Effectiveness of Feature Extraction in Neural Network Architectures for Novelty Detection

Jonathan Francis Dale Addison, Stefan Wermter, John MacIntyre
Centre for Adaptive Systems
School of Computing, Engineering and Technology
University of Sunderland
St Peters Campus, St Peters Way
Sunderland, SR6 0DD, UK

e-mail dale.addison@sunderland.ac.uk

Abstract

This paper examines the performance of seven neural network architectures in classifying and detecting novel events contained within data collected from turbine sensors. Several different multi-layer perceptrons were built and trained using back propagation, conjugate gradient and Quasi-Newton training algorithms. In addition, Linear networks, Radial Basis Function networks, Probabilistic networks and Kohonen self organising feature maps were also built and trained, with the objective of discovering the most appropriate architecture. Because of the large input set involved in practice, feature extraction is examined to reduce the input features, the techniques considered being stepwise linear regression and a genetic algorithm. The results of these experiments have demonstrated an improvement in classification performance for multi layer perceptrons, Kohonen and probabilistic networks, using both genetic algorithms and stepwise linear regression over other architectures considered in this work. In addition, linear regression also performed better than a genetic algorithm for feature extraction. For classification problems involving a clear two class structure we consider a synthesis of stepwise linear regression with any of the architectures listed above to offer demonstrable improvements in performance for important real world tasks.

Introduction

Novelty detection is the task of observing changes of state in a process. 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 [1]. The monitoring task under investigation has a normal operating state, as well as several other abnormal states. Typically, the causes of these abnormal states differ, which means their performance characteristics can be observed, classified and recorded for future study.

One approach to the problem has been to use rule-based expert systems. Such systems become highly unstable if they are confronted with new data which does not correspond to anything contained within the system's rules or facts [2] [3]. Highly dynamic systems, such as gas turbines or medical diagnosis problems frequently contain aspects which make exact classification of states difficult, and our problem embodies all of them. In particular: i) The high dimensionality of this problem, owing to the number of input parameters. ii) The extremely dynamic nature of the problem. Current modeling techniques involve the use of extremely complex third order differential equations. iii) The uncertainty regarding the interaction between parameters and their inter-relationship with the output values.

Recently there has been much interest in modeling condition monitoring tasks using neural networks: Chowdhury and Wang describe a fault detector/classifier using a Kohonen network, using current and voltage signals obtained from high voltage electricity transmission lines [4]. Cirrincione *et al* examine the use of a Kohonen network which acts as a diagnostic system for small and medium sized machines [5]. In contrast Kevyan *et al* describe a prototype signal monitoring system which utilises adaptive-resonance theory networks and a fault identification database for fault detection in a fast breeder nuclear reactor [6]. Wu Yan and Upadhyaya investigate the use of data compression methods and neural networks for eddy current inspection of steam generator tubing, using non-destructive testing [7].

These researchers have considered in some depth the use of neural networks for novelty or fault detection. However, extensive comparisons between different architectures have not yet been performed. This paper considers seven different architectures in conjunction with feature extraction methods. Furthermore, the role of dimensionality reduction techniques has not been covered by any of these authors, even though in some cases the training sets contain as many as 50 dimensions. In our work we obtain comparable classification performance by eliminating those input features which do not contribute to the classification.

Also, there is a trend to work with data sets which have been created by time series processing using either the values at *t-1* or *t+1* as inputs to the neural networks, which can better extract or amplify the error cases contained in the data sets. In this work we investigate the raw signals without the benefit of any preprocessing.

Description of Task

This work aims at identifying a neural network architecture which can most effectively distinguish between normal and abnormal readings taken from a gas turbine. Another goal is the determination of how effectively dimensionality reduction of the input features improves the networks ability to classify. Existing rule-based expert systems have proven ineffective in dealing with the dynamic nature of such a problem, where the interaction between multiple variables and the significance of each parameter to the regressed value is not fully understood. An example of such problems has been documented by Milne and colleagues during the TIGER project [8].

Our problem domain allows the turbine to be in one of two states: normal (defined here as a normal reading of the electrical output produced by the turbine generating shaft); or abnormal, which can be in either the electrical output level or the two fuel/air sensors in the combustion chamber. The sensors provide the data used in the experiments. Examples of normal and abnormal sensor readings can be found in Figure 1 which shows readings for the fuel induction rate sensor. This figure demonstrates normal operating conditions, signified by a straight line. The second reading begins at a significantly lower reading and then develops a fault at one minute and 46 seconds.

