Exact Calculation of the Hessian Matrix for the Multi-layer Perceptron

Christopher M. Bishop
Current address:
Microsoft Research,
7 J J Thomson Avenue,
Cambridge, CB3 0FB, U.K.
`cmishop@microsoft.com`
`http://research.microsoft.com/∼cmbishop/`

Published in *Neural Computation* **4** No. 4 (1992) 494 – 501

Abstract
The elements of the Hessian matrix consist of the second derivatives of the error measure with respect to the weights and thresholds in the network. They are needed in Bayesian estimation of network regularization parameters, for estimation of error bars on the network outputs, for network pruning algorithms, and for fast re-training of the network following a small change in the training data. In this paper we present an extended back-propagation algorithm which allows all elements of the Hessian matrix to be evaluated exactly for a feed-forward network of arbitrary topology. Software implementation of the algorithm is straightforward.

1 Introduction

Standard training algorithms for the multi-layer perceptron use back-propagation to evaluate the first derivatives of the error function with respect to the weights and thresholds in the network. There are, however, several situations in which it is also of interest to evaluate the second derivatives of the error measure. These derivatives form the elements of the Hessian matrix.

Second derivative information has been used to provide a fast procedure for re-training a network following a small change in the training data (Bishop, 1991). In this application it is important that all elements of the Hessian matrix be evaluated accurately. Approximations to the Hessian have been used to identify the least significant weights as a basis for network pruning techniques (Le Cun et al., 1990), as well as for improving the speed of training algorithms (Becker and Le Cun, 1988; Ricotta et al., 1988). The Hessian has also been used by MacKay (1991) for Bayesian estimation of regularization parameters, as well as for calculation of error bars on the network outputs and for assigning probabilities to different network solutions. MacKay found that the approximation scheme of Le Cun et al. (1990) was not sufficiently accurate and therefore included off-diagonal terms in the approximation scheme.

In this paper we show that the elements of the Hessian matrix can be evaluated exactly using multiple forward propagation through the network, followed by multiple backward propagation. The resulting algorithm is closely related to a technique for training networks whose error functions contain derivative terms (Bishop, 1990). In Section 2 we derive the algorithm for a network of arbitrary feed-forward topology, in a form which can readily be implemented in software. The algorithm simplifies somewhat for a network having a single hidden layer, and this case is described in Section 3. Finally a brief summary is given in Section 4.

2 Evaluation of the Hessian Matrix

Consider a feed-forward network in which the activation z_i of the i^{th} unit is a non-linear function of the input to the unit:

$$z_i = f(a_i) \tag{1}$$

in which the input a_i is given by a weighted linear sum of the outputs of other units

$$a_i = \sum_j w_{ij} z_j + \theta_i \tag{2}$$

where w_{ij} is the synaptic weight from unit j to unit i, and θ_i is a bias associated with unit i. Since the bias terms can be considered as weights from an extra unit whose activation is fixed at $z_k = +1$, we can simplify the notation by absorbing the bias terms into the weight matrix, without loss of generality.

We wish to find the first and second derivatives of an error function E, which we take to consist of a sum of terms, one for each pattern in the training set,

$$E = \sum_p E_p \tag{3}$$

where p labels the pattern. The derivatives of E are obtained by summing the derivatives obtained for each pattern separately.

To evaluate the elements of the Hessian matrix, we note that the units in a feed-forward network can always be arranged in 'layers', or levels, for which there are no intra-layer connections and no feed-back connections. Consider the case in which unit i is in the same layer as unit n, or in a lower layer (i.e. one nearer the input). The remaining terms, in which unit i is above unit n, can be obtained from the symmetry of the Hessian matrix without further calculation. We first write

$$\frac{\partial^2 E_p}{\partial w_{ij} \partial w_{nl}} = \frac{\partial a_i}{\partial w_{ij}} \frac{\partial}{\partial a_i} \left(\frac{\partial E_p}{\partial w_{nl}} \right) = z_j \frac{\partial}{\partial a_i} \left(\frac{\partial E_p}{\partial w_{nl}} \right) \tag{4}$$

