Novelty Detection and Neural Network Validation

Chris M. Bishop

Neural Computing Research Group
Department of Computer Science
Aston University
Birmingham, B4 7ET, U.K.
c.m.bishop@aston.ac.uk

(May 1994)

To be published in IEE Proceedings,
Special Issue on Applications of Neural Networks

Abstract

One of the key factors limiting the use of neural networks in many industrial applications has been the difficulty of demonstrating that a trained network will continue to generate reliable outputs once it is in routine use. An important potential source of errors arises from *novel* input data, that is input data which differ significantly from the data used to train the network. In this paper we investigate the relationship between the degree of novelty of input data and the corresponding reliability of the outputs from the network. We describe a quantitative procedure for assessing novelty, and we demonstrate its performance using an application involving the monitoring of oil flow in multi-phase pipelines.

1 Introduction

Neural networks have been shown to have a useful degree of performance in a wide range of industrial and medical applications. However, a key factor limiting the widespread implementation of neural network solutions in many areas has been the difficulty of demonstrating that the outputs generated by the network in the field are reliable. In general, the problem of network validation is a difficult one, and involves many issues, some of which are generic to any software system. Here we consider only those aspects which are specific to neural networks and related methods.

It is useful to distinguish between two broad levels of network validation. At the first level we provide, in addition to the network outputs, some associated measure of confidence in order that potentially unreliable outputs can be detected. In many applications, this will give a useful degree of validation, since it could allow the bulk of the input data to be processed with high confidence, with remaining data either being discarded or processed by other means. An example would be the automation of medical image interpretation for mass screening programmes in which images which gave network outputs having a low confidence could be rejected by the network and interpreted instead by human experts. The second level of validation requires a guarantee that the network outputs are reliable for any achievable input vector. This represents a much more rigorous degree of validation, and would be appropriate for networks used in safety critical applications, for example. In this paper we shall consider only the first level of validation.

Intuitively we expect that a network will generate reliable results when presented with data which are similar to those used during training, but that, when substantially *novel* data are presented, the network outputs will be prone to serious error. In Section 2 we investigate the relationship between novelty of input data and validity of network outputs, and we use this as the basis of a practical system for network validation [2]. The technique is illustrated in Section 3 using an example from the monitoring of multi-phase flows in oil pipelines. Possible extensions of this approach are discussed in Section 4.

2 Network Validation

Consider a feedforward network trained by minimizing a sum-of-squares error function. If we denote the joint probability density functions for the training data by $p(\mathbf{x}, t_j)$, then we can write the error in the form

$$E = \sum_{j=1}^{m} \int [y_j(\mathbf{x}; \mathbf{w}) - t_j]^2 \, p(\mathbf{x}, t_j) \, d\mathbf{x} \, dt_j \tag{1}$$

where $j = 1, \ldots, m$ labels the output units, $\mathbf{x}$ is the input vector to the network, y_j denotes the output from unit j, and t_j is the target value for that unit. The network corresponds to a set of functional mappings $y_j(\mathbf{x}; \mathbf{w})$, parameterised by a set of weights and biases $\mathbf{w}$ whose values are found by minimizing E.

We note that the joint density $p(\mathbf{x}, t_j)$ can be factored into the product of the unconditional density of the input data $p(\mathbf{x})$ and the conditional density of the target data $p(t_j \mid \mathbf{x})$. After some simple algebra, we can then rewrite the error (1) in the form

$$\begin{aligned} E &= \sum_{j=1}^{m} \int [y_j(\mathbf{x};\mathbf{w}) - \langle t_j \mid \mathbf{x}\rangle]^2 \, p(\mathbf{x}) \, d\mathbf{x} \\ &+ \sum_{j=1}^{m} \int \left\{ \langle t_j^2 \mid \mathbf{x}\rangle - \langle t_j \mid \mathbf{x}\rangle^2 \right\} \, p(\mathbf{x}) \, d\mathbf{x} \end{aligned} \tag{2}$$

where we have defined the conditional averages of the target data as

$$\langle t_j \mid \mathbf{x}\rangle \equiv \int t_j \, p(t_j \mid \mathbf{x}) \, dt_j \tag{3}$$

$$\langle t_j^2 \mid \mathbf{x}\rangle \equiv \int t_j^2 \, p(t_j \mid \mathbf{x}) \, dt_j \tag{4}$$

Note that only the first term in (2) depends on the network weights. Provided the functions $y_j(\mathbf{x};\mathbf{w})$ have sufficient flexibility, for instance if they correspond to a network with a sufficient number of hidden units, then the minimum of this error function occurs when

$$y_j(\mathbf{x};\mathbf{w}) = \langle t_j \mid \mathbf{x}\rangle \tag{5}$$

so that the network outputs represent the *regression* of the target data, conditioned on the input vector. Note that we are implicitly assuming an infinitely large data set, as we have replaced sums over data points with integrals over distributions. For finite data sets, it is necessary to limit the complexity of the network in order to control the trade-off between the bias and the variance of the solution. In principle, the solution given by equation (5) would then be obtained by considering a sequence of ever larger data sets, and correspondingly more flexible network functions, such that, in the limit, both the bias and the variance of the solution are made to vanish [3].

