Lecture 1: Neural Networks

Linear Regression

Most scientists are familiar with regression analysis where data are best–fitted to a specified relationship which is usually linear. The result is an equation in which each of the inputs x_j is multiplied by a weight w_j; the sum of all such products and a constant θ then gives an estimate of the output $y = \sum_j w_j x_j + \theta$. As an example, the temperature at which a particular reaction starts (T_S) in steel may be written:

$$T_S(\,^\circ\mathrm{C}) = \underbrace{830}_{\theta}\ \underbrace{-270}_{w_C} \times c_C\ \underbrace{-37}_{w_{Ni}} \times c_{Ni}\ \underbrace{-70}_{w_{Cr}} \times c_{Cr} \qquad (1)$$

where c_i is the wt% of element i which is in solid solution in austenite. The term w_i is then the best–fit value of the *weight* by which the concentration is multiplied; θ is a constant. Because the variables are assumed to be independent, this equation can be stated to apply for the concentration (wt%) range:

Carbon 0.1-0.55 Nickel 0.0-5.0 Chromium 0.0-3.5

and for this range the start temperature can be estimated with 90% confidence to $\pm 25\,^\circ$C.

There are difficulties with ordinary linear regression analysis as follows:

(a) A relationship has to be chosen before analysis.

(b) The relationship chosen tends to be linear, or with non–linear terms added together to form a pseudo-linear equation.

(c) The regression equation applies across the entire span of the input space.

NEURAL NETWORKS

A general method of regression which avoids these difficulties is neural network analysis, illustrated at first using the familiar linear regression method. A network representation of linear regression is illustrated in Fig. 1a. The inputs x_i (concentrations) define the *input nodes*, the bainite–start temperature the *output node*. Each input is multiplied by a random weight w_i and the products are summed together with a constant θ to give the output $y = \sum_i w_i x_i + \theta$. The summation is an operation which is hidden at the hidden node. Since the weights and the constant θ were chosen at random, the value of the output will not match with experimental data. The weights are systematically changed until a best–fit description of the output is obtained as a function of the inputs; this operation is known as *training* the network.

The network can be made non–linear as shown in Fig. 1b. As before, the input data x_j are multiplied by weights ($w_j^{(1)}$), but the sum of all these products forms the argument of a hyperbolic tangent:

$$h = \tanh\left(\sum_j w_j^{(1)} x_j + \theta\right) \qquad \text{with} \qquad y = w^{(2)} h + \theta^{(2)} \tag{2}$$

where $w^{(2)}$ is a weight and $\theta^{(2)}$ another constant. The strength of the hyperbolic tangent *transfer function* is determined by the weight w_j. The

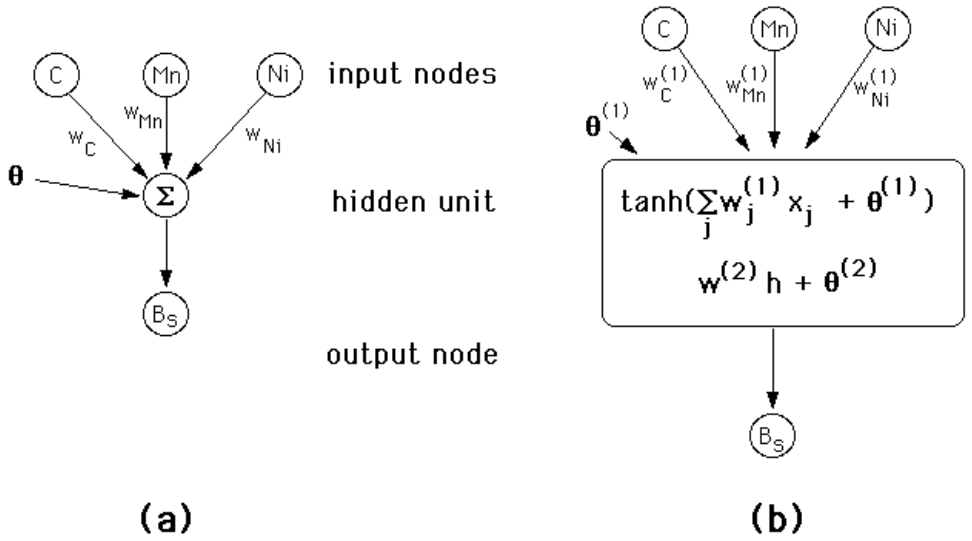

Fig. 1: (a) A neural network representation of linear regression. (b) A non–linear network representation.

output y is therefore a non–linear function of x_j, the function usually chosen being the hyperbolic tangent because of its flexibility. The exact shape of the hyperbolic tangent can be varied by altering the weights (Fig. 2a). Difficulty (c) is avoided because the hyperbolic function varies with position in the input space.

