Fast Exact Multiplication by the Hessian

Barak A. Pearlmutter
Dept. of Computer Sci. & Eng.
Oregon Graduate Institute
19600 NW von Neumann Drive
Beaverton, OR 97006-1999
bap@cse.ogi.edu

Abstract

Just storing the Hessian H (the matrix of second derivatives $\partial^2 E/\partial w_i \partial w_j$ of the error E with respect to each pair of weights) of a large neural network is difficult. Since a common use of a large matrix like H is to compute its product with various vectors, we derive a technique that directly calculates Hv, where v is an arbitrary vector. This allows H to be treated as a generalized sparse matrix. To calculate Hv, we first define a differential operator $\mathcal{R}\{f(w)\} = (\partial/\partial r) f(w + rv)|_{r=0}$, note that $\mathcal{R}\{\nabla_w\} = Hv$ and $\mathcal{R}\{w\} = v$, and then apply $\mathcal{R}\{\cdot\}$ to the equations used to compute ∇_w. The result is an exact and numerically stable procedure for computing Hv, which takes about as much computation, and is about as local, as a gradient evaluation. We then apply the technique to backpropagation networks, recurrent backpropagation, and stochastic Boltzmann Machines. Finally, we show that this technique can be used at the heart of many iterative techniques for computing various properties of H, obviating the need for direct methods.

1 Introduction

Efficiently extracting second-order information from large neural networks is an important problem, because properties of the Hessian appear frequently. For instance, in the analysis of the convergence of learning algorithms (Widrow et al., 1979; le Cun et al., 1991; Pearlmutter, 1992); in some techniques for estimating generalization in neural networks (MacKay, 1991; Moody, 1992); in techniques for enhancing generalization by pruning weights (le Cun et al., 1990; Hassibi and Stork, 1993); and in full second-order numerical optimization methods (Watrous, 1987).

There exist algorithms for calculating the full Hessian H (the matrix of second derivative terms $\partial^2 E/\partial w_i \partial w_j$ of the error E with respect to the weights w) of a backpropagation network (Bishop, 1992; Buntine and Weigend, 1991), or reasonable estimates thereof (MacKay, 1991)—but just storing the full Hessian is impractical for large networks. There is also an algorithm for efficiently computing just the diagonal of the Hessian (Becker and le Cun, 1989; le Cun et al., 1990). This is useful when the trace of the Hessian is needed, or when the diagonal approximation is being made—but there is no reason to believe that the diagonal approximation is good in general, and it is reasonable to suppose that, as the system grows, the diagonal elements of the Hessian become less and less dominant. Further, the inverse of the diagonal approximation to the Hessian is known to be a poor approximation to the diagonal of the inverse of the Hessian.

Here we derive an efficient technique for calculating the product of an arbitrary vector v with the Hessian H. This allows information to be pulled out of the Hessian without ever calculating of storing the Hessian itself. A common use for an estimate of the Hessian is to take its product with various vectors. This takes $O(n^2)$ time when there are n parameters. The technique we derive here finds this product in $O(n)$ time and space,[1] and does not make any approximations.

We first operate in a very general framework, to develop the basic technique, and then proceed to apply it to a series of more and more complicated systems, starting with a simple backwards propagation of error to calculate gradients, *i.e.* a backpropagation network, and then proceeding toa deterministic relaxation network, and then to a stochastic Boltzmann Machine.

2 The Relation Between the Gradient and the Hessian

The basic technique is to note that the Hessian matrix appears in the expansion of the gradient about a point in parameter space,

$$\nabla_w(w + \Delta w) = \nabla_w(w) + H\Delta w + O(||\Delta w||^2)$$

where E is the error, w is a point in parameter space, Δw is a perturbation of w, ∇_w is the gradient, the vector of partial derivatives $\partial E/\partial w_i$, and H is the Hessian, the matrix of second derivatives of E with respect to each pair of elements of w. This equation has been used to analyze the convergence properties of some variants of gradient descent (Pearlmutter, 1992), and to approximate the effect of deleting a weight from the network (le Cun et al., 1990; Hassibi and Stork, 1993). Here we instead use it by choosing $\Delta w = rv$, where v is a vector and r is a small number. We wish to compute Hv. Now we note that

$$H(rv) = rHv = \nabla_w(w + rv) - \nabla_w(w) + O(r^2)$$

or, dividing by r,

$$Hv = \frac{\nabla_w(w + rv) - \nabla_w(w)}{r} + O(r). \tag{1}$$

This in itself provides a simple approximation algorithm for finding Hv for any system whose gradient can be efficiently computed, in time less than double that required to compute the gradient, assuming that the gradient at w has already been computed. Also, applying the technique requires minimal programming effort. This approximation was used to good effect in (le Cun et al., 1993).

