Second-order backpropagation algorithms for a stagewise-partitioned separable Hessian matrix

Eiji Mizutani  
Department of Computer Science  
Tsing Hua University  
Hsinchuen, 300 Taiwan  
E-mail: eiji@wayne.cs.nthu.edu.tw

Stuart E. Dreyfus  
Industrial Engineering & Operations Research  
University of California, Berkeley  
Berkeley, CA 94720 USA  
E-mail: dreyfus@ieor.berkeley.edu

James W. Demmel  
Mathematics & Computer Science  
University of California, Berkeley  
Berkeley, CA 94720 USA  
E-mail: demmel@cs.berkeley.edu

Abstract—Recent advances in computer technology allow the implementation of some important methods that were assigned lower priority in the past due to their computational burdens. Second-order backpropagation (BP) is such a method that computes the exact Hessian matrix of a given objective function. We describe two algorithms for feed-forward neural-network (NN) learning with emphasis on how to organize Hessian elements into a so-called stagewise-partitioned block-arrow matrix form: (1) stagewise BP, an extension of the discrete-time optimal-control stagewise Newton of Dreyfus 1966; and (2) nodewise BP, based on direct implementation of the chain rule for differentiation attributable to Bishop 1992. The former, a more systematic and cost-efficient implementation in both memory and operation, progresses in the same layer-by-layer (i.e., stagewise) fashion as the widely-employed first-order BP computes the gradient vector. We also show intriguing separable structures of each block in the partitioned Hessian, disclosing the rank of blocks.

I. INTRODUCTION

In multi-stage optimal control problems, second-order optimization procedures (see [8] and references therein) proceed in a stagewise manner since the number of stages, is often very large. Naturally, those methods can be employed for optimizing multi-stage feed-forward neural networks: In this paper, we focus on an N-layered multilayer perceptron (MLP), which gives rise to an N-stage decision making problem. At each stage s, we assume there are P_s (s = 1, 2, ..., N) states (or nodes) and n_s (s = 1, 2, ..., N−1) decision parameters (or weights), denoted by an n_s-vector \( \theta^{s+1} \) (between layers \( s \) and \( s+1 \)). No decisions are to be made at terminal stage \( N \) (or layer \( N \)); hence, the \( N-1 \) decision stages in total. To compute the gradient vector for optimization purposes, we employ the “first-order” backpropagation (BP) process [5], [6], [7], which consists of two major procedures: forward pass and backprop pass [see later Eq. (2)]. A forward-pass situation in MLP-learning, where the node outputs in layer \( s-1 \) (denoted by \( y^{s-1} \)) affect the node outputs in the next layer \( s \) (i.e., \( y^s \)) via connection parameters (denoted by \( \theta^{s-1,s} \) between those two layers), can be interpreted as a situation in optimal control where state \( y^{s-1} \) at stage \( s-1 \) is moved to state \( y^s \) at the next stage \( s \) by decisions \( \theta^{s-1,s} \). In the backward pass, sensitivities of the objective function \( E \) with respect to states (i.e., node sensitivities) are propagated from one stage back to another while computing gradients and Hessian elements. However, MLPs exhibit a great deal of structure, which turns out to be a very special case in optimal control; for instance, the “afternode” outputs (or states) are evaluated individually at each stage as \( y^s_j = f^s_j(x^s_j) \), where \( f(\cdot) \) denotes a differentiable state-transition function of nonlinear dynamics, and \( x^s_j \), the “before-node” net input to node \( j \) at layer \( s \), depends on only a subset of all decisions taken at stage \( s-1 \). In spite of this distinction and others, using a vector of states as a basic ingredient allows us to adopt analogous formulas available in the optimal control theory (see [8]). The key concept behind the theory resides in stagewise implementation; in fact, first-order BP is essentially a simplified stagewise optimal-control gradient formula developed in early 1960s [6]. We first review the important “stagewise concept” of first-order BP, and then advance to stagewise second-order BP with particular emphasis on our organization of Hessian elements into a stagewise-partitioned block-arrow Hessian matrix form.

