CHOOSING AN APPROPRIATE MODEL FOR NOVELTY DETECTION

Alexandre Nairac, Timothy A. Corbett-Clark, Ruth Ripley, Neil W. Townsend and Lionel Tarassenko

Neural Networks Research Group, Oxford University Engineering Department
19 Parks Road, Oxford OX1 3PJ, UK
E-mail: {alex,tcc,ruth,neil,lionel}@robots.ox.ac.uk

ABSTRACT

For **novelty detection**, a description of normality is learnt by fitting a *model* to the set of normal examples, and previously unseen patterns are then tested by comparing their novelty score (as defined by the model) against some threshold. Models need to be assessed not only according to their ability to separate normal and abnormal examples but also according to the extent of overfitting relative to the training set. When the amount of training data is small, the effects of overfitting can be estimated by using cross-validation to determine how many *previously unseen* normal patterns would be classified correctly. This is illustrated by considering the problem of identifying unusual jet engine vibration signatures. With a larger training database, a principled approach can be adopted which provides both a measure of the quality of the model and a means of determining the value of the novelty threshold.

NOVELTY DETECTION

In a probabilistic sense, novelty detection is equivalent to deciding whether a test pattern was produced by the *underlying data generator* which produced the training database of normal patterns. This underlying data generator can be defined by its unconditional probability density function $p(\mathbf{x})$, so that a test pattern which belongs to a region of data space where $p(\mathbf{x})$ is low is likely to be novel. A novelty detection network produces an estimate $\hat{p}(\mathbf{x})$ of the unconditional density function $p(\mathbf{x})$, learnt from the patterns in the training database. A test pattern $\mathbf{x}_t$ is then classified as novel if $\hat{p}(\mathbf{x}_t)$ is below some *novelty threshold*. Approaches to novelty detection differ in the structure of the model $\hat{p}(\mathbf{x})$, in whether the novelty threshold is local or global, and in how the novelty threshold is computed.

For example, Bishop [1] models $p(\mathbf{x})$ with a Parzen windows estimator:

$$\hat{p}(\mathbf{x}) = \frac{1}{N(2\pi)^{D/2}\sigma^D} \sum_{j=1}^{N} \exp - \frac{(\mathbf{x}-\mathbf{x}_j)^2}{2\sigma^2} \quad (1)$$

which fits one spherical gaussian kernel of radius σ to each of the N training patterns $\mathbf{x}_j$; note that σ is a *global* smoothing parameter, which is the same for all Gaussians. Tarassenko *et al* [4] argue that a *local* representation may be preferable for some problems when there are significant variations in data density for the patterns in the training database. For example, data space could be partitioned using clustering techniques and a different value of the smoothing parameter σ could then be used in each partition.

Alternatively, the probability density function $p(\mathbf{x})$ can be modelled by a mixture of Gaussians:

$$\hat{p}(\mathbf{x}) = \sum_j p(\mathbf{x} \mid j)\, P(j), \quad (2)$$

where

$$p(\mathbf{x} \mid j) = \frac{\exp -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}-\boldsymbol{\mu}_j)}{(2\pi)^{D/2}\,|\boldsymbol{\Sigma}_j|^{1/2}} \quad (3)$$

In this case, it becomes necessary to determine suitable values of $P(j)$, $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ by maximising the log likelihood $\sum \log \hat{p}(\mathbf{x})$ over all training patterns.

CASE STUDY: JET ENGINE VIBRATION DATA

This section presents a solution to the problem of recognising unusual jet engine vibration signatures, as an example of how to apply the principles of novelty detection when the amount of available training data is limited.

The vibration data

Before a newly-built jet engine is delivered to the customer, it is submitted to a number of pass-off tests, one of which records carcass vibrations over the full range of jet engine operating speeds (see [2] for more details). As the engine speed is changed, shaft tachometers are monitored continuously and data are collected from the vibration gauges at consecutive engine speeds. Each set of vibration data is then transformed into an amplitude spectrum.

