NEURAL NETWORKS FOR CORROSION DATA REDUCTION

I. S. Helliwell and M.A. Turega
Dept. of Computation
R. A. Cottis
Corrosion and Protection Centre
UMIST
PO Box 88
Manchester, UK
M60 1QD

ABSTRACT

Artificial Neural Networks are being used increasingly in many fields of industry and commerce. Their ability to "learn by example" and to generalise this knowledge so as to give correct predictions to previously unseen data make them extremely attractive. There is great potential for applying this technology to corrosion problems as, in principle, artificial neural networks can 'learn' the behaviour of materials in a range of corrosive environments and thereby predict their behaviour.

In practice, data in the literature is derived from differing sources, it is often of poor quality and important parameters are sometimes omitted. Also, care has to be exercised when applying artificial neural networks to ensure that their operational range is bounded and that some measure of output "quality" is given to the user. This paper will discuss the particular problems and limitations of applying neural networks to corrosion data and suggest ways in which these limitations can be overcome.

Keywords: neural networks, error bounds, factors of confidence, corrosion data.

INTRODUCTION

In recent years there has been an explosion in the application of artificial neural networks in many fields of industry and commerce. Their ability to "learn by example" and to generalise this knowledge so as to give correct predictions to previously unseen data make them extremely attractive. Corrosion prediction is an area which has great potential for the use of neural networks and, indeed, there is a small, but growing, list of successful applications (for a review of current applications of neural networks in corrosion see Helliwell et al [1]).

One attractive feature of them is that, for many applications, it is possible to treat them as a "black box" (albeit a black box with control knobs on). That is, although an understanding of the workings of neural networks is advantageous, it is possible to create working applications using software packages where the internals of the neural network are transparent. There are guidelines, both heuristic and theoretically based, that allow the user to optimise the settings of the "knobs" for a given application (for an introduction to neural networks from a corrosion perspective see Helliwell et al [1]).
Neural networks, for all their intrinsic appeal, are not without their limitations. It is all too easy to apply neural networks without consideration of the possible sources of error. Used without caution, a neural network can give answers which are, at the least, inaccurate but are more likely to be extremely misleading or just plain wrong.

In a given MLP (Multi-layer Perceptron - the most frequently used neural network model) application, in response to a given generalisation (unknown) point, there will be, typically, no indication as to the “quality” of the outputs of the network. Yet, in many industrial applications there would be considerable financial implications attached to the subsequent use of inaccurate, incorrect or misleading corrosion data. Hence, neural networks need to be made accountable. Industrial users need to be able to assess the ‘usefulness’ (or inversely - the risk) of utilising the data obtained from the outputs of such a network.

It would be useful, at this point, to review the possible sources of inaccuracy in neural network outputs (for a more detailed discussion see Hellwell et al [2]).

**SOURCES OF ERROR**

The quality of the output data from any given neural network for a given generalisation (unknown) point is dependent upon the following:

1. The representative nature of the original training data in relation to the scope of the problem domain.
2. The properties of the network - network topology, training algorithm, degree of training etc.
3. Whether the generalisation point lies within the domain of validity of the network.
4. The density of training data, and hence the goodness of fit, in the region of the generalisation point.
5. The quality of the training data in the region of the generalisation point.

The first of these is application dependent and relies upon: (i) local domain knowledge, (ii) an appropriate mapping of that domain onto the inputs and outputs of a neural network, and (iii) the availability of a sufficient quantity of suitable data encompassing the possible range of values of those variables. Obviously, the neural network can only be as good as the original data used for training. Poor choice of variables due to a lack of understanding of the problem and/or insufficient badly chosen training data can only lead to uncertainty in the outputs from the network.

The second point is dependent upon the use of general training and validation principles covered in most good neural network textbook and tutorial papers (for a list of recommended texts see Hellwell et al. [1]).

In discussing the other three points, we shall outline specific implications for the use of corrosion data and briefly review some methods available to help the interested reader and, for the final point, we shall outline current research being carried out by the authors in this area.

**TECHNIQUES AND TIPS (OR THE BLACK ART OF USING THE BLACK BOX)**

Once we have trained our neural network, it is necessary to establish the range of input data over which the resultant output data can be believed. This is known as the domain of validity of the network. A neural network is a universal function approximator (Hornik et al. [3]) and, in this respect, can be viewed as a multi-dimensional non-linear interpolator and, common to many other forms of interpolators, the validity of the model formed by the neural network is closely related to the boundaries of the training data (Courrien [4])). Once outside of these bounds it can usually be assumed that the ‘performance’ of the model will degenerate rapidly with the distance from the boundary. Unfortunately, a MLP type network gives us no clues as to when it is interpolating and when it is extrapolating.
A simple, intuitive approach would be to bound the network by the hyperrectangle formed by max. and min. of each of the input variables (fig 1). Unfortunately, in practice, this is often a gross over estimation and forms only a very rough approximation of the actual interpolation domain (fig 2).

