To appear in *Neural Information Processing Systems-92)*

Second Order Derivatives for Network Pruning: Optimal Brain Surgeon

Babak Hassibi and **David G. Stork**
Ricoh California Research Center
2882 Sand Hill Road, Suite 115
Menlo Park, CA 94025-7022
stork@crc.ricoh.com

and

Department of Electrical Engineering
Stanford University
Stanford, CA 94305

ABSTRACT

We investigate the use of information from *all* second order derivatives of the error function to perform network pruning (i.e., removing unimportant weights from a trained network) in order to improve generalization and increase the speed of further training. Our method, Optimal Brain Surgeon (OBS), is significantly better than magnitude-based methods, which can often remove the wrong weights. OBS also represents a major improvement over other methods, such as Optimal Brain Damage [Le Cun, Denker and Solla, 1990], because ours uses the full off-diagonal information of the Hessian matrix **H**. Crucial to OBS is a recursion relation for calculating $\mathbf{H}^{-1}$ from training data and structural information of the net. We illustrate OBS on standard benchmark problems — the MONK's problems. The most successful method in a recent competition in machine learning [Thrun et al., 1991] was backpropagation using weight decay, which yielded a network with 58 weights for one MONK's problem. OBS requires only 14 weights for the same performance accuracy. On two other MONK's problems, our method required only 38% and 10% of the weights found by magnitude-based pruning.

INTRODUCTION

A central problem in machine learning is minimizing the system complexity (description length, VC-dimension, etc.) consistent with the training data. In neural networks this problem is usually cast as minimizing the number of connection weights. It is well-known that without such elimination of weights overfitting problems and thus poor performance on untrained patterns result, i.e., poor generalization. On the other hand, if there are too few weights, the network might not be able to learn the training data.

The question then becomes, which weights should be eliminated? How should we adjust the remaining weights (if at all) for best performance? How can this be done in a computationally efficient way?

Magnitude based methods [Hertz, Krogh and Palmer, 1991] eliminate weights that have the smallest magnitude. This simple and naively plausible idea unfortunately often leads to the elimination of the wrong weights — small weights can be necessary for low error. Both Optimal Brain Damage [Le Cun, Denker and Solla, 1990] and our Optimal Brain Surgeon use instead a criterion of minimal increase in error on the training data. The superiority of OBS lies in great part to the fact that it makes no restrictive assumptions about the form of the network's Hessian that OBD does — assumptions that we find are invalid for many networks. Moreover, unlike OBD, OBS does not demand (typically slow) retraining.

OPTIMAL BRAIN SURGEON

In deriving our method we begin, as do Le Cun, Denker and Solla [1990], by considering a network trained to a local minimum in error. The functional Taylor series of the error with respect to weights is:

$$\delta \mathrm{E} = \underbrace{\left(\frac{\partial \mathrm{E}}{\partial \mathbf{w}}\right)^{\mathrm{T}} \cdot \delta \mathbf{w}}_{\approx 0} + \tfrac{1}{2}\, \delta \mathbf{w}^{\mathrm{T}} \cdot \mathbf{H} \cdot \delta \mathbf{w} + \underbrace{O(\|\delta \mathbf{w}\|^{3})}_{\approx 0} \tag{1}$$

where $\mathbf{H} = \dfrac{\partial^2 \mathrm{E}}{\partial \mathbf{w}^2}$ is the Hessian matrix (containing all second order derivative information) and the superscript T denotes vector transpose. For a network trained to a local minimum in error, the first (linear) term vanishes; we also ignore the third and all higher order terms. Our goal is then to set one of the weights to zero (which will be called $\mathrm{w_q}$) to minimize the increase in error given by Eq. 1. Eliminating $\mathrm{w_q}$ can be expressed as:

$$\delta \mathrm{w}_q + \mathrm{w}_q = 0 \qquad \text{or} \qquad \mathbf{e}_q^{\mathrm{T}} \cdot \delta \mathbf{w} + \mathrm{w}_q = 0 \tag{2}$$

where $\mathbf{e}_\mathrm{q}$ is the unit vector in weight space corresponding to (scalar) weight $\mathrm{w_q}$. Our goal is then to solve:

$$Min_q \{ Min_{\delta \mathbf{w}} \{ \tfrac{1}{2}\, \delta \mathbf{w}^{\mathrm{T}} \cdot \mathbf{H} \cdot \delta \mathbf{w} \} \quad \text{such that} \quad \mathbf{e}_q^{T} \cdot \delta \mathbf{w} + \mathrm{w}_q = 0 \} \tag{3}$$