Unfortunately, this formula is succeptible to numeric and roundoff problems. The constant r must be small enough that the $O(r)$ term is insignificant. But as r becomes small, large numbers are added to tiny ones in $w + rv$, causing a loss of precision of v. A similar loss of precision occurs in the subtraction of the original gradient from the perturbed one, because two nearly identical vectors are being subtracted to obtain the tiny difference between them.

3 The $\mathcal{R}\{\cdot\}$ Technique

Fortunately, there is a way to make an algorithm which exactly computes Hv, rather than just approximating it, and simultaneously rid ourselves of these numeric difficulties. To do this, we first

[1] Or $O(pn)$ time when, as is typical for supervised neural networks, the full gradient is the sum of p gradients, each for one single exemplar.

take the limit of equation (1) as $r \to 0$. The left hand side stays Hv, while the right hand side matches the definition of a derivative, and thus

$$Hv = \lim_{r \to 0} \frac{\nabla_w(w + rv) - \nabla_w(w)}{r} = \frac{\partial}{\partial r} \nabla_w(w + rv) \Big|_{r=0} \tag{2}$$

As we shall see, there is a simple transformation to convert an algorithm that computes the gradient of the system into one that computes this new quantity. The key to this transformation is to define the operator

$$\mathcal{R}\{f(w)\} = \frac{\partial}{\partial r} f(w + rv) \Big|_{r=0} \tag{3}$$

We can then take all the equations of the procedure for calculating the gradient, *e.g.* the backpropagation procedure, or the deterministic Boltzmann Machine gradient calculation, or whatever the gradient calculation involves, and we can apply this $\mathcal{R}\{\cdot\}$ operator to each equation. Because $\mathcal{R}\{\cdot\}$ is a differential operator, it obeys the usual rules for differential operators, such as:

$$\begin{aligned}
\mathcal{R}\{cf(w)\} &= c\mathcal{R}\{f(w)\} \\
\mathcal{R}\{f(w) + g(w)\} &= \mathcal{R}\{f(w)\} + \mathcal{R}\{g(w)\} \\
\mathcal{R}\{f(w)g(w)\} &= \mathcal{R}\{f(w)\}\, g(w) + f(w)\mathcal{R}\{g(w)\} \\
\mathcal{R}\{f(g(w))\} &= f'(g(w))\mathcal{R}\{g(w)\} \\
\mathcal{R}\left\{\frac{df(w)}{dt}\right\} &= \frac{d\mathcal{R}\{f(w)\}}{dt}
\end{aligned} \tag{4}$$

Also note that

$$\mathcal{R}\{w\} = v. \tag{5}$$

These rules are sufficient to derive, from the equations normally used to compute the gradient, a new set of equations about a new set of $\mathcal{R}$-variables. These new equations make use of variables from the original gradient calculation on their right hand sides. This can be thought of as an adjoint system to the gradient calculation, just as the gradient calculation of backpropagation can be thought of as an adjoint system to the forward calculation of the error measure. This new adjoint system computes the vector $\mathcal{R}\{\nabla_w\}$, which is precisely the vector Hv which we desire.

4 Application of the $\mathcal{R}\{\cdot\}$ Technique to Various Networks

Let us utilize this new technique for transforming the equations that compute the gradient into equations that compute Hv, the product of a vector v with the Hessian H. We will, rather mechanically, derive appropriate algorithms for some standard sorts of neural networks. Of course, the same process can be used to mechanically derive similar algorithms for most neural network gradient calculation procedures not covered below .

Usually the error E is the sum of the errors for many patterns, $E = \sum_p E_p$. Therefore ∇_w and H are sums over all the patterns, $H = \sum_p H_p$, and $Hv = \sum_p H_p v$. As is usual, for clarity this outer sum over patterns is not shown except where necessary, and the gradient and Hv procedures are shown for only a single exemplar.