where we have made use of equation 2. The first equality in equation 4 follows from the fact that, as we shall see later, the first derivative depends on w_{ij} only through a_i. We now introduce a set of quantities σ_n defined by

$$\sigma_n \equiv \frac{\partial E_p}{\partial a_n} \tag{5}$$

Note that these are the quantities which are used in standard back-propagation. The appropriate expressions for evaluating them will be obtained shortly. Equation 4 then becomes

$$\frac{\partial^2 E_p}{\partial w_{ij} \partial w_{nl}} = z_j \frac{\partial}{\partial a_i} (\sigma_n z_l) \tag{6}$$

where again we have used equation 2. We next define the quantities

$$g_{li} \equiv \frac{\partial a_l}{\partial a_i} \tag{7}$$

$$b_{ni} \equiv \frac{\partial \sigma_n}{\partial a_i} \tag{8}$$

The second derivatives can now be written in the form

$$\frac{\partial^2 E_p}{\partial w_{ij} \partial w_{nl}} = z_j \sigma_n f'(a_l) g_{li} + z_j z_l b_{ni} \tag{9}$$

where $f'(a)$ denotes df/da. The $\{g_{li}\}$ can be evaluated from a forward propagation equation obtained as follows. Using the chain rule for partial derivatives we have

$$g_{li} = \sum_r \frac{\partial a_r}{\partial a_i} \frac{\partial a_l}{\partial a_r} \tag{10}$$

where the sum runs over all units r which send connections to unit l. (In fact, contributions only arise from units which lie on paths connecting unit i to unit l). Using equations 1 and 2 we then obtain the forward propagation equation

$$g_{li} = \sum_r f'(a_r) w_{lr} g_{ri} \tag{11}$$

The initial conditions for evaluating the $\{g_{li}\}$ follow from the definition of equation 7, and can be stated as follows. For each unit i in the network, (except for input units, for which the corresponding $\{g_{li}\}$ are not required), set $g_{ii} = 1$ and set $g_{li} = 0$ for all units $l \neq i$ which are in the same layer as unit i or which are in a layer below the layer containing unit i. The remaining elements of g_{li} can then be found by forward propagation using equation 11. The number of forward passes needed to evaluate all elements of $\{g_{li}\}$ will depend on the network topology, but will typically scale like the number of (hidden plus output) units in the network.

The quantities $\{\sigma_n\}$ can be obtained from the following back-propagation procedure. Using the definition in equation 5, together with the chain rule, we can write

$$\sigma_n = \sum_r \frac{\partial E_p}{\partial a_r} \frac{\partial a_r}{\partial a_n} \tag{12}$$

where the sum runs over all units r to which unit n sends connections. Using equations 1 and 2 then gives

$$\sigma_n = f'(a_n) \sum_r w_{rn} \sigma_r \tag{13}$$

This is just the familiar back-propagation equation. Note that the first derivatives of the error function are given by the standard expression

$$\frac{\partial E_p}{\partial w_{ij}} = \sigma_i z_j \tag{14}$$

which follows from equations 2 and 5. The initial conditions for evaluation of the $\{\sigma_n\}$ are given, from equations 2 and 5, by

$$\sigma_m = f'(a_m) \frac{\partial E_p}{\partial z_m} \tag{15}$$

where m labels an output unit.

Similarly, we can derive a generalised back-propagation equation which allows the $\{b_{ni}\}$ to be evaluated. Substituting the back-propagation formula 13 for the $\{\sigma_n\}$ into the definition of b_{ni}, equation 8, we obtain

$$b_{ni} = \frac{\partial}{\partial a_i} \left\{ f'(a_n) \sum_r w_{rn} \sigma_r \right\} \tag{16}$$

which, using equations 7 and 8, gives

$$b_{ni} = f''(a_n) g_{ni} \sum_r w_{rn} \sigma_r + f'(a_n) \sum_r w_{rn} b_{ri} \tag{17}$$

where again the sum runs over all units r to which unit n sends connections. Note that, in a software implementation, the first summation in equation 17 will already have been computed in evaluating the $\{\sigma_n\}$ in equation 13.