II. STAGEWISE FIRST-ORDER BACKPROPAGATION

The backward pass in MLP-learning starts evaluating the so-called terminal after-node sensitivities (also known as costates or multipliers in optimal control) \( \xi^N = \frac{\partial E}{\partial y^N} \) (defined as partial derivatives of an objective function \( E \) with respect to \( y^N \), the output of node \( k \) at layer \( N \) for \( k = 1, ..., P_N \), yielding a \( P_N \)-vector \( \xi^N \)). Then, at each node \( k \), the after-node sensitivity is transformed into the before-node sensitivity (called delta in ref. [5]; see pages 325–326) \( \delta_h^k = \frac{\partial E}{\partial h^k} \) (defined as partial derivatives of \( E \) with respect to \( x_N^k \), the before-node “net input” to node \( k \)) by multiplying by node-function derivatives as \( \delta_h^k = f^N_k(x_N^k)\xi^N_k \). The well-known stagewise first-order BP (i.e., generalized delta rule; see Eq.(14), p.326 in [5]) for intermediate stage \( s = 2, ..., N-1 \) can be written out with \( \delta \) or \( \xi \) as the recurrence relation below

\[
\begin{align*}
\delta^s_{P_x} & \equiv \frac{\partial E}{\partial y^s} \\
& = \sum_{P_x} \theta^{s+1}_{P_x} \frac{\partial E}{\partial h^s} \\
& = \sum_{P_x} \theta^s_{P_x} \frac{\partial E}{\partial h^s} \\
& = \sum_{P_x} \left[ \frac{\partial E}{\partial h^s} \right]_{\theta^{s+1}_{P_x}} \delta^s_{P_x}, \\
& = \sum_{P_x} \left[ \frac{\partial E}{\partial h^s} \right]_{\theta^{s+1}_{P_x}} \delta^s_{P_x}, \\

\delta^s_{P_x} & = \frac{\partial E}{\partial y^s} \\
& = \sum_{P_x} \theta^{s+1}_{P_x} \frac{\partial E}{\partial h^s} \\
& = \sum_{P_x} \theta^s_{P_x} \frac{\partial E}{\partial h^s} \\
& = \sum_{P_x} \left[ \frac{\partial E}{\partial h^s} \right]_{\theta^{s+1}_{P_x}} \delta^s_{P_x}.
\end{align*}
\]

(1)

where \( E \) is a given certain objective function (to be minimized), and two \( P_{s+1} \)-by-\( P_s \) matrices, \( N^{s+1}_{s+1} \) and \( W^{s+1}_{s+1} \), are defined as \( N^{s+1}_{s+1} \equiv \frac{\partial E}{\partial \theta^{s+1}_{P_x}} \) and \( W^{s+1}_{s+1} \equiv \frac{\partial E}{\partial \theta^{s+1}_{P_x}} \).
These two are called before-node and after-node sensitivities, respectively, for they translate the node-sponsored transition matrices, respectively, from one stage to another; e.g., we can readily verify $\delta_{s+1} = N_{s+1}^{-1} s_{s+1}^T \delta_{s+1} = N_{s+1}^{-1} s_{s+1}^T \delta_{s+1}$. Note that those two forms of sensitivity vectors become identical when node functions $f(.)$ are linear identity functions usually employed only at terminal layer $N$ in MLP-learning.

The forward and backwards passes in first-order stagewise BP for the standard MLP-learning can be summarized below:

**Forward pass:**

\[
\begin{cases}
\hat{y}^{s+1}_{s+1} = y_{s+1}^{s+1} \delta_{s+1}^T \\
\delta_{s+1} = \delta_{s+1}^T \theta_{s+1}^{s+1} \\
\delta_{s} = \delta_{s}^T \theta_{s}^{s+1}
\end{cases}
\]  

(2)

**Backward pass:**

\[
\begin{cases}
\hat{y}^{s+1}_{s+1} = y_{s+1}^{s+1} \delta_{s+1}^T \\
\delta_{s+1} = \delta_{s+1}^T \theta_{s+1}^{s+1} \\
\delta_{s} = \delta_{s}^T \theta_{s}^{s+1}
\end{cases}
\]  

(2)

Here, $y_{s+1}^{s+1}$ (with subscript + on $y$) includes a scalar constant output $y^{s+1}_0$ of a bias node (denoted by node 0) at layer $s$ leading to $y_{s+1}^{s+1} = [y_{s+1}^{s+1}, y_{s+1}^{s+1}]$, a $(1 + P_s)$-vector of outputs at layer $s$; $\theta_{s+1}^{s+1}$ is a $P_{s+1}$-vector of the parameters linked to node $i$ at layer $s$; $\theta_{s}^{s+1}$ is a $(1 + P_s)$-vector of the parameters linked to node $s$ at layer $s + 1$ (including a threshold parameter linked to bias node 0 at layer $s$); $\Theta_{s+1}^{s+1}$ in forward pass, a $P_{s+1}$-by-$(1 + P_s)$ matrix of parameters between layers $s$ and $s + 1$, includes the $P_{s+1}$ threshold parameters (i.e., the $P_{s+1}$-vector $\theta_{s+1}^{s+1}$ linked to bias node 0 at layer $s$ in the first column, whereas $\Theta_{s}^{s+1}$ in backward pass excludes the threshold parameters. Note that a matrix can always be reshaped into a vector for our convenience; for instance, $\Theta_{s+1}^{s+1}$ can be reshaped to $\theta_{s+1}^{s+1}$, an $n_s$-vector of $\Theta_{s+1}^{s+1}$, as shown next:

\[
\Theta_{s+1}^{s+1} = \begin{bmatrix}
\theta_{s+1,1}^{s+1} & \theta_{s+1,2}^{s+1} & \cdots & \theta_{s+1,n_s}^{s+1}
\end{bmatrix}
\]

