Neural network predictions with error bars*

William D. Penny and Stephen J. Roberts

Neural Systems Research Group,
Department of Electrical and Electronic Engineering,
Imperial College of Science, Technology and Medicine,
London SW7 2BT., U.K.
w.penny@ic.ac.uk, s.j.roberts@ic.ac.uk

February 21, 1997

ABSTRACT

When a neural network makes a prediction it will have an error that can be decomposed into the six following sources: (1a) model bias from data, (1b) model bias from training, (2a) model variance from data, (2b) model variance from training, (3) target noise, (4) input noise. We discuss methods for estimating each of these components of error in linear and nonlinear neural networks. In the nonlinear case, the prediction errors are estimated by using a combination of methods; the 'delta method' to capture model variance from data, a committee of networks to capture the model variance from training and an auxiliary network to capture the target noise. This is motivated by considering each neural net prediction to be a 'mixture' of predictions from different networks where the output of each network is modelled as a Gaussian.

1 Introduction

In this paper we investigate methods for estimating the errors associated with predictions from neural networks. Error estimates are useful as they allow confidence intervals to be placed around network predictions. This is particularly important in medical and other safety critical applications.

We restrict our attention to regression problems. That is, the network is given a set of inputs and targets (outputs) which are samples of an underlying data function that the network must learn. The inputs and targets are both real continuous numbers. The data is noisy which means that the inputs do not perfectly describe the targets. Therefore, the network is trained to estimate the conditional mean of the data. This is done through a training (optimization) process whereby the network weights (parameters) are adjusted so as to minimize the discrepancy between the network outputs and the target outputs.

Geman *et al.* [4] have shown that the prediction error can be decomposed into three terms; model bias, model variance and target noise. This is illustrated in Figure 1 which shows the output of two trained networks. The 'bias' and 'variance' terms are defined with respect to an ensemble of networks trained from the *same* random network initialisation, but on different fixed-size samples of the data set. In this paper, we also take into account the variance due to training from *different* random initialisations of the networks. Consequently, the prediction error may be decomposed into the following terms:

**Submitted to IEEE Transactions on Neural Networks*

1. **Model Bias** *(a) From Data:* This occurs in regions of input space where the network output is not equal to the conditional mean of the data. If the new data point falls in a region of high bias the network prediction will be wrong. This contribution to prediction error is greatest for simple networks. *(b) From Training:* This contribution arises if the network is sufficiently structurally complex to learn the conditional mean of the data in a given region of input space, but the optimization algorithm never finds the appropriate solution. This can occur, for example, if training is stopped too early.

2. **Model Variance** *(a) From Data:* If the network were to be trained on a different data set (sampled from the same underlying function) the network response would be different in areas of low training data density. Consequently, if the new data point is far away from the training data it is likely that the prediction error will be large. This contribution to prediction error is greatest for complex networks. *(b) From Training:* Whatever the training method, e.g. gradient descent, conjugate gradients etc., the network will reach a local minimum solution. Given that there are typically many local minima when training multilayer neural networks, different randomly-chosen initial weight settings will therefore lead to different solutions.

3. **Target noise** If the input variables do not inherently predict the target variable *exactly* then there will be some irreducible uncertainty in the predicted value of that target. We might imagine, for example, that if we had a few more input variables the target could be predicted exactly. This irreducible uncertainty is referred to as target noise and is the conditional variance of the data. Target noise may vary non-uniformly over the input spac i.e. there are some regions of input space where the inherent prediction error is larger than elsewhere.

4. **Input noise** This represents intrinsic uncertainty in the values of the input variables. such as experimental measurement error. This leads to uncertainty in the predicted value, as the noise is propagated through the network.

Each of these contributions to prediction error may vary non-uniformly over the input space. In Figure 1, for example, the model is biased at $x = 20$ but not at $x = 10$ or $x = 5$, has high model variance at $x = 5$ but not at $x = 10$ and the target noise is greater at $x = 13$ than at $x = 10$.

In this paper we investigate the following methods for estimating each of the contributions to prediction error. To estimate target noise we train two networks: a regression network and an auxiliary network. The regression network learns the conditional mean of the data. The auxiliary network is trained on the squared errors from the first network and so learns the conditional variance of the data. Model variance due to data (also known as weight uncertainty) is estimated analytically by the delta method approximation. Model variance due to training (also known as committee uncertainty) is estimated by a committee of networks. The contributions to prediction error from model bias and input noise are assumed to be negligible. The general approach we have taken is to choose estimation methods which make as few assumptions as possible about the data and require a minimum of computational effort.

We first review methods for estimating prediction errors in linear regression models (section 2). These methods are then discussed in the context of generalized linear regression. Finally, we look at methods for nonlinear regression (section 3).

We do not discuss confidence intervals in the paper as they can be derived directly from estimates of the prediction error, given that we make some assumption about the distribution of the errors (e.g. that they are normally distributed).