Each amplitude spectrum contains a small number of peaks, corresponding to the important harmonics of the rotation frequency of

one of the engine shafts[1]. A *tracked order* is the amplitude of the signal in a narrow frequency band centered on a harmonic of the rotation frequency of a shaft, measured as a function of engine speed. It tracks the frequency response of the engine to the energy injected by the rotating shaft. Tracked orders are named after the harmonic which they represent, so that, for example, the 2HP tracked order corresponds to the second harmonic of the high pressure (HP) shaft. Since tracked orders account for a significant proportion of the total vibration energy, they provide useful diagnostic information. It is known that a number of engine anomalies may be characterised by abnormal shapes of the main tracked orders. A feature vector $\mathbf{x}$ is therefore constructed to describe the shape of the three main tracked orders (1HP, 2HP and 1LP). Data from normal engines is then collected in order to construct the estimator $\hat{p}(\mathbf{x})$ of normality.

Data preprocessing

It is important for the dimension D of the feature vector $\mathbf{x}$ to be kept relatively small, because the number of training examples required to produce a good general model is typically much larger than the input dimensionality.

The shape information for each of the three tracked order is encoded as a low-dimensional vector by calculating a weighted average over **six** different speed ranges according to:

$$\tilde{A}_j = \int_{-\infty}^{+\infty} a(s)\,\omega_j(s)\,ds, \qquad (4)$$

where s is the engine speed, $a(s)$ is the tracked order value and $\omega_j(s)$ is the weighting function for the j^{th} speed range ($1 \leq j \leq 6$). The partly overlapping weighting functions cover the range of speeds over which the engine is tested (see figure 1).

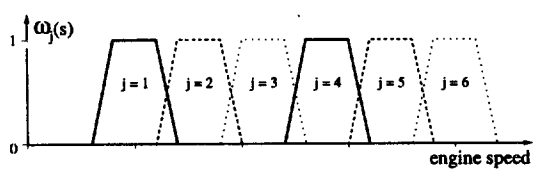

Figure 1: *The six partly-overlapping trapezoidal weighting functions used to encode tracked order shape.*

The tracked order information from each of the 52 engines available in the database of normal engines is in this way compressed into an 18D shape vector (3 tracked orders with 6 components each). Despite the data compression, the small number of normal examples prevents us from training a full Gaussian mixture model (see equations 2 and 3). Instead, a normalising transformation is applied to the whole of the training data and a small number of spherical basis functions (kernels) are then fitted to the transformed data. Hence the complexity of the model is largely determined by the normalising transformation which is applied to the data. Two linear transformations are considered, both of which have the same general form:

$$\mathbf{x}' = \mathbf{\Lambda}^{-1/2}\mathbf{V}^T(\mathbf{x} - \boldsymbol{\mu}), \qquad (5)$$

where $\mathbf{\Lambda}$ is a diagonal matrix and $\mathbf{V}^T$ is a rotation matrix. The two transformations differ in how the matrices $\mathbf{\Lambda}$ and $\mathbf{V}$ are computed:

- For *component-wise normalisation*, $\mathbf{\Lambda}$ is the diagonal matrix of component-wise variances and $\mathbf{V} = \mathbf{I}$.
- For the *whitening transformation*, $\mathbf{\Lambda}$ is the matrix of eigenvalues and $\mathbf{V}$ is the matrix of eigenvectors of the full covariance matrix $\mathbf{\Sigma}$ of the data set. In this case Mahalanobis distance in the original space becomes equivalent to Euclidean distance in the transformed space.

With component-wise normalisation, each vector component in the target space is distributed with zero-mean and unit-variance. Correlations between vector components in the original space are **preserved** by this transformation. A minimum of two vectors is needed to compute the component-wise variances used to define the transform.

With the whitening transformation, each vector component in the target space is also distributed with zero-mean and unit-variance. However, correlations between vector components in the original space are **removed** by this transformation. If the dimension of the vector space is D, then a minimum of $D + 1$ vectors is needed to compute a non-singular full covariance matrix, which is needed to define the transform.

