# Combining neural networks with input-dependent weighting factors


William D. Penny, Dirk Husmeier and Stephen J. Roberts

Technical Report,
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, d.husmeier@ic.ac.uk, s.j.roberts@ic.ac.uk*


May 5, 1998


**Abstract**

This paper introduces a method for calculating the 'Cross-Covariance Weight Matrix', that is, the covariance between different neural network solutions. It is based on a generalisation of the Delta method for calculating the network Hessian and generates what we call the 'Cross-Hessian'. Using the cross-covariance weight matrix we are able to calculate the pointwise covariance between network predictions, that is, the covariance at each point in the input space. This is a generalisation of the well known Delta method for calculating error bars.

We apply the method to the problem of finding optimal linear combinations of predictors. The linear combination that minimises the pointwise expected prediction error can be calculated using our estimate of the pointwise covariance matrix. This results in an input-dependent weighting scheme which we call covariance-based weighting. This scheme is compared to a variance-based weighting scheme and a uniform weighting scheme.

If the networks in the committee are unbiased then the covariance and variance based weighting schemes both result in substantially more accurate committee predictions than the uniform weighting scheme. The covariance scheme then shows an improvement over the variance scheme in areas of input space where the predictions are correlated and when the networks have different prediction variances. The covariance-based weighting scheme never gives worse results than variance-based weighting.


## 1 Introduction

It is well known that combining predictions from many neural networks to form a single 'committee' prediction often results in reduced prediction error (Perrone and Cooper, 1993). This can be understood by decomposing the average prediction error, averaged over many possible training data sets, into a bias component and a variance component (Geman, Bienenstock and Doursat, 1992). Neural networks are low-bias, high-variance predictors and combining predictions acts to reduce the variance component whilst not affecting the bias component (Breiman, 1994).

For regression problems, the simplest combining method is to take the mean prediction. More accurate committee predictions can however be made by linear combinations of network predictions. The problem then remains of how to assign 'weights' to the contributions from individual networks.

This can be tackled by again considering a decomposition of the average prediction error into bias and variance components, but this time taking the averages over the different networks in the ensemble (as opposed to averaging over different data sets, Krogh and Vedelsby, 1995). This 'ensemble-averaged' prediction error is then averaged again over the input space (where predictions are to be made) to produce a 'global ensemble-averaged' bias-variance decomposition. This leads to weighting schemes that are based on a 'global' covariance matrix of network predictions. This covariance matrix, however, cannot be calculated from the training data directly, as the data have already been used to optimise the individual networks and to use them again would give biased results. But it can be estimated using cross-validation or bootstrap methods (LeBlanc and Tibshirani, 1993) or by using unlabelled data sampled from non-parametric models of the input data density (Krogh and Vedelsby, 1995).

An alternative approach is to consider a 'pointwise ensemble-averaged' bias-variance decomposition where the averaging takes place over different networks in the ensemble but is *not* averaged over the input space. This leads to weighting schemes that are based on a pointwise covariance matrix of network predictions. Consideration of this input-dependent covariance matrix leads to an input-dependent weighting scheme.

Tresp and Taniguchi (1995) consider a diagonal approximation to this input-dependent covariance matrix which results in a weighting scheme where each predictor is weighted in proportion to the inverse variance of its prediction. The prediction variances are calculated using the delta method approximation (Tibshirani, 1996). The diagonal approximation assumes that the network predictions are uncorrelated; clearly an unrealistic assumption considering that the networks have been trained on the same data to perform the same task. In later work, Taniguchi and Tresp (1997) trained different networks on different bootstrap samples of the original data set. This helped to decorrelate the predictions of individual networks and so made the diagonal approximation more realistic.

In this paper, we show how a Taylor series expansion can be used to approximate the pointwise covariance matrix of network predictions. This is a generalisation of the well known delta method for calculating error bars (Tibshirani, 1996). We then go on to show how the covariance matrices are used to define a covariance-based committee weighting scheme and compare it to variance-based weighting and uniform weighting.

## 2 A Single Network

In this section, we show how the variance of a prediction from a single network can be approximated by a Taylor series expansion. The derivations are based on an analysis of the linear regression case. The material is well known (Seber, 1989; Tibshirani, 1996) but we include it here because the derivation of prediction *co*-variances follows similar reasoning. The derivations use results on estimating variances of linear functions of random variables and on results from linear regression. We include the results in the appendices.

We consider the nonlinear regression framework where $D = \boldsymbol{x}_i, y_i$ is a data set consisting of input-output pairs and $e_i$ is independent and identically distributed (IID) additive target noise with variance $\sigma^2$. The target noise captures the notion that we may not be able to model the data perfectly. Data are generated from

$$y_i = f(\boldsymbol{x}_i, \boldsymbol{w}) + e_i \tag{1}$$

The underlying deterministic function, $f$, is parameterised by the vector $\boldsymbol{w}$ which is of dimension $p$. We consider $f$ to be a neural network where $\boldsymbol{w}$ is the (unknown) optimal weight setting. The data are modelled with the regression equation

$$\hat{y}_i = f(\boldsymbol{x}_i, \hat{\boldsymbol{w}}) \tag{2}$$

where $\hat{y}_i$ is the output of a neural network with weight vector $\hat{\boldsymbol{w}}$. The goodness of fit of the model to the data can be assessed by the least squares cost function

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

where $N$ is the number of training points. 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 (Bishop, 1995). These algorithms will find only local minima solutions. This is not considered a problem, however, as the committee weighting scheme will be used to combine predictions from many different local-minima solutions in an optimal way.

## 2.1 Estimate of weight covariance

Different instantiations of target noise will result in a different estimated weight vector. The following methodology allows us to calculate the consequent variance around a single local minimum solution.

To find the variance of the estimated parameter, $\hat{\boldsymbol{w}}$, we assume that the nonlinear function can be locally approximated by a linear function. From a first order Taylor series we obtain

$$\hat{y}_i = f(\boldsymbol{x}_i, \hat{\boldsymbol{w}}) = f(\boldsymbol{x}_i, \boldsymbol{w}) + \boldsymbol{g}_i^T \Delta\hat{\boldsymbol{w}} \tag{4}$$

where $\Delta\hat{\boldsymbol{w}} = \hat{\boldsymbol{w}} - \boldsymbol{w}$ and $\boldsymbol{g}_i = \partial f(\boldsymbol{x}_i, \boldsymbol{w})/\partial \boldsymbol{w}$.