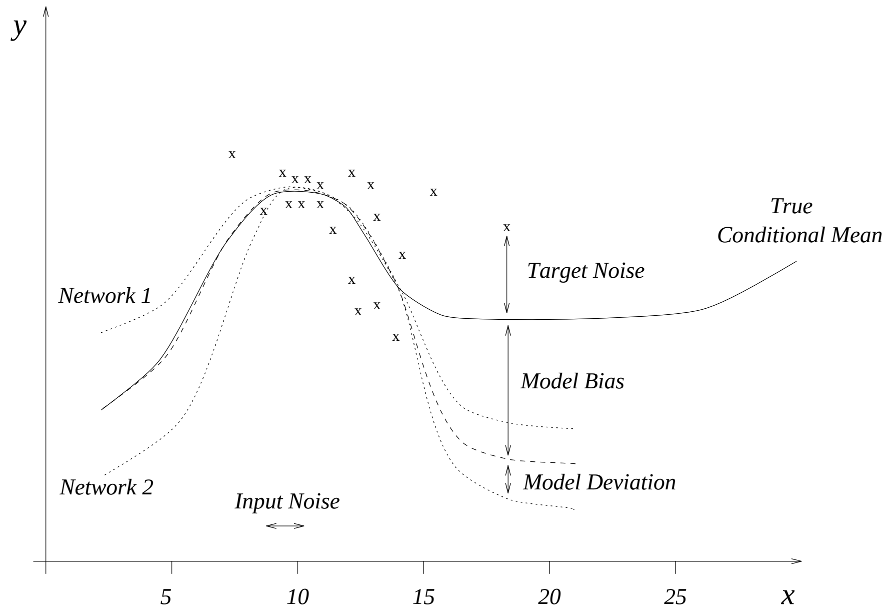

Figure 1: *Prediction error in a neural network regression problem may be decomposed into the four illustrated components. Model bias and variance may be further decomposed into contributions from data and contributions from the training process. The solid line is the true conditional mean of the function to be learnt, the data points, 'x', are noisy samples of that function, network outputs are shown by the dotted lines and the mean of the two network outputs is shown by the dashed line.*

2 Linear regression models

Linear regression models are equivalent to a single linear node in a feedforward neural network. It is therefore instructive to see how prediction errors are estimated in this more straightforward context.

Linear regression models are not subject to prediction errors from model variance from training. This is because the error space (for a least squares cost function, anyway) has a single global minimum. They are, however, particularly susceptible to model bias error as a linear model is the simplest model we can postulate and very often this will be an over-simplification. This fact must be borne in mind when considering the analyses that follow which assume that the linear model is unbiased for the data at hand.

We now look at methods for estimating prediction errors due to target noise and model variance from data. We first consider the case where the target noise is uniform over the input space and then we allow it to vary non-uniformly. Press *et al.* [10] (p. 666) review methods for estimating errors due to input noise via the system's Jacobian matrix.

2.1 Uniform target noise

Consider data generated from the following univariate linear function

$$y_i = ax_i + b + e_i \tag{1}$$

where e_i is independent, identically distributed (iid) random noise drawn from a distribution having standard deviation σ. For the moment, we assume σ to be uniform over the input space x.

This can be modelled with the linear regression equation

$$\hat{y}_i = \hat{a} x_i + \hat{b} \tag{2}$$

The goodness of fit of the model to the data may be measured by the least squares cost function

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3}$$

The values of $\hat{a}$ and $\hat{b}$ that minimize the above cost function can be calculated by setting the first derivatives of the cost function to zero and solving the resulting simultaneous equations [17]. This enables least squares fitting of a regression line to a data set as shown in Figure 2.

If there was no noise on the y_i's (Eq. 1) then $\hat{a}$ and $\hat{b}$ could be calculated exactly, provided we had at least two data points. They would be equal to a and b. Because there is noise, however, then there is some variation in the calculated values of $\hat{a}$ and $\hat{b}$. If we were to repeat the regression process on a number of data sets drawn from Eq. 1 the mean values of $\hat{a}$ and $\hat{b}$ would be equal to a and b. It is also possible to calculate the variances, and from these, the variance of a prediction, $\tilde{y}$, on a new data point $\tilde{x}$. From [17] we have

$$\sigma_{\tilde{y}}^2 = \sigma^2 + \sigma^2 [\frac{1}{N} + \frac{1}{N-1} \frac{(\tilde{x} - \bar{x})^2}{\sigma_x^2}] \tag{4}$$

where $\bar{x}$ and σ_x^2 are the mean and variance of x as measured on the training set. The variance of predictions on new data points has two parts; the part due to the intrinsic noise on the new target, σ, and the part due to the uncertainty in estimating the weights (model variance from data). The second factor increases the further away the new data point is from $\bar{x}$. This can be seen in Figure 2 which shows linear prediction with error bars at $\pm 2\sigma_{\tilde{y}}$. Note that data points further away from the training set are assigned larger error bars.

If the true variance of σ^2 is unknown, as is usually the case, then it can be estimated by the sample variance

$$\hat{\sigma}^2 = \frac{\sum (y_i - \hat{y}_i)^2}{N-2} \tag{5}$$

where the denominator is $N-2$ because two degrees of freedom are lost in calculating the regression line weights.

For a multivariate linear data set, the dependent variable y_i depends on the p dimensional input vector, $\boldsymbol{x}_i$. The ith output is hence generated as

$$y_i = \boldsymbol{w}^T \boldsymbol{x}_i + e_i \tag{6}$$

This may be modelled by the regression equation

$$\hat{y}_i = \hat{\boldsymbol{w}}^T \boldsymbol{x}_i \tag{7}$$

The goodness of fit can be assessed by the least squares cost function Eq. 3. The weight vector that minimises this cost function can again be calculated by setting the first derivatives of the cost function to zero and solving the resulting simultaneous equations. The solution is given by a matrix inversion [17] which is best solved by calculating the pseudo-inverse [10]. Due to the noise on the targets, a different realisation of the data set will result in a different weight vector. These different weight vectors will give different predictions, $\tilde{y}$, on a new data point $\tilde{\mathbf{x}}$. The variance of these predictions is given by [17]