The neural network solution represented by equation (5) can be regarded as optimal in the sense that, if the training data were generated from a deterministic function with superimposed zero-mean noise, then the network will average over the noise and learn the underlying function. Similarly, for classification problems in which the training data are labelled using a 1–of–N target coding scheme, equation (5) implies that the network outputs represent the *posterior* probabilities of the input vector $\mathbf{x}$ belonging to the corresponding classes, and so again they can be regarded as optimal. Note that the residual value of the error function at the minimum is given by the second term in (2), which represents the average variance of the noise on the data. Since this term is independent of the network weights, it plays no rôle in network training.

The key point to note is that the first term in the error function of equation (2) is weighted by $p(\mathbf{x})$ which represents the *unconditional* density of the input data. In a practical application, the goal of network training is to find a good approximation to the regression by minimization of a sum-of-squares error defined over a finite training set. From equation (2) we expect this approximation to be most accurate in regions of input space for which the density $p(\mathbf{x})$ is high, since only then does the error function penalize the network mapping if it differs from the regression. In regions of input space where $p(\mathbf{x})$ is small, there is little contribution to E if the network outputs $y_j(\mathbf{x};\mathbf{w})$ and the regression functions $\langle t_j \mid \mathbf{x}\rangle$ differ significantly. A similar argument applies to other error functions such as the cross entropy.

This suggests that the unconditional density can provide an appropriate quantitative measure of novelty with respect to the training data set. If a new input vector falls in a region of input space for which the density $p(\mathbf{x})$ is high then the network is effectively interpolating between training data points and the network performance will generally be good. If the input vector falls in a region of input space for which $p(\mathbf{x})$ is low, then the input data is essentially *novel* and the network could easily generate erroneous outputs.

We therefore arrive at the following procedure for validating network outputs. The data which are used to train the network are also used to construct an estimate $\widehat{p}(\mathbf{x})$ of the (unknown) density $p(\mathbf{x})$. Standard cross-validation (using partitions of the training set) may be used at this point to optimize the network topology and the values of regularization parameters, and this can also serve to optimize any smoothing parameters in the density model $\widehat{p}(\mathbf{x})$. When the network is in use, each new input is presented to the trained network, and is also used to evaluate $\widehat{p}(\mathbf{x})$ to provide a quantitative measure of the degree of novelty of the input vector. Input vectors $\mathbf{x}$ which have relatively small values of $\widehat{p}(\mathbf{x})$ are those which are likely to generate spurious outputs. This is illustrated schematically in Figure 1.

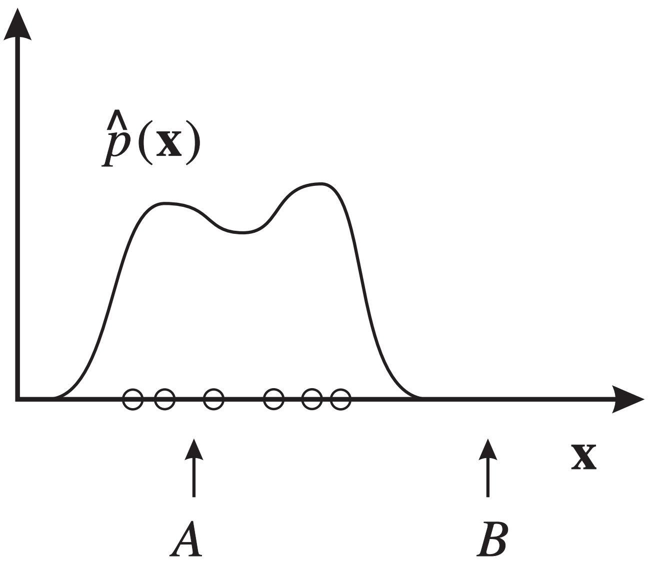

Figure 1: Schematic illustration of the use of the unconditional probabilty density to measure the degree of novelty of new input vectors. Here $\mathbf{x}$ denotes the vector of input variables to the network, the circles denote the training data, and the curve depicts a model $\widehat{p}(\mathbf{x})$ for the unconditional density estimated from the training data points. A new input A, for which $\widehat{p}(\mathbf{x})$ is large, is considered to be similar to the training data, while an input B, for which $\widehat{p}(\mathbf{x})$ is small, is considered *novel*.

From (2) we see that the sum-squared error between the network outputs and the regression is weighted by a variance factor of the form

$$\sigma_y(\mathbf{x}) = \{p(\mathbf{x})\}^{-1/2} \tag{6}$$

and this can be used to assign $\mathbf{x}$-dependent error bars to the network outputs $y_j(\mathbf{x})$, via the model $\widehat{p}(\mathbf{x})$. Thus, for each new input vector $\mathbf{x}$, we obtain the values of the network outputs y_j, together with a value σ_y for the confidence associated with this value. Note that this confidence value reflects the uncertainty in the value of the network estimate for the regression, and is distinct from the variance of the target values around their conditional mean as determined by the intrinsic noise on the target data itself.