A one hidden-unit model may not however be sufficiently flexible. Further degrees of non–linearity can be introduced by combining several of the hyperbolic tangents (Fig. 2b), permitting the neural network method to capture almost arbitrarily non–linear relationships. The number of tanh functions per input is the number of hidden units; the structure of a two hidden unit network is shown in Fig. 3.

The function for a network with i hidden units is given by

$$y = \sum_i w_i^{(2)} h_i + \theta^{(2)} \tag{3}$$

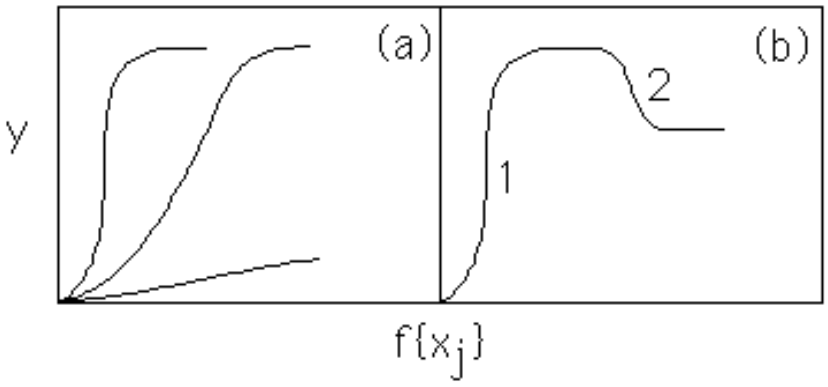

Fig. 2: (a) Three different hyperbolic tangent functions; the "strength" of each depends on the weights. (b) A combination of two hyperbolic tangents to produce a more complex model.

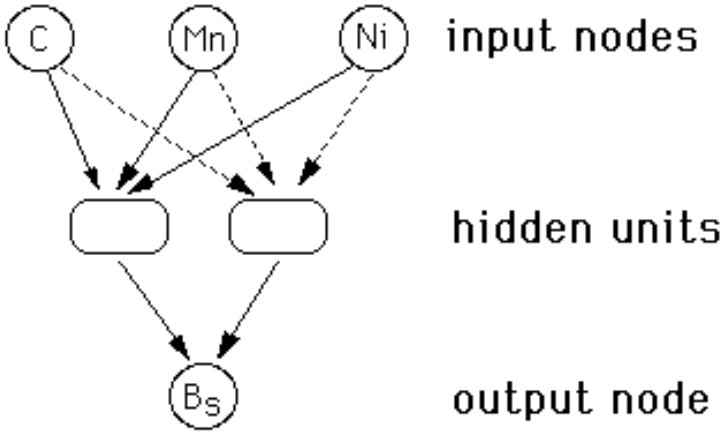

Fig. 3: The structure of a two hidden–unit neural network.

where

$$h_i = \tanh\left(\sum_j w_{ij}^{(1)} x_j + \theta_i^{(1)}\right) \tag{4}$$

Notice that the complexity of the function is related to the number of hidden units. The availability of a sufficiently complex and flexible function means that the analysis is not as restricted as in linear regression

where the form of the equation has to be specified before the analysis.

The neural network can capture interactions between the inputs because the hidden units are nonlinear. The nature of these interactions is implicit in the values of the weights, but the weights may not always be easy to interpret. For example, there may exist more than just pairwise interactions, in which case the problem becomes difficult to visualise from an examination of the weights. A better method is to actually use the network to make predictions and to see how these depend on various combinations of inputs.

Overfitting

A potential difficulty with the use of powerful non–linear regression methods is the possibility of overfitting data. To avoid this difficulty, the experimental data can be divided into two sets, a *training* dataset and a *test* dataset. The model is produced using only the training data. The test data are then used to check that the model behaves itself when presented with previously unseen data. This is illustrated in Fig. 4 which shows three attempts at modelling noisy data for a case where y should vary with x^3. A linear model (Fig. 4a) is too simple and does not capture the real complexity in the data. An overcomplex function such as that illustrated in Fig. 4c accurately models the training data but generalises badly. The optimum model is illustrated in Fig. 4b. The training and test errors are shown schematically in Fig. 4d; not surprisingly, the training error tends to decrease continuously as the model complexity increases. It is the minimum in the test error which enables that model to be chosen which generalises best to unseen data.