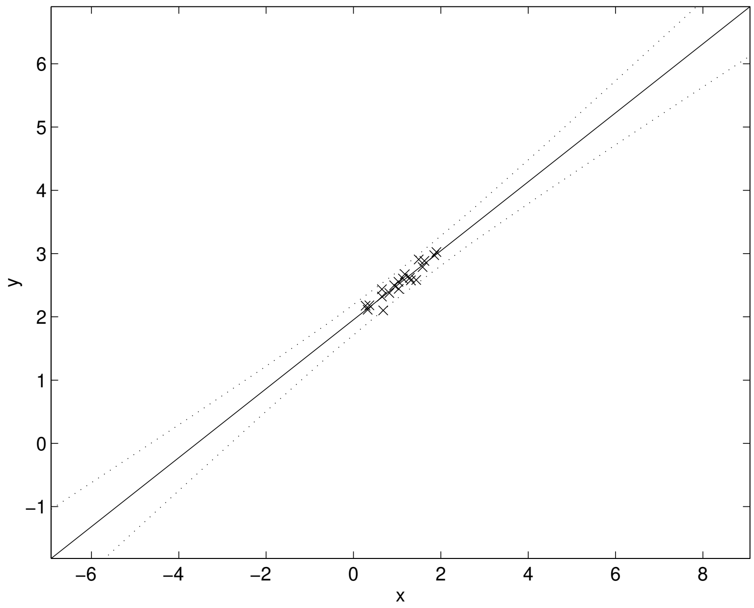

Figure 2: *Linear regression (-) with error bars at $\pm 2\sigma_{\tilde{y}}$ (..). The target noise variance is uniform over the input space.*

$$\sigma_{\tilde{y}}^2 = \sigma^2 + \sigma^2 [\frac{1}{N} + \frac{1}{N-1} d(\tilde{x})] \tag{8}$$

where $d(\tilde{x})$ is the Mahalanobis distance between $\tilde{x}$ and $\bar{x}$. This measures the distance of the new input vector from the centre of the training data and is given by

$$d(\tilde{\boldsymbol{x}}) = (\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}})^T \boldsymbol{C}^{-1} (\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}) \tag{9}$$

where $\boldsymbol{C}$ is the covariance matrix of input data in the training set.

2.2 Non-uniform target noise

We now consider making predictions from a data set that is linear but has additive noise with a variance that itself varies as a function of x. Thus, some points in the input space will be inherently less predictable than others. This more realistic assumption is known in the statistics literature as heteroskedasticity.

Consider the univariate case, where the ith data point is generated as

$$y_i = a x_i + b + e_i \tag{10}$$

Again we model the data with the linear regression equation

$$\hat{y}_i = \hat{a} x_i + \hat{b} \tag{11}$$

The goodness of fit of this model to the data may be measured by the general least squares cost function

$$E = \sum_{i=1}^{N} w_i (y_i - \hat{y}_i)^2 \tag{12}$$

where $w_i = 1/\sigma_i^2$. This cost function can be derived from a maximum likelihood paradigm when the errors are known to be normally distributed. Each squared difference in the cost function is weighted by the inverse variance of the noise at that point. This is known as 'weighted least squares'. Consequently, Equation (3) defines the 'ordinary least squares' cost function.

The values of $\hat{a}$ and $\hat{b}$ that minimize the above cost function can again be found by setting the first derivatives of the cost function to zero and solving the resulting simultaneous equations. It is also possible to calculate the variance of the weight values and from these the variance of a prediction on a new data point, $\tilde{x}$. From [17] we have

$$\sigma_{\tilde{y}}^2 = \sigma^2(\tilde{x}) + \left[\frac{1}{\sum w_i} + \frac{1}{N-1} \frac{(\tilde{x} - \bar{x}_w)^2}{\sigma_{x_w}^2} \right] \tag{13}$$

where $\bar{x}_w$ and $\sigma_{x_w}^2$ are the weighted mean and variance of the input data

$$\bar{x}_w = \sum w_i x_i \tag{14}$$

$$\sigma_{x_w}^2 = \sum w_i (x_i - \bar{x}_w)^2 \tag{15}$$

Thus, the further the new data point is from the weighted mean input in the training data set, the larger the error bar.

If the target noise variance function is unknown then it can be estimated. These estimates then define the weight values that can be plugged into weighted least squares.

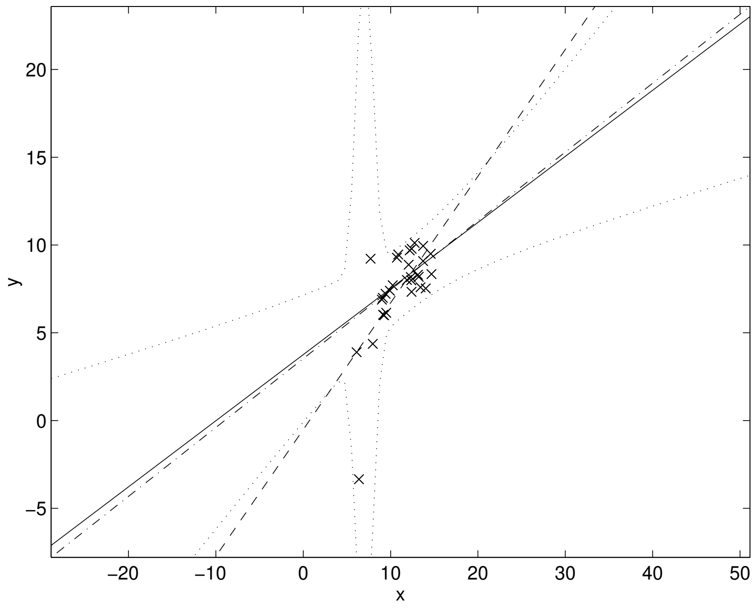

Figure 3: *Weighted linear regression with true weight values (solid line), weighted linear regression with estimated weight values (dash-dotted line), linear regression (dashed line). The error bars (dotted lines) are at $2\sigma_{\tilde{y}}$ above and below the estimated weighted least squares regression line.*

Wilcox [18] compares a number of methods for estimating the noise variance for the case of univariate linear regression. Among the best methods is a running interval smoother.

The method proceeds as follows. Firstly, the normalised median absolute deviation, σ_{norm}, of the input data is calculated. This is a robust statistic that estimates the standard deviation if the data are normally distributed. All input points within absolute distance σ_{norm} to a training data point, x_i, are then defined to be within that training point's neighbourhood. The estimated noise variance corresponding to a particular training point is then defined to be the variance of all y values within the neighbourhood of x_i.