Selecting kernel centres

The distribution of the transformed vectors $\mathbf{x}'$ corresponding to normal engines is modelled by a few spherical kernels. Placement of the kernel centres is performed by the batch K-means algorithm (see Duda and Hart [3]) with the Euclidean distance in the *transformed* space as the metric used in defining the error E to be

[1] Two-shaft engines are considered in this case study.

minimised:

$$E = \sum_{j=1}^{N} \min_i \left((\mathbf{x}'_j - \mathbf{c}_i)^2 \right), \qquad (6)$$

where $\mathbf{c}_i$ is the centre of the i^{th} kernel. The results obtained from a number of different initial choices for the kernel centres are compared based on the value of the final error E and on the distribution of data vectors among the kernels.

After initial experimentation, it was decided to fix the number of kernels K at $K = 4$, for both the component-wise normalised data and the whitened data. Once the optimal partitioning of the data has been found for the $\mathbf{x}'$ vectors, the representation can be mapped back into the original data space by using equation 5 again. The kernel centres obtained with the component-wise normalised data are found to be fairly well separated and can be associated with broad engine types such as 'high HP vibrations' and 'high LP but low HP vibrations'. In comparison, the centres obtained with the whitening transformation are less clearly separated and cannot easily be associated with any engine types. The difference arises because component-wise normalisation retains correlations between vector components in the transformed space, whereas the whitening transformation eliminates these correlations. For unimodal data[2], the positioning of the kernel centres is strongly affected by correlations, and therefore the resulting placement of kernel centres is more arbitrary under whitening than component-wise normalisation.

Novelty detection results

For each kernel, we compute a *width* defined as the average distance σ_i between the data vectors (denoted by $\mathbf{x}_{ij}$) belonging to that kernel and the kernel centre $\mathbf{c}_i$; if kernel i contains N_i data vectors:

$$\sigma_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \sqrt{(\mathbf{x}_{ij} - \mathbf{c}_i)^2}. \qquad (7)$$

This completes the "training phase", the construction of a highly simplified model of normality. The novelty of a test vector $\mathbf{x}_j$ is assessed by computing d_j, the shortest normalised distance to a kernel centre:

$$d_j = \min_i \frac{1}{\sigma_i} \sqrt{(\mathbf{x}_j - \mathbf{c}_i)^2}, \qquad (8)$$

[2]In work not described here, the distribution of tracked order shapes for the normal engines in the database was found to be approximately unimodal.

where d_j is the measure of the *novelty* of vector $\mathbf{x}_j$. It is a locally defined measure of novelty, since it is based on the distance to the nearest kernel and is normalised by the average *spread* of data within that kernel (approximated by σ_i). Note that if the number of kernels is reduced to $K = 1$, the measure of novelty is simply proportional to the distance of the data vector $\mathbf{x}_j$ from the mean of the training database.

Once the representation of normality has been derived from the database of 52 normal engines, novelty scores are calculated for these 52 engines as well as for 33 other engines known to have at least one unusual tracked order shape. (In most cases, this "abnormality" would not be serious enough for the engine to fail its pass-off test. The extent of the abnormality does not concern us here, only that it exists). The ability of the system to distinguish between these two groups of engines can be assessed from the degree of overlap between the novelty values of the two groups. Table 1 summarises the results obtained for both component-wise transformed and whitened data, with $K = 1$ and $K = 4$. For comparison, novelty detection results based on Parzen windows are also included in table 1 for both types of normalising transformations; in the case of the Parzen windows model, the *novelty score* of each test vector $\mathbf{x}_j$ is $-\log(p(\mathbf{x}_j))$ (minus the log-likelihood), where $p(\mathbf{x})$ is computed according to equation 1.

It is interesting to note that for component-wise transformed data, the separation between the two groups of engines is much better for $K = 4$ than for $K = 1$. This is not unexpected as the transformation preserves correlations within the data, and more than one spherical kernel is needed to produce a sufficiently specific representation of normality. By contrast, changing from $K = 1$ to $K = 4$ does not affect the quality of the results for whitened data, since there is no need for more than one spherical kernel after whitening. Note also that, with this small training database ($N = 52$), the Parzen windows method produces results very similar to those obtained from a single kernel ($K = 1$). The overall conclusion from this section is that the whitening transformation appears to produce marginally better results than component-wise normalisation.