Reshape

where scalar $\theta_{s+1,1}^{s+1}$ denotes a parameter between node $i$ at layer $s$ and node $k$ at layer $s + 1$. At each stage $s$, the $n_s$-length gradient vector associated with $\theta_{s+1}^{s+1}$ can be written as

\[
\xi_{s+1}^{s+1} = \left[ \frac{\partial y_{s+1}^{s+1}}{\partial \theta_{s+1}^{s+1}} \right]^T \delta_{s+1}^{s+1} = \left[ \frac{\partial \Theta_{s+1}^{s+1}}{\partial \theta_{s+1}^{s+1}} \right]^T \delta_{s+1}^{s+1},
\]

(4)

where the transposed matrices are sparse in a block-diagonal form; for instance, see $\frac{\partial y_{s+1}^{s+1}}{\partial \theta_{s+1}^{s+1}}$ later in Eqs. (25) and (26). Yet, the stagewise computation by first-order BP can be viewed in such a way that the gradients are efficiently computed (without forming such sparse block-diagonal matrices explicitly) by the outer product $\delta_{s+1}^{s+1} y_{s+1}^{s+1}$, which produces a $P_{s+1}$-by-$(1 + P_s)$ matrix $G_{s+1}^{s+1}$ of gradients [7] associated with the same-sized matrix $\Theta_{s+1}^{s+1}$ of parameters; here, column $i$ of $G_{s+1}^{s+1}$ is given as a $P_{s+1}$-vector $g_{s+1}^{s+1}$ for $\theta_{s+1}^{s+1}$. Again, the resulting gradient matrix $G_{s+1}^{s+1}$ can be reshaped to an $n_s$-length gradient vector $g_{s+1}^{s+1}$ in the same manner as shown in Eq. (3).

Furthermore, the before-node sensitivity vector $\delta_{s+1}^{s+1}$ (used to get $g_{s+1}^{s+1}$ in the outer-product operation) is backpropagated by $\xi_{s+1}^{s+1} = \Theta_{s+1}^{s+1} \delta_{s+1}^{s+1}$, as shown in Eq. (2), which makes stagewise “first-order” BP forms neither $N$ nor $W$ explicitly.

Such matrices as $N_{s+1}$ for adjacent layers also play an important role as a vehicle for bucket-brigading second-order information [see later Eqs. (10) and (11)] necessary to obtain the Hessian matrix $H$. Stagewise second-order BP computes one block after another in the stagewise-partitioned $H$ without forming $N_{s+1}$ explicitly in the same way as stagewise first-order BP, which we shall describe next.

III. THE STAGEWISE-PARTITIONED HESSIAN MATRIX

Given an $N$-layered MLP, let the total number of parameters be denoted by $n = \sum_{s=1}^{N-1} n_s$, and let each $n_s$-$by$-$n_t$ block include Hessian elements with respect to pairs of one parameter at stage $s$ (in the space of $n_s$ parameters $\theta_{s+1}^{s+1}$ between layers $s$ and $s+1$) and another parameter at stage $t$ (in the space of $n_t$ parameters $\theta_{s+1}^{s+1}$ between layers $t$ and $t+1$). Then, the $n$-$by$-$n$ symmetric Hessian matrix $H$ of a certain objective function $E$ can be represented as a partitioned form among $N$ layers (i.e., $N-1$ decision stages) in such a stagewise format as shown next:

\[
\begin{bmatrix}
H_{N-1}^1 & H_{N-2}^1 & \cdots & H_{1}^1
\end{bmatrix}
\]

By symmetry, we need to form only the lower (or upper) triangular part of $H$; totally $\frac{N(N-1)}{2}$ blocks including $n_s$-$by$-$n_t$ rectangular “off-diagonal” blocks $H_{s,t}^{s+1}(1 \leq s \leq t \leq N-1)$ as well as $N-1$ symmetric $n_s$-$by$-$n_s$ “diagonal” blocks $H_{s,s}^{s+1}(1 \leq s \leq N-1)$, of which we need only the lower half.