Substituting this into the cost function gives

$$\begin{aligned} E &= \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i, \boldsymbol{w}) - \boldsymbol{g}_i^T \Delta\hat{\boldsymbol{w}})^2 \\ &= \sum_{i=1}^{N} (e_i - \boldsymbol{g}_i^T \Delta\hat{\boldsymbol{w}})^2 \end{aligned} \tag{5}$$

This can be written in matrix form as

$$E = (\boldsymbol{e} - \boldsymbol{G}\Delta\hat{\boldsymbol{w}})^T (\boldsymbol{e} - \boldsymbol{G}\Delta\hat{\boldsymbol{w}}) \tag{6}$$

where $\boldsymbol{G}$ is an N-by-p matrix whose $i$th row consists of the vector $\boldsymbol{g}_i$ and $\boldsymbol{e}$ is the corresponding target noise vector.

This is the same form as equation 46 in Appendix B where $\boldsymbol{e}$ takes the place of $\boldsymbol{y}$, $\Delta\hat{\boldsymbol{w}}$ takes the place of $\hat{\boldsymbol{w}}$ and $\boldsymbol{G}$ takes the place of $\boldsymbol{X}$. Hence, by analogy with equation 49

$$\Delta\hat{\boldsymbol{w}} = (\boldsymbol{G}^T \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{e} \tag{7}$$

and by analogy with equation 53

$$\begin{aligned} Var(\Delta\hat{\boldsymbol{w}}) &= \sigma^2 (\boldsymbol{G}^T \boldsymbol{G})^{-1} \\ &= \boldsymbol{H}^{-1} \end{aligned} \tag{8}$$

where $\boldsymbol{H}$ is the outer-product approximation to the Hessian matrix (Bishop, 1995 p.152; Press 1992, p. 682)

$$\boldsymbol{H} = \frac{1}{\sigma^2}(\boldsymbol{G}^T\boldsymbol{G}) \tag{9}$$

Equation 8 summarises the variance of the neural network solution after training. Specifically, it shows how sensitive the local-minima solution is to small changes in the training set target values. When the network is presented with a new data point this variance will give rise to a variance in the prediction.

### 2.2 Estimate of prediction variance

The prediction of a neural network on a new data point, $\tilde{\mathbf{x}}$, is

$$\tilde{y} = f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}) + \tilde{e} \tag{10}$$

Because $\hat{\boldsymbol{w}}$ and $\tilde{e}$ are independent, the variance of the predicted value is

$$Var(\tilde{y}) = Var(f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}})) + Var(\tilde{e}) \tag{11}$$

This captures the variance due to model uncertainty which has its original source in the target noise on the training data set. The second term is due to the inherent target noise at the predicted point. This accounts for the fact that, even if we had a perfect model, we could only predict the conditional mean of the data ($f(\tilde{\boldsymbol{x}}, \boldsymbol{w})$ - see equation 1); there would still be some original uncertainty due to the target noise on the test point. In this respect, our treatment differs from that of Tibshirani (1996) who considers only the first term.

The first term is estimated by looking at the change in predicted value that results from a change in the model's estimated parameters. From a local linear approximation

$$\Delta f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}) = \tilde{\boldsymbol{g}}^T \Delta\hat{\boldsymbol{w}} \tag{12}$$

where $\tilde{\boldsymbol{g}} = \partial f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}})/\partial\hat{\boldsymbol{w}}$. Hence,

$$\begin{aligned} Var(f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}})) &= Var(\Delta f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}})) \\ &= \tilde{\boldsymbol{g}}^T Var(\Delta\hat{\boldsymbol{w}})\tilde{\boldsymbol{g}} \\ &= \tilde{\boldsymbol{g}}^T \boldsymbol{H}^{-1}\tilde{\boldsymbol{g}} \end{aligned} \tag{13}$$

Therefore,

$$Var(\tilde{y}) = \sigma^2 + \tilde{\boldsymbol{g}}^T\boldsymbol{H}^{-1}\tilde{\boldsymbol{g}} \tag{14}$$

## 3 Two networks

We now look at the case where two networks have been trained to solutions, $\hat{\boldsymbol{w}}_1$ and $\hat{\boldsymbol{w}}_2$. These may correspond to different local minima in the same network or to solutions in networks with different structures.

### 3.1 Estimates of weight covariance

From the analysis in the previous section we can arrive at the equivalent of equation 7 for each of the two solutions

$$\begin{aligned} \Delta\hat{\boldsymbol{w}}_1 &= (\boldsymbol{G}_1^T\boldsymbol{G}_1)^{-1}\boldsymbol{G}_1^T\boldsymbol{e} \\ \Delta\hat{\boldsymbol{w}}_2 &= (\boldsymbol{G}_2^T\boldsymbol{G}_2)^{-1}\boldsymbol{G}_2^T\boldsymbol{e} \end{aligned} \tag{15}$$

These approximations arise from a first order Taylor series expansion of each solution about the *same* true solution, $\boldsymbol{w}$. If the different solutions are two different local minima in the same network then it is unlikely that such an approximation can be valid in both cases. For two networks with different structures, however, the approximation is more plausible. The covariance of the two solutions is given by

$$\begin{aligned} \boldsymbol{C} &= Var(\Delta\hat{\boldsymbol{w}}_1, \Delta\hat{\boldsymbol{w}}_2) \\ &= Var((\boldsymbol{G}_1^T\boldsymbol{G}_1)^{-1}\boldsymbol{G}_1^T\boldsymbol{e}, (\boldsymbol{G}_2^T\boldsymbol{G}_2)^{-1}\boldsymbol{G}_2^T\boldsymbol{e}) \end{aligned} \tag{16}$$