MODEL VALIDATION

The results presented above seem to suggest that the representation of normality for tracked order shape should be based on whitened data. However, with only 52 normal engines in our

transformation / representation	component-wise			whitening		
	spherical kernels		Parzen windows	spherical kernels		Parzen windows
	$K=1$	$K=4$		$K=1$	$K=4$	
overlapping normal engines (out of 52)	6	3	6	0	0	0
overlapping "abnormal" engines (out of 33)	**5**	**1**	**4**	**0**	**0**	**0**

Table 1: *Novelty detection results: the ability to distinguish between normal and "abnormal" engines is measured by the overlap between the novelty scores of both classes.*

database, the estimate of the covariance matrix (needed for the whitening transform) is highly dependent on the sample[3] of engines used as training data — recall that a minimum of 19 vectors is required to build a non-singular covariance matrix. This section investigates the extent to which this dependence affects the description of normality: how would normal engines not belonging to the training database be classified? The question is answered in a number of experiments in which the existing database of normal engines is deliberately split into a training set and a *validation* set.

The validation test

The database of $N = 52$ normal engines is split into $N_T = 42$ engines for training and $N_V = 10$ engines for validation. Experiments are conducted as follows:

1. N_V validation engines are chosen at random.
2. The remaining N_T engines are used to produce the representation of normality as previously described; this includes computing the component-wise or whitening transformations, kernels placement and computation of normalised distances.
3. The *novelty threshold* is set at the novelty value of the 'most novel' of the N_T engines in the training set[4].
4. The N_V validation engines are tested against the representation of normality: they are deemed to be correctly classified if their novelty score is below the novelty threshold.

The number of correctly classified normal engines (out of N_V) are averaged over 100 experiments with different random splits of the training database.

[3] The 52 engines in the database are assumed to have been sampled from the underlying distribution of normal engines.

[4] For such a small training database, it is reasonable to expect all training engines to be classified as normal.

value of X	ordering of novelty values
N_V	· · · · · · · · · · · · · · · △ |
$N_V - 1$	· · · · · · · · · · · · · · △ | ○
$N_V - 2$	· · · · · · · · · · · · · △ | ○○

Table 2: *This table shows the relative ordering of the novelty values for training and validation engines which is required to classify correctly X validation engines for $X = N_V$, $N_V - 1$ and $N_V - 2$; a '△' represents a training set engine, a '○' represents a validation engine and a '·' represents either. The novelty threshold is indicated by a '\|'.*

The average number of correctly classified normal engines in the validation set provides an indication of the extent to which the representation is overfitting the training dataset. We now derive an estimate of what this number should be if the representation was exact (and thus not overfitting). The evidence available suggests that the distribution in input-space of the 18-D feature vectors representing normal engines is unimodal. Therefore 'novelty' is expected to be a monotonically increasing function of the 'distance' from the mean of this unimodal distribution. Assuming that there are sufficient numbers of training patterns to provide a good estimate of the distribution (*i.e.* avoid overfitting), the number of correctly classified validation engines in a given experiment becomes a random variable X, which depends on the particular split of the database into a training and a validation set. The probability of k engines in the validation set being correctly identified is denoted by $P(X = k)$.

The engines in the database can be ordered according to their novelty values. An *exact* representation should reproduce the ordering correctly, regardless of the choice of training and validation set. Therefore, the value of X depends on how many engines in the validation set have larger novelty values than any engine in the training set. For example, table 2 shows

transformation	component-wise		whitening	
model	$K = 4$ kernels	Parzen windows	$K = 1$ kernel	Parzen windows
experiments	correctly classified (out of $N_V = 10$)			
group 1	**8.82**	**9.68**	**5.43**	**5.48**
group 2	**8.79**	**9.63**	**5.23**	**5.23**

