MULTI-LAYER PERCEPTRONS AND RADIAL BASIS FUNCTION NETWORKS FOR CORROSION MONITORING

J F D ADDISON, STEFAN WERMTER, JOHN MacINTYRE

University of Sunderland, Centre for Adaptive Systems, School of Computing, Engineering and Technology, St Peters Campus, St Peters Way, Sunderland, SR6 0DD
England

ABSTRACT

Developing a model of aqueous corrosion of metal has proven to be a complex and intractable problem. Although the electro-chemistry of the exchange of electrons is well documented, the influence of other factors such as changes in water temperature, oxygen levels, and the levels of pH and alkalinity on the corrosion process are less well understood. As yet there is no model which adequately explains this interaction, because of the extreme non-linearity of the problem. One method of achieving this is through the use of artificial neural networks which are well established as a means of mapping complex non-linear relationships onto a desired output. However the best architecture for the extrapolation of data is very problem dependant. Because of the high dimensionality of the data sets, we have compared and contrasted two methods for eliminating highly correlated data sets. Our claim is that accurate regression modeling on such a complex problem can best be achieved using radial basis function networks, which have demonstrated a superiority over multi layer perceptrons for modeling highly non-linear problem surfaces, combined with genetic algorithms as an initial pre-processing step.

.

KEY WORDS: Neural networks, regression analysis, corrosion monitoring

1. INTRODUCTION

Corrosion monitoring is one of the basic functions required for safe and efficient industrial operations. Corrosion represents the reduction of the metal to its original ore, via an exchange of electrons between the metal and its surroundings. Although the anodic and cathodic process which drives the electron exchange is well understood, and can be modelled by differential equations derived from circuit diagrams, the influence of other parameters such as changes in the temperature and level of oxygen, is less understood. Neural networks provide a means of modelling the relationship between parameters known to influence the corrosion process. Previous studies have concentrated on the use of multi-layer perceptrons (MLP's) which are well documented in regression problems [1] [2] [3]. However comparative studies of other architectures are sparse.

This paper is structured as follows. Section 1 gives a brief overview of the problem, a consideration of the necessity for dimensionality reduction, and a brief discussion of the methods involved. Section 2 discusses issues relating to regression problems. Section 3 outlines the main neural network architectures used in this work, section 4 outlines the experimental method used, and sections 5 and 6 consider the results, and discuss the implications of our findings for others interested in applying neural networks to regression problems respectively.

Our source data is taken from the domestic water supply in Finland. Water towers process water intended for reservoir systems, which supply the west and eastern halves of Helsinki respectively. Each water tower site carries out treatment and readings before forwarding the supply to a specific town or city. The water treatment towers are particularly interested in values for the following parameters, pH, alkalinity, temperature, conductivity and chloride levels. The inter-relationship of these parameters was examined by the use of a genetic algorithm which ensures superfluous data is eliminated from training sets presented to the neural networks. The data provided for this study came in two formats, a spreadsheet of laboratory readings from the water treatment plants, containing pH, alkalinity, temperature, chloride and conductivity readings, and a second spreadsheet containing readings for corrosion, pH temperature and chloride. Figure 1 below shows that the corrosion readings appear to follow a linear pattern, but are prone to disturbances by a number of outliers. From this data we have attempted to create a means of predicting corrosion in cast iron water pipes, using data for temperature, and chloride recorded at the same time.

2. APPLICATION OF DIMENSIONALITY REDUCTION TECHNIQUES

In real world scenarios of developing new algorithms the dimensionality the problem space is a crucial factor as stated by Bishop [5] and many techniques have been used to ensure that only parameters with a significant influence on the target output are retained. These methods can range from simple linear techniques such as stepwise linear regression and principal components analysis [4] [13] to more exotic technologies such as genetic algorithms. The latter has been used extensively on our data sets, and the results compared to stepwise linear regression which, in addition to being an effective regression technique, has been shown to be an effective technique for reducing data dimensionality. [4]