From Appendix A we see that this can be written

$$\begin{aligned} \boldsymbol{C} &= (\boldsymbol{G}_1^T\boldsymbol{G}_1)^{-1}\boldsymbol{G}_1^T[Var(\boldsymbol{e})]\left((\boldsymbol{G}_2^T\boldsymbol{G}_2)^{-1}\boldsymbol{G}_2^T\right)^T \\ &= (\boldsymbol{G}_1^T\boldsymbol{G}_1)^{-1}\boldsymbol{G}_1^T\sigma^2\boldsymbol{I}\left((\boldsymbol{G}_2^T\boldsymbol{G}_2)^{-1}\boldsymbol{G}_2^T\right)^T \end{aligned} \tag{17}$$

Re-arranging gives,

$$\boldsymbol{C} = \sigma^2(\boldsymbol{G}_1^T\boldsymbol{G}_1)^{-1}\boldsymbol{G}_1^T\boldsymbol{G}_2(\boldsymbol{G}_2^T\boldsymbol{G}_2)^{-1} \tag{18}$$

We refer to $\boldsymbol{C}$ as the cross-covariance weight matrix. If $\hat{\boldsymbol{w}}_1 = \hat{\boldsymbol{w}}_2$ then $\boldsymbol{G}_1 = \boldsymbol{G}_2 = \boldsymbol{G}$ and we recover the estimate of weight variance for a single network given in equation 8.

The covariance between two different network solutions is captured in equation 18. It is this covariance that will give rise to correlations in predictions when the networks are presented with new data points.

In practice $\sigma^2$ may be unknown. In these cases we can substitute the estimate

$$\sigma^2 = \hat{\sigma}_1\hat{\sigma}_2 \tag{19}$$

where $\hat{\sigma}_i$ is the root mean squared training error of the $i$th network.

### 3.2 Estimate of prediction covariance

The predictions on a new data point $\tilde{\mathbf{x}}$ are

$$\begin{aligned} \tilde{y}_1 &= f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_1) + \tilde{e} \\ \tilde{y}_2 &= f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_2) + \tilde{e} \end{aligned} \tag{20}$$

Because $\tilde{e}$ is independent of $\hat{\boldsymbol{w}}_1$ and $\hat{\boldsymbol{w}}_2$, the covariance of the predictions is

$$Var(\tilde{y}_1, \tilde{y}_2) = Var(f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_1), f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_2)) + Var(\tilde{e}) \tag{21}$$

Now, from a local linear approximation

$$f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_1) = f(\tilde{\boldsymbol{x}}, \boldsymbol{w}) + \tilde{\boldsymbol{g}}_1^T \Delta\hat{\boldsymbol{w}}_1 \qquad (22)$$
$$f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_2) = f(\tilde{\boldsymbol{x}}, \boldsymbol{w}) + \tilde{\boldsymbol{g}}_2^T \Delta\hat{\boldsymbol{w}}_2$$

The covariance of these quantities is induced solely by the target noise in the training set. Hence,

$$\begin{aligned} Var(f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_1), f(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{w}}_2)) &= Var(\tilde{\boldsymbol{g}}_1^T \Delta\hat{\boldsymbol{w}}_1, \tilde{\boldsymbol{g}}_2^T \Delta\hat{\boldsymbol{w}}_2) \\ &= \tilde{\boldsymbol{g}}_1^T Var(\Delta\hat{\boldsymbol{w}}_1, \Delta\hat{\boldsymbol{w}}_2)\tilde{\boldsymbol{g}}_2 \\ &= \tilde{\boldsymbol{g}}_1^T \boldsymbol{C}\tilde{\boldsymbol{g}}_2 \end{aligned} \qquad (23)$$

Therefore,

$$Var(\tilde{y}_1, \tilde{y}_2) = \sigma^2 + \tilde{\boldsymbol{g}}_1^T \boldsymbol{C}\tilde{\boldsymbol{g}}_2 \qquad (24)$$

where $\boldsymbol{C}$ is the cross-covariance weight matrix defined in equation 18. When $\tilde{y}_1 = \tilde{y}_2$ the formula reduces to the prediction variance of a single network given in equation 14. The above formula captures the covariance in network predictions due to the correlation in network solutions induced by the random variation of the target noise.

Again, $\sigma^2$ may be unknown but can be estimated with equation 19.

# 4 Committees of networks

Given that $L$ networks are to be used in a committee, we can use the results from the last section to write down the estimated pointwise covariance matrix of predictions, $\hat{\boldsymbol{\Omega}}$. Its $j,k$th element is the covariance between the $j$th and $k$th networks in the committee and, for input $\boldsymbol{x}$, is given by

$$\hat{\boldsymbol{\Omega}}_{jk}(\boldsymbol{x}) = \sigma^2 + \boldsymbol{g}_j(\boldsymbol{x})^T \boldsymbol{C}(j,k,\boldsymbol{x})\boldsymbol{g}_k(\boldsymbol{x}) \qquad (25)$$

where $\boldsymbol{g}_j(\boldsymbol{x})$ is the vector of first derivatives of network $j$'s output with respect to its weights evaluated for input vector $\boldsymbol{x}$ and $\boldsymbol{C}(j,k,\boldsymbol{x})$ is the cross-covariance matrix for networks $j$ and $k$

$$\boldsymbol{C}(j,k,\boldsymbol{x}) = \sigma^2(\boldsymbol{G}_j^T\boldsymbol{G}_j)^{-1}\boldsymbol{G}_j^T\boldsymbol{G}_k(\boldsymbol{G}_k^T\boldsymbol{G}_k)^{-1} \qquad (26)$$

where the matrix $\boldsymbol{G}_j$ has $N$ rows, one for each data point, the $i$th row of which is $\boldsymbol{g}_j(\boldsymbol{x})$. Note: $\boldsymbol{C}(j,k,\boldsymbol{x})$ is not the $j$,$k$ th element of a matrix.

## 4.1 Optimal weighting of network predictions

In committees of networks, the predictions are weighted and summed to provide a single overall prediction

$$\hat{t}(\boldsymbol{x}) = \sum_{j=1}^{L} \alpha_j(\boldsymbol{x}) f_j(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x})^T \boldsymbol{f}(\boldsymbol{x}) \qquad (27)$$

where $f_j(\boldsymbol{x})$ is the output of network $j$ and $\alpha_j(\boldsymbol{x})$ is the weight for the $j$th network.