A. Stagewise second-order backpropagation

Stagewise second-order BP computes the entire Hessian matrix by one forward pass followed by backward processes per training datum in a stagewise block-by-block fashion. The Hessian blocks are computed from stage $N-1$ in a stagewise manner in the order of

\[
\begin{bmatrix}
H_{N-1}^1 & H_{N-2}^1 & \cdots & H_{1}^1
\end{bmatrix}
\]

Stage $N-1$

Stage $N-2$

Stage $N-3$

Stage $N$

Stage $1$
In what follows, we describe algorithmic details step by step:

**Algorithm: Stagewise second-order BP** (per training datum).

(i) **(Step 0)** Do forward pass from stage 1 to \( N \) to obtain node outputs, and evaluate the objective function value \( E \).

(ii) **(Step 1)** At terminal stage \( N \), compute \( \xi^N = \left[ \frac{\partial E}{\partial y^N} \right] \), the \( P_N \)-length *after-node* sensitivity vector (defined in Sec. II), and

\[
Z^N_{P_N \times P_N} = \begin{bmatrix} \frac{\partial^2 E}{\partial y^N \partial y^N} & \frac{\partial E}{\partial y^N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 y^N}{\partial x^N \partial x^N} & \frac{\partial y^N}{\partial x^N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 y^N}{\partial x^N \partial x^N} & \frac{\partial y^N}{\partial x^N} \end{bmatrix} \xi^N. \tag{6}
\]

The \((i, j)\)-element of the last symmetric matrix is obtainable from the following special \( \langle \cdot, \cdot \rangle \)-operation (set \( s = N \) below):

\[
\left\langle \frac{\partial^2 y^N}{\partial x^N \partial x^N} \right\rangle_{ij} = \sum_{k=1}^{P_s} \sum_{j=1}^P \sum_{i=1}^P \xi^N_k \frac{\partial^2 y^N}{\partial x^N \partial x^N}, \tag{7}
\]

which is just a diagonal matrix in standard MLP-learning.

- Repeat the following Steps 2 to 6, starting at stage \( s = N - 1 \):

- **(Step 2)** Obtain the diagonal Hessian block at stage \( s \) by

\[
H_{n_x \times n_x}^{s+1} = \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{8}
\]

where \( F_x^{s+1, u} \) is not needed initially when \( s = N - 1 \); hence, defined later in Eq. (11).

- **(Step 3)** Only when \( 2 \leq N - s \) holds, obtain \( (N - s - 1) \)-off-diagonal Hessian blocks by

\[
H_{n_x \times n_x}^{s+1} = \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{9}
\]

where \( F_x^{s+1, u} \) is not needed initially when \( s = N - 1 \); hence, defined later in Eq. (11).

- **(Step 4)** When \( 2 \leq N - s \) holds, update previously-computed rectangular matrices \( F_x^{s+1, u} \) for the next stage by:

\[
F_x^{s+1, u} \leftarrow \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{10}
\]

- **(Step 5)** Compute a new \( P_s \)-by-\( n_s \) rectangular matrix \( F_x^{s, u} \) at the current stage \( s \) by

\[
F_x^{s, u}_{P_s \times n_s} = \begin{bmatrix} \theta_x^{s+1} \end{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{11}
\]

- **(Step 6)** Compute a \( P_s \)-by-\( P_s \) matrix \( Z^{s+1} \) by

\[
Z^{s+1} = \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{12}
\]

where the last matrix is obtainable from the \( \langle \cdot, \cdot \rangle \)-operation defined in Eq. (7).

- Go back to Step 2 by setting \( s = s - 1 \). *(End of Algorithm)*

**Remarks:** The \( \langle \cdot, \cdot \rangle \)-operation defined in Eq. (12) yields a matrix of only first derivatives below:

\[
\begin{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} \end{bmatrix} \frac{\partial h_{s+1}}{\partial \theta_{s+1}^{(1)}} + \begin{bmatrix} \frac{\partial^2 h_{s+1}}{\partial \theta_{s+1}^{(1)} \partial \theta_{s+1}^{(1)}} \end{bmatrix} \tag{13}
\]

Here, the \( n_s \)-column space of the resulting \( P_s \)-by-\( n_s \) matrix has totally \( P_{s+1} \) partitions, each of which consists of \( (1 + P_s) \) columns since \( n_s = (1 + P_s)P_{s+1} \), and each partition has a \( P_s \) vector of zeros, denoted by \( 0 \), in the first column. The posed sparsity is tied to particular applications like MLP-learning.