To this end we form a Lagrangian from Eqs. 1 and 2:

$$L = \tfrac{1}{2}\, \delta \mathbf{w}^{\mathrm{T}} \cdot \mathbf{H} \cdot \delta \mathbf{w} + \lambda (\mathbf{e}_q^{T} \cdot \delta \mathbf{w} + \mathrm{w}_q) \tag{4}$$

where λ is a Lagrange undetermined multiplier. We take functional derivatives, employ the constraints of Eq. 2, and use matrix inversion to find that the optimal weight change and resulting change in error are:

$$\delta \mathbf{w} = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} \cdot \mathbf{e}_q \qquad \text{and} \qquad L = \frac{1}{2} \frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}} \tag{5}$$

Note that neither **H** nor $\mathbf{H}^{-1}$ need be diagonal (as is assumed by Le Cun et al.); moreover, our method recalculates the magnitude of *all* the weights in the network, by the left side of Eq. 5. Figure 1 illustrates the basic idea.

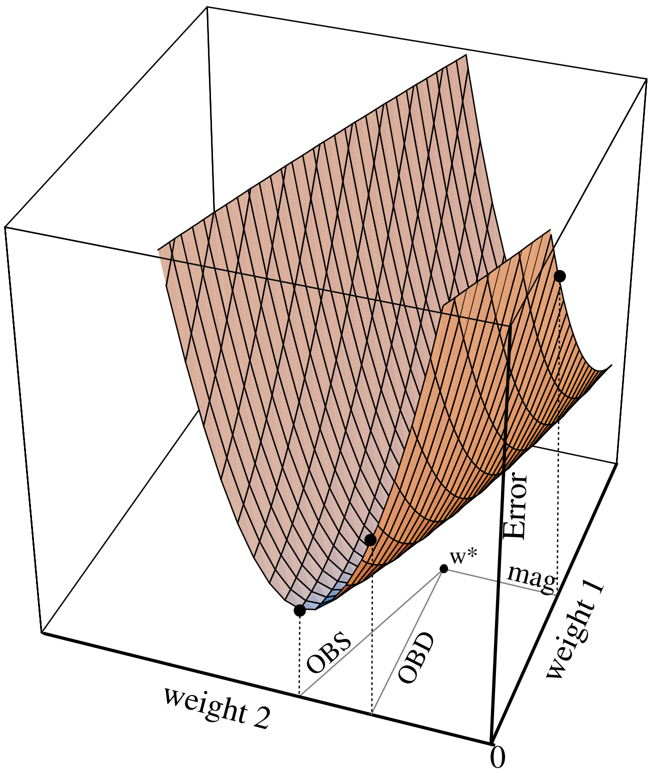

Figure 1: Error as a function of two weights in a network. The (local) minimum occurs at weight w*, found by gradient descent learning. In this illustration, a magnitude based pruning technique (mag) then removes the smallest weight, weight 2; Optimal Brain Damage before retraining (OBD) removes weight 1. In contrast, our Optimal Brain Surgeon method (OBS) removes weight 1 but *also* automatically adjusts the value of weight 2 to minimize the error. The error surface here is general in that it has different curvatures (second derivatives) along different directions, a minimum at a non-special weight value, and a non-diagonal Hessian (i.e., principal axes are *not* parallel to the weight axes). The relative magnitudes of the error after pruning (before retraining, if any) depend upon the particular problem, but to second order obey: E(mag) ≥ E(OBD) ≥ E(OBS), which is the key to the superiority of OBS. We call our method Optimal Brain *Surgeon* because in addition to cutting out weights, it calculates and *changes* the strengths of other weights without the need for retraining.

Thus we have the following algorithm:

Optimal Brain Surgeon procedure

1. Train a "reasonably large" network to minimum error.
2. Compute $\mathbf{H}^{-1}$
3. Find the q that gives the smallest $L = w_q^2/(2[H^{-1}]_{qq})$ (cf. Eq. 5). If this candidate error increase is much smaller than *E*, then the q^{th} weight should be deleted, and we proceed to step 4; otherwise go to step 5.
4. Use the q from step 3 to update *all* weights (Eq. 5). Go to step 2.
5. No more weights can be deleted without large increase in E. (At this point it may be desirable to retrain the network.)

COMPUTING THE INVERSE HESSIAN

The difficulty appears to be step 2 in the procedure, since inverting a matrix of hundreds or thousands of terms seems computationally intractable. (In fact, for very small problems the Hessian and its inverse can be calculated explicitly by standard matrix methods; dramatic results on such toy problems gives us confidence in our approach, but sheds little light on inverting **H** for large, realistic problems.) We show here that for backpropagation nets the Hessian has an interesting structure which permits its inverse to be calculated efficiently and robustly, even for large problems.

