Science of Materials, Volume 0, Number 0, Pages 1-5.

Bayesian Neural Networks as Implemented in Model Manager

Mathew James Peet

1 January 2015

Department of Materials Science and Metallurgy, University of Cambridge, 27 Charles Babbage Road, Cambridge, CB3 0FS, UK.

Abstract

This document gives some details about the use of Bayesian neural networks in materials science, as implemented in Neuromat's Model Manager. Model Manager provides a graphical interface to David MacKays bigback program, used extensively in the Phase Transformations and Complex Properties group at the University of Cambridge.

Keywords— Bayesian Neural Networks, Modelling

1 Introduction

Model Manager implements a system following the work of MacKay [1], who applied neural networks models in a Bayesian framework – this makes it possible to 'quantitatively embody Occam's razor' [2, 3, 4].

2 Main Matter

Model Manager provides a graphical interface to code developed by MacKay and also incorporates further practical methods which further contribute to the success of modelling. Splitting the data into testing and training sets

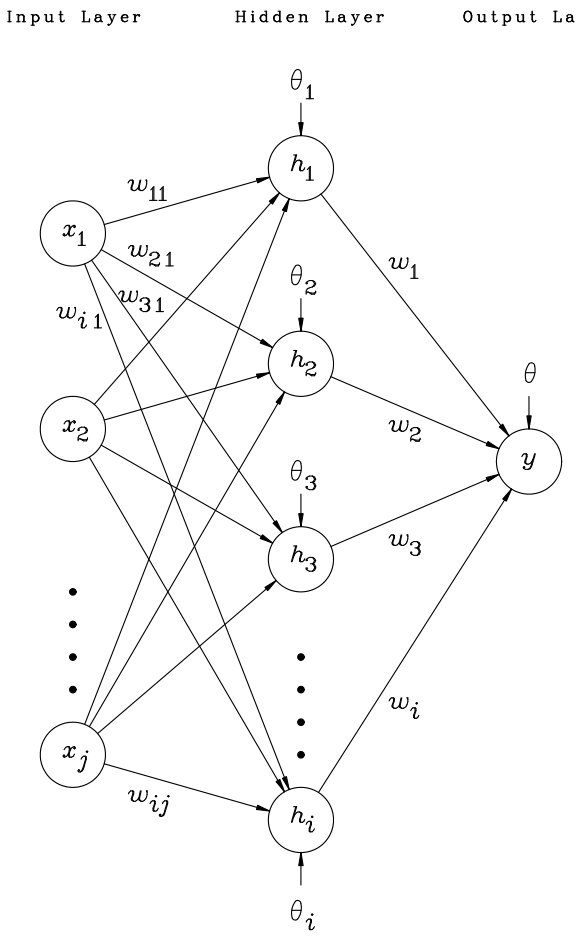

Figure 1: Structure of a three layer neural network.

allows the selection of models which are best able to generalise the trends in the data, so as to reasonably predict the test set. A final model is built from a committee of multiple models which converge from different initial positions in weight space [5]. These methods are further attempts to find the appropriate level of complexity from the data, and to ensure a robust solution is found. The approach used has been applied previously in materials science [5, 6, 7, 8].

Bhadeshia has published two comprehensive reviews on the use and performance of neural networks in materials science [9, 10], with the objective of encouraging the proper application of this method.

There is an implicit assumption in neural network modelling that the input variables are related to the output by a continuous and differentiable function. In training the network the aim is to mimic the behaviour of the underlying equation. MacKay has shown that a sufficiently complex three–layer network (having one hidden layer), as shown in figure 1, using hyperbolic tangent functions in the form of equation 1, is able to imitate any such function.

The neural network in this case is simply a flexible equation that can be fit to any data set by adjusting the parameters or 'weights'. It is usual for the equation 1 to be the sum of several hyperbolic tangent functions equation 2.

The output variable is expressed as a linear summation of activation functions, h_i, weighted by the weights w_i and the bias θ:

$$y = \sum_i w_i h_i + \theta \tag{1}$$

in the case of a hyperbolic tangent formulation the activation function for a neuron i in the hidden layer is given by:

$$h_i = \tanh\left(\sum_j w_{ij} x_j + \theta_i\right) \tag{2}$$

with weights w_{ij} and biases θ_i. In order to simplify the weightings, inputs are linearly normalised within the range ± 0.5 using the normalisation function, $x_j = (x - x_{\min})/(x_{\max} - x_{\min})$ - 0.5, where x is the value of the input and x_j is normalised value.