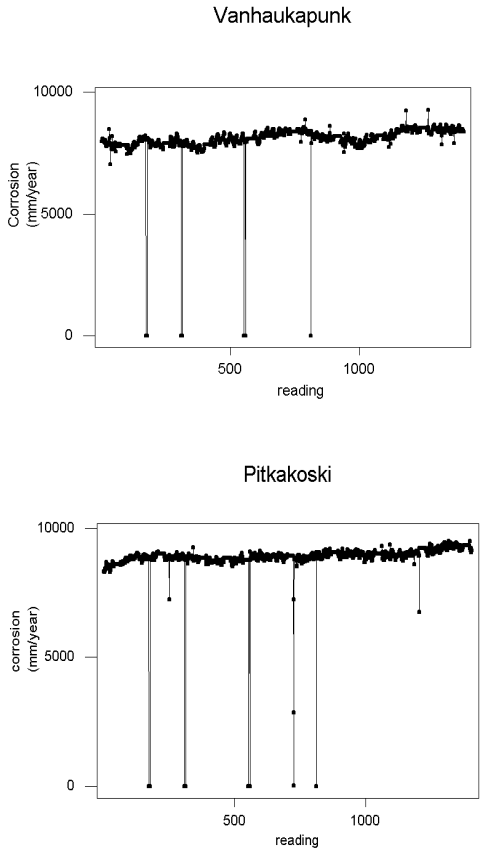

Figure 1: Typical corrosion readings from sample sites showing the influence of outliers

3. REGRESSION PROBLEMS

In regression problems, the purpose of the neural network is to learn a mapping from the input variables to a continuous output variable, or variables. A network is successful at regression if it makes predictions more accurate than a simple estimate. The simplest way to construct an estimate, given training data, is to calculate the mean of the training data, and use that mean as the predicted value for all previously unseen cases. The average expected error from this procedure is the standard deviation of the training data. The aim in using a regression network is therefore to produce an estimate that has a lower prediction error standard deviation than the training data standard deviation. A particularly important issue in regression are output scaling and extrapolation effects. The most common neural network architectures have outputs in a limited range (e.g., (0,1) for the logistic activation function). This presents no difficulty for classification problems, where the desired output is in such a range. However, for regression problems there clearly is an issue to be resolved, and some of the consequences are quite subtle. This subject is discussed below.

Using the example given in figure 1, the hope is to fit a curve allowing a value *y* to be predicted using a collection of data points x. From the figure below, it is obvious that the extrapolation curve will fit the cases based upon the shape of the existing curve. However, what about a point well to the right of the data points? There are two possible approaches to estimating *y* for this point. First, we might decide to extrapolate: projecting the trend of the fitted curve onwards. Second, we might decide that we do not really have sufficient evidence to assign any value, and therefore assign the mean output value (which is probably the best).

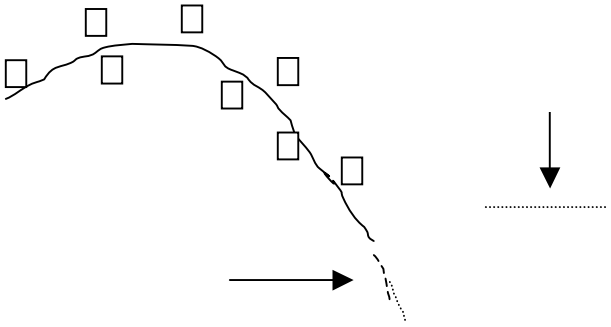

Figure 2: Obtaining the correct function to predict the next values in time series problems

4. ARCHITECTURES FOR REGRESSION PROBLEMS

We will now consider two neural network architectures for extrapolating data. Multi layer perceptrons, and Radial basis function networks.

4.1. MULTI-LAYER PERCEPTRONS

Using a scaling algorithms such as minimax is highly restrictive. First, the curve is not extrapolated independent of how close to the training data we may be. Second, it does not estimate the mean either - it actually saturates at either the minimum or maximum, depending on whether the estimated curve was rising or falling when approaching this region. There are a number of approaches to correct this deficiency in a multi layer perceptron [15].

First, we can replace the logistic output activation function with a linear activation function, which simply passes on the activation level unchanged (N.B. only the activation functions in the output layer are changed; the hidden layers still use logistic or hyperbolic activation functions). The linear activation function does not saturate, and so can extrapolate further (the network will still saturate eventually as the hidden units saturate). Second, we can alter the target range for the minimax scaling function (for example, to [0.25,0.75]). The training cases are then all mapped to levels which correspond to only the middle part of the output units' output range.