Better methods exist of indicating extrapolation (for example Courrieu [4], Bishop [5], Leonard et al.[6][7]) but, unfortunately, they tend to involve a fair amount of mathematics and some involve writing a pre-processor to reject points outside of the domain of validity. In our opinion, though, this is amply repaid by the ability to reject generalisation queries where the unknown point lies outside the domain of validity of the network.

Two of these methods (Bishop [5], Leonard et al [6][7]) also give an indication of data "voids" within the domain of validity (such as the unpopulated region within the interpolation boundary of fig. 2) and so, by choosing sensible thresholds, points can be rejected that lie in any unpopulated region of the input space.

Once we have accepted that a generalisation query (unknown point) lies within the domain of validity of our network we still need a factor of confidence or ideally an error bar on the outputs of the network to reflect the accuracy of the prediction. Again, methods already exist to generate error bars and these are based upon the "noise" of the training data and/or the density of the training data in the local region of input space.

Of the available methods, the following three can be applied using most neural network simulators without resorting to editing of the source code:

i. Bartlett and Kim [8] propose a system which applies Stacked Generalisation to an Artificial Neural Network to provide error bounds on the outputs. In this method a standard MLP neural network is trained in the conventional way and then by following a methodical, if somewhat time consuming procedure, a training set is derived for a secondary network which gives, as its output, the error bounds on the output of the first network. Unfortunately, this method requires the calculation of the distance from any unknown point to its nearest neighbour in the training set. This implies not only storing the training set but also writing a pre-processor to do the calculation.

ii. A similar but simpler method to derive statistical moments is proposed by Satchwell [10]. This involves training a secondary network using data derived easily and quickly from the primary network (which is trained in the standard way). When presented with a generalisation point, the secondary network expresses the variance of the output of the primary network in terms of the input variables. If Gaussian noise is assumed, an estimate of the 95% confidence interval is $2\sqrt{\text{variance}}$ on either side of the mean. Unfortunately, this method, for all its attractions, only reflects the inaccuracies due to the "noise" of the training data and takes no account of local regions of poor fit due to sparse data.

iii. For Radial Basis Function networks, Leonard et al [7][8] have developed a method of extending the network to contain additional neurons. They term their network: "The Validity Index Network" or "Vi-net". The additional neurons give as their output: i. the 95% confidence limits, ii. the local training data density, and iii. the maximum activation of the hidden units (a useful extrapolation measure - see the discussion of the domain of validity above).

Methods for training networks assume the availability of a ready supply of "perfect" data. Published corrosion data, however, does not tend to be perfect. It has typically originated from many sources and is of variable quality. It is finite in quantity and frequently sparse in relation to the possible range of the variables. Often it is the case that data which is known to be of poor quality cannot be rejected due to the lack of anything better.

Commonly Encountered Data Problems

The types of problems commonly encountered within published corrosion data are outlined below:

A. Missing data - Values of one or more primary variables (i.e. network inputs) may not be available for certain data items.
B. Conflicting data - Similar (or identical) published corrosion tests may give widely differing corrosion rates. It is not sufficient in many cases to take the mean as there may be grounds to have more confidence in one test over the other.

C. Inaccurate data collection - Variance in standards and inherent inaccuracies in the test techniques can lead to systematic as well as stochastic errors in the data. Also certain techniques, such as the ASTM-G48 method of determining pitting temperatures, introduce a degree of uncertainty (albeit of known range) due to their incremental nature.

D. Inaccurate data representation - data from short duration tests are often extrapolated to give yearly corrosion rates. It is not uncommon for the results of 1 hour tests to be published (requiring an extrapolation factor of about 8000 to convert to a yearly rate!). If the potential magnification of errors was no problem enough, time variance of the corrosion rate can give rise to large inaccuracies in the data.

Hence, it is common for the training set to contain items that are known a-priori to be doubtful. This may lead, in sparsely populated regions of input space, to predictions being made based wholly upon poor quality data. The problem is, therefore, how to convey to the user the confidence that can be placed in the output of the network based on the “quality” of the training data close (in input space) to the unknown data point, and how can we optimise the training of the network to make the best use of our prior knowledge.

As a first approximation it is tempting to associate a simple factor of confidence with each training point (i.e. as an extra output). This would give, for a given generalisation point, an output and an associated measure of the uncertainty of that output. In practice this proves to be inadequate for two major reasons:

i. A network capable of good generalisation is incapable of producing the very localised representation of data quality required, and

ii. Depending upon the type of data defect, by just associating a simple factor of confidence with a point we are not incorporating our prior knowledge of the nature of the defect directly into the training of the network.

The authors are, at present, engaged in research aimed at developing a new neural network model specifically tailored to maximise the use of, and reduce the errors caused by, the type of data defects illustrated above. This model will aim to give the user maximum accountability by embodying techniques that reflect our knowledge of the potential sources of error. This includes sources of error that are both intrinsic in the use of neural networks and specific to the problem domain. In the meantime, however, it is possible to improve the situation by applying our knowledge of corrosion coupled with a little effort and common sense.