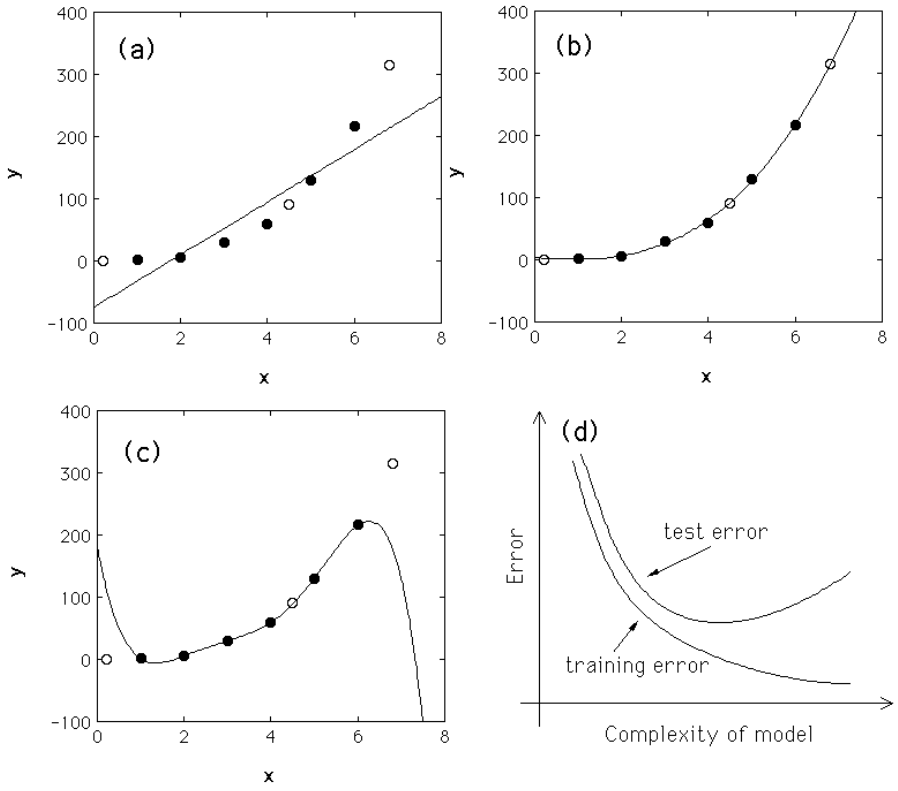

Fig. 4: Test and training errors as a function of model complexity, for noisy data in a case where y should vary with x^3. The filled points represent training data, and the circles the test data. (a) A linear function which is too simple. (b) A cubic polynomial with optimum representation of both the training and test data. (c) A fifth order polynomial which generalises poorly. (d) Schematic illustration of the variation in the test and training errors as a function of the model complexity.

Error Estimates

The input parameters are generally assumed in the analysis to be precise and it is normal to calculate an overall error, E_D, by comparing

the predicted values (y_j) of the output against those measured (t_j), for example,

$$E_D \propto \sum_j (t_j - y_j)^2. \tag{5}$$

E_D is expected to increase if important input variables have been excluded from the analysis. Whereas E_D gives an overall perceived level of noise in the output parameter, it is, on its own, an unsatisfying description of the uncertainties of prediction. Fig. 5 illustrates the problem; the practice of using the best–fit function (*i.e.* the most probable values of the weights) does not adequately describe the uncertainties in regions of the input space where data are sparse (B), or where the data are particularly noisy (A).

MacKay has developed a particularly useful treatment of neural networks in a Bayesian framework, which allows the calculation of error bars representing the uncertainty in the fitting parameters. The method recognises that there are many functions which can be fitted or extrapolated into uncertain regions of the input space, without unduly compromising the fit in adjacent regions which are rich in accurate data. Instead of calculating a unique set of weights, a probability distribution of sets of weights is used to define the fitting uncertainty. The error bars therefore become large when data are sparse or locally noisy, as illustrated in Fig. 5.

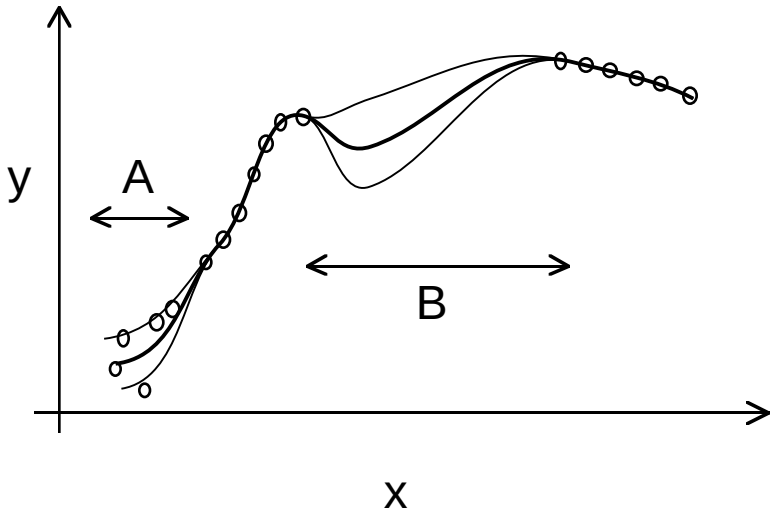

Fig. 5: Schematic illustration of the uncertainty in defining a fitting function in regions where data are sparse (B) or where they are noisy (A). The thinner lines represent error bounds due to uncertainties in determining the weights.