In some applications it will be more appropriate to place a threshold on $\widehat{p}(\mathbf{x})$ and to reject all new data points for which $\widehat{p}(\mathbf{x})$ falls below this threshold. The justification for such an approach can be seen as follows. In applying a threshold we are implicitly classifying all new input vectors into one of two classes: those which are similar the training data, which we denote by class $\mathcal{C}_1$, and those which are *novel*, which we denote by class $\mathcal{C}_2$. The training data are assumed to be drawn entirely from $\mathcal{C}_1$, while new data could arise from either class, with *prior* probabilities $P(\mathcal{C}_1)$ and $P(\mathcal{C}_2)$ respectively, where $P(\mathcal{C}_1) + P(\mathcal{C}_2) = 1$. Given a new input vector $\mathbf{x}$ we wish to assign it to one of the two classes in such a way as to minimize the probability of misclassification. This is achieved when the vector is assigned to the class having the larger *posterior* probability [4], so that the vector is assigned to class $\mathcal{C}_1$ if

$$P(\mathcal{C}_1 \mid \mathbf{x}) > P(\mathcal{C}_2 \mid \mathbf{x}) \tag{7}$$

and to class $\mathcal{C}_2$ otherwise. From Bayes theorem, the posterior probability of the vector $\mathbf{x}$ belonging to class $\mathcal{C}_i$ is given by

$$P(\mathcal{C}_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_i)P(\mathcal{C}_i)}{\pi(\mathbf{x})} \tag{8}$$

where the unconditional density $\pi(\mathbf{x})$ is given by

$$\pi(\mathbf{x}) = P(\mathbf{x} \mid \mathcal{C}_1)P(\mathcal{C}_1) + P(\mathbf{x} \mid \mathcal{C}_2)P(\mathcal{C}_2) \tag{9}$$

which acts as a normalization factor and ensures that the posterior probabilities sum to unity

$$P(\mathcal{C}_1 \mid \mathbf{x}) + P(\mathcal{C}_2 \mid \mathbf{x}) = 1 \tag{10}$$

Since the denominator in (8) is the same for both classes, it can be omitted from the comparison of probabilites needed to assign $\mathbf{x}$ to a particular class. Using (7) and (8), we therefore assign $\mathbf{x}$ to class $\mathcal{C}_1$ if

$$p(\mathbf{x} \mid \mathcal{C}_1) > \frac{p(\mathbf{x} \mid \mathcal{C}_2)P(\mathcal{C}_2)}{P(\mathcal{C}_1)} \tag{11}$$

and otherwise to class $\mathcal{C}_2$. The quantity $p(\mathbf{x} \mid \mathcal{C}_1)$ is the probabilty density from which the training data were drawn, which we have modelled by $\widehat{p}(\mathbf{x})$. The density $p(\mathbf{x} \mid \mathcal{C}_2)$ represents the distribution of novel data, and so by definition we can have little idea of what form it should take. The simplest assumption is to take it to be constant over some large region of input space, falling to zero outside this region to ensure that the density function can be normalized. (In many applications, such as the one discussed in the next section, the range of possible input values will in fact be bounded). In this case the condition (11) is then equivalent to a threshold on $\widehat{p}(\mathbf{x})$. This is illustrated in Figure 2.

More generally, we might suppose that the density of novel data should be smaller in regions of input space where the density $\widehat{p}(\mathbf{x})$ is large. Thus we could choose $p(\mathbf{x} \mid \mathcal{C}_2)$ to be of the

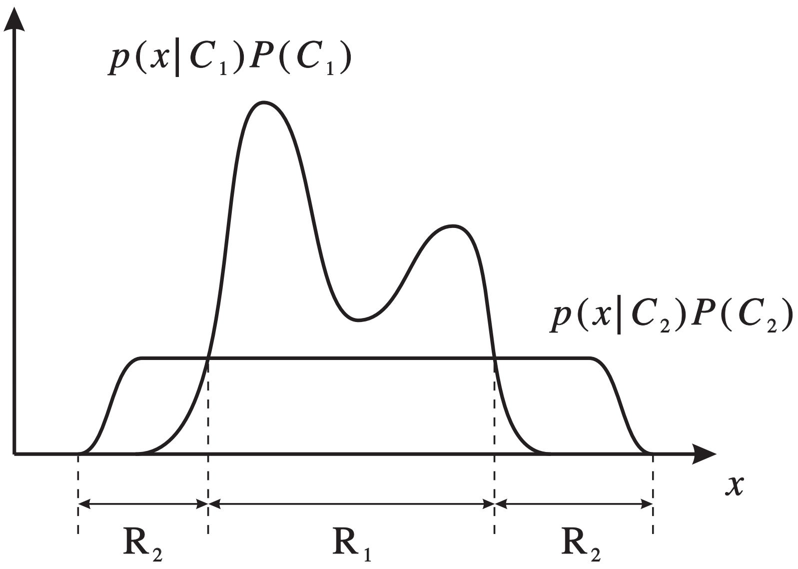

Figure 2: Illustration of the Bayesian formalism for determining whether a new input vector is novel. We assume that the training data has arisen from class $\mathcal{C}_1$ with prior probability $P(\mathcal{C}_1)$ and distribution $p(\mathbf{x} \mid \mathcal{C}_1)$ (which we model using $\widehat{p}(\mathbf{x})$). Similarly, we assume that novel data is drawn from class $\mathcal{C}_2$ with prior probability $P(\mathcal{C}_2)$ whose distribution function $p(\mathbf{x} \mid \mathcal{C}_2)$ is taken to be constant over some large region of input space. From Bayes theorem, the optimal decision boundary for classifying new data as novel is equivalent to a threshold on $p(\mathbf{x} \mid \mathcal{C}_1)$. This divides the input space into two (possibly disjoint) regions $\mathcal{R}_1$ and $\mathcal{R}_2$ such that new input vectors which fall in the region $\mathcal{R}_2$ are classified as novel.