The pointwise expected squared error between the committee prediction and the true value is

$$E = \varepsilon_D[(\hat{t}(\boldsymbol{x}) - t(\boldsymbol{x}))^2] \tag{28}$$

where $\varepsilon_D$ denotes an expectation over the training data set, $D = \{\boldsymbol{x}, y\}$. In this paper the $\boldsymbol{x}$-values in the training set are considered fixed, but the $y$-values are not. The expectation is therefore in effect taken just over the $y$-values, that is, over different realisations of the target noise.

In what follows we drop the dependence on $\boldsymbol{x}$, the test point, to simplify the notation although this dependence is implicitly assumed. By substituting for $\hat{t}$ and introducing the term $\varepsilon_D[\boldsymbol{\alpha}^T\boldsymbol{f}]$ we can decompose the expected prediction error into a variance term and a bias term (Taniguchi and Tresp, 1997; Meir 1995)

$$E = \varepsilon_D[(\boldsymbol{\alpha}^T\boldsymbol{f} - \varepsilon_D[\boldsymbol{\alpha}^T\boldsymbol{f}])^2] + \varepsilon_D[(\varepsilon_D[\boldsymbol{\alpha}^T\boldsymbol{f}] - t)^2] \tag{29}$$

By introducing the notation $m_j = \varepsilon_D[f_j]$ and $\boldsymbol{m} = (m_1, m_2, ...m_L)'$ the above equation can be written as

$$E = \varepsilon_D[(\boldsymbol{\alpha}^T\boldsymbol{f} - \boldsymbol{\alpha}^T\boldsymbol{m})^2] + (\boldsymbol{\alpha}^T\boldsymbol{m} - t)^2 \tag{30}$$

If we now introduce the covariance matrix

$$\boldsymbol{\Omega}_{jk} = \varepsilon_D[(f_j - m_j)(f_k - m_k)] \tag{31}$$

then

$$E = \boldsymbol{\alpha}^T\boldsymbol{\Omega}\boldsymbol{\alpha} + (\boldsymbol{\alpha}^T\boldsymbol{m} - t)^2 \tag{32}$$

This can be written expilcity as the sum of a variance term and a bias term

$$E = V + B \tag{33}$$

where $V$ is the committee variance

$$V = \boldsymbol{\alpha}^T\boldsymbol{\Omega}\boldsymbol{\alpha} \tag{34}$$

and $B$ is the committee bias

$$B = (\boldsymbol{\alpha}^T\boldsymbol{m} - t)^2 \tag{35}$$

If the networks are unbiased and the weights are constrained to sum to one ($\sum_i \alpha_i = 1$), then the weight values that give rise to the minimum prediction error are (Tresp and Taniguchi, 1995; Meir 1995)

$$\boldsymbol{\alpha} = \frac{\boldsymbol{\Omega}^{-1}\boldsymbol{u}}{\boldsymbol{u}^T\boldsymbol{\Omega}^{-1}\boldsymbol{u}} \tag{36}$$

If the networks are biased (and, again, the weights are constrained to sum to one) then the optimal weight values are given by (Tresp and Taniguchi, 1995; Meir 1995)

$$\boldsymbol{\alpha} = \frac{(\boldsymbol{\Omega} + \boldsymbol{\Phi})^{-1}\boldsymbol{u}}{\boldsymbol{u}^T(\boldsymbol{\Omega} + \boldsymbol{\Phi})^{-1}\boldsymbol{u}} \tag{37}$$

where $\boldsymbol{u}$ is a vector of ones and $\boldsymbol{\Phi}$ is a bias matrix having elements

$$\boldsymbol{\Phi}_{jk} = \varepsilon_D[(f_j - t)(f_k - t)] \tag{38}$$

By plugging in our estimate of $\mathbf{\Omega}$ from equation 25 into the above equations we have an optimal method for weighting networks in a committee. We call this 'covariance-based' weighting. In previous work (Tanigughi and Tresp, 1997) $\mathbf{\Omega}$ has been approximated by a diagonal matrix resulting in a 'variance-based' weighting scheme. We note that both variance and covariance-based weighting can only act to reduce the component of prediction error due to the variance term. Reduction of the bias term will require a separate estimate of bias.

A problem with the above methods is that the weights can take on large negative and positive values giving rise to committee predictions which are beyond the range of predictions from individual networks (Breiman, 1993; Bishop, 1995). To overcome this we can impose the additional non-negativity constraint, $\alpha_i > 0$. This can be implemented by expressing the weighting factors as softmax functions of hidden variables which are chosen so as to minimise the expected pointwise prediction error (equation 33). This is, however, a nonlinear optimization problem requiring iterative solution. An alternative approach is to use a non-iterative regularization technique (Golub and Van Loan, 1996).

Following Tresp and Taniguchi (1995) we illustrate the various weighting schemes in committees composed of just two networks.

# 5 Results

## 5.1 Effect of correlation between predictors

The potential benefit of the covariance-based weighting scheme can be seen by looking at the effect of correlations between predictors in the ensemble. This effect is shown in Figure 1 for a committee composed of just two networks.

The stronger the positive correlation between networks, the stronger the weight on just one of the networks; the one with the smallest variance. This is shown in Figures 1(a) and (b) where the low-variance network (network 1) is assigned a weight of 1 if the correlation is high. This makes sense because knowing the prediction from the second (high variance) network gives you an increasingly smaller amount of extra information.

Figure 1(c) shows the committee prediction variance for each of the three weighting schemes; covariance-based, variance-based and uniform. The covariance-based weighting scheme results in a committee with the lowest prediction variance. The effect is greater as the correlation between predictors increases. Figure 1(d) shows the same curves but for networks where the difference in prediction variance is less. The difference between weighting schemes is less apparent. If both networks have the same variance then all weighting schemes are the same.

We can summarise this section by saying that the covariance-based weighting scheme will be beneficial if the variance of predictors is uneven and if the correlation is high.