Table 3: *The results in each group are averages from 10 validation experiments.*

the relative ordering of training and validation novelty values to obtain $X = N_V$, $X = N_V - 1$ and $X = N_V - 2$. From the table, it can be seen that the probability $P(X = N_V)$ of classifying all validation engines correctly is equal to the probability of the engine with the largest novelty value belonging to the training set:

$$P(X = N_V) = \frac{N_T}{N}, \text{ where } N = N_V + N_T.$$

The probability $P(X = N_V - 1)$ of misclassifying a single validation engine is equal to the probability of the most novel engine being a validation set engine and the second most novel being a training set engine:

$$P(X = N_V - 1) = \frac{N_V}{N} \cdot \frac{N_T}{N - 1}.$$

The probabilities $P(X)$ for other values of X are derived in a similar manner. The mean of this distribution is:

$$\mathrm{E}[X] = N_V \frac{N_T}{N_T + 1}, \qquad (9)$$

where it is apparent that each validation engine has got a probability $1/(N_T + 1)$ of being identified as novel. However the distribution is not binomial because the probability of a validation engine being found novel is *not* independent of the number of other validation engines found to be novel.

A random variable Y is defined as the average of the outcomes X of 100 independent trials. The distribution of Y is computed for $N_V = 10$ and $N_T = 42$ by recursive convolution of the probability distribution $P(X)$. Its mean is $\mathrm{E}[Y] = 9.77$ and the 95% and 99.98% confidence intervals are:

$$\mathrm{I}_{95\%}(Y) = [9.67, 9.87], \qquad (10)$$

$$\mathrm{I}_{99.98\%}(Y) = [9.55, 9.93], \qquad (11)$$

where $\mathrm{I}_z(Y)$ is the interval within which Y will fall with probability z.

Validation results

Table 3 presents the results from two groups of 100 validation experiments for different representations of normality. The number of correctly classified validation engines is much smaller with the whitening transformation than with the component-wise normalisation. This confirms that the whitening transform leads to a representation heavily dependent on the choice of which engines belong to the training set. The component-wise normalisation gives much better results: with 4 kernels, 88% of the validation engines are correctly classified; with the Parzen windows model, the *gold standard* set out in equation 11 is effectively reached, since the success rate is 96.5%.

The marginal superiority of the whitening transform in detecting "abnormal" engines cannot compensate for its inability to identify correctly previously unseen normal engines. The results presented in this section show that, for a limited amount of training data (52 examples here), the representation of normality should use as simple a model as possible, namely one based on the component-wise normalisation. Because the Parzen windows model produces better results than the spherical kernels representation on unseen normal engines but worse results on "abnormal" engines (see table 1), the particular choice of model will depend on the application.

ESTIMATING THE QUALITY OF A NOVELTY DETECTION MODEL

We have shown that with a *small* training database, the adequacy of the representation of normality can be determined by validation experiments. This section proposes an alternative test of the quality of novelty detection models which define a probability density function (pdf) over the data. This technique, suitable for *larger* databases, also provides a principled solution to the problem of choosing a global threshold value for novelty.

An overview of the method is given in figure 2. It is assumed that there is some unknown underlying distribution $p(\mathbf{x})$ from which the real data was sampled. A probability model $\hat{p}(\mathbf{x})$ is then fitted to the training data and used as an estimate for $p(\mathbf{x})$. We would now like to assess the accuracy of such a model. One possibility is to integrate $\hat{p}(\mathbf{x})$ over a region $\mathcal{R}$

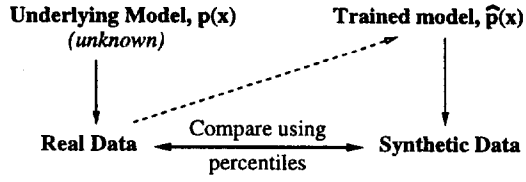

Figure 2: *Method for estimating the quality of a novelty detection model.*

of data space and hence find the probability, *according to our model*, of a data point being contained within this region, $P(\mathbf{x} \in \mathcal{R})$. This defines a binomial distribution which can be compared with the number, n, of real data points found to be in $\mathcal{R}$. If the model is a good fit then one would expect n to be close to the mean of the binomial distribution. However if $\hat{p}(\mathbf{x})$ underestimates the pdf of the real data in $\mathcal{R}$ then n will be greater than the mean; conversely if $\hat{p}(\mathbf{x})$ is an overestimate. Repeating this procedure for different $\mathcal{R}$ provides an indication of the quality of the fit between $\hat{p}(\mathbf{x})$ and the real data.