form $F(p(\mathbf{x} \mid \mathcal{C}_1))$ where $F(\cdot)$ is a monotonically decreasing function. It is easily seen that this too gives rise to a threshold criterion on $p(\mathbf{x} \mid \mathcal{C}_1)$, which takes the form

$$p(\mathbf{x} \mid \mathcal{C}_1) > G^{-1}\left(\frac{P(\mathcal{C}_1)}{P(\mathcal{C}_2)}\right) = \text{const.} \tag{12}$$

where $G(z) \equiv F(z)/z$. Since $F(\cdot)$ is monotonic, so too is $G(\cdot)$, and hence G^{-1} is unique. In a practical application of neural networks, an independent test set (one not used in the training and cross-validation process) is generally used to confirm the performance of the trained network. The same test set can also be used to find a suitable threshold value for $\widehat{p}(\mathbf{x})$. This is achieved by evaluating $\widehat{p}(\mathbf{x})$ for all points in the test set, and then choosing a value for the threshold such that most (or all) of these points are classified as belonging to $\mathcal{C}_1$. This procedure will be illustrated in the next section.

Many conventional techniques exist for constructing probability density estimates $\widehat{p}(\mathbf{x})$ from finite samples [4, 9] and various adaptive 'neural' approaches have also been suggested. In the next section we illustrate these ideas using one of the standard methods for density estimation.

3 An Example Application

In order to illustrate the concept of novelty detection as the basis of a practical system for network validation, we consider a specific industrial application of neural networks concerned with the determination of the oil fraction in a multi-phase oil pipeline. Full details of this application can be found in ref. [1].

In order to reduce costs, the oil industry makes use of multi-phase pipelines to transfer mixtures of oil, water and gas directly from offshore production fields to an onshore facility where the various components can be separated. This has led to the need for an effective non-invasive method for monitoring the oil fraction in such pipes, with sufficient accuracy for reservoir monitoring and custody transfer purposes. The problem is a complex one due to the fact that the multi-phase flow can exhibit a wide variety of configurations, and that numerical modelling of such flows is notoriously difficult.

One technique for determining the phase fractions in the pipeline uses multiple-beam dual-energy gamma densitometry [1], which is based on the attenuation properties of gamma beams passing through the pipe. The degree of attenuation of a gamma beam depends on the particular material in the path of the beam (expressed in terms of an absorption coefficient), the length of material through which the beam passes, and the gamma energy (i.e. wavelength). Figure 3 shows a schematic illustration of a collimated gamma beam passing through a circular cross-section pipe containing oil, water and gas in a stratified configuration.

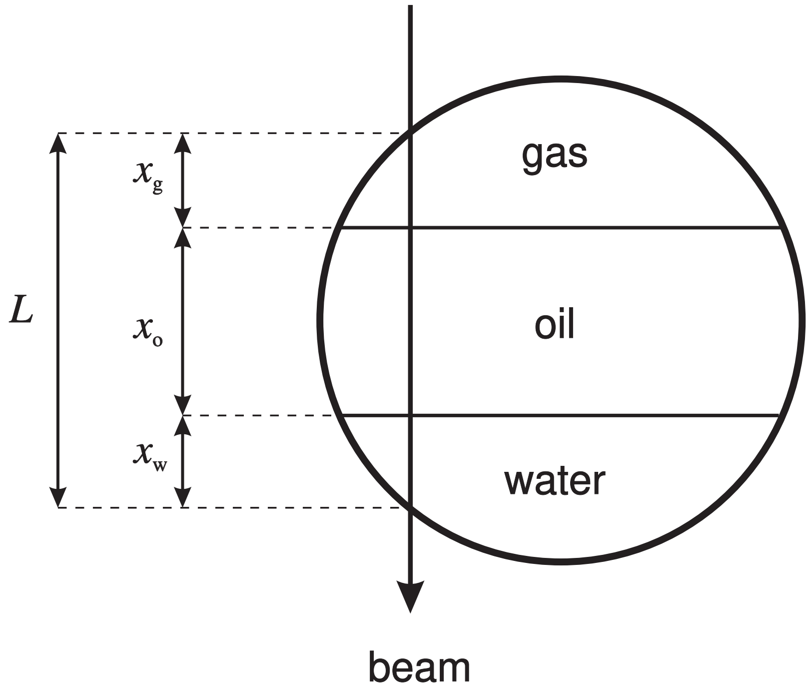

Figure 3: Schematic cross-section of a pipe containing water, oil and gas in a stratified configuration, showing the path of a gamma beam together with the definitions of the path lengths x_w, x_o, x_g and L.

We wish to determine the path lengths x_w, x_o, x_g in each of the three phases. By measuring the attenuation of the beam at two energies we obtain two independent pieces of information. The constraint that the three path lengths must sum to the total path length, so that $x_{\mathrm{w}} + x_{\mathrm{o}} + x_{\mathrm{g}} = L$, provides a third piece of information. Knowing the absorption coefficients of the gamma beam in each of the three phases at each energy, it is then straightforward to evaluate the required path lengths [1].

If measurements could be made along many transverse paths through the pipe it would in principle be possible to perform tomographic reconstruction of the phase configuration and hence calculate the oil fraction. In a practical system, however, only a few lines of sight will be available, and so alternative analysis methods need to be used.