4.1 Simple Backpropagation Networks

Let us apply the above procedure to a simple backpropagation network, to derive the $\mathcal{R}\{\text{backpropagation}\}$ algorithm, a set of equations that can be used to efficiently calculate Hv for a backpropagation network. This $\mathcal{R}\{\text{backpropagation}\}$ algorithm was independently discovered by Martin Møller (unpublished manuscript) using a different derivation.

For convenience, we will now change our notation for indexing the weights w. Let w be the weights, now doubly indexed by their source and destination units' indices, as in w_{ij}, the weight from unit i to unit j. Because v is of the same dimension as w, its elements will be identically indexed. All sums over indices are limited to weights that exist in the network topology. As is usual, quantities which occur on the left sides of the equations are treated computationally as variables, and calculated in topological order, which is assumed to exist because the matrix w_{ij} is zero-diagonal and can be put into triangular form (Werbos, 1974).

The forward computation of the network is[2]

$$\begin{aligned} x_i &= \sum_j w_{ji} y_j \\ y_i &= \sigma_i(x_i) + I_i \end{aligned} \tag{6}$$

where $\sigma_i(\cdot)$ is the nonlinearity of the ith unit, x_i is the total input to the ith unit, y_i is the output of the ith unit, and I_i is the external input (from outside the network) to the ith unit.

Let the error measure be $E = E(y)$, and its simple direct derivative with respect to y_i be $e_i = dE/dy_i$. We assume that e_i depends only on y_i, and not on any y_j for $j \neq i$. This is true of most common error measures, such as sum square error or cross entropy.[3] We can thus write $e_i(y_i)$ as a simple function. The backward pass is then

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= e_i(y_i) + \sum_j w_{ij} \frac{\partial E}{\partial x_j} \\ \frac{\partial E}{\partial x_i} &= \sigma_i'(x_i) \frac{\partial E}{\partial y_i} \\ \frac{\partial E}{\partial w_{ij}} &= y_i \frac{\partial E}{\partial x_j} \end{aligned} \tag{7}$$

Applying $\mathcal{R}\{\cdot\}$ to the above equations gives

$$\begin{aligned} \mathcal{R}\{x_i\} &= \sum_j \left(w_{ji} \mathcal{R}\{y_i\} + v_{ji} y_j \right) \\ \mathcal{R}\{y_i\} &= \mathcal{R}\{x_i\} \, \sigma_i'(x_i) \end{aligned} \tag{8}$$

for the forward pass, and, for the backward pass,

$$\mathcal{R}\left\{ \frac{\partial E}{\partial y_i} \right\} = e_i'(y_i) \mathcal{R}\{y_i\} + \sum_j \left(w_{ij} \mathcal{R}\left\{ \frac{\partial E}{\partial x_j} \right\} + v_{ij} \frac{\partial E}{\partial x_j} \right) \tag{9}$$

[2]This compact form of the backpropagation equations, due to Fernando Pineda, unifies the special cases of input units, hidden units, and output units. In the case of a unit i with no incoming weights, an input unit, it simplifies to $y_i = \sigma_i(0) + I_i$, allowing the value to be set entirely externally. For a hidden unit or output i, the term $I_i = 0$. In the corresponding equations for the backward pass (7) only the output units have nonzero direct error terms e_i, and since such output units have no outgoing weights, the situation for an output unit i simplifies to $\partial E / \partial y_i = e_i(y_i)$.

[3]If this assumption is violated then in equation (9) the $e_i'(y_i)\mathcal{R}\{y_i\}$ term generalizes to $\sum_j (\partial e_i / \partial y_j) \mathcal{R}\{y_j\}$.

$$
\begin{aligned}
\mathcal{R}\left\{\frac{\partial E}{\partial x_i}\right\} &= \mathcal{R}\{x_i\}\,\sigma_i''(x_i)\frac{\partial E}{\partial y_i} + \sigma_i'(x_i)\mathcal{R}\left\{\frac{\partial E}{\partial y_i}\right\} \\
\mathcal{R}\left\{\frac{\partial E}{\partial w_{ij}}\right\} &= y_i\mathcal{R}\left\{\frac{\partial E}{\partial x_j}\right\} + \mathcal{R}\{y_i\}\,\frac{\partial E}{\partial x_j}
\end{aligned}
$$

The vector whose elements are the terms $\mathcal{R}\{\partial E/\partial w_{ij}\}$ is just $\mathcal{R}\{\nabla_w\} = Hv$, the quantity we wish to compute.