We illustrate the method by generating data from a linear model with noise variance defined by the function

$$\sigma^2(x) = 100\sigma^2[\exp(-(x-7)^2)+1] \tag{16}$$

which gives an 'error hotspot' at $x = 7$.

Figure 3 shows the weighted least squares regression using estimated weights from the Wilcox method. In the particular data realisation shown, there are more points below the mean than above the mean in the region of high noise variance. This causes the ordinary least squares linear regression line to overestimate the gradient. The estimated weighted least squares line largely ignores the data in this region and so is better able to estimate the true underlying function.

2.3 Generalized linear regression

Generalized linear regression models may be written as

$$\hat{y}_i = \hat{\boldsymbol{w}}^T \Phi(\boldsymbol{x}_i) \tag{17}$$

where $\Phi(\boldsymbol{x}_i)$ are nonlinear basis functions. These might be, for example, radial basis functions with fixed centres and covariances or quadratic or higher order combinations of the original input variables.

The main attraction of generalized linear regression is that the models are nonlinear in the inputs but linear in the weights. This linearity property means that training is equivalent to linear regression but in the space of 'hidden' unit functions, $\Phi(\boldsymbol{x})$, instead of in the original input space, $\boldsymbol{x}$. Thus, the formula for calculating the prediction error due to model variance and target noise is the same as before but with $\boldsymbol{x}$ replaced by $\Phi(\boldsymbol{x})$. For the case of uniform target noise we have,

$$\sigma^2_{\tilde{y}} = \sigma^2 + \sigma^2\left[\frac{1}{N} + \frac{1}{N-1} d(\Phi(\tilde{\boldsymbol{x}}))\right] \tag{18}$$

where $d(\Phi(\tilde{\boldsymbol{x}}))$ is the Mahalanobis distance in the space of hidden units,

$$d(\Phi(\tilde{\boldsymbol{x}})) = (\Phi(\tilde{\boldsymbol{x}}) - \bar{\Phi}(\boldsymbol{x}))^T \boldsymbol{C}^{-1}(\Phi(\tilde{\boldsymbol{x}}) - \bar{\Phi}(\boldsymbol{x})) \tag{19}$$

and $\boldsymbol{C}$ is the hidden unit covariance matrix with elements

$$\boldsymbol{C}_{ij} = Cov[\phi_i(x)\phi_j(x)] \tag{20}$$

where $\phi_j(x)$ is the output of the jth basis function. These results may be used to construct elliptical contours of constant prediction error in the hidden unit space. Using an iterative inversion algorithm [6], these contours can be mapped back into the original input space.

Estimates of model bias and non-uniform target noise are considered in [12].

3 Nonlinear regression

In the general nonlinear regression case the exact form of the function generating the data (the conditional mean) , $\bar{f}$, is unknown and there is iid additive noise, e_i

$$y_i = \bar{f}(\boldsymbol{x}_i) + e_i \tag{21}$$

The data can be modelled with a nonlinear neural network model

$$\hat{y}_i = f(\boldsymbol{x_i}, \hat{\boldsymbol{w}}) \tag{22}$$

where

$$f(\boldsymbol{x}_i, \hat{\boldsymbol{w}}) = \sum_j^M \hat{c}_j \phi(\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}_j) \tag{23}$$

is a multi-layer perceptron (MLP), if ϕ is the sigmoid or hyperbolic tangent function, or a radial basis function (RBF) if ϕ is a gaussian, for example. The vector $\hat{\boldsymbol{\theta}}_j$ denotes the hidden-layer weights of the jth node and $\hat{c}_j$ is the output layer weight connecting the jth node to the output.

Geman *et al.* [4] have shown that the prediction error can be decomposed into three terms; model bias, model variance and target noise. The 'bias' and 'variance' terms are defined with respect to an ensemble of networks trained on different fixed-size samples of the data set. Each network is considered to be trained from the *same* randomly initialised weight vector.

In this paper, we also take into account the variance due to training from *different* random initialisations of the networks. This is achieved by considering the prediction, $\hat{y}$, to be a 'mixture' of predictions from different networks, $\hat{y}_j$. That is, $\hat{y} = \hat{y}_j$ with probability p_j where p_j is the probability of selecting network j. The prediction from each network is itself a random variable that captures the variation from different data sets. The 'mixing' of these predictions captures variation due to the training process. Each of these variations can be decomposed into bias and variance terms.

We first decompose the prediction error by taking an average over all different network initialisations, $j = 1..C$

$$E[(\hat{y} - y)^2] = E[(y - \bar{y})^2] + (E_C[\hat{y}] - \bar{y})^2 + Var_C[\hat{y}] \tag{24}$$

It is important to note that each term is dependent on $\boldsymbol{x}$. We have left out this dependence in order to make the equations more readable.

Now, each $\hat{y}_j$ is itself a random variable that captures the variation due to different data realisations. It has mean $E_D[\hat{y}_j]$ and variance $Var_D[\hat{y}_j]$.
Hence

$$E_C[\hat{y}] = \sum_j p_j E_D[\hat{y}_j] \tag{25}$$

and

$$Var_C[\hat{y}] = Var_C[E_D[\hat{y}_j]] + E_C[Var_D[\hat{y}_j]] \tag{26}$$

where the second equation is arrived at by noting that the variance of a mixture of distributions is the variance of the means plus the mean of the variances. Substituting this into our earlier decomposition (Eq. 24) gives

$$E[(\hat{y}-y)^2] = E[(y-\bar{y})^2] + (E_C[\hat{y}]-\bar{y})^2 + Var_C[E_D[\hat{y}_j]] + E_C[Var_D[\hat{y}_j]] \tag{27}$$

which we rewrite as

$$\sigma_{\hat{y}}^2 = \sigma_t^2 + \sigma_b^2 + \sigma_C^2 + \sigma_{WU}^2 \tag{28}$$

where σ_t^2 is the target noise, σ_b^2 is the model bias (from training and data), σ_C^2 is the model variance from training (or committee variance) and σ_{WU}^2 is the average model variance due to data (or average weight uncertainty).