We have investigated linear mapping methods, as well as neural networks based on the multi-layer perceptron (MLP), to analyse the outputs from the densitometer [1]. It is found that the non-linearity offered by the MLP gives a useful improvement in performance over the linear approach, reducing the RMS error on new data by about a factor of 2. We have considered a system having 3 vertical and 3 horizontal beams, with each of the 6 detectors generating two signals corresponding to the attenuation at each of the two wavelengths. These are first pre-processed to extract the fractional path lengths along the lines of sight in the oil and water phases. (The fractional path lengths in the gas phase are not considered since they represent redundant information). The resulting 12 numbers are used as inputs to a multilayer perceptron having a single hidden layer of sigmoidal units followed by two linear output units whose activations represent the fractions of oil and water in the pipeline.

For the purposes of this study, synthetic data have been generated using configurations selected at random from the four examples shown in Figure 4. Note that these configurations are chosen mainly for computational simplicity, and they do not necessarily give accurate representations of real multi-phase flows. They are, however, sufficiently representative for the purposes of the present study. An important feature of this synthetic data is that the distribution of noise can be accurately modelled, as discussed shortly. To generate a data point, a configuration is chosen at random with equal probability from the four shown in Figure 4. The oil, water and gas fractions are then also selected at random by choosing three numbers F_{o}, F_{w} and F_{g} at random with uniform distributions in the range $(0, 1)$ and then calculating the oil fraction f_{o} using

$$f_{\mathrm{o}} = \frac{F_{\mathrm{o}}}{F_{\mathrm{o}} + F_{\mathrm{w}} + F_{\mathrm{g}}} \tag{13}$$

with similar expressions for the water and gas fractions. This ensures that the three fractions sum to unity. The path lengths are then calculated, allowing for the effects of noise. In a practical system, the dominant contribution to the noise arises from the photon statistics due to the limited integration time of the detectors (necessary to avoid averaging over temporal variations in the flow) and the limited source strength of the gamma beams, and has been accurately modelled with the correct Poisson statistics.

For a training set with 1,000 points, the best generalization performance is obtained from a network having 8 hidden units. Figure 5 shows a plot of the predicted oil fraction versus the actual oil fraction for 1,000 points in an independent test set. It is clear that the network can predict the oil fraction to high accuracy. In a practical application of this technique the network would be trained using data from a multi-phase flow test rig in which a variety of

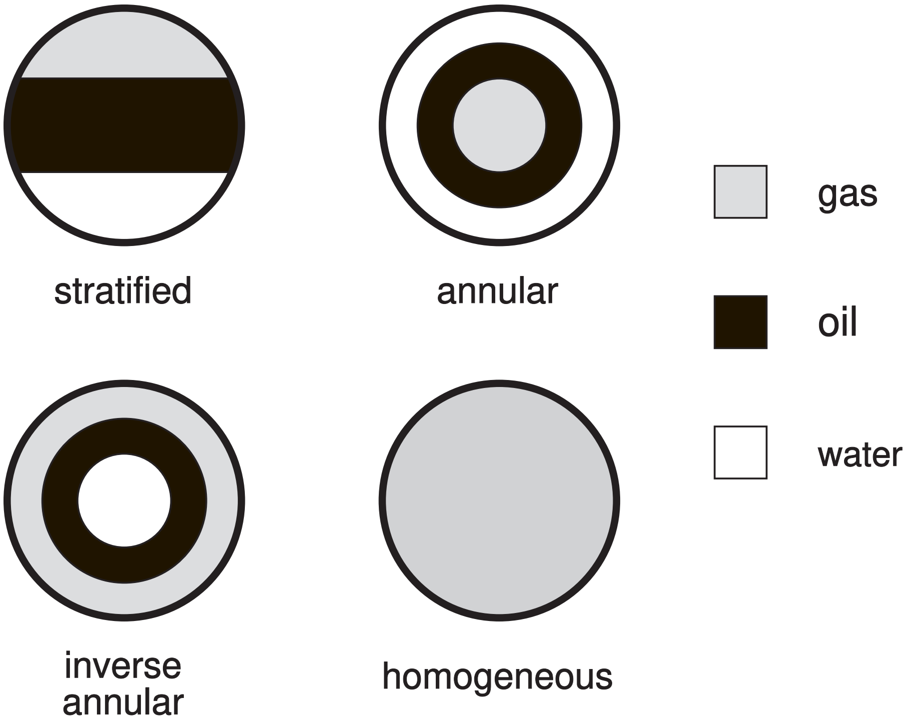

Figure 4: Cross-sections of a pipe illustrating the four multi-phase configurations used for network training. Data generated from a fifth configuration, 'inverted stratified', is used to represent novel data.

conditions, typical of those which arise in practice, can be produced.

