Supplementary Materials for “Distributed Newton Methods for Deep Learning”

Chien-Chih Wang,¹ Kent Loong Tan,¹ Chun-Ting Chen,¹ Yu-Hsiang Lin,² S. Sathiya Keerthi,³ Dhruv Mahajan,⁴ S. Sundararajan,³ Chih-Jen Lin¹
¹Department of Computer Science, National Taiwan University, Taipei 10617, Taiwan
²Department of Physics, National Taiwan University, Taipei 10617, Taiwan
³Microsoft
⁴Facebook Research

I List of Symbols

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y^i$</td>
<td>The label vector of the $i$th training instance.</td>
</tr>
<tr>
<td>$x^i$</td>
<td>The feature vector of the $i$th training instance.</td>
</tr>
<tr>
<td>$l$</td>
<td>The number of training instances.</td>
</tr>
<tr>
<td>$K$</td>
<td>The number of classes.</td>
</tr>
<tr>
<td>$\theta$</td>
<td>The model vector (weights and biases) of the neural network.</td>
</tr>
<tr>
<td>$\xi$</td>
<td>The loss function of a single instance.</td>
</tr>
<tr>
<td>$\xi_i$</td>
<td>The loss function of the $i$th instance.</td>
</tr>
<tr>
<td>$f$</td>
<td>The objective function.</td>
</tr>
<tr>
<td>$C$</td>
<td>The regularization parameter.</td>
</tr>
<tr>
<td>$L$</td>
<td>The number of layers of the neural network.</td>
</tr>
<tr>
<td>$n_m$</td>
<td>The number of neurons in the $m$th layer.</td>
</tr>
<tr>
<td>$n_0$</td>
<td>The number of input neurons (the dimension of the feature vector).</td>
</tr>
<tr>
<td>$n_L$</td>
<td>The number of output neurons (the number of classes, except for binary classification one may use $n_L = 1$).</td>
</tr>
<tr>
<td>$W^m$</td>
<td>The weight matrix in the $m$th layer (with dimension $\mathbb{R}^{n_{m-1} \times n_m}$).</td>
</tr>
<tr>
<td>$w^m_{tj}$</td>
<td>The weight between neuron $t$ in the ($m-1$)th layer and neuron $j$ in the $m$th layer.</td>
</tr>
<tr>
<td>$w^m$</td>
<td>The vector obtained by concatenating the columns of $W^m$.</td>
</tr>
<tr>
<td>$b^m$</td>
<td>The bias vector in the $m$th layer.</td>
</tr>
<tr>
<td>$s^{m,i}$</td>
<td>The affine function $(W^m)^T z^{m-1,i} + b^m$ in the $m$th layer for the $i$th instance.</td>
</tr>
<tr>
<td>$z^{m,i}$</td>
<td>The output vector (element-wise application of the activation function on $s^{m,i}$) in the $m$th layer for the $i$th instance.</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>The activation function.</td>
</tr>
<tr>
<td>$n$</td>
<td>The total number of weights and biases.</td>
</tr>
<tr>
<td>$J^i$</td>
<td>The Jacobian matrix of $z^{L,i}$ with respect to $\theta$.</td>
</tr>
<tr>
<td>$J_p^i$</td>
<td>The local component of the $J^i$ in the partition $p$.</td>
</tr>
</tbody>
</table>
Notation | Description
--- | ---
$B^i$ | The Hessian matrix of the loss function of the \( i \)-th instance with respect to \( z_{L,i} \) (the matrix element is \( B_{is} = \frac{\partial^2 \xi(z_{L,i}; y^i)}{\partial z^L_{i,s} \partial z^L_{s,t}} \)).
$\theta^k$ | The model vector \( \theta \) at the \( k \)-th iteration.
$H^k$ | The Hessian matrix \( \nabla^2 f(\theta^k) \) at the \( k \)-th iteration.
$G$ | The Gauss-Newton matrix of \( f(\theta) \).
$G^k$ | The Gauss-Newton matrix of \( f(\theta^k) \) at the \( k \)-th iteration.
$P$ | The number of partitions to the model variables.
$T_m$ | A subset in \( \{1, 2, \ldots, n_m\} \).
$\mathcal{P}_m$ | The set of \( T_m \).
$S$ | A subset in \( \{1, 2, \ldots, l\} \).
$S_k$ | A subset in \( \{1, 2, \ldots, l\} \) chosen at the \( k \)-th iteration.
$G^S$ | The subsampled Gauss-Newton matrix of \( f(\theta) \).
$G^N_k$ | The subsampled Gauss-Newton matrix at the \( k \)-th iteration.
$g_p^k$ | The local component of the gradient in the partition \( p \) at the \( k \)-th iteration.
$d^k$ | The minimizer of \( f \) at the \( k \)-th iteration.
$v$ | An arbitrary vector in \( \mathbb{R}^n \).
$v_p$ | An arbitrary vector in \( \mathbb{R}^{|T_{m-1}| \times |T_m|} \) in the partition \( p \).
$I$ | An identity matrix.
$\alpha_k$ | A step size at the \( k \)-th iteration.
$\rho_k$ | The ratio between the actual function reduction and the predicted reduction at the \( k \)-th iteration.
$\lambda_k$ | A LM constant.
$\mathcal{N}(\mu, \sigma^2)$ | A Gaussian distribution with mean \( \mu \) and variance \( \sigma^2 \).