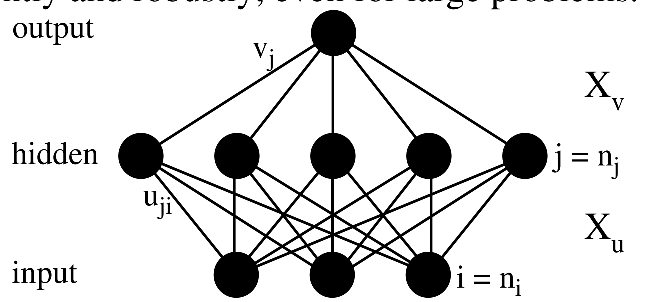

Figure 2: Backpropagation net with n_i inputs and n_j hidden units. The input-to-hidden weights are u_{ji} and hidden-to-output weights v_j. The vectors $\mathbf{X}_v$ and $\mathbf{X}_u$ refer to second derivative information (cf. Eqs. 13-14).

We use the standard terminology from backpropagation [Rumelhart, Hinton and Williams, 1986], here for a single output network. (The generalization to more output units is straightforward.) The error is:

$$E = \frac{1}{2P}\sum_{k=1}^{P}(t^{[k]} - o^{[k]})^2 \tag{6}$$

where P is the total number of training patterns, [*k*] is a pattern index and t and o are the teaching and network output signals. We use the notation of Fig. 2 and after some tedious calculations find that the three types of second derivatives (two intra- and one inter-weight-level) are:

$$\frac{\partial^2 E}{\partial v_j \partial v_{j'}} = \frac{1}{P}\sum_{k=1}^{P}\{f'(net^{[k]})^2 - (t^{[k]} - o^{[k]})f''(net^{[k]})\}o_j^{[k]}o_{j'}^{[k]} \tag{7}$$

$$\begin{aligned}\frac{\partial^2 E}{\partial v_j \partial u_{j'i'}} = \frac{1}{P}\sum_{k=1}^{P}\{\{\{f'(net^{[k]})^2 - (t^{[k]} - o^{[k]})f''(net^{[k]})\}v_{j'}f'(net_{j'}^{[k]})o_{i'}^{[k]}o_j^{[k]}\} - \\ (t^{[k]} - o^{[k]})f'(net^{[k]})f'(net_{j'}^{[k]})\delta_{jj'}o_{i'}^{[k]}\}\end{aligned} \tag{8}$$

$$\begin{aligned}\frac{\partial^2 E}{\partial u_{ji} \partial u_{j'i'}} = \frac{1}{P}\sum_{k=1}^{P}\{\{f'(net^{[k]})^2 - (t^{[k]} - o^{[k]})f''(net^{[k]})\}v_j v_{j'}f'(net_j^{[k]})f'(net_{j'}^{[k]})o_i^{[k]}o_{i'}^{[k]} - \\ (t^{[k]} - o^{[k]})f'(net^{[k]})v_j f''(net_j^{[k]})\delta_{jj'}o_i^{[k]}o_{i'}^{[k]}\}\end{aligned} \tag{9}$$

where all unsubscripted terms refer to the output unit, and δ is the Kronecker delta. We need only ever evaluate these second derivatives after training, when the actual output is approximately equal to the teaching signal. In this limit of $t^{[k]} = o^{[k]}$ Eqs. 7-9 simplify to:

$$\frac{\partial^2 E}{\partial v_j \partial v_{j'}} = \frac{1}{P}\sum_{k=1}^{P}f'(net^{[k]})^2 o_j^{[k]}o_{j'}^{[k]} \tag{10}$$

$$\frac{\partial^2 E}{\partial v_j \partial u_{j'i'}} = \frac{1}{P}\sum_{k=1}^{P} f'(net^{[k]})^2 v_{j'} f'(net_{j'}^{[k]}) o_{i'}^{[k]} o_j^{[k]} \tag{11}$$

$$\frac{\partial^2 E}{\partial u_{ji} \partial u_{j'i'}} = \frac{1}{P}\sum_{k=1}^{P} f'(net^{[k]})^2 v_j v_{j'} f'(net_j^{[k]}) f'(net_{j'}^{[k]}) o_i^{[k]} o_{i'}^{[k]} \tag{12}$$