In order to determine the degree of novelty of the input data we need a procedure for estimating the unconditional probability density $p(\mathbf{x})$ corresponding to the training data set. For simplicity we have chosen a standard Parzen window approach with Gaussian kernel functions [9]. We shall discuss alternative approaches in Section 4. The Parzen estimator has one kernel function for each point in the training set with its centre given by the corresponding input vector. For Gaussian kernels, this gives a density model of the form

$$\widehat{p}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}\sigma^d} \sum_{q=1}^{n} \exp\left\{ -\frac{|\mathbf{x} - \mathbf{x}^q|^2}{2\sigma^2} \right\} \tag{14}$$

where $\mathbf{x}^q$ represents a data point from the training set (which consists of n points in total), and d is the dimensionality of input space (so that here $d = 12$). The smoothing parameter σ controls the degree of smoothness of the estimated density function and its value must be neither too large (since this gives the estimate a large bias) nor too small (since this gives the estimate a large variance). We have adopted the simple heuristic of setting σ to the average distance of the ten nearest neighbours from each point in the training set, averaged over the whole training set. This gave a value of σ close to that obtained using cross-validation.

In order to test the performance of this novelty detector we have generated a further data set, consisting of 1,000 examples with randomly chosen oil and water fractions, corresponding to a 5$^{\text{th}}$ 'inverted-stratified' configuration which is obtained by inverting the stratified

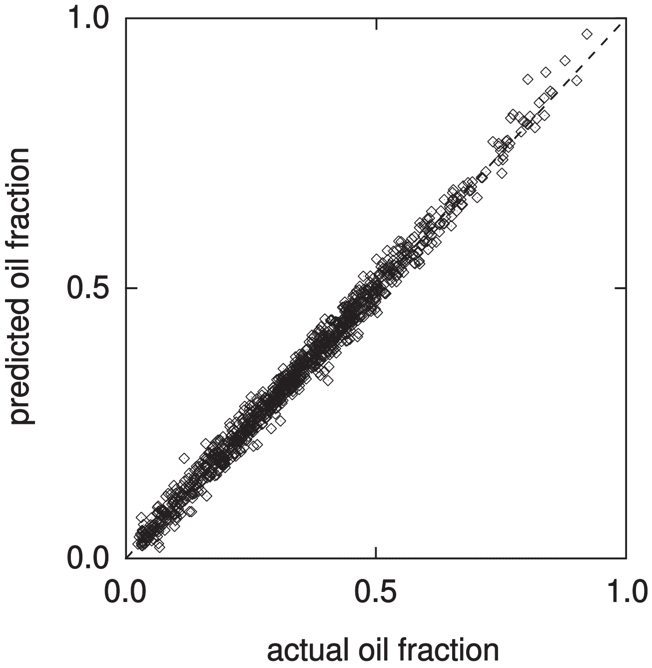

Figure 5: Scatter plot of the oil fraction predicted by the neural network versus the actual oil fraction, for the 1,000 points in the test set.

configuration of Figure 4

The value of the density function $\widehat{p}(\mathbf{x})$ for a new value of $\mathbf{x}$ is generally called a likelihood. Figure 6 shows a plot of the log likelihood versus the magnitude of the error between the oil fraction predicted by the neural network and the true value obtained from the data set, for the 1,000 points from the test set as used to plot Figure 5. We see that the points all have large values for the log likelihood, and that the network output errors are correspondingly small. By comparison, Figure 7 shows the corresponding plot (on the same scale) for the 1,000 examples of the novel multi-phase configuration. The majority of these points have log likelihood values which are substantially smaller than those of the test set points, with a correspondingly larger range of oil fraction errors. We see that the network can indeed generate poor results when presented with data from this new configuration. Such data points can, however, easily be rejected on the basis of their log likelihood values. Setting a threshold anywhere between -5 and -10 would reject all data points having significant phase fraction errors.

It can also be seen from Figure 7 that there are some inverted-stratified points which have log likelihood values comparable to those from the original test set. Examination of the phase fractions for these points shows that they represent configurations which could also be classified as stratified configurations. For instance, if the oil phase fraction is sufficiently large then the three horizontal beam lines pass through oil only, and there exist stratified and inverted-stratified configurations which have the same phase fractions and which give rise to the same 12 path length measurements. The novelty detector 'correctly' interprets these as being similar to the training data, and indeed the network predicts the phase fraction to high accuracy.

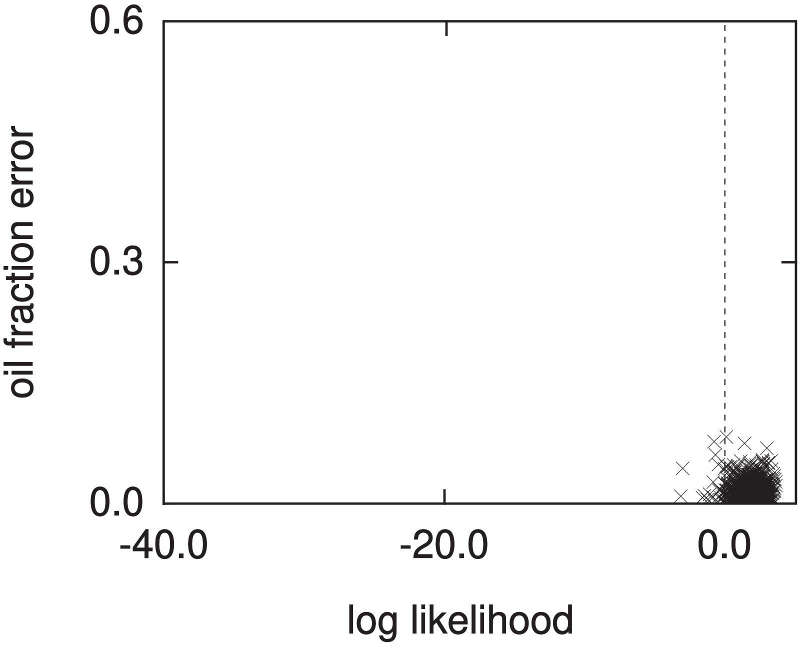

Figure 6: Plot of the magnitude of the oil fraction error from the neural network prediction versus the log likelihood from the novelty detector, showing the 1,000 points from the original test set. These points have large values of the log likelihood (they are not novel) and have correspondingly small values for the network error.