## 5.2 Effect of bias

The variance-based and covariance-based weighting schemes will only act to reduce the component of prediction error due to committee variance (equation 34). They will not affect the bias term (equation 35). If the individual predictors are biased both variance and covariance-based weighting will be less effective. Indeed, uniform weighting mey be superior to both.

*[See Graph ...... to do]*

This is especially important in our formulation of the expected prediction error as we are taking the expectation only over different training *targets* - the training inputs are

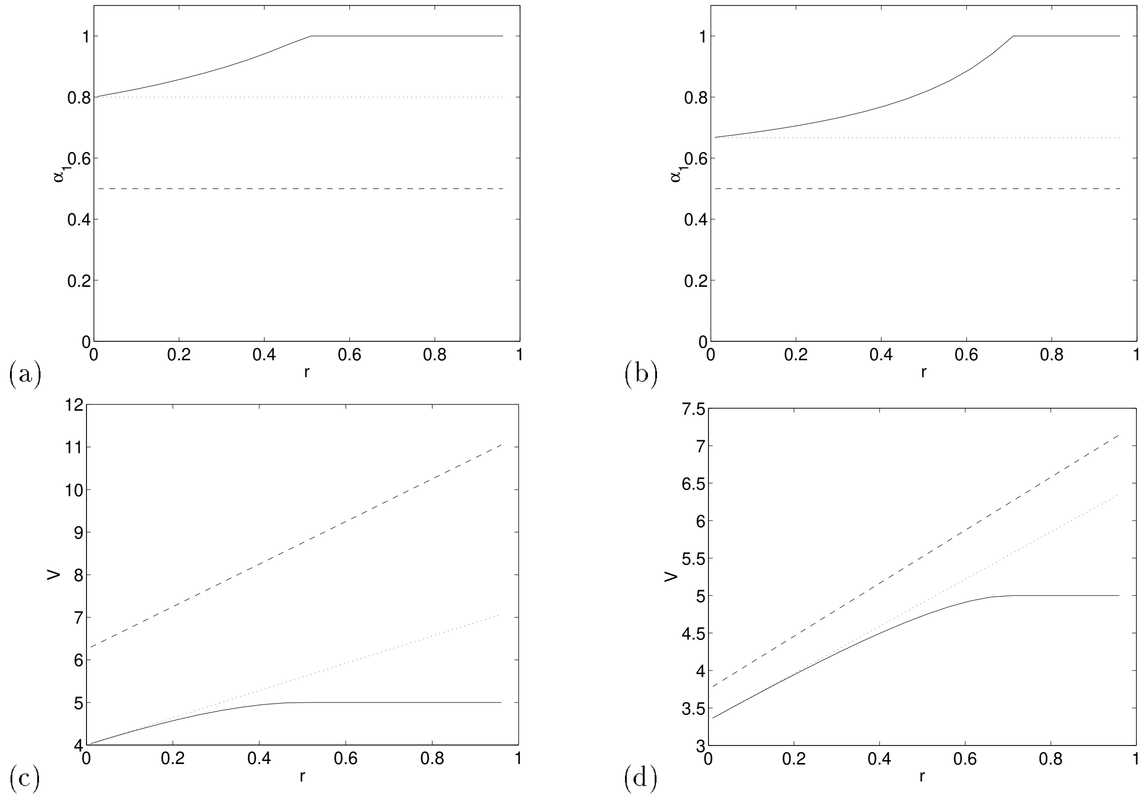


Figure 1: Effect of correlation between predictors; (a,b) Weighting of first predictor versus correlation, (c,d) committee variance versus correlation. Graphs (a) and (c) are for a committee where the first predictor has a variance of 5 and the second predictor 20. Graphs (b) and (d) are for variances of 5 and 10. The solid lines are for covariance-based weight values (from equation 36), dotted lines for variance-based weight values (from equation 36 using a diagonal $\mathbf{\Omega}$), and dashed lines for uniform weights. If the variances of the predictors are equal all curves are the same. The stronger the correlation between predictors the greater the benefit of using the covariance-based weighting scheme.

considered fixed (see section 4.1). Therefore $\varepsilon_D(f_j) = f_j$. It is therefore much more likely that our networks will be biased. (Recall, that in the standard bias-variance decomposition (Geman, Bienenstock and Doursat, 1992) the expectation is taken over all data sets of a fixed size, ie. both training targets *and* inputs).

### 5.3 Sinewave data

Figure 2a shows the output of a Multi-Layer Perceptron (MLP) with 2 hidden units that has been trained on samples of a noisy sine wave function. The error bars at plus and minus 2 standard deviations are also shown. These have been calculated according to equation 14. Figure 2b is the same plot but for an MLP with 4 hidden units. In regions of low data-density the 2-hidden unit network has smaller error bars. In regions of high data-density the 4-hidden unit network has smaller error bars and is seen to fit the data better.

Equation **??** tells us how to estimate the covariance between model predictions. By normalising this covariance with respect to the estimate of variance for each model it is possible to estimate the pointwise correlation, that is, the correlation between model predictions at each point in input space. This is shown in Figure 3a. In regions of high data-density the model predictions are positively correlated. In regions of low data-density they are not correlated at all. Thus, in regions of low data-density the pointwise covariance matrix becomes diagonal and the covariance-based weighting scheme reduces to the variance-based weighting scheme.

Figure 3b shows the weight coefficients derived from the variance-based and the covariance-based weighting schemes. In regions of low data-density both schemes do indeed give rise to similar weighting factors. In regions of high data-density, the covariance-based weighting scheme relies more heavily on the predictions of the 4-hidden unit MLP; the more accurate network.