The remarkably elegant structure of Eqs. 10-12 permits us to calculate the Hessian and its inverse simply. To see this we first define vectors of second derivative information for the weight layers v and u (cf. Fig. 2):

$$[\mathbf{X}_v^{[k]}]^T = \left(f'(net^{[k]}) o_{j=1}^{[k]}, \ldots, f'(net^{[k]}) o_{n_j}^{[k]} \right) \tag{13}$$

$$\begin{aligned}[\mathbf{X}_u^{[k]}]^T = \Big(& f'(net^{[k]}) f'(net_1^{[k]}) v_1^{[k]} o_{i=1}^{[k]}, \ldots, f'(net^{[k]}) f'(net_1^{[k]}) v_1^{[k]} o_{n_i}^{[k]}, \ldots, \\ & f'(net^{[k]}) f'(net_{n_j}^{[k]}) v_{n_j}^{[k]} o_1^{[k]}, \ldots, f'(net^{[k]}) f'(net_{n_j}^{[k]}) v_{n_j}^{[k]} o_{n_i}^{[k]} \Big)\end{aligned} \tag{14}$$

where for $\mathbf{X}_u$ we employ lexico-graphical ordering over the input and hidden units. We concatenate these vectors to form a full vector of second derivative information throughout the net (evaluated for a single pattern $[k]$):

$$\mathbf{X}^{[k]} = \begin{pmatrix} \mathbf{X}_v^{[k]} \\ \mathbf{X}_u^{[k]} \end{pmatrix} \tag{15}$$

The dimensionality of $\mathbf{X}$ is equal to the total number of weights in the net. The Hessian, evaluated at a local minimum, is then simply:

$$\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w}^2} = \frac{1}{P}\sum_{k=1}^{P} \mathbf{X}^{[k]} \cdot \mathbf{X}^{[k]T} \tag{16}$$

We can calculate the full Hessian by sequentially adding in successive "component" Hessians (Eq. 16) as:

$$\mathbf{H}_{m+1} = \mathbf{H}_m + \frac{1}{P}\mathbf{X}^{[m+1]} \cdot \mathbf{X}^{[m+1]T} \quad \text{with} \quad \mathbf{H}_0 = \alpha \mathbf{I} \text{ and } \mathbf{H}_P = \mathbf{H} \tag{17}$$

But Optimal Brain Surgeon requires the *inverse* of $\mathbf{H}$ (Eq. 5). This inverse can be calculated using a standard matrix inversion formula [Kailath, 1980]:

$$(\mathbf{A} + \mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\cdot\mathbf{B}\cdot(\mathbf{C}^{-1} + \mathbf{D}.\mathbf{A}^{-1}\cdot\mathbf{B})^{-1}\cdot\mathbf{D}\cdot\mathbf{A}^{-1} \tag{18}$$

applied to each term in the analogous sequence in Eq. 17:

$$\mathbf{H}_{m+1}^{-1} = \mathbf{H}_m^{-1} - \frac{\mathbf{H}_m^{-1}\cdot\mathbf{X}^{[m+1]}\cdot\mathbf{X}^{[m+1]T}\cdot\mathbf{H}_m^{-1}}{P + \mathbf{X}^{[m+1]T}\cdot\mathbf{H}_m^{-1}\cdot\mathbf{X}^{[m+1]T}} \quad \text{with} \quad \mathbf{H}_0^{-1} = \alpha^{-1}\mathbf{I} \text{ and } \mathbf{H}_P^{-1} = \mathbf{H}^{-1} \tag{19}$$

and α ($10^{-8} \le \alpha \le 10^{-4}$) a small constant needed to make $\mathbf{H}_0^{-1}$ meaningful, and to which our method is insensitive [Hassibi and Stork, 1993].

Equation 19 is a pivotal result, which permits the calculation of $\mathbf{H}^{-1}$ using a *single* sequential pass through the training data $1 \le m \le P$.

We note that the approximations used for Eqs. 10-12 induce several very nice computational properties in Eq. 19. The first is that because of the approximation $t^{[k]} = o^{[k]}$, $\mathbf{H}^{-1}$ is being evaluated *at the desired error minimum*, despite the possibility that the network has not been fully trained to this minimum. Moreover, there are no singularities in the calculation of $\mathbf{H}^{-1}$. Finally, Eq. 19 is an $O(n^2)$ calculation that may possibly admit efficient $O(n)$ parallel implementations in VLSI.

SIMULATION RESULTS

We applied OBS to three of the Monk's problems and compared our results to those of Thrun et al. [1991], whose backpropagation network outperformed all other approaches (network and rule-based) on these benchmark problems in an extensive machine learning competition.

		Accuracy		# weights
		training	test	
MONK 1	**BPWD**	**100**	**100**	**58**
	OBS	**100**	**100**	**14**
MONK 2	**BPWD**	**100**	**100**	**39**
	OBS	**100**	**100**	**15**
MONK 3	**BPWD**	**93.4**	**97.2**	**39**
	OBS	**93.4**	**97.2**	**4**

Table 1: The accuracy and number of weights found by backpropagation with weight decay found by Thrun et al. [1991], and by OBS on three MONK's problems.

