig. 1(a), it can be seen that some mean vectors of the homoscedastic PNN were merged together. The two heteroscedastic PNNs had better layouts of the pattern units. The heteroscedastic PNN, however, failed when the number of pattern units was increased to six or more. The reason for the heteroscedastic PNN to collapse was that a mean vector approached the outlier inserted at the edge of the data space with a variance getting too small. Fig. 2(a) depicts the learning trajectory of this mean vector where the radii of the dotted circles denote the square roots of the related variance values. Fig. 2(b) shows



(a)



(b)



(c)

Fig. 1. The distribution of training patterns (○ and ●) and the layout of the pattern units (△ and ▲) for the XOR problem (case A). The number of pattern units is six, and the arrow points to the outlier. (a) Homoscedastic PNN; (b) heteroscedastic PNN; (c) robust heteroscedastic PNN.

that for the robust heteroscedastic PNN, although the same pattern unit was still approaching the outlier, its variance did not become too small.

7.1.2. Case B

A different training set of 60 patterns was randomly generated using Eqs. (37) and (38). The testing set was identical to the case A. The percentage of correct classification on the testing set using the original PNN was 96.67%. Table 2 summarises the testing results for the other three PNNs, and Fig. 3 shows the layouts of pattern units for these three PNNs. Again, the performance of the homoscedastic PNN was the poorest and the robust heteroscedastic PNN the best. For this case, the heteroscedastic PNN collapsed when the number of pattern units was increased to 10 or

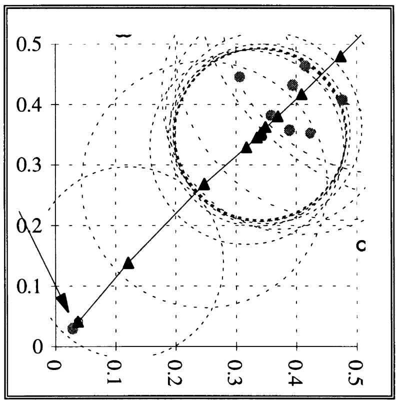
Table 1
The percentage of correct classification for the XOR problem (case A)

Pattern units	Homoscedastic PNN	Heteroscedastic PNN	Robust heteroscedastic PNN
4	89.03%	95.16%	88.71%
6	87.42%	Fail	96.77%
8	89.03%	Fail	98.39%
10	89.03%	Fail	100%

more. The root of this numerical problem was a variance approaching zero, as illustrated in Fig. 4. The robust heteroscedastic PNN did not suffer from the same problem.

7.2. The company prediction problem

The objective of company prediction is to build a model based on historical financial data of companies for forecasting an issuing company's future. In this study, we focus on the UK private construction industry. The raw data set contained annual financial statements of 2408 UK private construction companies from 1989 to 1994. Among this set of data, there were 2244 successful companies and 164 failed companies. Based on these raw data, eight key financial ratios were computed from each year's financial statement of a company (Yang et al., 1997) to provide a pattern.


(a)

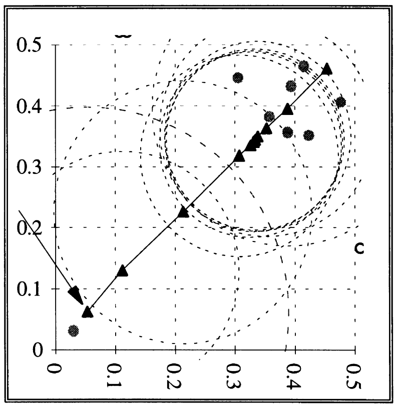
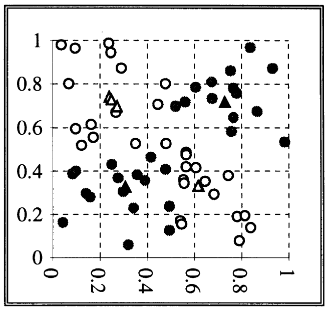
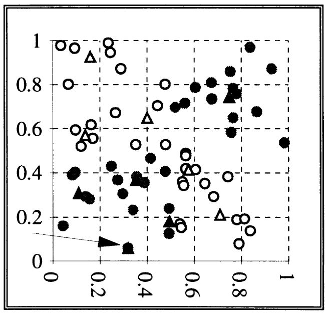

(b)

Fig. 2. The learning trajectory (triangles linked by line) of the mean vector which approaches an isolated pattern. Radii of dotted circles are the corresponding standard deviations. (a) Heteroscedastic PNN; (b) robust heteroscedastic PNN. In (a), the arrow points to the position just before the algorithm collapsed. In (b), the arrow points to the final position.