For simple squared error $e_i(y_i) = y_i - d_i$ where d_i is the desired output for unit i, so $e_i'(y_i) = 1$, which simplifies (9) for simple output units to $\mathcal{R}\{\partial E/\partial y_i\} = \mathcal{R}\{y_i\}$. Note that, in the above equations, the topology of the neural network sometimes results in some $\mathcal{R}$-variables being guaranteed to be zero when v is sparse, and in particular, when $v = (0 \cdots 0\ 1\ 0 \cdots 0)$, which can be used to compute a single desired column of the Hessian. In fact, using a sequence of such vectors to pull out the Hessian column by column is the most efficient known technique to compute the Hessian for networks of this sort.

4.2 Recurrent Backpropagation Networks

The recurrent backpropagation algorithm (Almeida, 1987; Pineda, 1987) consists of a set of forward equations which relax to a solution for the gradient,

$$
\begin{aligned}
x_i &= \sum_j w_{ji} y_j \qquad (10) \\
\frac{dy_i}{dt} &\propto -y_i + \sigma_i(x_i) + I_i \\
\frac{dz_i}{dt} &\propto -z_i + \sigma_i'(x_i)\sum_j (w_{ij} z_j) + e_i(y_i) \\
\frac{\partial E}{\partial w_{ij}} &= y_i z_j|_{t=\infty}
\end{aligned}
$$

Adjoint equations for the calculation of Hv are obtained by applying the $\mathcal{R}\{\cdot\}$ operator, yielding

$$
\begin{aligned}
\mathcal{R}\{x_i\} &= \sum_j \left(w_{ji}\mathcal{R}\{y_i\} + v_{ji} y_j\right) \qquad (11) \\
\frac{d\mathcal{R}\{y_i\}}{dt} &\propto -\mathcal{R}\{y_i\} + \sigma_i'(x_i)\mathcal{R}\{x_i\} \\
\frac{d\mathcal{R}\{z_i\}}{dt} &\propto -\mathcal{R}\{z_i\} + \sigma_i'(x_i)\sum_j \left(v_{ij} z_j + w_{ij}\mathcal{R}\{z_j\}\right) + e_i'(y_i)\mathcal{R}\{y_i\} \\
\mathcal{R}\left\{\frac{\partial E}{\partial w_{ij}}\right\} &= y_i\mathcal{R}\{z_j\} + \mathcal{R}\{y_i\}\, z_j|_{t=\infty}
\end{aligned}
$$

These equations specify a relaxation process for computing Hv. Just as the relaxation equations for computing ∇_w are linear even though those for computing y and E are not, these new relaxation equations are linear.

4.3 Stochastic Boltzmann Machines

One might ask whether this technique can be used to derive a Hessian multiplication algorithm for a classic Boltzmann Machine (Ackley et al., 1985), which is discrete and stochastic, unlike its continuous and deterministic cousin to which application of $\mathcal{R}\{\cdot\}$ is simple. A classic BM operates stochastically, with its binary unit states s_i taking on random values according to the probability

$$\begin{aligned} P(s_i = 1) &= p_i = \sigma(x_i/T) \\ x_i &= \sum_j w_{ji} s_j \end{aligned} \tag{12}$$

At equilibrium, the probability of a state α of all the units is related to its energy

$$E_\alpha = \sum_{i<j} s_i^\alpha s_j^\alpha w_{ij} \tag{13}$$

by $P(\alpha) = Z^{-1} \exp -E_\alpha/T$, where the partition function Z is defined by $Z = \sum_\alpha \exp -E_\alpha/T$. The system's equilibrium statistics are sampled because, at equilibrium,

$$\frac{\partial G}{\partial w_{ij}} = \tfrac{1}{T}\left(p_{ij}^+ - p_{ij}^-\right) \tag{14}$$

where $p_{ij} = \langle s_i s_j \rangle$, G is an information-theoretic error term, the asymmetric divergence, T is the temperature, and the $+$ and $-$ superscripts indicate varying environmental distributions.