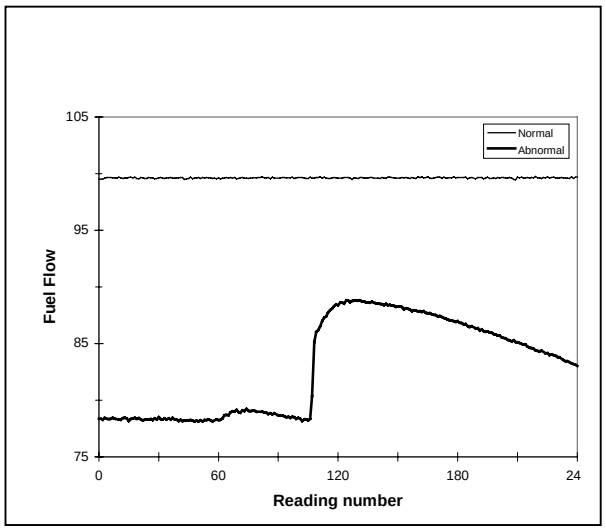

Figure 1: Chart showing normal (top) and abnormal (bottom) readings for gas turbine fuel flow.

The training set was organised as follows. The fault file used for the output readings was collected from the fuel/air ratio sensors. The input mappings were established by using 18 thermal exhaust sensors, combined with the readings for fuel, level of electrical output, and level of turbine steam output. Each training set consisted of 240 patterns in the following order: (1) The input pattern set without any form of pre-processing consists of 21 features and 240 patterns. (2) Following the application of a genetic algorithm, the training features are reduced slightly to 19 features. (3) Following stepwise linear regression, six features are retained in the 240 patterns.

Training Algorithms

The architectures used for these experiments are multi-layer perceptrons, Kohonen self-organising mapping, radial basis function networks, probabilistic networks and linear networks. It is not intended to go into a detailed exposition of each network here, but we will focus on the most distinctive features.

Multi-Layer Perceptrons

Typically MLP networks [9] consist of a set of sensory units (source node) constituting an input layer, one or more hidden layers of computational nodes and an output layer of computational nodes. The input signal moves forward through the network, layer by layer. The following three training algorithms can be used.

Back Propagation

Back propagation provides two passes through the node structures [10]. In the forward pass, a pattern is propagated through the network layer by layer, producing a set of responses. In the backward pass, the network weights are adjusted according to the value of the error correction rule by subtracting the response value from the actual target value in the output layer to produce an error value. The weights are then adjusted to make the actual response of the network move closer to the desired response. Various training times were used with back propagation, but the most effective combination was found to be a small number of epochs (100) combined with a very low learning rate (0.2).

Conjugate Gradient

Conjugate gradient descent works by constructing a series of line searches across the error surface [11]. The direction of steepest descent is computed by projecting a straight line in that direction, then locating

a minimum along this line. Further line searches are conducted (one per epoch). The direction of the line searches (the conjugate directions) are chosen to try to ensure that the directions that have already been minimized stay minimized.

Quasi-Newton

To assist the Quasi-Newton [12] algorithm in escaping from local minima, it was decided to train in two stages. First, back propagation using a small number of epochs (20-30) was used to train the network. Then, a longer period of Quasi-Newton was utilised (100 epochs).

Kohonen Networks

In Kohonen networks, the post-processing definition always includes a single nominal output variable, with one nominal value for each class [13]. The units in the output layer are labeled to correspond to the classes; when the network is executed, the winning node (i.e. the one with the lowest activation, indicating greatest proximity to the input case) is selected, and its label is used to assign the class. Consequently, the Kohonen network's output layer is user-defined, and can have any number of units. The Kohonen network was trained in two phases. An initial short learning phase, incorporating a high learning rate, and a large neighbourhood. The second phase is longer, using 10000 epochs, a very small learning rate, and no neighbourhoods.

Linear Networks

In terms of function approximation, the simplest model is the linear model [14], where the fitted function is a hyperplane. A linear model is typically represented using an *NxN* matrix and an *Nx1* bias vector. A linear neural network has no hidden layers, but an output layer with fully linear units (that is, linear units with linear activation function). The weights correspond to the matrix, and the thresholds to the bias vector. When the network is executed, it effectively multiplies the input by the weight matrix, then adds the bias vector. For these experiments, the pseudo-inverse procedure was used. This optimizes the last layer in any network, assuming it is a linear layer. Not only does it guarantee location of the absolute minimum error, it is also relatively quick.