In the previous section is was noted that the covariance-based weighting scheme will only be beneficial if the variance of predictors is uneven and if the correlation is high. Now, experiments with the sinewave data tell us that in regions of high training data-density the predictors will be correlated but in regions of low data density they will not. We can therefore only expect to see an improvement in regions of high data density where the predictors have unequal variance. This fact was borne out by experiments on the sinewave data - the two and four hidden unit networks have similar prediction variances when they are correlated and therefore covariance-based weighted committees give only a marginal reduction in prediction variance over variance-based committees. If the committee is, however, composed of one network that overfits the data and one that underfits it then a significant reduction in prediction variance can be achieved. Figure 5 shows the relative prediction variance in such a case. The committee consists of two four hidden unit networks, one trained to a normalised mean squared error (NMSE) of 0.05 and the other to 0.2. This is calculated from the first term in equation 33. The covariance scheme results in networks with the lowest prediction variance, especially in areas of high data density. The covariance scheme also resulted in committees with the lowest actual test error; the normalised mean squared test errors were 0.344 for uniform weighting, 0.135 for variance-based weighting and 0.116 for covariance-based weighting.

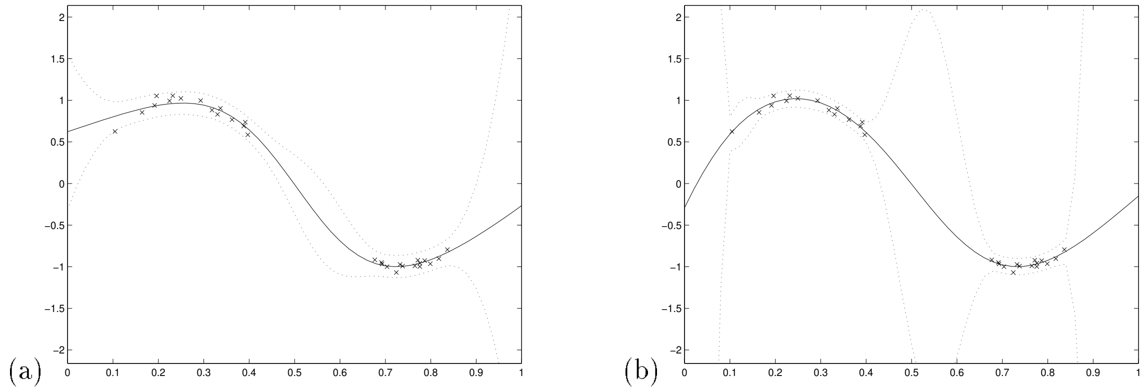

Figure 2: (a) MLP with two hidden units, (b) MLP with four hidden units. The crosses are training data points, the solid line is the prediction and the dotted lines are error bars at plus and minus 2 standard deviations.

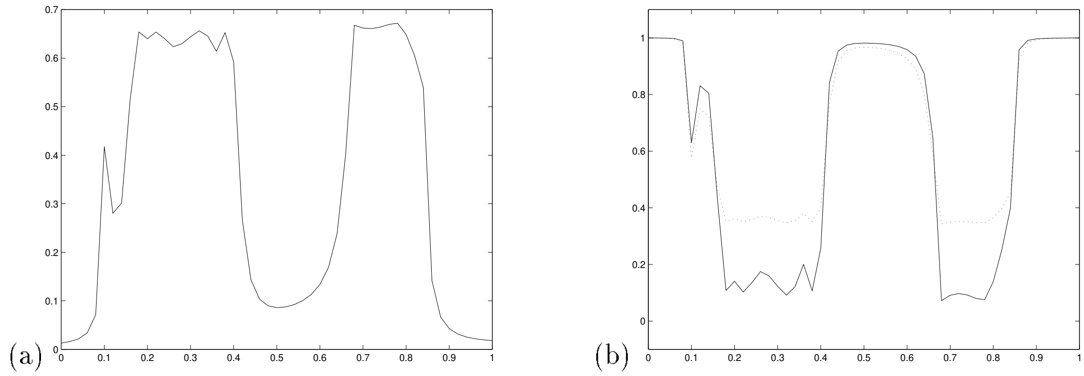

Figure 3: (a) Pointwise correlation between the predictions of the 2-unit MLP and the 4-unit MLP as approximated by a normalised version of equation 25. The predictions are positively correlated in regions of high data-density, (b) Committee weighting factors for the 2-unit MLP from equation 36 and a non-negativity constraint; covariance-based (solid line) and variance-based (dotted line). For both weighting schemes, the 2-unit MLP is relied upon more in regions of low data-density.

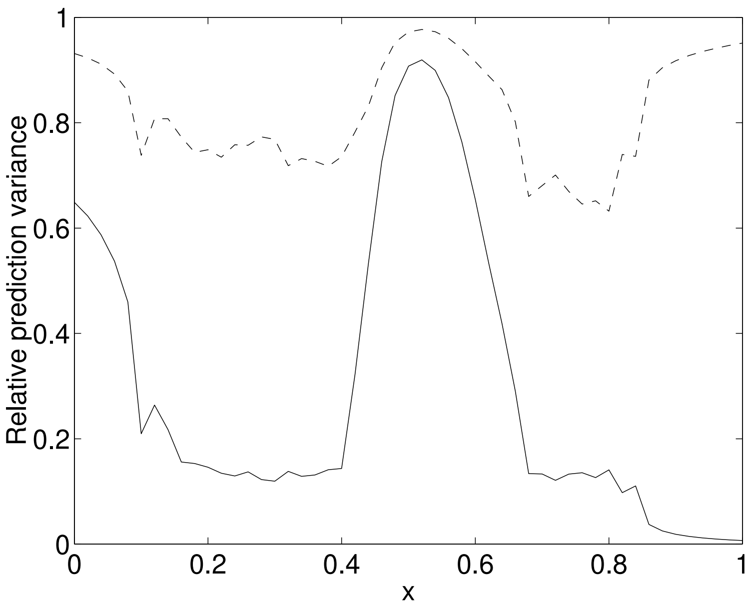


Figure 4: Prediction variance of covariance-based weighted committee relative to unweighted committee (solid line) and variance-based weighted committee (dotted line). Committees using the covariance-based weights have lower prediction variance in areas of higher data density.