Applying the $\mathcal{R}\{\cdot\}$ operator, we obtain

$$\mathcal{R}\left\{\frac{\partial G}{\partial w_{ij}}\right\} = \tfrac{1}{T}\left(\mathcal{R}\left\{p_{ij}^+\right\} - \mathcal{R}\left\{p_{ij}^-\right\}\right). \tag{15}$$

We shall soon find it useful if we define

$$D_\alpha = \mathcal{R}\{E_\alpha\} = \sum_{i<j} s_i^\alpha s_j^\alpha v_{ij} \tag{16}$$

$$q_{ij} = \langle s_i s_j D \rangle \tag{17}$$

(with the letter D chosen because it has the same relation to v that E has to w) and to note that

$$\langle D \rangle = \sum_{i<j} p_{ij} v_{ij}. \tag{18}$$

With a little calculus we see that $\mathcal{R}\{\exp -E_\alpha/T\} = -P(\alpha) Z D_\alpha/T$, and thus $\mathcal{R}\{Z\} = -Z \langle D \rangle / T$. Using these and the relation between the probability of a state and its energy, we find that

$$\mathcal{R}\{P(\alpha)\} = \tfrac{1}{T} P(\alpha)(\langle D \rangle - D_\alpha) \tag{19}$$

which can be used to simplify

$$\begin{aligned} \mathcal{R}\{p_{ij}\} &= \sum_\alpha \mathcal{R}\{P(\alpha)\} s_i^\alpha s_j^\alpha \\ &= \tfrac{1}{T} \sum_\alpha P(\alpha)(\langle D \rangle - D_\alpha) s_i^\alpha s_j^\alpha \\ &= \tfrac{1}{T}(\langle s_i s_j \rangle \langle D \rangle - \langle s_i s_j D \rangle) \\ &= \tfrac{1}{T}\left(p_{ij} \langle D \rangle - q_{ij}\right). \end{aligned} \tag{20}$$

This beautiful formula[4] gives an efficient way to compute Hv for a BM, or at least as efficient a way as is used to compute the gradient, simply by using sampling to estimate q_{ij} and $\langle D \rangle$. It requires the additional calculation and broadcast of the single global quantity D, but is otherwise local.

The collection of statistics for the gradient is sometimes accelerated by using the equation

$$p_{ij} = \langle s_i \rangle \langle s_j | s_i = 1 \rangle = \langle p_i \rangle \langle p_j | s_i = 1 \rangle \,. \tag{21}$$

The analogous identity for accelerating the computation of q_{ij} is

$$q_{ij} = \langle p_i \rangle \langle s_j D | s_i = 1 \rangle \tag{22}$$

or

$$q_{ij} = \langle p_i \rangle \langle p_j (D + (1 - s_j) \Delta D_j) | s_i = 1 \rangle \tag{23}$$

where $\Delta D_i = \sum_j s_j v_{ji}$ is defined by analogy with $\Delta E_i = E|_{s_i=1} - E|_{s_i=0} = \sum_j s_j w_{ji}$.

The derivation here was for the simplest sort of Boltzmann Machine, with binary units and only pairwise connections between the units. However, the technique is immediately applicable to higher order Boltzmann Machines (Hinton, 1987), as well as to Boltzmann Machines with non-binary units (Movellan and McClelland, 1991).

4.4 Weight Perturbation

In weight perturbation (Jabri and Flower, 1991; Alspector et al., 1993; Flower and Jabri, 1993; Kirk et al., 1993; Cauwenberghs, 1993) the gradient ∇_w is approximated using only the globally broadcast result of the computation of $E(w)$. This is done by adding a random zero-mean perturbation vector Δw to w repeatedly and approxmating the resulting change in error by

$$E(w + \Delta w) = E(w) + \Delta E = E(w) + \nabla_w \cdot \Delta w.$$

From the viewpoint of each individual weight w_i

$$\Delta E = \Delta w_i \frac{\partial E}{\partial w_i} + \textit{noise}. \tag{24}$$

Because of the central limit theorem it is reasonable to make a least-squares estimate of $\partial E / \partial w_i$, which is $\partial E / \partial w_i = \langle \Delta w_i \Delta E \rangle / \langle \Delta w_i^2 \rangle$. The numerator is estimated from corresponding samples of Δw_i and ΔE, and the denominator from prior knowledge of the distribution of Δw. This requires only the global broadcast of ΔE, while each Δw_i can be generated and used locally.