4.2. RADIAL BASIS FUNCTION NETWORKS

Radial networks are inherently incapable of extrapolation. As the input case gets further from the points stored in the radial units, the activation of the radial units decays and (ultimately) the output of the network decays. An input case located far from the radial centers will generate a zero output. The tendency not to extrapolate may be regarded as good (depending on the problem-domain and viewpoint), but the tendency to decay to a zero output (at first sight) is not. If we decide to ignore extrapolation, then what we would like to see reported at highly novel input points is the mean. This is achieved by using the mean/SD scaling function with radial networks in regression problems. The training data is scaled so that its output mean corresponds to 0.0, with other values scaled according to the output standard deviation. As input points are executed outside the range represented in the radial units, the output of the network tends back towards the mean. The performance of a regression network can be examined in a number of ways.

First, the output of the network for each case can be submitted to the network. If part of the data set, the residual error is also displayed. Second, summary statistics can be generated. These include the mean and standard deviation of both the training data values and the prediction error. One would generally expect to see a prediction error mean extremely close to zero (it is, after all, possible to get a zero prediction error mean simply by estimating the mean training data value, without any recourse to the input variables or a neural network at all). The most significant value is the prediction error standard deviation. If this is no better than the training data standard deviation, then the network has performed no better than a simple mean estimator.

Thirdly, a view of the response surface can be generated. The network's actual response surface is, of course, constructed in N+1 dimensions, where N is the number of input units, and the last dimension plots the height. It is clearly impossible to directly visualize this surface where N is anything greater than two (which it invariably is). Radial Basis Function networks [16] represent a means of producing a smooth interpolation function, in which the number of basis functions is determined by the complexity of the mapping to be represented rather than by the size of the data set.

To ensure the accuracy of this mapping several modifications are required, specifically the number of basis functions is much less than the number of data points. The centers of basis functions are now determined during the training process, and each basis function contains its own width parameter σ_j. With these changes the mapping function for the radial basis neural network mapping becomes:

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{k0} \quad (1)$$

The biases w_{k0} can be incorporated into the summation by including additional extra basis function ϕ_0 whose activation is set to 1. For Gaussian basis functions this gives:

$$\phi_j(x) = \exp\left(-\frac{\|x - u_j\|^2}{2\sigma_j^2}\right) \quad (2)$$

where x is the *d*-dimensional input with elements x_i, and μ_j is the vector determining the center of basis function ϕ_j and has elements u_{ji}. Further improvements can be obtained via the use of kernal regression, a

technique for estimating regression functions from noisy data. If a mapping from input vector x, to output vector y is required, the statistical properties of the data generator is given by probability density $p(x,t)$ in the joint input-target space. This can then be modeled via parzen kernal estimator constructed from the data set, using Gaussian kernal functions the estimator looks as follows.

$$\hat{p}(x,t)=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{(d+c)/2}}\exp\left\{-\frac{\left\|x-x^n\right\|^2}{2h^2}-\frac{\left\|t-t^n\right\|^2}{2h^2}\right\} \quad (3)$$

With *d* and *c* being the dimensionalities of the input and output spaces respectively

5. EXPERIMENTAL METHOD

The parameters were pre-processed using the genetic algorithm, after several experiments, a smoothing factor of 0.13 with 100 populations and 100 generations (this provides for 10000 matches on the search string). This resulted in a suggested list of parameters which should be retained for training. Specifically, chloride, alkalinity and temperature. Several neural network architectures were subsequently created using chloride, alkalinity and temperature as inputs, and the values for corrosion as output. The criteria for effective learning was the standard deviation ratio, between the training, and the validation set. Tables 1 and 2 gives the result for each architecture, the number of neurons used, and their respective standard deviation ratios.

6. RESULTS

Tables 1 and 2 show the results for each of the architectures considered, which include two types of multi-layer perceptrons, as well radial basis function, probabilistic and generalized regression networks. The tables also show the results for multi-layer perceptrons and radial basis function networks which include the 24 hour period which is an additional means of smoothing times series input.