Table 1: *Committee prediction variance on Robot Arm Data.*

| Weighting scheme | Interp | Extrap | Overall |
|---|---|---|---|
| Uniform | 0.118 | 0.341 | 0.229 |
| Variance | 0.062 | 0.099 | 0.080 |
| Covariance | 0.043 | 0.081 | 0.062 |

## 5.4 Robot arm data

The methods were also tested on the Robot arm data set (Mackay, 1992) which describes the mapping for a two-joint robot arm. There are two input variables and one target (originally there were two targets but we only use the first one). The data was then grouped into two regions. Half of the points where $1.0 \leq x2 \leq 2.8$ were used for training and half for testing. All points outside this region were also placed in the test set. Thus, at half of the test points the networks will be interpolating and at the other half they will be extrapolating.

Two four hidden unit networks were then trained to 0.05 and 0.2 NMSE training error, representing one good fit to the data and one bad fit. The networks were then tested. In the interpolation regions the average correlation of predictions, as calculated from the estimates of the covariance matrix (equation **??**), was 0.963. In the extrapolation region this reduced to 0.669. Thus, the correlation measure is quantitatively correct.

Table 1 shows that the covariance-based committee weighting scheme results in a committee with a lower prediction variance, especially in interpolation regions. The actual prediction errors were 0.318 for uniform, 0.236 for variance and 0.222 for covariance. This implies that network bias is not a significant factor for this network/data combination.

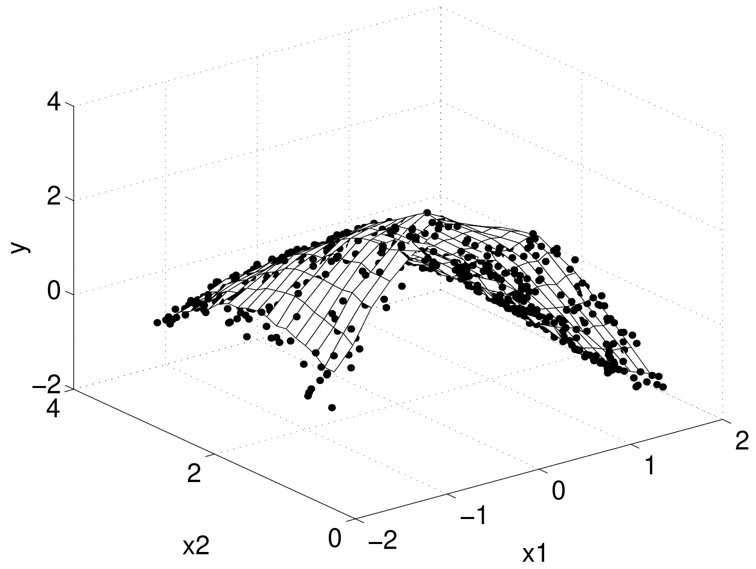


Figure 5: Robot arm data. Half of the points where $1.0 \leq x2 \leq 2.8$ were used for training and half for testing. All points outside this region were also placed in the test set.

### 5.5 Boston housing data

We also tested the methods on the Boston housing data set (13 inputs, one output) . The data was first divided into 60 groups using k-means clustering. Twenty of the clusters made up half of the test set. Patterns in the remaining forty clusters were then randomly divided into a training set and further test set examples. Thus, half of the test patterns are similar to the training patterns - in these regions we might expect that the networks are interpolating. For the remaining test patterns the networks are more likely to be extrapolating. Two eight hidden unit networks were then trained to NMSE errors of 0.05 and 0.2. In regions where the networks are thought to be interpolating the average correlation of predictions, as calculated from estimates of the covariance matrix (equation **??**), was 0.358. In the extrapolation region this reduced to 0.200. Thus, the correlation measure is quantitatively correct.

Table 2 shows that the covariance-based committee weighting scheme results in a committee with a lower prediction variance. The actual prediction errors, however, were 0.409 for uniform, 0.420 for variance and 0.426 for covariance. This implies that network bias is a significant factor with this network/data combination. If the network biases are measured and used in equation 37 to calculate a new set of committee weights then the prediction errors become 0.409 for uniform (unchanged), 0.341 for variance and 0.340 for covariance. This is cheating however, as to calculate the bias requires knowledge of the true test value. An alternative method is to estimate the network bias as the difference between each network output and the committee output. Using this method the prediction errors are 0.409 for uniform, 0.410 for variance and 0.410 for covariance. Although better results could be obtained using a better estimate of network bias (eg. from a larger committee) these results show that the bias term can be more important than the variance term. If this is the case then the use of variance and covariance-based input-dependent weighting schemes will have no impact on the actual test error of committees; estimates of bias are also important.

Table 2: *Committee prediction variance on Boston Housing Data.*

| Weighting scheme | Interp | Extrap | Overall |
|---|---|---|---|
| Uniform | 0.914 | 1.753 | 1.336 |
| Variance | 0.274 | 0.486 | 0.381 |
| Covariance | 0.267 | 0.483 | 0.375 |

## 6 Summary

This paper has introduced a method for calculating the 'Cross-Covariance Weight Matrix', that is, the covariance between different network solutions. Using this matrix we are able to calculate the pointwise covariance between network predictions, that is, the covariance between predictions at each point in the input space. This is a generalisation of the well known delta method for calculating error bars.

We have applied these methods to the problem of finding optimal linear combinations of neural network predictors. This results in a covariance-based weighted committee, where the weights are input-dependent. If the individual networks are unbiased (in the sense that the expected network output averaged over different training sets is equal to the true value) then this the resulting committee has minimum expected prediction error. If, however, the networks are biased then this weighting is optimal in the sense that the committee has minimum prediction variance.

Our experiments have shown that the covariance-based weighting scheme can be superior to variance-based weighting and uniform (non input-dependent) weighting. If the networks in the committee are unbiased then the covariance and variance based weighting schemes both result in substantially more accurate committee predictions. The covariance scheme then shows an improvement over the variance scheme in areas of input space where the networks are correlated (ie. in regions of high training data density) and when the networks have different prediction variances. This suggests using a range of networks with different complexities, or networks trained to fit the data with a range of accuracies. The covariance-based scheme does, however, require more computation and storage than the variance-base scheme as the cross-Hessians for each network pair in the committee must be calculated and stored.