### B. Nodewise second-order backpropagation

In the NN literature, the best-known second-order BP is probably Bishop’s method [1], [3], where for every node individually one must run a forward pass to the terminal output layer followed by a backward pass back to the node to get information necessary for Hessian elements; here, that node is one of the variables differentiated with respect to [for seeking node sensitivity in Eq. (1)]. This is what we call nodewise **BP**, a nodewise implementation of the chain rule for differentiation, which yields a Hessian element below with respect to two parameters \( \theta_{s,j}^{u} \) and \( \theta_{s,k}^{u} \) for \( 1 < s \leq u \leq N \) using \( n_{s-1}^{u,v} \) and \( z_{s,l}^{u,v} \) [cf. Eqs. (6) and (13)]:

\[
\frac{\partial^2 E}{\partial \theta_{s,j}^{u} \partial \theta_{s,k}^{v}} = y_i^{1} \frac{\partial y_i^{v}}{\partial \theta_{s,j}^{u}} \frac{\partial \theta_{s,j}^{u}}{\partial \theta_{s,k}^{v}} + y_i^{1} \frac{\partial y_i^{v}}{\partial \theta_{s,j}^{u}} \frac{\partial \theta_{s,j}^{u}}{\partial \theta_{s,k}^{v}} - y_i^{1} \frac{\partial y_i^{1}}{\partial \theta_{s,k}^{v}} \frac{\partial \theta_{s,k}^{v}}{\partial \theta_{s,j}^{u}} \tag{14}
\]

This is Eq. (4.71), p. 155 in [2] rewritten with stages introduced and denoted by superscripts, and \( n_{s,k}^{u-1} \) is the \((k, j)\)-element of \( P_{u-1} \)-by-\( P_s \) matrix \( N_{s-1}^{u-1} \) in Eq. (1). The basic idea of Bishop’s nodewise BP is as follows: Compute all the necessary quantities: \( \delta_i^u \) by stagewise first-order BP, \( n_{s,k}^{u-1} \) by forward pass, and \( z_{s,j}^{u-1} \) by backward pass in advance; then,
use Eq. (14) to evaluate Hessian elements. Unfortunately, this nodewise implementation of chain rules (14) does not exploit 
stagewise structure unlike first-order BP (see Section II); in addition, it has no implication about how to organize Hessian elements into a stagewise-partitioned “block-arrow” Hessian matrix [see Eq. (5) and Fig. 1]: To this end, it would be of much greater value to rewrite the nodewise algorithm posed by Bishop (outlined on p. 157 in [2]) in matrix form below.

**Algorithm: Nodewise second-order BP** (per training datum).

(Step 0) Do forward pass from stage 1 to stage $N$.

(Step 1) Initialize $N^{s,0} = I$ (identity matrix) and $N^{u,s} = 0$ (matrix of zeros) for $1 < s < u \leq N$ (see pages 155 & 156 in [2] for this particular initialization), and then do forward pass to obtain a $P_t$-by-$P_t$ non-zero dense matrix $N^{s,t}$ (for $s < t; s = 2, \ldots, N - 1$) by the following computation:

$$n^{s,t}_{j,l} = \sum_{k=1}^{P_{t-1}} f_k(x^{t-1}_k)\theta_{k,l}^{t-1} n^{s,t-1}_{j,k} \quad \Leftarrow \quad N^{s,t} = N^{s,t-1} \odot N^{s,t-1} \quad \text{(15)}$$

(Step 2) At terminal stage $N$, compute $\delta^N = [\partial E / \partial x^N_k]$, the $P_N$-length before-node sensitivity vector, and matrix $Z^N$ [defined in Eq. (6)], and then obtain the following for $2 \leq s \leq N$:

$$z^{s,N}_{j,l} = \sum_{m=1}^{P_N} P^{s,N}_{j,m} \left( \frac{\partial^2 E}{\partial x^t_j \partial x^m_l} \right) \Leftrightarrow Z^{s,N} = N^{s,N} \cdot Z^N \quad \text{(16)}$$

(Step 3) Compute $\delta^s$ using first-order BP: $\delta^{s+1}_{j,l} = \sum_{l=1}^{P_{s+1}} P_{s+1}^{l} \delta^{s+1}_{l} + f_k(x^{s+1}_k) \sum_{l=1}^{P_{s+1}} \theta_{k,l}^{s+1} \delta^{s+1}_{j,l}$, and obtain the next for $1 < s < t < N$:

$$z^{s,t}_{j,k} = n^{s,t}_{j,k} f_k(x^{t}_k) \sum_{l=1}^{P_{s+1}} \theta_{k,l}^{t} \delta^{s+1}_{l} + f_k(x^{t}_k) \sum_{l=1}^{P_{s+1}} \theta_{k,l}^{t} \delta^{s+1}_{j,l} \quad \Leftrightarrow \quad Z^{s,t} = N^{s,t} \cdot \begin{bmatrix} \frac{\partial^2 y}{\partial x^t_j \partial x^s_l} & \epsilon^t \end{bmatrix} + Z^{s+1,t} \quad \text{(17)}$$