Radial Basis Function Networks (RBF)

A RBF network has a hidden layer of radial units, each modeling a Gaussian response surface [15]. The hidden radial unit outputs are blended into the network outputs by using a linear combination of these outputs (i.e. a weighted sum of the Gaussians). The RBF has an output layer containing linear units with linear activation function. Random sampling to assign the kernel density functions was found to be just as effective as *k-nearest neighbours*, which has been utilised in the experiments performed so far. The typical number of radial units is 24, which represents approximately ten percent of the pattern set.

Probabilistic Neural Networks (PNN)

In the PNN, there are at least three layers: input, radial, and output layers [16]. The radial units are copied directly from the training data, one per case. Each unit models a Gaussian function centered at the training case. There is one output unit per class. Each is connected to all the radial units belonging to its class, without connections from all other radial units. Hence, the output units simply add up the responses of the units belonging to their own class. The outputs are each proportional to the kernel-based estimates of the probability density functions of the various classes, and by normalizing these to sum to 1.0, estimates of class probability are produced. The main issue here is concerned with assigning a random sampling value to determine the distribution of the Gaussian function. The default smoothing constant used in our experiments is 0.3.

Dimensionality Reduction Techniques

Later in the paper we will show our benchmark results for the seven network architectures described above. However, before that we would like to introduce our synthesis of neural network architectures with two different dimensionality reduction techniques. The need for dimensionality reduction methods has been stated by Bishop [17]. In our task, although many parameters are known to influence the output of electrical energy, their exact inter-relationship (and more importantly the degree of importance of each) is difficult to extract or understand. In addition, to avoid issues related to the “curse of dimensionality”[17], it was decided to utilise feature extraction techniques to determine which parameters should be retained for the neural nets, and which should be ignored. Two techniques have been used here to create the network training sets: genetic algorithms and stepwise linear regression. Both will be described below.

Genetic Algorithms

Genetic algorithms are an optimisation technique based upon the classical Holland algorithm [18] that can search efficiently for binary strings. Reproduction takes place when individual strings are copied according to an objective function, which measures each string’s fitness. These qualities will be passed to the next

generation during the breeding process. Reproduction in these experiments is achieved by simulating a roulette wheel biased in favour of those strings which have the highest fitness value. After reproduction, simple crossover takes place in two stages. First, the strings are mated at random, then the integer position *k* along the string is selected uniformly at random between one and the string length less than one [1, *l*-1]. Two new strings are created by swapping all characters between position *k* + 1 and *l* inclusively. Mutation allows a bit to be changed from 0 to 1 and vice versa, with a mutation probability p_m to prevent premature convergence [19].

Genetic algorithms are well-suited for feature selection as they are very good at recognising subsets of inter-related bits (in this case, correlated or mutually-required inputs). The time requirements are high, but are mostly unaffected by the number of variables, whereas forward and backward selection have time requirements proportional to the square of the number of variables. There are a large number of parameters that can be altered in a genetic algorithm.

Stepwise Linear Regression

Another technique for dimensionality reduction is the use of stepwise linear regression. Here we are attempting to regress two or more independent variables on to a single dependent variable. The formula used is:

$$Y' = a + b_1 X_1 + b_2 X_2 + ... b_k X_k \qquad [1.1]$$

Here *Y'* is the predicted value of *Y*, *X* are the input variables, and *a* and *b* are parameters used in determining the degree of correlation. This formula makes several assumptions, particularly that all the *Y* values are independent of each other, and *Y* is a linear function of *X*. For any fixed combination of *X* values, the variance of *Y* is fixed, and for any fixed combination of X values, Y is normally distributed. The maximum value of the correlation coefficient between observed *Y* values and predicted *Y'* values will be obtained if we find the values of *a*, b_1, $b_2 \ldots b_k$. that minimise the sum of squared errors (SS) of prediction of the residual sum of squares.

$$SS_{res} = (Y - Y')^2 \qquad [1.2]$$

Given that there are 21 potential inputs, for the sake of clarity and readability, we will explain the concept using only two *X* variables. The required value of a is given by the equation.

$$a = \overline{Y} - b_1 \overline{X_1} - b_2 \overline{X_2} \qquad [1.3]$$

Where $\overline{X}$ and $\overline{Y}$ represent the mean values of the input parameters *X*, and the regressed *Y* value.