We have also assumed that the contribution to prediction error from input noise is negligible. For a treatment of input noise see Webb [16].

We now estimate each of the other components in turn.

3.1 Model variance from data

To estimate the prediction error due to model variance from data we first assume that the target noise is distributed as $N(0, \sigma^2(\boldsymbol{x}))$ where $\sigma^2(\boldsymbol{x})$ is the input-dependent target noise variance, and $N(\mu, v)$ denotes a normal distribution with mean μ and variance v.

Learning occurs by minimization of the weighted least squares error function

$$E = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{\sigma_i^2} \tag{29}$$

The choice of $\hat{\boldsymbol{w}}$ that minimises the above function can be found by an iterative optimization procedure such as steepest descent, conjugate gradients or pseudo-Newton methods [2].

We now derive the 'delta method' [3] (p.313) approximation for the prediction error. Depending on the particular realisation of the target noise, e_i, a different solution $\hat{\boldsymbol{w}}$ will be found. Following Press *et al.* [10] (p.695) we assume that the targets can be approximated by a first order Taylor series expansion about the network output

$$y_i = \hat{y}_i(\boldsymbol{x}_i, \boldsymbol{w}) = \hat{y}_i(\boldsymbol{x}_i, \hat{\boldsymbol{w}}) + \boldsymbol{g}(\boldsymbol{x}_i, \hat{\boldsymbol{w}})\Delta\boldsymbol{w} \tag{30}$$

where $\boldsymbol{g}$ is the weight sensitivity vector with elements $d\hat{y}/d\hat{w}$ and $\Delta\boldsymbol{w} = \boldsymbol{w} - \hat{\boldsymbol{w}}$.

If we make the further assumption that the network is unbiased then $\hat{y}_i(\boldsymbol{x}_i, \hat{\boldsymbol{w}}) = \bar{f}(\boldsymbol{x}_i)$. Hence we may write

$$e_i = y_i - \bar{f}(\boldsymbol{x}_i) = \boldsymbol{g}(\boldsymbol{x}_i, \hat{\boldsymbol{w}})\Delta\boldsymbol{w} \tag{31}$$

The weight change $\boldsymbol{\Delta w}$ is then linearly related to e_i. Therefore the weights are distributed normally, according to

$$p(\Delta\boldsymbol{w}) \propto exp[-\frac{1}{2}\Delta\boldsymbol{w}^T Cov(\hat{\boldsymbol{w}})^{-1}\Delta\boldsymbol{w}] \tag{32}$$

where $Cov(\hat{\boldsymbol{w}})$ is the weight covariance matrix. The associated change in the cost function is given by

$$\Delta E = \frac{1}{2}\Delta\boldsymbol{w}^T Cov(\hat{\boldsymbol{w}})^{-1}\Delta\boldsymbol{w} \tag{33}$$

The error can also be expanded as a second order Taylor series, giving

$$\Delta E = \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{H} \Delta\boldsymbol{w} \tag{34}$$

where the first order term, ∇E, is zero at the error minimum and $\boldsymbol{H}$ is the Hessian matrix given by $\nabla\nabla E$. Equating these two equations we have,

$$Cov(\hat{\boldsymbol{w}}) = \boldsymbol{H}^{-1} \tag{35}$$

This is an approximation that is valid as long as the noise on the targets is normally distributed, and as long as the deviations in $\hat{\boldsymbol{w}}$ do not extend outside the region in which the network can be replaced by a suitably linearized network. We have also assumed that the network is unbiased.

The variance in the prediction on a new data point due to this weight uncertainty is calculated as

$$Var_D[\hat{y}_j] = \boldsymbol{g}^T \boldsymbol{H}_j^{-1} \boldsymbol{g} \tag{36}$$

where $\boldsymbol{g}$ is the weight sensitivity vector with elements $d\tilde{y}/d\hat{w}$ and the subscript j denotes network j. The Hessian matrix has elements given by [10]

$$h_{lk} = \sum_i^N \frac{1}{\sigma_i^2}\left[\frac{\partial \hat{y}(\boldsymbol{x}_i, \hat{\boldsymbol{w}})}{\partial \hat{w}_l}\frac{\partial \hat{y}(\boldsymbol{x}_i, \hat{\boldsymbol{w}})}{\partial \hat{w}_k} + [y_i - \hat{y}_i]\frac{\partial^2 \hat{y}(\boldsymbol{x}_i, \hat{\boldsymbol{w}})}{\partial \hat{w}_l \partial \hat{w}_k}\right] \tag{37}$$

Therefore, larger noise levels increase the weight uncertainty. This is because large noise levels decrease the size of Hessian elements. The weight uncertainty term is also larger for networks with more weights and for inputs in regions of lower data density.

Thus, by training a single network on a single data set, the prediction error due to model variance from a hypothesised ensemble of data sets can be estimated.

The output from the jth trained network, $\hat{y}(\boldsymbol{x}, \boldsymbol{w}_j)$ is assumed to be equal to the mean over this ensemble, $E_D[\hat{y}_j]$. If the network output is linearized according to a first order Taylor series, then the normal distribution of weight values maps onto a normal distribution of output errors which are distributed as $N(0, \boldsymbol{g}^T \boldsymbol{H}_j^{-1} \boldsymbol{g})$. Hence, the output from the jth network is distributed as $N(\hat{y}(\boldsymbol{x}, \hat{\boldsymbol{w}}_j), \boldsymbol{g}^T \boldsymbol{H}_j^{-1} \boldsymbol{g})$.