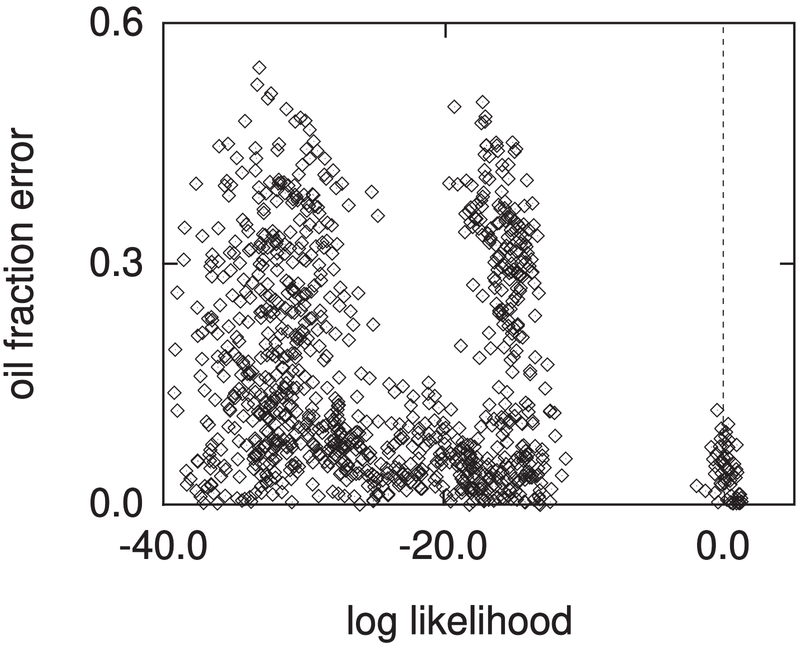

Figure 7: As in Figure 6, but showing the 1,000 points corresponding to inverted-stratified configurations. The majority of these points have small values of likelihood (and hence are regarded as novel) and the corresponding network outputs have a wide range of error values. It is clear that all points with significant errors can be rejected by setting a threshold on the log likelihood anywhere in the range -5 to -10.

4 Discussion

In this paper we have considered the problem of validating the outputs generated by a trained neural network, once it is installed in an application. One of the most important sources of high errors in network predictions arises from *novel* input data, i.e. data which differ significantly from those used to train the network. We have shown that novelty can be quantified by modelling the unconditional probability density of the input data used during training, and subsequently evaluating this density for all new data points. This can be used to assign error bars to the network outputs, or can be used to classify input vectors on the basis of a novelty threshold.

We have illustrated the technique of novelty detection using a simple kernel-based estimator of the unconditional input density. Such an estimator is easy to implement, and fast to train, but is computationally intensive to evaluate for new data points, since the number of kernel functions equals the number of data points in the training set. An alternative approach is to use a semi-parametric estimator, constructed from a Gaussian mixture model of the form

$$\widehat{p}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \phi_i(\mathbf{x}) \tag{15}$$

where

$$\phi_i(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp\left\{-(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right\} \tag{16}$$

where $\boldsymbol{\Sigma}_i$ is the covariance matrix, and the *mixing coefficients* α_i satsify

$$\sum_{i=1}^{m} \alpha_i = 1 \tag{17}$$

The parameters α_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are adaptive parameters of the model, and their values are chosen by maximizing the log likelihood of the training data with respect to the model $\widehat{p}(\mathbf{x})$, given by

$$\log \mathcal{L} = \sum_{q=1}^{n} \log \widehat{p}(\mathbf{x}^q) \tag{18}$$

This can be done using standard optimization algorithms (such as conjugate gradients) or by using re-estimation techniques based on the EM algorithm [6]. The optimum value for the number of components m can be found by cross-validation. By comparison with the simple Parzen method used in Section 3, the Gaussian mixture model requires significantly more computational effort to train. However, since the value of m is generally much smaller than the number of training data points, the computation time needed to evaluate $p(\mathbf{x})$ will typically be much less than with the Parzen estimator. Techniques for density estimation in which the number of basis functions is grown incrementally as part of the learning process have been proposed in ref. [10], and these can also be used to measure novelty. In this case an appropriate threshold for detecting novelty arises naturally as part of the training process.

We have discussed the validation of network outputs in terms of the detection of novelty with respect to the network input vectors. However, many practical applications of neural networks require that the raw input data be preprocessed before presentation to the network. This is especially important for problems with many input variables, from which a relatively small number of features are usually extracted. In such cases the novelty detection should be performed at the inputs to the *preprocessing* stage. This is important since the preprocessing itself is generally not information-preserving (for instance it will often involve a reduction in dimensionality). Thus an input vector which is novel could be mapped to a vector at the inputs to the network which is similar to the training data and which therefore would not be detected as novel.