(Step 4) Evaluate the Hessian blocks by Eq. (14) in matrix form for $1 < s < u \leq N$:

$$H^{s,u}_{n_{s-1} \times n_{u-1}} = \left[ \begin{array}{c} \frac{\partial^2 y}{\partial x^s_j \partial x^u_k} \\ \frac{\partial^2 y}{\partial x^s_j \partial x^u_k} \end{array} \right] \begin{bmatrix} P^s_{n_{s-1}} & P^u_{n_{u-1}} \\ P^u_{n_{u-1}} & P^s_{n_{s-1}} \end{bmatrix} \delta^s + Z^{u,u}_{n_{u-1} \times n_{u-1}} \frac{\partial^2 u}{\partial x^s_j \partial x^u_k} = \begin{bmatrix} I_{P_s \times P_s} & 0_{P_s \times P_u} \\ 0_{P_u \times P_s} & I_{P_u \times P_u} \end{bmatrix} \delta^s + Z^{u,u}_{n_{u-1} \times n_{u-1}} \frac{\partial^2 u}{\partial x^s_j \partial x^u_k} \quad \text{(18)}$$

where Eq. (12) is used for evaluating a $\langle \cdot , \cdot \rangle$-term. $\circ (End)$$\circ$

**Remarks:** Eqs. (15), (16), and (17) correspond to Eqs.(4.75), (4.79), and (4.78), respectively, on pages 155 & 156 in ref. [2].

**IV. TWO HIDDEN-LAYER MLP LEARNING**

In optimal control, $N$, the number of stages, is arbitrarily large. In MLP-learning, however, use of merely one or two hidden layers is by far the most popular at this stage. For this reason, we consider standard two-hidden-layer MLP-learning. This is a four-stage ($N=4$; three decision stages plus a terminal stage) problem, in which the total parameters (or decision variables) is given as:

$$n = n_3 + n_2 + n_1 = P_4(1 + P_3) + P_3(1 + P_2) + P_2(1 + P_1) \quad \text{including threshold parameters.}$$

In this setting, we have a three-block by three-block stagewise symmetric Hessian matrix $H$ in a nine-block partitioned form below as well as a three-block-partitioned gradient vector $g$ defined in Eq. (4):

$$H = \begin{bmatrix} H^{3,3} & H^{3,2} & H^{3,1} \\ H^{2,3} & H^{2,2} & H^{2,1} \\ H^{1,3} & H^{1,2} & H^{1,1} \end{bmatrix}, \quad g = \begin{bmatrix} g^{1,2} \\ g^{2,3} \\ g^{3,4} \end{bmatrix}. \quad \text{(19)}$$

Here, we need to form three off-diagonal blocks and only the lower (or upper) triangular part of three diagonal blocks; totally, six blocks $H^{s,t}$ ($1 \leq s \leq t \leq 3$). Each block $H^{s,t}$ includes Hessian elements with respect to pairs of one parameter at stage $s$ and another at stage $t$.

A. Algorithmic behaviors

We describe how our version of nodewise second-order BP algorithm in Section III-B works:

(Step 1): By initialization, set $N^{4,4} = I$, $N^{3,3} = I$, $N^{2,2} = I$, $N^{1,1} = 0$, $N^{1,2} = 0$, and $N^{3,2} = 0$. By forward pass in Eq. (15), get three dense blocks: $N^{3,3}, N^{2,2},$ and $N^{2,3}, N^{3,4}$. (Step 2): Get $Z^4$ by Eq. (6) and $Z^{4,4} = N^{4,4}Z^4$ by Eq. (16); similarly, obtain $Z^{3,4}$ and $Z^{4,3}$ as well.

(Step 3): Use Eq. (17) to get $Z^{3,3}, Z^{2,3},$ and $Z^{2,2}$; for instance, by $Z^{3,3} = N^{3,3} \cdot \begin{bmatrix} \frac{\partial^2 y}{\partial x^3_j \partial x^3_k} \end{bmatrix} \epsilon^3 + Z^{4,4}N^{4,4}$. (Step 4): Use Eq. (18) [i.e., Eq. (14)] to obtain the desired six Hessian blocks.

All those nine $N$ blocks can be pictured in an augmented “upper triangular” before-node sensitivity transition matrix $\tilde{N}$ defined below together with $\tilde{x}$, a $P$-dimensional augmented vector, which consists of all the before-node net-inputs per datum at three layers except the first input layer ($N = 1$) because $x^1$ is a fixed vector of given inputs; hence, $\tilde{P} = P_1 + P_2 + P_3$:

$$N_{\tilde{P} \times \tilde{P}} \sigma \begin{bmatrix} \frac{\partial y}{\partial x^{\tilde{P}}} \\ \frac{\partial y}{\partial x^{\tilde{P}}} \end{bmatrix} = \begin{bmatrix} I_{P_1 \times P_1} & 0_{P_1 \times P_2} & 0_{P_1 \times P_3} \\ 0_{P_2 \times P_1} & I_{P_2 \times P_2} & 0_{P_2 \times P_3} \\ 0_{P_3 \times P_1} & 0_{P_3 \times P_2} & I_{P_3 \times P_3} \end{bmatrix} \delta^s + \begin{bmatrix} \tilde{x}^T \\ \tilde{x} \end{bmatrix} \quad \text{(20)}$$