The problem with this method is that it requires $\hat{p}(\mathbf{x})$ to be integrable analytically; two of the most popular models, Gaussian mixture models and Parzen windows, cannot be integrated analytically over arbitrary contours in a high dimension. However, Monte Carlo simulation can be used to provide an accurate estimate of the integral so long as a large enough sample of synthetic data is drawn from the model $\hat{p}(\mathbf{x})$.

A convenient choice for the family of regions to be tested can be made by choosing appropriate thresholds for the value of $\hat{p}(\mathbf{x})$. The method can be summarised as follows:

1. Train a model $\hat{p}(\mathbf{x})$ on some (or all) of the real data (normal patterns only).

2. Generate a large set of synthetic data by sampling from $\hat{p}(\mathbf{x})$. Calculate $\hat{p}(\mathbf{x})$ for each of the points generated.

3. Choose a threshold t on $\hat{p}(\mathbf{x})$ and then define a region $\mathcal{R}$ by $x \in \mathcal{R} \equiv \hat{p}(\mathbf{x}) > t$. Note that t is a threshold on a pdf and can therefore be any positive real number.

4. Estimate $p = P(\mathbf{x} \in \mathcal{R}) = \int_{\mathcal{R}} \hat{p}(\mathbf{x}) dV$ by dividing the number of synthetic data points in $\mathcal{R}$ by the total number of synthetic data points generated in 2 above. Note that p is a probability and is therefore in $[0, 1]$.

5. If the trained model, $\hat{p}(\mathbf{x})$, is a good approximation to the real distribution $p(\mathbf{x})$, then the number N_R of real data points which are in $\mathcal{R}$ should follow a binomial distribution $B(N, p)$, where N is the total number of real data points and p is as defined in 4 above.

6. Plot a graph of the mean (and variance) of the binomial distribution vs the number of real data points in $\mathcal{R}$ for different values of the threshold t.

One modification to the above method is convenient: rather than using t as the controlling parameter which fixes $\mathcal{R}$ and p, it would be more intuitive to be able to choose p directly. This is easily achieved by sorting the synthetic data into decreasing values of $\hat{p}(\mathbf{x})$ and choosing $t = \hat{p}(\mathbf{x}_p)$, where $\mathbf{x}_p$ is the data point at position $p \times N_S$ in the sorted list of N_S synthetic data points. Thus by construction, a Monte Carlo simulation using the synthetic data would give the desired value of p.

Finally, if the above method leads to a model which is a good fit[5], then the value of the threshold corresponding to a value of $p = 0.99$ gives a principled solution to the problem of finding a global novelty threshold which classifies 99% of the distribution of training data as non-novel (*i.e.* normal).

ACKNOWLEDGEMENTS

The work described in this paper was supported by EPSRC and Rolls-Royce plc.

REFERENCES

[1] C.M. Bishop. Novelty detection and neural network validation. *IEE Proc. - Vis. Image Signal Process.*, 141(4):217–222, August 1994.

[2] P.H. Cowley and H.R. Carr. Application of neural networks to aero engine vibration monitoring. In *5th European Propulsion Forum, Italy.* Rolls-Royce plc, 1995.

[3] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[4] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proc. 4th IEE Int. Conf. on Artificial Neural Networks, Cambridge*, pages 442–447. IEE Conf. Publication No. 409, 1995.

[5] To ensure that we have a good definition of novelty we need to have a good model $\hat{p}(\mathbf{x})$ for the a priori distribution of data. To see why this is so consider the case where the model $\hat{p}(\mathbf{x})$ has a high value in regions where there is very little data: this will be classified as normal, when it should clearly be classified as novel.