It is hard to see a way to mechanically apply $\mathcal{R}\{\cdot\}$ to this procedure, but we can nonetheless derive a suitable procedure for estimating Hv. We note that a better approximation for the change in error would be

$$E(w + \Delta w) = E(w) + \nabla_w \cdot \Delta w + \tfrac{1}{2} \Delta w^T \widehat{H} \Delta w \tag{25}$$

where $\widehat{H}$ is an estimate of the Hessian H. We wish to include in $\widehat{H}$ only those properties of H which are relevent. Let us define $z = Hv$. If $\widehat{H}$ is to be small in the least squares sense, but also $\widehat{H} v = z$, then the best choice would then be $\widehat{H} = z v^T / ||v||^2$, except that then $\widehat{H}$ would not be symmetric,

[4] Equation (20) is similar in form to that of the gradient of the entropy (Geoff Hinton, personal communication.)

and therefore the error surface would not be well defined. Adding the symmetry requirement, the least squares $\widehat{H}$ becomes

$$\widehat{H} = \frac{1}{||v||^2}\left(zv^T + vz^T - \frac{v \cdot z}{||v||^2}vv^T\right). \tag{26}$$

Substituting this in and rearranging the terms, we find that, from the perspective of each weight,

$$\Delta E = \Delta w_i \frac{\partial E}{\partial w_i} + \frac{\Delta w \cdot v}{||v||^2}\left(\Delta w_i - \frac{\Delta w \cdot v}{2||v||^2}v_i\right) z_i + \textit{noise}. \tag{27}$$

This allows both $\partial E/\partial w_i$ and z_i to be estimated in the same least-squares fashion as above, using only locally available values, v_i and Δw_i, and the globally broadcast ΔE, plus a new quantity which must be computed and globally broadcast, $\Delta w \cdot v$. The same technique applies equally well to other perturbative procedures, such as unit perturbation, and a similar derivation can be used to find the diagonal elements of H, without the need for any additional globally broadcast values.

5 Applications

The $\mathcal{R}\{\cdot\}$ technique makes it possible to calculate Hv efficiently. This can be used in the center of many different iterative algorithms, in order to extract particular properties of H. In essence, it allows H to be treated as a generalized sparse matrix.

5.1 Finding Eigenvalues and Eigenvectors

Standard variants of the power method allow one to

- Calculate the largest few eigenvalues of H, and their eigenvectors.
- Calculate the smallest few eigenvalues of H, and their eigenvectors.
- Sample H's eigenvalue spectrum.

A clever algorithm (Skilling, 1989) based on choosing a random vector v_0, calculating $v_i = H^i v_0$ for $0 < i \leq m$, using the dot products $v_i \cdot v_j$ to estimate the moments of the eigenvalue spectrum, and using these moments to recover the shape of the eigenvalue spectrum, is made applicable to the Hessian by the $\mathcal{R}\{\cdot\}$ technique, in both deterministic and, with minor modifications, stochastic settings.

5.2 The Inverse Hessian

It is frequently necessary to compute $H^{-1}b$, the key calculation of all Newton's-method second-order numerical optimization techniques, and also used in the Optimal Brain Surgeon technique and in some techniques for predicting the generalization rate. The $\mathcal{R}\{\cdot\}$ technique does not directly solve this problem, but instead one can solve $Hx = b$ for x by minimizing $||Hx - b||^2$ using the conjugate-gradient method, thus exactly computing $x = H^{-1}b$ in n iterations without calculating or storing H^{-1}. This squares the condition number, but if H is known to be positive definite, one can instead minimize $x^T Hx/2 + x \cdot b$, which does not square the condition number. See (Press et al., 1988, page 78) for details.

5.3 Step Size and Line Search

Many optimization techniques repeatedly choose a direction v, and then proceed along that direction some distance μ, which takes the system to the constrained minimum of $E(w + \mu v)$. Finding the value for μ which minimizes E is called a line search, because it searches only along the line $w + \mu v$. There are many techniques for performing a line search. Some are approximate while others attempt to find an exact constrained minimum, and some use only the value of the error, while others also make use of the gradient.

In particular, the line search used within the Scaled Conjugate Gradient (SCG) optimization procedure, in both its deterministic (Møller, 1993a) and stochastic (Møller, 1993b) incarnations, makes use of both first- and second-order information at w to determine how far to move. The first order information used is simply $\nabla_w(w)$, while the second-order information is precisely Hv, calculated with the one-sided finite difference approximation of equation (1). It can thus benefit immediately from the exact calculation of Hv. In fact, the $\mathcal{R}\{\text{backpropagation}\}$ procedure was independently discovered for precisely that application by Martin Møller (personal communication.)