3.2 Model variance from training

We now assume that each of the C network solutions is arrived at with probability $p(j)$. Hence the full distribution of network outputs is given by the Gaussian mixture

$$p(\hat{y}(\boldsymbol{x})) = \sum_{j=1}^{C} p(j) p(\hat{y}(\boldsymbol{x}, \hat{\boldsymbol{w}}_j)) \tag{38}$$

If we believe that the conditional mean of the data is multi-modal, as would be the case for inverse problems, then the data is best modelled with a Mixture Density Network [2]. If we believe, however, that the true data function is unimodal, as is usually the case for regression problems, then we are interested only in the conditional mean and variance. The conditional mean is given by the output of a committee of networks

$$\hat{y}_{COM} = E_C[\hat{y}] = \sum_j p(j)\hat{y}(\boldsymbol{x}, \hat{\boldsymbol{w}}_j) \tag{39}$$

The model variance from training is then the variance of the committee

$$\sigma_C^2 = \sum_j p(j)(\hat{y}(\boldsymbol{x}, \hat{\boldsymbol{w}}_j) - \hat{y}_{COM})^2 \tag{40}$$

The average weight uncertainty variance is given by

$$\sigma^2_{WU} = \sum_j p(j) \boldsymbol{g}^T \boldsymbol{H}_j^{-1} \boldsymbol{g} \tag{41}$$

All of these terms are input-dependent. At the beginning of section 3 we considered that the final prediction was given by a mixture of predictions from each of the networks, where each network prediction was itself a random variable. If we believe, however, that the true data function is unimodal then we can say that the final prediction is given by the output of the committee, y_{COM}. For simplicity, we assume that $p(j) = 1/C$ so that individual networks are weighted equally.

3.3 Target noise

The target noise term can be estimated by training an auxiliary network on the residuals of the committee predictions

$$\hat{v}(\boldsymbol{x}) = (\hat{y}_{COM}(\boldsymbol{x}) - y(\boldsymbol{x}))^2 \tag{42}$$

which can be rewritten by expanding about $\bar{y}$ to give the first two terms in Eq. 27. Consequently, the output of the auxiliary network will be equal to a noise term and a bias term

$$\hat{v}(\boldsymbol{x}) = \sigma_t^2(\boldsymbol{x}) + \sigma_b^2(\boldsymbol{x}) \tag{43}$$

If we assume that the network is unbiased then $\hat{v}(\boldsymbol{x}) = \sigma_t^2(\boldsymbol{x})$. To ensure that $\hat{v}(\boldsymbol{x})$ is positive, we use an MLP with an exponential output function.

3.4 Example 1: Bump data

The methods for estimating prediction error are illustrated by training an MLP on data points generated from the function

$$y_i = \phi(x_i - 4) + \phi(-x_i + 6) - 1 + e_i \tag{44}$$

where ϕ is the sigmoid function

$$\phi(a) = 1/[1 + \exp(-a)] \tag{45}$$

and e_i is iid random noise with non-uniform variance given by

$$\sigma^2(x_i) = \exp(9.2\phi(3x_i - 16.5) + 9.2\phi(-3x_i + 19.5) - 18.4) \tag{46}$$

The underlying data function corresponds to a 'bump' at x=5. There is a target noise 'hot spot' at x=6. One hundred data training points were generated uniformly in the range $0 \leq x \leq 12$.

The data was then modelled with a committee of MLPs. Networks were trained to learn the conditional mean of the data. The accuracy of the approximation is assessed with the Normalised Mean Squared Error

$$NMSE = \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{47}$$

Training was stopped at $NMSE \leq 0.01$. An auxiliary network was then trained to learn the conditional variance of the data by training on the squared residuals of the committee predictions. Training was stopped at $NMSE \leq 0.5$. The stopping points were chosen so that the networks were seen to capture the main features of the data sets. In larger problems, where learning cannot be observed visually, the stopping points must be chosen by cross-validation.

If a network failed to reach the convergence criteria within $100W$ function evaluations (where W is the number of weights in each network) it was re-initialised from a different random initial weight vector.

Each network had 4 sigmoidal hidden units and was trained with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [10].

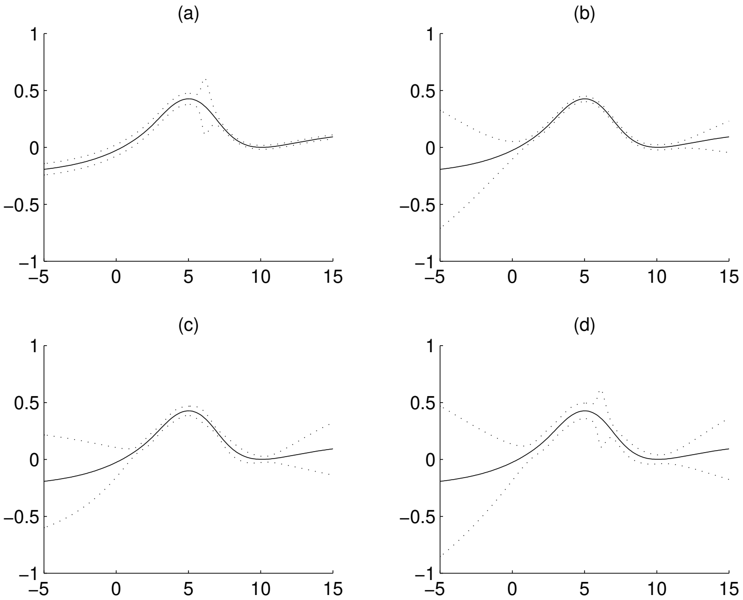

Figure 4: *Regression function and error bars: (a)* $\hat{y} \pm 2\sigma_t$ *(b)* $\hat{y} \pm 2\sigma_{WU}$ *(c)* $\hat{y} \pm 2\sigma_C$ *(d)* $\hat{y} \pm 2\sigma_{\hat{y}}$ *(See Equation 28)*

Figure 4 shows the learnt regression function and each of the components of prediction error. The weight uncertainty (Figure 4b) and committee uncertainty (Figure 4c) error bars are larger when the networks are extrapolating. The auxiliary network (Figure 4a) picks out the region of high target noise.