## II  Details of Using \( \mathcal{R} \) Operators for Gauss-Newton Matrix Vector Products

The matrix-vector product can be obtained by following the settings in [Schraudolph 2002](https://www.schraudolph.org/public/gradient.pdf), [Martens and Sutskever 2012](https://arxiv.org/abs/1212.5701). We consider the first-order approximation of \( z_{L,i}(\theta) \) to define

\[
\hat{z}_{L,i}(\theta) = z_{L,i}(\theta^k) + \hat{J}^i(\theta - \theta^k) \approx z_{L,i}(\theta), \quad i = 1, \ldots, l,
\]

where

\[
\hat{J}^i = J^i|_{\theta=\theta^k}.
\]

Next we define

\[
\hat{f}(\theta) = \frac{1}{2C} \theta^T \theta + \frac{1}{l} \sum_{i=1}^l \xi(\hat{z}_{L,i}; y^i).
\]

The gradient vector of \( \hat{f}(\theta) \) is

\[
\nabla \hat{f}(\theta) = \frac{1}{C} \theta + \frac{1}{l} \sum_{i=1}^l (\hat{j}^i)^T \frac{\partial \xi(\hat{z}_{L,i}; y^i)}{\partial \hat{z}_{j}^{L,i}}
\]

(II.1)
and the Hessian matrix of $\hat{f}(\theta)$ is

$$\frac{1}{C}I + \frac{1}{l} \sum_{i=1}^{l} (\hat{j}_i)^T \hat{B}_i \hat{j}_i$$

where

$$\hat{B}_i = \frac{\partial^2 \xi(z_{L,i}; y_i)}{\partial z_{L,i}^2}, \quad t = 1, \ldots, n_L, \quad s = 1, \ldots, n_L.$$

Therefore, from (18) we can derive that

$$\nabla \hat{f}(\theta) \bigg|_{\theta = \theta^k} = \nabla f(\theta) \bigg|_{\theta = \theta^k} \text{ and } \nabla^2 \hat{f}(\theta) \bigg|_{\theta = \theta^k} = G(\theta = \theta^k). \quad (II.2)$$

We can apply the $R$ operator to $\nabla \hat{f}(\theta)$ and derive the Gauss-Newton matrix-vector product. That is,

$$R\{ \nabla \hat{f}(\theta) \bigg|_{\theta = \theta^k} \} = \nabla^2 \hat{f}(\theta^k) v = Gv. \quad (II.3)$$

From (II.2), instead of using

$$R\{ \nabla \hat{f}(\theta) \bigg|_{\theta = \theta^k} \},$$

we can use

$$R\{ \nabla f(\theta) \bigg|_{\theta = \theta^k} \}.$$

Then in (13)-(16), $z_{t-1,i}^{m-1,i}$, $s_j^{m,i}$, and $w_{ta}^m$ can be considered as constant values after substituting $\theta$ with $\theta^k$. Therefore, the following results can be derived.