The SCG line search proceeds as follows. Assuming that the error E is well approximated by a quadratic, the product Hv at w and the gradient $\nabla_w(w)$ allows one to predict the gradient at any point along the line $w + \mu v$ by

$$\nabla_w(w + \mu v) = \nabla_w(w) + \mu H v + O(\mu^2). \tag{28}$$

Disregarding the $O(\mu^2)$ term, if we wish to choose μ to minimize the error, we take the dot product with v and set equal to zero, as the gradient at the constrained minimum must be perpendicular to the space under consideration. This gives $v \cdot \nabla_w(w) + \mu v^T H v = 0$ or

$$\mu = -\frac{v \cdot \nabla_w(w)}{v^T H v}. \tag{29}$$

Equation (28) then gives a prediction of the gradient at $w + \mu v$. To access the accuracy of the quadratic approximation we might wish to compare this with a gradient measurement taken at that point, or we might even preemptively take a step in that direction.

Divorced from the SCG algorithm, another application for this way of calculating μ is to eliminate the step size η of conventional gradient descent, which uses

$$w_{t+1} = w_t - \eta \nabla_w(w_t)$$

to gradually minimize E. Gradient descent suffers not only from a poor convergence rate, but also from the need to constantly tune η for rapid convergence as the minimization proceeds. The above simple line search suggests the use of $\eta = -\mu$ at each step, or

$$w_{t+1} = w_t - \frac{||\nabla_w(w_t)||^2}{\nabla_w(w_t)^T H \nabla_w(w_t)} \nabla_w(w_t). \tag{30}$$

The necessary modifications for gradient descent with momentum are trivial, as are the appropriate modifications for a stochastic gradient setting. Of course, this simple procedure needs to be augmented by mechanisms to check that $v^T H v > 0$, and that the quadratic assumption is not inaccurate enough to cause failure to reduce E at each step.

5.4 Eigenvalue Based Learning Rate Optimization for Stochastic Gradient Descent

The technique descibed in the previous section is, at least as stated, suitable only for deterministic gradient descent. Typically, for large systems, deterministic gradient descent is impractical, and only noisy estimates of the gradient are available. In joint work with colleagues at AT&T Bell Labs (le Cun et al., 1993), the approximation technique of equation (1) enabled H to be treated as a generalized sparse matrix, and properties of H were extracted in order to accelerate the convergence of stochastic gradient descent.

Information accumulated online, in particular eigenvalues and eigenvectors of the principle eigenspace, was used to linearly transform the weight space in such a way that the ill-conditioned off-axis long narrow valleys in weight space, which slow down gradient descent, become well-conditioned circular bowls. This work did not use an exact value for Hv, but rather a stochastic unbiased estimate of the Hessian based on just a single exemplar at a time. Computations of the form $x(t) = H(t)v$ were replaced with relaxations of the form $x(t) = (1-\alpha)x(t-1) + \alpha\widehat{H}(t)v$, where α is chosen to trade off steady-state noise against speed of convergence.

6 Summary and Conclusion

Second-order information about the error is of great practical and theoretical importance. It allows sophisticated optimization techniques to be applied, appears in many theories of generalization, and is used in sophisticated weight pruning procedures. Unfortunately, the Hessian matrix H, whose elements are the second derivative terms $\partial^2 E/\partial w_i \partial w_j$, is unwieldy. We have derived the $\mathcal{R}\{\cdot\}$ technique, which directly computes Hv, the product of the Hessian with a vector. The technique is

- *exact:* no approximations are made.
- *numerically accurate:* there is no drastic loss of precision.
- *efficient:* it takes about the same amount of computation as a gradient calculation.
- *flexible:* it applies to all current gradient calculation procedures.
- *robust:* if the gradient calculation gives an unbiased estimate of ∇_w then our procedure gives an analogous unbiased estimate of Hv.

Procedures that result from the applications of the $\mathcal{R}\{\cdot\}$ technique are about as local, parallel, and efficient as the original untransformed gradient calculation. The technique applies naturally to backpropagation networks, recurrent networks, relaxation networks, and stochastic Boltzmann Machines. To give an idea of its efficiency, we can compute H one column at a time by cycling v through the unit basis vectors. This is as efficient as the best algorithms to calculate the full Hessian of a backpropagation network, a well studied problem. Hopefully, application of this technique will facilitate the construction of efficient algorithms that make use of exact second-order information without calculating or storing the full Hessian.