Figure 5 shows a plot of estimated prediction error and actual absolute prediction error, for a test data set drawn uniformly in the range $-5 \leq x \leq 15$. In regions where the networks are extrapolating ($x < 0$, $x > 12$) the correlation between estimated and actual errors is very strong (correlation coefficient, r = 0.98). In the region where the target noise suddenly increases ($5 \leq x \leq 7$) the correlation is also strong (r = 0.65). In the remaining region, the networks are interpolating in a region of constant target noise. For this reason the estimated prediction error is approximately constant and is therefore not correlated to the actual errors, but it does provide an upper bound.

3.5 Example 2: Robot arm data

We now look at the 'robot arm data set' [8], which describes the mapping for a two-joint robot arm. There are two input variables and one target variable (originally there are two target variables, but we use only the first one).

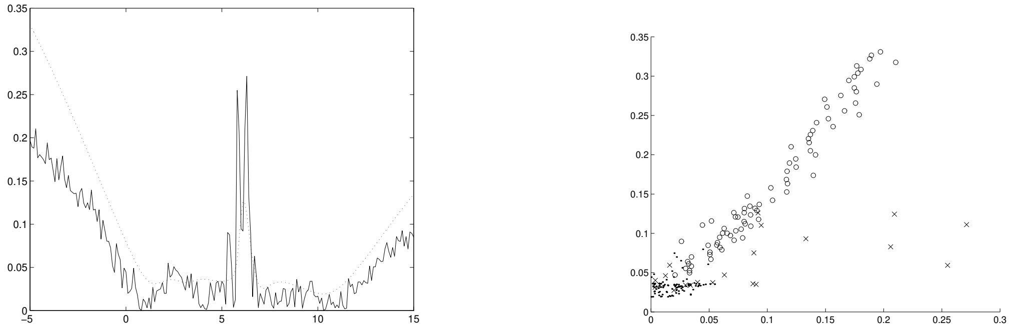

Figure 5: (a) Estimated prediction error, $\sigma_{\hat{y}}$, (dotted line) and actual absolute prediction errors (solid line), (b) Estimated versus actual prediction errors for $x < 0$, $x > 12$ (circles), $5 \leq x \leq 7$ (crosses),elsewhere (dots).

Our training and test sets were generated so as to force the networks to extrapolate during testing. To this end, the training data consisted of the 100 points near the top left and bottom right corners of the plots shown in Figure 6. The test data consisted of 100 examples from the same two areas and a further 100 examples from near the top right and bottom left corners.

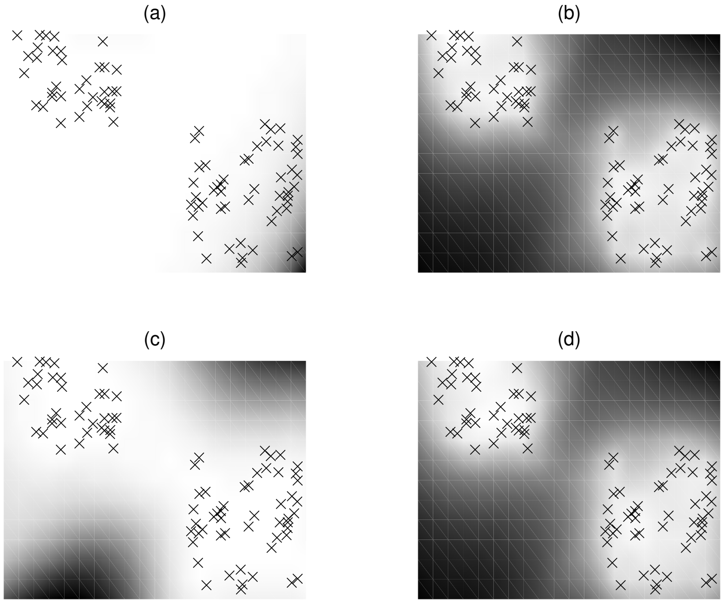

Figure 6: Estimated prediction error due to (a) target noise, (b) log (weight uncertainty), (c) committee uncertainty, (d) all components. Darker areas correspond to higher prediction error.

The data was then modelled with a committee of ten MLPs each with ten hidden units. An auxiliary network was then trained to learn the conditional variance of the data by training on the squared residuals of the committee predictions. The training method was the same as in the last example.

Figure 6 shows how each of the contributions to prediction error varies over the input space. The weight uncertainty and committee uncertainty terms are large when the networks are extrapolating ((b) and (c) respectively). For this data set, however, the weight uncertainty term increases to arbitrarily high values which are several orders of magnitude larger than the actual errors measured on the test set (we have therefore plotted the log of the weight uncertainty error in Figure 6b). Thus, although these estimates are strongly correlated with the actual errors they do not provide a useful quantitative estimate. The committee uncertainty errors, however, are in the correct range.

It is also worth noting that the auxiliary network shows high prediction error in the area of input space at the bottom right corner of Figure 6a. As we indicated in section 3.3, the auxiliary network will primarily model the target noise, but if the network is biased (i.e. it does'nt have enough hidden units) then it will pick up model bias error as well. As our data set is known to have uniform target noise [8] the auxiliary network is therefore picking up model bias error. This indicates that a more complex network would be more suitable for modelling the data set. This would eliminate the model bias error but at the cost of increasing the weight uncertainty error. In a companion paper [11] it is shown how to use estimates of model bias error and weight uncertainty error for selecting networks with optimal structures.

4 Discussion

This paper has described methods for estimating prediction errors in linear and nonlinear neural networks. These estimates are necessary as the prediction error may be non-uniform over the input space. For different problems, different components of the prediction error will be more or less important.