$$R\{ z_{t-1,i}^{m-1,i} \} = 0, \quad R\{ s_j^{m,i} \} = 0, \text{ and } \quad R\{ w_{ta}^m \} = 0.$$

The following equations are therefore used in a backward process to get the Gauss-Newton matrix-vector product.

$$R\{ \frac{\partial \xi}{\partial s_j^{m,i}} \} = \sigma'(s_j^{m,i}) R\{ \frac{\partial \xi}{\partial z_{m,i}^{m-1,i}} \}, \quad (II.4)$$

$$R\{ \frac{\partial \xi}{\partial z_{t,i}^{m-1,i}} \} = \sum_{j=1}^{n_m} w_{ij}^m R\{ \frac{\partial \xi}{\partial s_j^{m,i}} \},$$

$$R\{ \frac{\partial \hat{f}}{\partial w_{ij}^m} \} = \sum_{i=1}^{l} z_{t-1,i}^{m-1,i} R\{ \frac{\partial \xi}{\partial s_j^{m,i}} \},$$

$$R\{ \frac{\partial \hat{f}}{\partial b_{j}^m} \} = \sum_{i=1}^{l} R\{ \frac{\partial \xi}{\partial s_j^{m,i}} \}. \quad (II.5)$$

However, because

$$R\{ \frac{\partial \xi}{\partial z_{j}^{L,i}} \} = \frac{\partial^2 \xi}{\partial (z_{j}^{L,i})^2} R\{ z_{j}^{L,i} \},$$
\( \mathcal{R}\{z^t_{j_i}\} \) are also needed. They are not computed in the backward process. Instead, we can pre-calculate them in the following forward process.

\[
\mathcal{R}\{s^m_{j_i}\} = \mathcal{R}\{ \sum_{t=1}^{n_m-1} w_{t_j} z_{t-1} + b_{j_i} \} = \sum_{t=1}^{n_m-1} (w_{t_j} \mathcal{R}\{z_{t-1}\} + v_{t_j} z_{t-i}) + \mathcal{R}\{b^m_{j_i}\}
\]

(II.6)

\[
\mathcal{R}\{z^m_{j_i}\} = \mathcal{R}\{\sigma(s^m_{j_i})\} = \mathcal{R}\{s^m_{j_i}\}\sigma'(s^m_{j_i})
\]

where \( m = 1, \ldots, L \). Notice that \( \mathcal{R}\{z^0_{t,i}\} = 0 \) because \( z^0_{t,i} \) is a constant. For more details, see Section 6.1 of [Martens and Sutskever 2012].

### III Cost for Solving (68)

In Section 5.3, we omit discussing the cost of solving (68) after the CG procedure. Here we provide details.

To begin we check the memory consumption. The extra space needed to store the linear system (69) is negligible. For the computational and the communication cost, we analyze the following steps in constructing (68).

1. For \(-\nabla f(\theta^k)^T d^k\) and \(-\nabla f(\bar{\theta}^k)^T \bar{d}^k\), we sum up local inner products of sub-vectors (i.e., \(-g_p d^k_p\) and \(-g_p \bar{d}^k_p\) at the \( p \)th node) among all partitions.

2. For the 2 by 2 matrix in (69), we must calculate

\[
(d^k)^T G^S_k d^k = \frac{1}{C} (d^k)^T d^k + \frac{1}{|S_k|} \left( \sum_{p=1}^{P} J^S_k d^k_p \right)^T B^S_k \left( \sum_{p=1}^{P} J^S_k d^k_p \right),
\]

\[
(d^k)^T G^\bar{S}_k d^k = \frac{1}{C} (d^k)^T d^k + \frac{1}{|\bar{S}_k|} \left( \sum_{p=1}^{P} J^\bar{S}_k d^k_p \right)^T B^\bar{S}_k \left( \sum_{p=1}^{P} J^\bar{S}_k d^k_p \right),
\]