In the following section we shall discuss some simple heuristic methods that can be applied externally to most neural network simulators and which go some way to achieving our aims. In the discussion we shall continue in our view of neural networks as ‘black boxes’ and only consider techniques which can be implemented without the editing of source code.

In deciding the best course of action it is first necessary to establish exactly what it is that we require to aid us in our use of poor quality training data. It turns out that the requirements are, in fact, different for each of the above mentioned data defects. These can be achieved with careful assignment of factors of confidence and estimates of variance to individual data points and optimising network training to make the best use of our extra knowledge.

In general, we associate two extra values with each data point. These are namely:

1. Factor of Confidence - this is a measure of our belief in the data point and is used during training to control the amount of influence each data point has on regressions.

2. Variance - this is an estimate of the variance of the output based on our knowledge of the process used to obtain the data point.

In practice, these can be assigned to families of data points. All test points from a particular lab/site or that have been obtained by a particular technique can be assigned a factor of confidence and a variance measure that reflect our overall knowledge of that family of points.
Firstly we shall expand on the requirements for dealing with the data defects listed above followed by a heuristic approach to optimising network training.

For missing inputs (point A) we have a number of choices. We can:

I. seek expert opinion to obtain a value for the missing variable

ii. make a statistical estimate based on either the mean or a random value from an assumed distribution

iii. make an intelligent assumption about the missing parameter e.g. if the corrosion test temperature is not stated then assume 20°C (or maybe 25°C in the USA!)

Whichever of the three methods we use, there is a level of uncertainty associated with the new value. Obviously we have less confidence in these points and if possible we must reflect this in the way we utilise them in the training of the neural network. We achieve this by associating a lower factor of confidence with these points proportional to our belief in the new value.

For inaccurate data collection (point C) we wish the output to include error bars which reflect not only the generalisation properties of the network, but also a measure of the inherent inaccuracies of the training data points. In effect, we wish to include an error bar on each training data point, and the output of the trained network in response to a generalisation point should include a value that reflects the local measure of training data inaccuracy. With knowledge of specific techniques and experience of the expected variation between different labs/sites it is possible to assign error bars to families of like test points in the form of variance or standard deviation. In the case of short duration tests (point D) we can also apply a correction factor to account for time variance (as these tests generally give an artificially high yearly rate) before using them for training.

For inconsistent data (point B) we may, from expert opinion or statistical evidence, obtain a probabilistic measure of confidence in each point. We wish the influence of each point in the training of the network to reflect the confidence we have in each such point, i.e. if we have two conflicting training points (identical inputs but differing outputs) we may believe that the true value lies closer to one than to the other (we have a greater confidence in one measure than the other) and, therefore, we wish the one with the highest confidence measure to have a greater influence on regression. Again, this is achieved by associating an appropriate factor of confidence with the data points.

The first step in our approach is to train a network with the inputs and outputs required for our corrosion system. The network topology etc. should be optimised in the standard way but the training should be modified as below.

The standard method of training a MLP type neural network is to supply the simulator with a training set that contains each point once. This ensures that each point has equal chance of influencing the regression. In our case, though, we wish the regression to be weighted in favour of points with the highest factor of confidence. A simple method of achieving this is to include multiple copies of the points in the training set in proportion to the amount of confidence in each point.

If it is possible to modify the simulator program appropriately then a more elegant solution is to generate a new training set for each epoch (training cycle). Each point is drawn randomly according to a probability density based on the factor of confidence (actually proportional to the factor of confidence squared). This, again, has the required effect of weighting the regression in favour of points with a higher factor of confidence.

Next, a second network is trained using the same input values but the outputs replaced by their respective variances. This, when used in conjunction with the primary network, will give as its outputs a measure of the variance of the primary network outputs based on the local training data variance. It will be found that this second network will require a greater number of hidden units than the primary network (typically double) to give a local measure of output variance. The relation of the two networks is shown in fig. 3.
Relationship with standard error bar techniques

The output of the second network is only meaningful when the unknown point lies close in input space to at least one training point. If this is not the case, we are, in effect, in a data void and so our corrosion prediction should be treated with caution. To indicate this, it is necessary to combine our predicted variance (output from the second network) with information from one of the other techniques above (such as [5], [6], [7], [8]) that give an indication of local training data density.

CONCLUSIONS

In this paper we have discussed potential sources of error connected with the application of neural networks to corrosion data. Some of these are general to all neural network applications and we have reviewed a selection of published techniques for dealing with them. Other sources of error discussed in the paper are related to the specific data problems that are typified by published corrosion data. We have presented a heuristic “black box” method of both illustrating the inaccuracies of the data and of optimising network training in the light of our knowledge of corrosion data.

The authors are, at present, researching a new neural network model specifically tailored to the requirements of a corrosion prediction tool. This will embody the principles discussed in this paper of greater accountability to the user and maximum use of prior information.

REFERENCES


FIGURES

Figure 1. Two dimensional input space bounded by max-min rectangle.

Figure 2. Actual interpolation boundary.

Figure 3. Primary and Secondary networks.