(a)


(b)

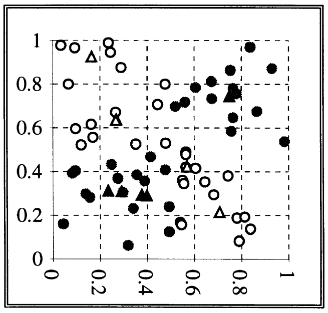

(c)

Fig. 3. The distribution of training patterns (○ and ●) and the layout of the pattern units (△ and ▲) for the XOR problem (case B). The number of pattern units is 10, and the arrow points to where the numerical problem occurred. (a) Homoscedastic PNN; (b) heteroscedastic PNN; (c) robust heteroscedastic PNN.

Table 2
The percentage of correct classification for the XOR problem (case B)

Pattern units	Homoscedastic PNN	Heteroscedastic PNN	Robust heteroscedastic PNN
4	93.33%	93.33%	98.33%
6	93.33%	98.33%	98.33%
8	93.33%	98.33%	98.33%
10	93.33%	Fail	98.33%
12	93.33%	Fail	98.33%
14	93.33%	Fail	96.67%
16	93.33%	Fail	98.33%
18	93.33%	Fail	100%

The pattern space was, thus, eight-dimensional. All the failed companies and those successful companies matching the failed companies in size and accounting years were pooled together. Overall, there were 171 (96 successful and 75 failed) patterns for training and 1262 (1144 successful and 118 failed) patterns for testing.

The original PNN was first applied to this real-data problem and the percentage of correct classification on the testing data set was 71.52%. Table 3 lists the classification accuracies on the testing set for the other three PNNs. Again, the homoscedastic PNN had the poorest performance, and the heteroscedastic PNN failed when the number of pattern units was increased to 10, whilst the robust heteroscedastic PNN did not suffer from this numerical problem. Table 4 summarises the classification errors for the three PNNs with four, six and eight pattern units, where a type I error is defined as a failed company being misclassified as a successful one, and a type II error is defined as a successful company being misclassified as a failed one. The distribution of the training patterns and the layouts of the pattern units for the three PNNs are plotted in Fig. 5, where an open circle denotes a successful company pattern and a filled circle a failed company pattern. For display purposes, the original eight-dimensional space was scaled into a two-dimensional one in Fig. 5 using the Sammon mapping (Sammon, 1969). The convergence rate of the robust ML training algorithm was also investigated.

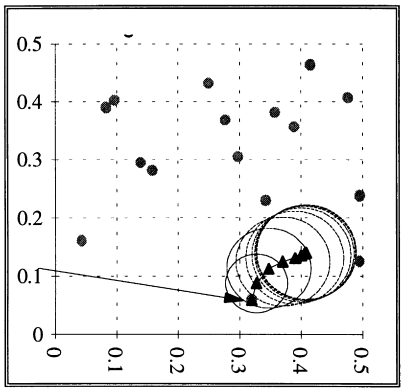


Fig. 4. The learning trajectory (triangles linked by line) of the mean vector which approaches an isolated pattern. Radii of dotted circles are the corresponding standard deviations. The arrow points to the position just before the algorithm collapsed.

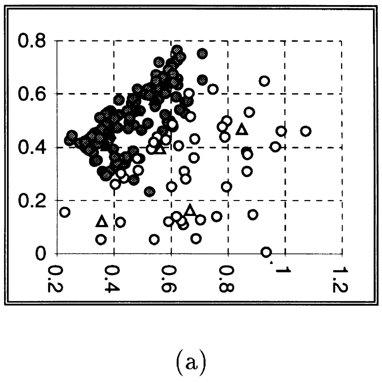
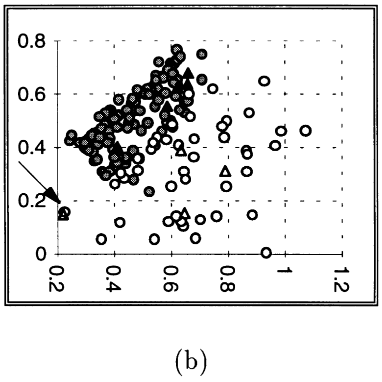
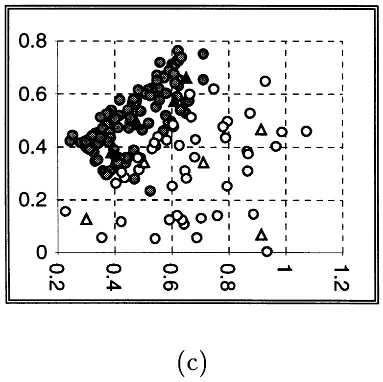




Fig. 5. The distribution of training patterns (○ and ●) and the layout of the pattern units (△ and ▲) for the company prediction problem. The original eight-dimensional space is scaled using Sammon mapping. The number of pattern units is 10, and the arrow points to where the numerical problem occurred. (a) Homoscedastic PNN; (b) heteroscedastic PNN; (c) robust heteroscedastic PNN.