The derivative $\partial/\partial a_i$ which appears in equation 16 arose from the derivative $\partial/\partial w_{ij}$ in equation 4. This transformation, from w_{ij} to a_i, is valid provided w_{ij} does not appear explicitly within the brackets on the right hand side of equation 16. This is always the case, because we considered only units i in the same layer as unit n, or in a lower layer. Thus the weights w_{rn} are always above the weight w_{ij} and so the term $\partial w_{rn}/\partial w_{ij}$ is always zero.

The initial conditions for the back-propagation in equation 17 follow from equations 7, 8 and 15,

$$b_{mi} = g_{mi} H_m \tag{18}$$

where we have defined

$$H_m \equiv \frac{\partial^2 E_p}{\partial a_m^2} = f''(a_m)\frac{\partial E_p}{\partial z_m} + \left(f'(a_m)\right)^2 \frac{\partial^2 E_p}{\partial z_m^2} \tag{19}$$

Thus, for each unit i (except for the input units), the b_{mi} corresponding to each output unit m are evaluated using equations 18 and 19, and then the b_{ni} for each remaining unit n (except for the input units, and units n which are in a lower layer than unit i) are found by back-propagation using equation 17.

Before using the above equations in a software implementation, the appropriate expressions for the derivatives of the activation function should be substituted. For instance, if the activation function is given by the sigmoid:

$$f(a) \equiv \frac{1}{1 + \exp(-a)} \tag{20}$$

then the first and second derivatives are given by

$$f'(a) = f(1-f) \qquad\qquad f''(a) = f(1-f)(1-2f) \tag{21}$$

For the case of linear output units, we have $f(a) = a$, $f'(a) = 1$, and $f''(a) = 0$, with corresponding simplification of the relevant equations.

Similarly, appropriate expressions for the derivatives of the error function with respect to the output unit activations should be substituted into equations 15 and 19. Thus, for the sum of squares error defined by

$$E_p = \frac{1}{2}\sum_m (z_m - t_m)^2 \tag{22}$$

where t_m is the target value for output unit m, the required derivatives of the error become

$$\frac{\partial E_p}{\partial z_m} = (z_m - t_m) \qquad\qquad \frac{\partial^2 E_p}{\partial z_m^2} = 1 \tag{23}$$

Another commonly used error measure is the relative entropy (Solla et al., 1988) defined by

$$\hat{E}_p = \sum_m \left\{ t_m \ln z_m + (1 - t_m)\ln(1 - z_m) \right\} \tag{24}$$

The derivatives of $\hat{E}_p$ take a particularly elegant form when the activation function of the output units is given by the sigmoid of equation 20. In this case, we have, from equations 15, 19 and 21,

$$\sigma_m = t_m - z_m \qquad\qquad H_m = -z_m(1 - z_m) \tag{25}$$

To summarise, the evaluation of the terms in the Hessian matrix can be broken down into three stages. For each pattern p, the $\{z_n\}$ are calculated by forward propagation using equations 1 and 2, and the $\{g_{li}\}$ are obtained by forward propagation using equation 11. Next, the $\{\sigma_n\}$ are found by back-propagation using equations 13 and 15, and the $\{b_{ni}\}$ are found by back-propagation using equations 17, 18, and 19. Finally, the second derivative terms are evaluated using equation 9. (If one or both of the weights is a bias, then the correct expression is obtained simply by setting the corresponding activation(s) to +1). These steps are repeated for each pattern in the training set, and the results summed to give the elements of the Hessian matrix.