Here, three diagonal identity blocks $I$ correspond to $N^{4,4}, N^{3,3}$, and $N^{2,2}$. At first glance, Bishop’s nodewise BP relies on using $\tilde{N}$ explicitly, requiring $N^{s,1}$ even for non-adjacent layers ($s + 1 < t$) as well as identity blocks $N^{s,s}$ and zero blocks. For adjacent blocks $N^{s,s+1}$, Eq. (15) just implies multiply by an identity matrix; hence, no need to use it in reality. Likewise, at Step 2, $Z^{4,4} = Z^4$ due to $N^{4,4} = I$. Furthermore, in (Eq. 18), $N^{3,3} = 0$ and $N^{3,2} = 0$ (matrices of zeros) are used when diagonal blocks $H^{s,s}$ are evaluated (but $N^{3,2} = 0$ is not needed at all). In this way, nodewise BP yields Hessian blocks by Eq. (18), a matrix form of Eq. (14), as long as $\tilde{N}$ in Eq. (20) is obtained correctly in advance by forward pass at Step 1 (according to pp.155–156 in [2]); yet, it is not very efficient to work on such zero entries and multiply by one.
On the other hand, stagewise second-order BP evaluates $N_{s+1}^{s+1}$ implicitly only for adjacent layers during the backward process (not by forward pass) essentially in the same manner as stagewise first-order BP does with no $N_{s+1}^{s+1}$ blocks required explicitly, and thus avoids operating on such zeros and ones [for Eq. (20)]. For off-diagonal Hessian blocks $H^{s,n}$ \((s < n)\), the parenthesized terms in Eq. (18) become the rectangular matrix $F^{s,n-1}$ in Eq. (11). That is, stagewise BP splits Eq. (18) into Eqs. (8) and (9) by exploitation of the stagewise MLP structure.  

B. Separable Hessian Structures

We next show the Hessian-block structures to be separable into several portions. Among the six distinct blocks in Eq. (19), due to space limitation we display below three Hessian blocks: two diagonal blocks and one off-diagonal block alone.

\[
H^{3,3}_{n_3 \times n_3} = \left[ \begin{array}{cccc} \frac{\partial^2 y^s}{\partial x^4 \partial x^3} & \vdots & \frac{\partial^2 y^s}{\partial x^3 \partial x^1} & \frac{\partial^2 y^s}{\partial x^4 \partial x^1} \\
\vdots & \ddots & \vdots & \vdots \\
\frac{\partial^2 y^s}{\partial x^3 \partial x^1} & \vdots & \frac{\partial^2 y^s}{\partial x^3 \partial x^2} & \frac{\partial^2 y^s}{\partial x^3 \partial x^1} \\
\frac{\partial^2 y^s}{\partial x^4 \partial x^1} & \vdots & \frac{\partial^2 y^s}{\partial x^4 \partial x^2} & \frac{\partial^2 y^s}{\partial x^4 \partial x^1} \end{array} \right] \\
+ \frac{\partial^2 y^s}{\partial x^3 \partial x^1} \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right] \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right]^T \frac{\partial^2 y^s}{\partial x^4 \partial x^1} \\
+ \frac{\partial^2 y^s}{\partial x^3 \partial x^1} \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right] \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right]^T \frac{\partial^2 y^s}{\partial x^4 \partial x^1} \\
+ \frac{\partial^2 y^s}{\partial x^3 \partial x^1} \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right] \left[ \begin{array}{c} \xi_4 \\
\vdots \\
\xi_2 \\
\xi_1 \end{array} \right]^T \frac{\partial^2 y^s}{\partial x^4 \partial x^1} \end{array} \right]
\]

Eq. (21) [i.e., $H^{3,3}_{n_3 \times n_3}$ placed at the upper-left corner in Eq. (5)] always becomes block-diagonal (with $P_4$ sub-blocks $\times$ below):

\[
H^{3,3}_{n_3 \times n_3} = \begin{bmatrix} x_1 & \cdots & x_{P_4} \\
\vdots & \ddots & \vdots \\
x_{P_3} & \cdots & x_{P_4} \end{bmatrix},
\]

where $x_i$ denotes a $C_A$-by-$C_A$ dense symmetric sub-block.

Furthermore, the last term in $H^{1,2}$ is a sparse matrix of only first derivatives due to Eq. (12); in the next section, we shall explain this finding in nonlinear least squares learning.