By substituting this value in [1.1] we have:

$$Y' = \overline{Y} + b_1 (X_1 - \overline{X_1}) + b_2 (X_2 - \overline{X_2}) \qquad [1.4]$$

or:

$$Y' = \overline{Y} + b_1 x_1 + b_2 x_2 \qquad [1.5]$$

Then if the residual sum of squares is to be minimized, the values of b_1 and b_2 (the slope of each independent variable) must satisfy the following equations.

$$b_1 \quad x_1^2 + b_2 \quad x_1 x_2 = \quad x_1 y \qquad [1.6]$$

and:

$$b_1 \quad x_1 x_2 + b_2 \quad x_2^2 = \quad x_2 y \qquad [1.7]$$

These two equations with unknown values for b_1 and b_2 can be solved in the following fashion. Multiplying [1.5] by x_1^2 and [1.6] by $x_1 x_2$ we obtain [20].

$$b_1 = \frac{\left(x_1 y\right)\left(x_2^2\right) - \left(x_2 y\right)\left(x_1 x_2\right)}{\left(x_1^2\right)\left(x_2^2\right) - \left(x_1 x_2\right)^2} \qquad [1.8]$$

Using a similar procedure we arrive at.

$$b_2 = \frac{\left(x_2 y\right)\left(x_1^2\right) - \left(x_1 y\right)\left(x_1 x_2\right)}{\left(x_1^2\right)\left(x_2^2\right) - \left(x_1 x_2\right)^2} \qquad [1.9]$$

Stepwise linear regression has not been used as a feature extraction method for neural network processing. An important benefit of the technique is that it produces several useful statistics such as the standard deviation and variance of the *X* values in relation to *Y*, which helps in identifying those parameters which should be retained for training. But most importantly linear regression deals with multiple correlation's between the data in the input set and the target regression value. Moreover by using the stepwise

feature the user can observe which of the input features should be retained, which makes creating the input sets for the neural networks easier.

Experiments and Analysis

The experiments were carried out in three stages:

a) The entire training set was used to train the networks without the benefit of any feature selection being applied. This establishes a benchmark of performance for each network.

b) A genetic algorithm was then applied to the input parameters used in a). Those deemed unfit for training were discarded, and a new input set was produced.

c) Finally stepwise linear regression was applied to the training set described in a) above and again unfit parameters were discarded, and a new training set built.

Tables 1-3 show the network architectures which performed best in terms of training, verification and test set performance. Table 1 shows the benchmark performance for the full input feature representation.

Architecture	Training	Verification	Test
MLP-BP	95	95	97
MLP-CG	100	98	100
MLP-QN	100	98	100
Kohonen	98	98	100
Linear	88	83	85
RBF	98	97	100
PNN	100	96	100

Table 1: Percentage of patterns correctly classified for training, verification and test sets.

The following acronyms are used here for different training algorithms in conjunction with multi-layer perceptrons: BP=back propagation, CG=Conjugate gradients, QN=Quasi-Newton.

Table 2 shows the results of preprocessing with a genetic algorithm. The most striking result is a decline in performance for linear and radial basis function networks. This is most noticeable in the test set figures for both networks. The decline in performance is most noticeable in the radial basis function network. The elimination of one feature from the training set reduces classification ability by nearly 23%, whilst the verification set (used to verify the accuracy of the training set predictions) is reduced by approximately 34%.

Architecture	Training	Verification	Test
MLP-BP	94	93	97
MLP-CG	100	97	100
MLP-QN	100	100	100
Kohonen	99	99	100
Linear	81	88	79
RBF	75	63	79
PNN	100	97	100

Table 2: Percentages of patterns correctly classified after processing with a genetic algorithm.

Table 3 shows the classification performance of each architecture after the application of stepwise linear regression. The performance of linear and radial basis function networks is improved on both verification and test sets, while the performance of the Kohonen network improves to 100%.

Architecture	Training	Verification	Test
MLP-BP	95	93	97
MLP-CG	95	93	97
MLP-QN	100	100	100
Kohonen	100	100	100
Linear	88	90	95
RBF	96	92	90
PNN	98	98	100

Table 3: Percentages of patterns correctly classified after processing with stepwise linear regression.