The total number of distinct forward and backward propagations required (per training pattern) is equal to twice the number of (hidden plus output) units in the network, with the number of operations for each propagation scaling like $\mathcal{N}$, where $\mathcal{N}$ is the total number of weights in the network. Evaluation of the elements of the Hessian using equation 9 requires of order $\mathcal{N}^2$ operations. Since the number of weights is typically much larger than the number of units, the overall computation will be dominated by the evaluations in equation 9.

3 Single Hidden Layer

Many applications of feed-forward networks make use of an architecture having a single layer of hidden units, with full interconnections between adjacent layers, and no direct connections from input units to output units. Since there is some simplification to the algorithm for such a network, we present here the explicit expressions for the second derivatives. These follow directly from the equations given in Section 2.

We shall use indices k and k' for units in the input layer, indices l and l' for units in the hidden layer, and indices m and m' for units in the output layer. The Hessian matrix for this network can be considered in three separate blocks as follows.

(A) Both weights in the second layer:

$$\frac{\partial^2 E_p}{\partial w_{ml}\partial w_{m'l'}} = z_l z_{l'} \delta_{mm'} H_{m'} \tag{26}$$

(B) Both weights in the first layer:

$$\frac{\partial^2 E_p}{\partial w_{lk}\partial w_{l'k'}} = z_k z_{k'} \left\{ f''(a_{l'})\delta_{ll'} \sum_m w_{ml'}\sigma_m + f'(a_{l'})f'(a_l) \sum_m w_{ml'} w_{ml} H_m \right\} \tag{27}$$

(C) One weight in each layer:

$$\frac{\partial^2 E_p}{\partial w_{lk}\partial w_{ml'}} = z_k f'(a_l) \left\{ \sigma_m \delta_{ll'} + z_{l'} w_{ml} H_m \right\} \tag{28}$$

where H_m is defined by equation 19.

If one or both of the weights is a bias term, then the corresponding expressions are obtained simply by setting the appropriate unit activation(s) to +1.

4 Summary

In this paper, we have derived a general algorithm for the exact evaluation of the second derivatives of the error function, for a network having arbitrary feed-forward topology.

The algorithm involves successive forward and backward propagations through the network, and is expressed in a form which allows for straightforward implementation in software. The number of forward and backward propagations, per training pattern, is at most equal to twice the number of (hidden plus output) units in the network, while the total number of multiplications and additions scales like the square of the number of weights in the network. For networks having a single hidden layer, the algorithm can be expressed in a particularly simple form. Results from a software simulation of this algorithm, applied to the problem of fast network re-training, are described in Bishop (1991).

References

Becker S. and Le Cun Y. 1988. Improving the Convergence of Back-Propagation Learning with Second Order Methods. In *Proceedings of the Connectionist Models Summer School*, Ed. D. S. Touretzky, G. E. Hinton and T. J. Sejnowski, Morgan Kaufmann, 29.

Bishop C. M. 1990. Curvature-Driven Smoothing in Feed-forward Networks. In *Proceedings of the International Neural Network Conference*, Paris, Vol 2, p749. Submitted to *Neural Networks.*

Bishop C. M. 1991. A Fast Procedure for Re-training the Multi-layer Perceptron. To appear in *International Journal of Neural Systems* **2** No. 3.

Le Cun Y., Denker J. S. and Solla S. A. 1990. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, Volume 2, Ed. D. S. Touretzky, Morgan Kaufmann, 598.

MacKay D. J. C. 1991. A Practical Bayesian Framework for Backprop Networks. Submitted to *Neural Computation.*

Ricotta L. P., Ragazzini S. and Martinelli G. 1988. Learning of Word Stress in a Sub-optimal Second Order Back-propagation Neural Network. In *Proceedings IEEE International Conference on Neural Networks*, San Diego, Vol 1, 355

Solla S. A., Levin E. and Fleisher M. 1988. Accelerated Learning in Layered Neural Networks. *Complex Systems* **2**, 625 – 640.