Table 1 shows that for the same performance, OBS (without retraining) required only 24%, 38% and 10% of the weights of the BP network, which, we stress was *already pruned by the most widely used method* (Fig. 3). The error increase L (Eq. 5) determined by OBS negligibly affected accuracy.

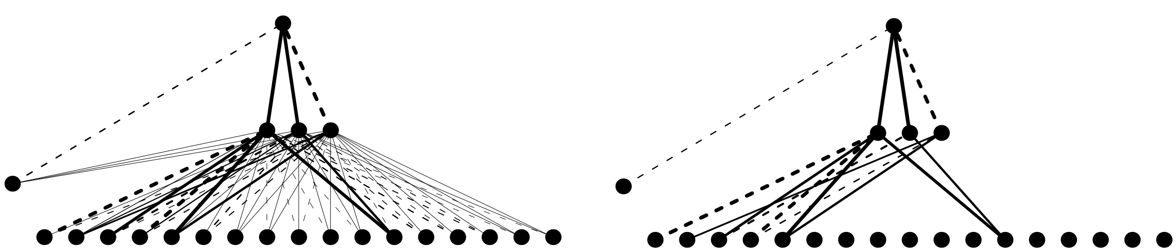

Figure 3: Optimal networks found by Thrun (left) and by OBS on MONK 1, which is based on logical rules. Solid (dashed) lines denote excitatory (inhibitory) connections; bias units are at left. Weight decay cannot reduce the small weights without increasing E.

ANALYSIS AND CONCLUSIONS

Why is Optimal Brain Surgeon so successful at reducing excess degrees of freedom? Or conversely, given this new standard in weight elimination, we can ask: Why are magnitude based methods so poor? Consider again Fig. 1. Starting from a the local minimum at w*, a magnitude based method deletes the wrong weight, weight 2, and through retraining, weight 1 will *increase*.

The final “solution” is weight 1 $\rightarrow$ large, weight 2 = 0. This is precisely the *opposite* of the solution found by OBS: weight 1 = 0, weight 2 $\rightarrow$ large. Although the actual difference in error shown in Fig. 1 may be small, in large networks, differences from many incorrect weight elimination decisions can add up to a significant increase in error. But most importantly, it is simply wishful thinking to believe that after the elimination of many incorrect weights by magnitude methods that the net can “sort it all out” through further training and reach a global optimum. Even gradual magnitude methods such as weight decay have this drawback.

The approximation necessary for Optimal Brain Damage [Le Cun, Denker and Solla, 1990] — that the diagonals of Hessian are dominant — does not seem to hold on the problems we have investigated. There are typically many off-diagonal terms that are comparable to their diagonal counterparts, and explains the improvement of OBS over OBD [Hassibi and Stork, 1993].

We note in closing that our method is quite general, and subsumes previous methods for weight elimination. In our terminology, magnitude based methods assume isotropic Hessian ($\mathbf{H} \propto \mathbf{I}$); OBD assumes diagonal $\mathbf{H}$; FARM [Kung and Hu, 1991] assumes linear f(net) and only updates the hidden-to-output weights. We have shown that none of those assumptions are valid nor sufficient for optimal weight elimination.

Acknowledgements
Thanks to Greg Wolff for useful discussions and display software, and to T. Kailath for constant encouragement and financial support of the first author.

REFERENCES

Hassibi, B. and Stork, D. G. (1993). Optimal Brain Surgeon (sub. for publ.).

Hertz, J., Krogh, A. and Palmer, R. G. (1991). ***Introduction to the Theory of Neural Computation*** Addison-Wesley.

Kailath, T. (1980). ***Linear Systems*** Prentice-Hall.

Kung, S. Y. and Hu, Y. H. (1991). A Frobenius approximation reduction method (FARM) for determining the optimal number of hidden units, ***Proceedings of the IJCNN-91*** Seattle, Washington.

Le Cun, Y., Denker, J. S. and Solla, S. A. (1990). Optimal Brain Damage, in ***NIPS 2***, D. S. Touretzky (ed.) 598-605, Morgan-Kaufmann.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Internal representations by error propagation, Chapter 8 (318-362) in ***Parallel Distributed Processing I*** D. E. Rumelhart and J. L. McClelland (eds.) MIT Press.

Thrun, S. B. and 23 co-authors (1991). The MONK's Problems — A performance comparison of different learning algorithms, CMU-CS-91-197 Carnegie-Mellon U. Department of Computer ScienceTech Report.