An exception to this, however, arises when the preprocessing is constructed on the basis of some form of prior knowledge which is believed to hold accurately. For example, we know that the classification of the image of a character should be invariant to translations. In such cases, novel inputs whose novelty arises solely from application of the transformation should be accepted since we expect the network to generate a satisfactory response, and so the novelty detection should be applied *after* the pre-processing.

The above discussion also makes it clear why in general it is incorrect to measure novelty in the set of activations of the hidden units in a network, since the transformation from inputs to hidden units is not in general information-preserving. There is one situation, however, in which the information provided by the activations of the hidden units can be used in the determination of novelty. The hidden units of a radial basis function network can be configured using unsupervised techniques based on density estimation in the input space [5, 8]. This might, for instance, be done using the approach discussed above for adapting the parameters of a Gaussian mixture model. In this case, the resulting basis functions $\phi_i(\mathbf{x})$ can be used directly as the hidden units in a radial basis function neural network, in which the second-layer weights are found by minimizing a sum-of-squares error function using linear techniques. If the values of the mixing coefficients α_i, found from the optimization of the basis functions, are retained they can be used to provide an estimate of the likelihood of new input data. In this case the α_i can be considered as weights to an extra output unit, whose activation, given by (15), represents the likelihood. Such a network combines in a natural way the generation of predictions, and the validation of those predictions, into a single network structure, as indicated in Figure 8 [1]. Note, however, that the optimum choice of basis functions for density estimation need not be the optimum for prediction of the values of y_k.

[1] A related scheme, called the 'validity index' network, was proposed in ref. [7]. An important difference is that the technique described above makes use of a principled method for density estimation based on maximum likelihood with respect to a Gaussian mixture model, rather than the more complex, heuristic, approach discussed in [7].

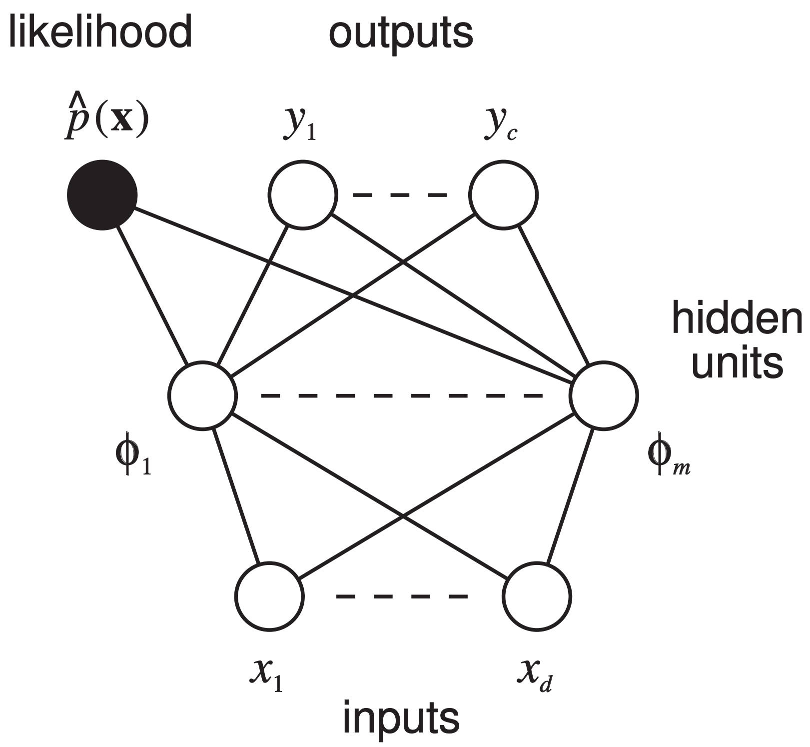

Figure 8: A radial basis function network in which the basis functions are used both in the evaluation of the network outputs $y_j(\mathbf{x})$, and to provide an estimate $\widehat{p}(\mathbf{x})$ of the likelihood of input data for validation of the outputs.

References

[1] Bishop C M and James G D (1993). Monitoring of Multi-phase Flows using Dual-Energy Gamma Densitometry and Neural Networks. *Nuclear Instruments and Methods in Physics Research* **A327** 580–593.

[2] Bishop C M (1993). Neural network validation: an illustration from the monitoring of multi-phase flows. In *Proceedings IEE Third International Conference on Artificial Neural Networks*, Brighton, UK, 41–45.

[3] C M Bishop (1994). *Neural Networks for Statistical Pattern Recognition*, Oxford University Press, to appear.

[4] Duda R O and Hart P E (1973). *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York.

[5] Lowe D (1991). On the iterative inversion of RBF networks: a statistical interpretation. In *Proceedings IEE Second International Conference on Artificial Neural Networks*, Bournemouth, UK, 29–33.

[6] Dempster A P, Laird M N and Rubin D B (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc.* **B 39** 1–38.

[7] Leonard J A, Kramer M A and Ungar L G (1992). Using radial basis functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks* **3** 624–627.

[8] Nowlan S J (1990). Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems* **2**, Ed. D S Touretzky, Morgan Kaufmann, 574–582.

[9] Silverman B W (1986). *Density Estimation*, Chapman and Hall, New York.

[10] Roberts S and Tarassenko L (1993). A probabilistic resource allocating network for novelty detection, *Neural Computation* **6** 270–284.