The target noise component was estimated by training an auxiliary network on the squared errors from the regression network. This method has been proposed by Satchwell [13] and Nix and Weigend [9]. If the regression network estimate is assumed to be unbiased then the auxiliary network will model the target noise term only. This estimate can be used to retrain the regression network with a weighted least squares error criterion. This results in more accurate predictions from the regression network as more computational effort can be focussed on fitting the function more accurately to those data points with low target noise [9]. More generally, however, the auxiliary network will pick up target noise *and* model bias error. For some problems, the target noise will be uniform over the input space so there is no point in training an auxiliary network to model its variation; the usual (global) residual training error will suffice as an estimate of target noise.

The model variance component due to averaging over data sets was estimated analytically by the 'delta method' approximation. This estimate assumes that the network is unbiased and that the target noise is normally distributed. The assumption that the network is unbiased may be relaxed if an alternative estimate, the Sandwich approximator, is used [15].

The model variance component due to the training process was estimated by a committee approach. This required training a number of networks. Similar approaches which may have a number of advantages are the bootstrap methods. Bootstrap estimates of prediction error [15] are based on sampling the training set with replacement to create 'bootstrap samples'. Each bootstrap sample is then used to train a neural network. Thus, many networks need to be trained. The bootstrap pairs approach resamples from the original training set and the bootstrap residuals approach resamples from the residuals of a trained network. Bootstrap estimates of prediction error estimate the components due to model variance from data *and* from training. This is because each network is trained from a different randomly initialised weight vector. Furthermore, in the Bootstrap residual approach we need no longer assume that the errors are normally distributed. In the Bootstrap pairs approach we need no longer assume that the model is unbiased. Thus, if we are going to take into account the component of prediction error due to model variance from training, as we have to train many networks anyway, it would seem worthwhile to use bootstrap methods. Heskes [5] investigates prediction errors by using boostrap methods to estimate model variance due to data and training, and the auxiliary network method to estimate input-dependent target noise. The combination of estimation methods provided more realistic error bars. For large data sets or large networks, however, the committee and bootstrap approaches may be too computationally demanding.

The delta method, in contrast, requires little computation. The delta method estimate of prediction error can also be derived from Mackay's Bayesian framework [7]. The derivation assumes that the errors are normally distributed and that deviations in the network weights (arising from a Taylor series expansion) do not extend outside the region in which the network can be replaced by a suitably linearized network. These are also the assumptions of the delta method. The Bayesian approach, however, has the advantage of determining regularization parameters automatically as part of the training process. This means the networks do not have to be cross-validated. Therefore, all of the data can be used in the training set.

The Bayesian approach has recently been extended to the case of non-uniform target noise [1]. The prediction error component due to model variance from training has also been estimated with a Bayesian committee [14]. A combination of the latter two methods would represent a Bayesian version of the maximum likelihood approach adopted in this paper. Computationally, they would differ only in their method for restricting model complexity; the Bayesian approach uses automatic regularization and the maximum likelihood method uses cross-validation.

In summary, realistic estimates of prediction error will only be achieved by taking into account the *many* contributions to prediction error. This can be achieved by using a combination of estimation methods.

5 Acknowledgements

William Penny is a research fellow funded by the UK EPSRC (GR/K79062), whose support we gratefully acknowledge.

References

[1] C. Bishop and C. Qazaz. Regression with input-dependent noise: A bayesian treatment. In M.Jordan M.Mozer and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, 1997. MIT Press.

[2] C. M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, 1995.

[3] B. Efron and R.J. Tibshirani. *An introduction to the bootstrap.* Chapman and Hall, 1993.

[4] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[5] T. Heskes. Practical confidence and prediction intervals. In M.Jordan M.Mozer and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, 1997. MIT Press.

[6] D. Lowe. On the iterative inversion of rbf networks: a statistical interpretation. In *First IEE Conference on artificial neural networks*, 1991.

[7] D.J.C. Mackay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[8] D.J.C. Mackay. A practical bayesian framework for backprop networks. *Neural computation*, 4(3):448–472, 1992.

[9] D.A. Nix and A.S. Weigend. Learning local error bars for nonlinear regression. In *Advances in Neural Information Processing Systems 7 (NIPS*94), edited by G.Tesauro, D.S.Touretzky and T.K. Leen*, pages 489–496. MIT Press, 1995.

[10] W. H. Press, S.A. Teukolsky, W.T. Vetterling, and B.VP. Flannery. *Numerical Recipes in C, second edition.* Cambridge, 1992.

[11] S.J. Roberts and W.D. Penny. A maximum certainty approach to feedforward neural networks. In Press: Electronics Letters. Available from http://www.ee.ic.ac.uk/hp/staff/sroberts.html, October 1996.

[12] S.J. Roberts and W.D. Penny. Novelty, confidence and errors in connectionist systems. Submitted to IEEE Transactions on Neural Networks. Available from http://www.ee.ic.ac.uk/hp/staff/sroberts.html, October 1996.

[13] C. Satchwell. Neural networks for stochastic problems: more than one outcome for the input space. Presentation at the Neural Computing Applications Forum conference, Aston University, September 1994.

[14] H.H. Thodberg. A review of bayesian neural networks with an application to near infrared spectroscopy. Technical Report Revised version of Manuscript 1132E-part I, The Danish Meat Research Institute, Maglegaardsvej 2, DK-4000 Roskilde, 1995.

[15] R. Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8:152–163, 1996.

[16] A.R. Webb. Functional approximation by feedforward networks: A least squares approach to generalization. *IEEE transactions on Neural Networks*, 5(3):363–371, May 1994.

[17] S. Weisberg. *Applied linear regression.* John Wiley, 1980.

[18] R.R. Wilcox. Estimation in the simple linear regression model when there is heteroskedasticity of unknown form. *Communication in statistics - theory and methods*, 1996.