'Training the network' is achieved by altering the parameters to fit the functions to the data using 'backpropagation' gradient descent optimisation procedures [11], to minimise an objective function. With efficient optimisation care has to be taken to avoid over–fitting, and fitting to local minima [12]. In MacKay's Bayesian treatment over–complex and under–regularised models are automatically inferred to be less probable, even though the flexibility of equation 1 allows them to fit the data better.

This is achieved by calculating and optimising an objective function $\mathrm{M}(w) = \beta E_D + \alpha E_W$ which combines an error term (E_D) to assess how good the fitting is and regularisation term (E_W) to penalise large weights, such as:

$$\mathrm{M}(w) = \beta\left(\frac{1}{2}\sum_i (t^{(i)} - y^{(i)})^2\right) + \alpha\left(\frac{1}{2}\sum_i w_i^2\right) \tag{3}$$

where β and α are complexity parameters which greatly influence the complexity of the model, $t^{(i)}$ and $y^{(i)}$ are the target and corresponding output values for one example input from the training data $x^{(i)}$.

The final values of the weights depend upon the initial guess made for the probability distribution of the weights and the number of hidden units. Therefore a large number of separate models are trained with different starting conditions.

These sub–models are best evaluated using the log predictive error (LPE), in comparison to calculating the test error (sum squared error) this function penalises wild predictions less when they are accompanied by large uncertainties. Assuming that for each example m the model gives a prediction according to the normal distribution with average $y^{(m)}$ and variance ${\sigma^{(m)}}^2$:

$$\text{LPE} = \sum_m \left[\frac{1}{2} \left(t^{(m)} - y^{(m)} \right)^2 / {\sigma^{(m)}}^2 + \log(\sqrt{2}\sqrt{\pi}\sigma^{(m)}) \right] \tag{4}$$

Cross–validation further guards against over–fitting. Splitting the data into a training and testing set, allows alternate sub–models to be compared by their performance in predicting the unseen data using equation 4. A collection of the best n models as ranked by the LPE (or TE) are combined to make a committee model. The prediction being the average of the predictions of the sub–models and the change in variance of predictions being the variance of the committee's predictive distribution plus the mean of the variances:

$$\text{Variance } \sigma^2 = \frac{1}{n} \sum_l {\sigma_y^{(l)}}^2 + \frac{1}{n} \sum_l (y^{(l)} - \bar{y})^2 \tag{5}$$

where n is the number of sub-models [1]. The errors for the different committees can be calculated and the one with the lowest perceived predictive error used.

Once trained the network captures interactions between the inputs because of the non–linearity of the activation function. The nature of the interactions are stored in the weights, however in comparison to linear regression they are difficult to interpret directly, the best way to identify trends is to make predictions using the model. Predictions are accompanied by an indication of the uncertainty which depends upon the confidence of the model for that prediction, and also the level of noise perceived in the data which is assumed to have a Gaussian distribution.

References

[1] D. J. C. MacKay. Bayesian non–linear modelling with neural networks. In H. Cerjak and H. K. D. H. Bhadeshia, editors, *Mathematical modelling of weld phenomena 3*. IOM, 1997.

[2] D. J. C. MacKay. *Bayesian methods for adaptive models*. PhD thesis, Caltech, 12 1992.

[3] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):41, 1992.

[4] D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[5] T. Sourmail, H. K. D. H. Bhadeshia, and D. J. C. MacKay. Neural network model of creep strength of austenitic stainless steels. *Materials Science and Technology*, 18:655–663, 2002.

[6] H. K. D. H. Bhadeshia, D. J. C. MacKay, and L.-E. Svensson. Impact toughness of C–Mn steel arc welds - bayesian neural network analysis. *Materials Science and Technology*, 11:1046–1051, 1995.

[7] M. A. Yescas-Gonzalez, H. K. D. H. Bhadeshia, and D. J. C. MacKay. Estimation of the amount of retained austenite in austempered ductile irons. *Materials Science and Engineering A*, A311:162–173, 2001.

[8] R. C. Dimitriu and H. K. D. H. Bhadeshia. Hot-strength of creep-resistant ferritic steels and relationship to creep-rupture data. *Materials Science and Technology*, 23:1127–1131, 2007.

[9] H. K. D. H. Bhadeshia. Neural networks in materials science. *ISIJ International*, 39:966–979, 1999.

[10] H. K. D. H. Bhadeshia, R. C. Dimitriu, S. Forsik, J. H. Pak, and J. H. Ryu. On the perfomance of neural networks in materials science. *Materials Science and Technology*, 25(4):504–510, 2009.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[12] D. J. C. MacKay. Bayesian methods for neural networks - faq.