Acknowledgments

I thank Yann le Cun and Patrice Simard for their encouragement and generosity. Without their work and enthusiasm I would not have derived the $\mathcal{R}\{\cdot\}$ technique or recognized its importance. Thanks also go to Nandakishore Kambhatla, John Moody, Akaysha Tang, Steve Rehfuss, and Andreas Weigend for their helpful comments and careful readings. This work was partially supported by grants NSF ECS-9114333 and ONR N00014-92-J-4062 to John Moody.

References

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann Machines. *Cognitive Science*, 9:147–169.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In (Caudill and Butler, 1987), pages 609–618.

Alspector, J., Meir, R., Yuhas, B., and Jayakumar, A. (1993). A parallel gradient descent method for learning in analog VLSI neural networks. In (Cowan and Giles, 1993). In press.

Becker, S. and le Cun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In Touretzky, D. S., Hinton, G. E., and Sejnowski, T. J., editors, *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann. Also published as Technical Report CRG-TR-88-5, Department of Computer Science, University of Toronto.

Bishop, C. (1992). Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation*, 4(4):494–501.

Buntine, W. and Weigend, A. (1991). Calculating second derivatives on feedforward networks. *IEEE Transactions on Neural Networks*. In submission.

Caudill, M. and Butler, C., editors (1987). *IEEE First International Conference on Neural Networks*, San Diego, CA.

Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimization. In (Cowan and Giles, 1993). In press.

Cowan, J. and Giles, L., editors (1993). *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann. In press.

Flower, B. and Jabri, M. (1993). Summed weight neuron perturbation: An $o(n)$ improvement over weight perturbation. In (Cowan and Giles, 1993). In press.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In (Cowan and Giles, 1993). In press.

Hinton, G. E. (1987). Connectionist learning procedures. Technical Report CMU-CS-87-115, Carnegie Mellon University, Pittsburgh, PA 15213.

Jabri, M. and Flower, B. (1991). Weight perturbation: An optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks. *Neural Computation*, 3(4):546–565.

Kirk, D., Kerns, D., Fleischer, K., and Barr, A. (1993). Analog VLSI implementation of gradient descent. In (Cowan and Giles, 1993). In press.

le Cun, Y., Denker, J., Solla, S., Howard, R. E., and Jackel, L. D. (1990). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann.

le Cun, Y., Kanter, I., and Solla, S. A. (1991). Second order properties of error surfaces: Learning time and generalization. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 918–924. Morgan Kaufmann.

le Cun, Y., Simard, P. Y., and Pearlmutter, B. A. (1993). Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors. In (Cowan and Giles, 1993). In press.

MacKay, D. J. C. (1991). A practical Bayesian framework for back-prop networks. *Neural Computation*, 4(3):448–472.

Møller, M. (1993a). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*. In press.

Møller, M. (1993b). Supervised learning on large redundant training sets. *International Journal of Neural Systems*. In press.

Moody, J. E. (1992). The *effective* number of parameters: an analysis of generalization and regularization in nonlinear learning systems. In (Moody et al., 1992), pages 847–854.

Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors (1992). *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann.

Movellan, J. R. and McClelland, J. L. (1991). Learning continuous probability distributions with the contrastive Hebbian algorithm. Technical Report PDP.CNS.91.2, Carnegie Mellon University Dept. of Psychology, Pittsburgh, PA.

Pearlmutter, B. A. (1992). Gradient descent: Second-order momentum and saturating error. In (Moody et al., 1992), pages 887–894.

Pineda, F. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 19(59):2229–2232.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Verrerling, W. T. (1988). *Numerical Recipes in C*. Cambridge University Press.

Skilling, J. (1989). The eigenvalues of mega-dimensional matrices. In Skilling, J., editor, *Maximum Entropy and Bayesian Methods*, pages 455–466. Kludwer Academic Publishers.

Watrous, R. (1987). Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization. In (Caudill and Butler, 1987), pages 619–627.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Widrow, B., McCool, J. M., Larimore, M. G., and Johnson, Jr., C. R. (1979). Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, 64:1151–1162.