## 7 Acknowledgements

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

## References

Breiman, L. (1994) Bagging predictors. Technical report 421, Department of Statistics, University of California at Berkeley.

Breiman, L. (1993) Stacked regression. Technical report, Department of Statistics, University of California at Berkeley.

Bevington, P.R. and Robinson,K. (1992) Data reduction and error analysis for the physical sciences. Mc-Graw Hill, pp 41-43.

Geman,S., Bienenstock,E. and Doursat, R. (1992) Neural Networks and the Bias/Variance Dilemma. Neural Computation. **4**:1-58.

Krogh, A.S. and Vedelsby, J. (1995) Neural network ensembles, cross validation and active learning. In G. Tesauro, D.S. Touretzky, and T.K. Leen (Eds.), Advances in Neural Information Processing Systems 7. Cambridge, MA: MIT Press.

LeBlanc, M. and Tibshirani, R. (1993) Combining estimates in regression and classification. Technical Report, Department of Preventive Medicine and Biostatistics, University of Toronto.

MacKay, D.J.C. (1992) A practical Bayesian framework for backprop networks. *Neural Computation* **4**(3):448-472.

Perrone, M.P. and Cooper, L. (1993). When networks disagree: ensemble methods for hybrid neural networks. In R.J. Mammone (Ed.), Artificial Neural Networks for Speech and Vision, pp. 126-142. London: Chapman and Hall.

Seber, G.A.F. and Wild, C.J. (1989). Nonlinear regression. New York: John Wiley.

Taniguchi, M. and Tresp, V. (1997). Averaging regularized estimators. Neural Computation, **9**:1163-1178.

Tibshirani, R. (1996) A comparison of some error estimates for neural network models. Neural Computation, **8**(1):152-163.

Tresp, V. and Taniguchi, M. (1995). Combining estimators using non-constant weighting functions. In G.Tesauro, D.S. Touretzky and T.K. Leen (Eds.), Advances in Neural Information Processing Systems 7. Cambridge, MA: MIT Press.

Weisberg, S. (1980). Applied linear regression. John Wiley.

## APPENDIX A. Statistics of linear functions of random variables

These results are taken from Weisberg (1980) and are approximations from a first order Taylor series. For a more detailed discussion see Bevington and Robinson (1992).

Given random variables, $u_i$ and constant values $a_i$, the variance of a sum of products is given by

$$Var\left(a_0 + \sum a_i u_i\right) = \sum_i a_i^2 Var(u_i) + 2\sum_i \sum_j a_i a_j Var(u_i, u_j) \tag{39}$$

where $Var(x, y)$ denotes the covariance of the random variables $x$ and $y$.
The covariance of a linear function of a pair of random variables is

$$Var\left(a_0 + a_1 u_1, a_3 + a_2 u_2\right) = a_1 a_2 Var(u_1, u_2) \tag{40}$$

For a vector of random variables, $\boldsymbol{z}$, and a matrix of constants, $\boldsymbol{C}$, and a vector of constants, $\boldsymbol{d}$, we have

$$Var(\boldsymbol{C}\boldsymbol{z}+\boldsymbol{d}) = \boldsymbol{C}[Var(\boldsymbol{z})]\boldsymbol{C}^T \tag{41}$$

where, here, Var() denotes a covariance matrix.

The covariance between a pair of random vectors is given by

$$Var(\boldsymbol{C}_1\boldsymbol{z}, \boldsymbol{C}_2\boldsymbol{z}) = \boldsymbol{C}_1[Var(\boldsymbol{z})]\boldsymbol{C}_2^T \tag{42}$$

# APPENDIX B. Multivariate linear regression

## 1. The regression problem

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

$$y_i = \boldsymbol{x}_i^T\boldsymbol{w} + e_i \tag{43}$$

where $\boldsymbol{w}$ is the $p$ dimensional true weight vector. This may be modelled by the regression equation

$$\hat{y}_i = \boldsymbol{x}_i^T\hat{\boldsymbol{w}} \tag{44}$$

where $\hat{\boldsymbol{w}}$ is the estimated weight vector. The goodness of fit can be assessed by the least squares cost function

$$\begin{aligned} E &= \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (45)\\ &= \sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T\hat{\boldsymbol{w}})^2 \end{aligned}$$

We can re-write this in matrix notation as

$$E = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}) \tag{46}$$

where $\boldsymbol{X}$ is an N-by-p matrix whose rows are made up of different training vectors, and $\boldsymbol{y}$ is the corresponding vector of targets.

## 1. Estimating the weights

The weight vector that minimises the above cost function can be calculated by setting the first derivative of the cost function to zero and solving the resulting equation.

By expanding the brackets and collecting terms (using the matrix identity $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$) we get

$$E = \boldsymbol{y}^T\boldsymbol{y} - 2\hat{\boldsymbol{w}}\boldsymbol{X}^T\boldsymbol{y} - \hat{\boldsymbol{w}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{w}} \tag{47}$$

Equating the first derivative to zero gives

$$(\boldsymbol{X}^T\boldsymbol{X})\hat{\boldsymbol{w}} = \boldsymbol{X}^T\boldsymbol{y} \tag{48}$$

Hence,

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{49}$$

## 2. Estimating the weight covariance matrix

Different instantiations of target noise will generate different estimated weight vectors according to the last equation. The corresponding weight covariance matrix is given by

$$Var(\hat{\boldsymbol{w}}) = Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) \tag{50}$$

Substituting for $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{e}$ gives

$$Var(\hat{\boldsymbol{w}}) = Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{e}) \tag{51}$$

This is in the form of equation 41 in Appendix A with $\boldsymbol{d}$ being given by the first term which is constant, $\boldsymbol{C}$ being given by $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ and $\boldsymbol{z}$ being given by $\boldsymbol{e}$. Hence,

$$\begin{aligned} Var(\hat{\boldsymbol{w}}) &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T[Var(\boldsymbol{e})][(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T]^T \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\sigma^2\boldsymbol{I})[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T]^T \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\sigma^2\boldsymbol{I})\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \end{aligned} \tag{52}$$

Re-arranging further gives

$$Var(\hat{\boldsymbol{w}}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{53}$$