C. Neural Networks Nonlinear Least Squares Learning

When our objective function $E$ is the sum over all the $d$ training data of squared residuals, we have $E(\theta) = \frac{1}{2}J^T J$, where $r = y^T(\theta) - t$; in words, an m-vector $r$ of residuals is the difference between an $m$-vector $t$ of the desired outputs and an $m$-vector $y^s$ of the terminal outputs of a two hidden-layer MLP (with $N = 4$), and $m \equiv P_a d (P_a > 1$, or multiple terminal outputs in general). The gradient vector of $E$ is given by $g = J^T r$; here, $J$ denotes the $m$-by-$n$ jacobian matrix of the residual vector $r$, which is $J$ of $y^s$ because $t$ is independent of $\theta$ by assumption. As shown in Eqs. (19)(right) and (4), $g$ is stagewise-partitioned as: $g^{s+1} = \left[ \frac{\partial y^{s+1}}{\partial \theta^{s+1}} \right] \xi^{s+1}$ for $s = 1, \ldots, 3$, where $\xi^s = r$. Likewise, $J$ can be given in stagewise column-partitioned form below in Eq. (25), or equivalently in block-angular form below in Eq. (26) [with $n_B \equiv n - n_3 = n_1 + n_2$]:

\[
J_{m \times n} = \begin{bmatrix} \frac{\partial y^1}{\partial \theta^1} & \cdots & \frac{\partial y^1}{\partial \theta^{n_2}} \\
\vdots & \ddots & \vdots \\
\frac{\partial y^2}{\partial \theta^1} & \cdots & \frac{\partial y^2}{\partial \theta^{n_2}} \\
\vdots & \ddots & \vdots \\
\frac{\partial y^3}{\partial \theta^1} & \cdots & \frac{\partial y^3}{\partial \theta^{n_2}} \end{bmatrix}_{m \times n_2}
\]

where $A_k$ is $d$-by-$C_A$ \((k = 1, \ldots, P_a)\) and $B_k$ $d$-by-$n_B$. The block-angular form is due to the same reason as Eq. (24); i.e., only $C_A$ parameters affect each terminal residual. Since $J$ has the block-angular form in Eq. (26), its cross-product matrix $J^T J$ has a so-called block-arrow form due to its appearance, as illustrated in Fig. 1, where $H = J^T J$ and $H^{1,3}$ in Eqs (21) and (24) consists of $P_4$ diagonal blocks $A_k^T A_k$ for $k = 1, \ldots, P_4$. If the terminal node functions are the linear
form because the right-front rectangular panel depicts the transposed block-angular residual identity function, then all the diagonal blocks learning, the full Hessian here, the lower-right block of the Hessian is: matrix $J$ as derivatives (called the Hessian can be obtained from Eq. (12)).

On the other hand, such nice sparsity may disappear when with its arrow-head pointing downwards to the sparsity with its arrow-head pointing downwards to the systemic sparsity $\mathbf{H}$ is still worth exploiting sparsity of weight-sharing and weight-pruning are applied (as usual in optimal control [8]) so that all the terminal parameters $\theta^{N-1}$ are shared among the terminal states $y^N$. In this way, MLP-learning exhibits a great deal of structure.

For the parameter optimization, we recommend trust-region globalization, which works even if $\mathbf{H}$ is indefinite [10], [9]. In large-scale problems, where $\mathbf{H}$ may not be needed explicitly, we could use sparse Hessian-matrix-vector multiply (e.g., [11]) to construct Krylov subspaces for optimization purposes, but it is still worth exploiting sparsity of $\mathbf{H}$ for pre-conditioning [10]. In this context, it is not recommendable to compute (or approximate) the inverse matrix of (sparse) block-arrow $\mathbf{H}$ (see Fig. 1) because it always becomes dense.

Our matrix-based algorithms revealed that blocks in the stagewise-partitioned $\mathbf{H}$ are separable into several distinct portions, and disclosed that sparse matrices of only first derivatives [see Eq. (27)] can be further identified. Furthermore, by inspection of the common matrix terms in block [e.g., see Eqs. (21) to (23)], we see that the Hessian part computed on each datum at stage $s$, which consists of blocks $\mathbf{H}^{s+1}$ (1 $\leq$ $s$ $\leq$ $N-1$), is at most rank $P_{s+1}$, where $P_{s+1}$ denotes the number of nodes at layer $s+1$. We plan to report in another opportunity more on those findings as well as the practical implementation issues of stagewise-second-order BP, for which the matrix recursive formulas may allow us to take advantage of level-3 BLAS (Basic Linear Algebra Subprograms; see http://www.netlib.org/blas/).

REFERENCES