An interesting point as a result of this work concerns the slightly superior classification ability of the probabilistic network over MLPs. In related work using vibration time series with Bayesian neural networks, they have proven highly effective [21]. But using a classification system based upon density estimation via a Gaussian kernel depends on selecting the correct value of the width parameter (h value) which acts as a smoothing parameter to enable an approximation to the density function. With a large kernel width the density function becomes too smooth, resulting in a loss of the bimodal nature of the distribution. Conversely too small a parameter allows the estimated density to represent the properties of a particular data set rather than the required distribution structure. The most important observation from these experiments is that combining neural network architectures with linear regression reaches comparable performance using 30% of the original input features compared to the original neural network architectures without linear regression

Conclusions

This work has evaluated seven neural network architectures and training algorithms for novelty detection. In addition two techniques for feature extraction and dimensionality reduction have been considered, as a means of improving classification performance. The strength of this work lies in a clear benchmark comparison for the real-world task of condition monitoring. We argue that a synthesis of various neural networks with linear regression provides comparable performance while using only 30% of the input features.

References

[1] Alexandre, N. T.; Corbett, C.; Ripley, R.; Townsend N. W.; and Tarassenko, T. 1997 Choosing an Appropriate Model for Novelty Detection, *Proceedings of IEEE Fifth International Conference on Artificial Neural Networks*, 117-123.

[2] McGarry, K.; Wermter, S.; and MacIntyre, J. 1999 Hybrid Neural Systems: From Simple Coupling to Integrated Neural Networks. *Neural Computing Survey* 2.; http://www.icsi.berkelyedu/~jagota/NCS.

[3] Wermter, S.; and Sun, R. 1999 Hybrid Neural Symbolic Systems. Springer, Heidelberg.; (to appear)

[4] Chowdhury, B. H.; and Wang, K. 1996 Fault Classification using Kohonen Feature Mapping, Electrical Engineering Department, University of Wyoming.

[5] Cirrincione, G.; Cirrincione, M.; Vitale, G. 1994 A Kohonen Neural Network for the Diagnosis of Incipient Faults in Induction Motors, *International Conference on Electrical Machines*, 364-73

[6] Kevyan, S.; Durg, A.; and Nagaraj J. 1997 Application of Artificial Neural Networks for the Development of a Signal Monitoring System, Nuclear Engineering Department, University of Missouri-Rolla, *Experts Systems*, 14(2):69-78.

[7] Yan, W.; and Upadhaya, R. 1996 An Integrated Signal Processing and Neural Networks System for Steam Generator Tubing Diagnostics using Eddy Current Inspection, *Annals Nuclear Energy*, 23(10): 813-825.

[8] Massuyes, L. T.; and Milne, R. 1996 Diagnosis of Dynamic Systems based on Explicit and Implicit Behavioural Models: An Application to Gas Turbines in Esprit project TIGER" *Applied Artificial Intelligence*, 10(3): 257-277.

[9] Haykin, S.; 1995 Neural Networks - A Comprehensive Foundation.; Maxwell Macmillan International Publishing Company, 138.

[10] Rumelhart, D. E.; Hinton, G. E.; and Williams, R J. 1986. Learning Internal Representations by Error Propagation in Parallel Distributed Processing: Explorations in the *Microstructure of Cognition:* 318-362.

[11] Hestenes, M. R.; and Stiefel, E. 1952 Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards* 49**(**6): 409-436.

[12] Barnard, E.; 1992 Optimization for Training Neural Nets. *IEEE Transactions on Neural Networks* 3(2), 232-240.

[13] Kohonen, T.; 1982 Self–organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43, 56-69.

[14] Golub, G.; and Kahan, W. 1965 Calculating the singular values and pseudo-inverse of a matrix, *Numerical Analysis, B* 2 (2): 205-224.

[15] Lowe, D.; 1995 Radial Basis Function Networks, The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press.

[16] Specht, D. F.; 1990 Probabilistic Neural Networks. *Neural Networks* 3 (1): 109-118.

[17] Bishop, C. M.; 1997 Neural Networks for Pattern Recognition, Oxford University Press.

[18] Holland, J.; 1975 Adaptation in Natural and Artificial systems. MIT Press.

[19] Goldberg, E.; 1989 Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley.

[20] Edwards, A. L.; 1976 An Introduction to Linear Regression and Correlation, W. H. Freeman and Company, San Francisco.

[21] Buntine, W.; 1991 Theory Refinement on Bayesian Networks *Proceedings of Uncertainty in Artificial Intelligence*,: 52-60.