Table 3
The percentage of correct classification for the company prediction problem

Pattern units	Homoscedastic PNN	Heteroscedastic PNN	Robust heteroscedastic PNN
4	62.13%	65.55%	65.61%
6	65.36%	68.10%	68.10%
8	66.60%	74.19%	74.50%
10	67.72%	Fail	74.19%
12	68.41%	Fail	68.10%
14	67.48%	Fail	74.19%
16	70.02%	Fail	74.19%

Table 4
Classification errors for the company prediction problem. The testing data set contains 118 failed company patterns and 1144 successful company patterns. The percentage of type I error is defined as the ratio of the type I errors to the number of failed patterns, and the percentage of type II error is defined as the ratio of the type II errors to the number of successful patterns

Pattern units	Homoscedastic PNN		Heteroscedastic PNN		Robust heteroscedastic PNN	
	Type I	Type II	Type I	Type II	Type I	Type II
4	34	444	41	394	40	400
	(28.81%)	(38.81%)	(34.75%)	(34.44%)	(33.90%)	(34.97%)
6	38	399	42	359	41	362
	(32.20%)	(34.88%)	(35.59%)	(31.38%)	(34.75%)	(31.64%)
8	61	361	16	310	12	310
	(51.69%)	(31.56%)	(13.56%)	(27.10%)	(10.17%)	(27.10%)

As illustrated in Fig. 6, the algorithm converged within 10 iterations for the company prediction problem.

8. Conclusions

The PNN is a powerful means to approximate the Bayes strategy for pattern classification. The heteroscedastic PNN is particularly attractive in practice because it can provide a very parsimonious model for PDF estimation. In this study, we have demonstrated that the optimal ML learning based on the EM algorithm often suffers from numerical difficulties when applied to heteroscedastic PNNs. To overcome this numerical problem, we have proposed a robust ML learning method by incorporating a robust statistical technique known as the Jack-knife with the EM algorithm. Applications to an artificial-data problem and a complex real-data problem have confirmed the effectiveness of this robust ML learning method.

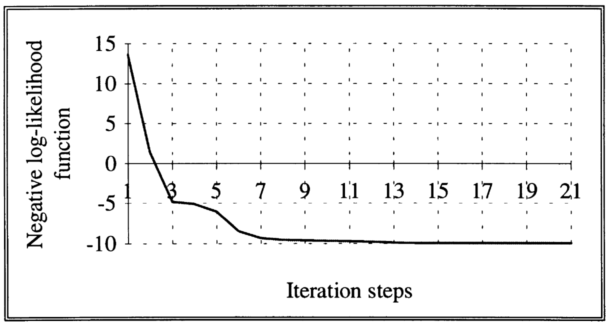


Fig. 6. The learning curve of the robust ML algorithm for the company prediction problem. The number of pattern units is 10.

Acknowledgements

One of the authors, Z.R.Y., thanks the support from the department of Land and Construction Management at the University of Portsmouth, in particular, the support from the head of the department, Professor Bill Seabrooke.

References

Chatfield, C. (1987). *The analysis of time series: An introduction*, 3rd ed. London: Chapman and Hall.
Dempster A. P., Laird N. M., & Rubin D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society (B)*, *39*, 1–38.
Devijer, P., & Kittler, J. (1982). *Pattern recognition: A statistical approach.* Englewood Cliffs, NJ: Prentice-Hall.
Efron B., & Gong G. (1983). A leisurely look at the bootstrap, the Jack-knife, and cross-validation. *The American Statistician*, *37*, 36–48.
Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap.* London: Chapman and Hall.
Lachenbruch P. A. (1967). An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. *Biometrics*, *9*, 639645.
Miller R. G. (1974). The jackknife—a review. *Biometrika*, *61*, 1–15.
Parzen E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, *3*, 1065–1076.
Sammon J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *18*, 401–409.

Specht D. F. (1988). Probabilistic neural networks for classification, mapping, or associative memory. *Proceedings of the IEEE International Conference on Neural Networks*, *1*, 525–532.

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

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

Tam, K. Y., & Kiang, M. Y. (1993). Managerial applications of neural networks: the case of bank failure predictions. In R. R. Trippi & E. Turban (Eds.), *Neural networks in finance and investing* (pp. 193–228). Chicago, IL: Probus Publishing Company.

Wu C. (1983). On the convergence properties of the EM algorithm. *Annals Statistics*, *11*, 95–103.

Xu L., & Jordan M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, *8*, 129–151.

Yang, Z. R., James, H., & Packer, A. (1997). The prediction of UK private construction industry. In *COBRA Construction and Building Research Conference of The Royal Institution of Chartered Surveyors*, September 10–12, University of Portsmouth, UK.