24 hour period inserted		**24 hour period absent**	
Network	Standard Deviation	Network	Standard Deviation
MLP (3-layer)	0.33	MLP (3-layer)	0.25
MLP (4-layer)	0.35	MLP (4-layer)	0.25
RBF	0.34	RBF	0.27
PNN/GRNN	0.33	PNN/GRNN	0.27

Table 1: Results of neural networks with 24 hour period inserted and absent for Llamala data set

24 hour period inserted		**24 hour period absent**	
Network	Standard Deviation	Network	Standard Deviation
MLP (3-layer)	0.93	MLP (3-layer)	0.85
MLP(4-layer)	0.93	MLP (4-layer)	0.99
RBF	0.94	RBF	0.99
PNN/GRNN	0.96	PNN/GRNN	0.64

Table 2: Results of neural networks with 24 hour period inserted and absent for Pitkakoski data set

7. CONCLUSIONS

We have focused on the important issues of whether it is better to attempt to model a response surface using either hyperplanes or hyperspheres. This is heavily dependant upon the nature of the data set. There are numerous advantages and disadvantages to each approach, but radial basis function networks do offer several advantages in regression modeling over multi layer perceptrons. Our results suggest that performance is comparable between the architectures when the number of outliers is small, but deteriorates rapidly as outliers increase. Certain architectures are assisted in their predictive ability by the use of a time smoothed seasonal component, but again this is dependant upon the number of outliers present. Crucially however, the question is raised “are multi layer perceptrons, better at extrapolating over the entire data set compared to radial basis networks?” Radial basis function

networks, are less likely to extrapolate beyond the known data, by contrast a multi layer perceptron tends to become more reliable over the entirety of the data set. This may not necessarily be an advantage if the data set contains a large number of outliers. In this work we have demonstrated that the predictive abilities of RBF networks are at least as accurate as MLPs confronted with significant outlier data.

REFERENCES

[1] I.S Helliwell, M.A Turega,. and. R.A Cottis, Accountability of Neural Networks Trained with Real World Data, Proc. *Fourth International Conference on Artificial Neural Networks;* Cambridge, UK; 26 - 28 June 1995, Conference Publication No 409, 218-222

[2] I.S Helliwell, M.A Turega. and R.A Cottis, Neural Networks for Corrosion Data Reduction, *Corrosion/96*, Paper 379.

[3] S.J Gartland, R.A Cottis, Q, Li and M, Turega. Neural Networks for the Prediction of Critical Pitting Temperatures, *Presented to Corrosion Science Symposium,* Sheffield, September 1996

[4] J. F. D. Addison, S. Wermter, J. MacIntyre, 1999 Effectiveness of Feature Extraction In Neural Network Architectures For Novelty Detection, *ICANN-99, Ninth International Conference on Artificial Neural Networks*, Edinburgh, UK, September 1999, 976-81.

[5] C.M, Bishop *Neural Networks for Pattern Recognition* (Oxford University press, 1997).

[6] J.Holland; *Adaptation in Natural and Artifical Systems*: (MIT Press, 1975)

[7] D. F. Specht, Probabilistic Neural networks, *Neural networks 3*(1), 1990, 109-118

[8] C.M. Bishop, *Neural networks for pattern recognition*, (Oxford University Press, 1997).

[9] S Haykin, *Neural Networks a comprehensive foundation* (Maxwell MacMillan, International Publishing Company 1995)

[10] D E, Rummelhart, G. E. Hinton, R. J. Wiliams Learning Internal Representations by Error Propagation in Parallel Distributed Processing, *Explorations in the Microstructure of Cognition*, 1996, 318-36

[11] M R Hestenes, and E. Steiffel, Methods of Conjugate Gradients for solving Linear Systems, *Journal of Research of the National Bureau of standards 49*(6), 1952

[12] E Barnard, Optimisation for Training Neural Nets, *IEEE transactions on Neural Networks 3*(2), 1992, 232-240

[13] O Taylor and D. Addison, Neural networks for Novelty detection *COMADEM 13th International Congress on Condition Monitoring and Diagnostic Engineering management*, Houston Texas December 3-8 2000, 731-741.

[14] T.Kohonen, *Self-Organization and Associative Memory*. (Springer-Verlag, Berlin, 1989).

[15] S.Haykin, *Neural Networks – A Comprehensive Foundation.* (Maxwell, MacMillan International Publishing Company, 1995)

[16] D.Lowe, *Radial Basis Function Networks, The Handbook of Brain Theory and Neural Networks*, (Cambridge, MA: MIT Press, 1995)