(III.7)

\[
(d^k)^T G^S_k \bar{d}^k = \frac{1}{C} (d^k)^T \bar{d}^k + \frac{1}{|S_k|} \left( \sum_{p=1}^{P} J^S_k \bar{d}^k_p \right)^T B^S_k \left( \sum_{p=1}^{P} J^S_k \bar{d}^k_p \right),
\]

\[
(d^k)^T G^\bar{S}_k \bar{d}^k = \frac{1}{C} (d^k)^T \bar{d}^k + \frac{1}{|\bar{S}_k|} \left( \sum_{p=1}^{P} J^\bar{S}_k \bar{d}^k_p \right)^T B^\bar{S}_k \left( \sum_{p=1}^{P} J^\bar{S}_k \bar{d}^k_p \right),
\]

where

\[
J^S_k d^k_p = \begin{bmatrix} \vdots \\ J^i_p d^k_p \\ \vdots \end{bmatrix}, \quad J^S_k \bar{d}^k_p = \begin{bmatrix} \vdots \\ J^i_p \bar{d}^k_p \\ \vdots \end{bmatrix}, \quad i \in S_k.
\]

From (III.7), \( J^S_k d^k_p \) and \( J^S_k \bar{d}^k_p, \forall p \) must be summed up. We reduce all these \( O(n_L) \) vectors to one node and obtain

\[
\sum_{p=1}^{P} J^S_k d^k_p \text{ and } \sum_{p=1}^{P} J^S_k \bar{d}^k_p.
\]

For \((d^k)^T d^k\), \((d^k)^T \bar{d}^k\), and \((\bar{d}^k)^T \bar{d}^k\), we also sum up local inner products in all partitions by a reduce operation. Then, the selected partition can compute the three values in (III.7) and broadcast them to all other partitions.
The computation at each node for $J^S_k d_p^k$ and $J^S_k d_{p'}^k$ is comparable to two matrix-vector products in the CG procedure. This cost is relatively cheap because the CG procedure often needs more matrix-vector products.

For the reduce and broadcast operations, by the analysis in Section 5.3 for the binary tree implementation, a rough cost estimation of the reduce operation is

$$\mathcal{O}(\alpha + 2 \times (\beta + \gamma) \times (|S| \times n_L) \times \log_2 \left( \sum_{m=1}^{L} n_{m-1} \frac{n_m}{|T_m|} \right) \right). \quad \text{(III.8)}$$

Note that one reduce operation is enough because we can concatenate all local values as a vector. The broadcast operation is cheap because only three scalars in (III.7) are involved.

By comparing (III.8) with the communication cost for function/gradient evaluations in (81)-(82), the cost here is not significant. Note that here we need only one reduce/broadcast operation, but in each function/gradient evaluation, multiple operations are needed because of the forward/backward process.

### IV Effectiveness of Using Levenberg-Marquardt Methods

We mentioned in Section 4.5 that the Levenberg-Marquardt method is often not applied together with line search. Here we conduct a preliminary investigation by considering the following two settings.

1. $\text{diag} + \text{sync} 50\%$: the proposed method in section 8.1
2. $\text{noLM} + \text{diag} + \text{sync} 50\%$: it is the same as $\text{diag} + \text{sync} 50\%$ except that the LM method is not considered.

The experimental results are shown in Figure IV.2. We can make the following observations.

1. For testing accuracy/AUC versus number of iterations, the setting without applying LM converges faster. The possible explanation is that for $\text{noLM} + \text{diag} + \text{sync} 50\%$, by solving

   $$G^S d = -\nabla f(\theta^k) \quad \text{(IV.9)}$$

   without the term $\lambda_k I$, the direction leads to a better second-order approximation of the function value.

2. For testing accuracy/AUC versus training time, we observe the opposite result. The setting with the Levenberg-Marquardt method is faster for all problems except HIGGS1M. It is faster because of fewer CG steps in the CG procedure. A possible explanation is that with the term $\lambda_k I$, the linear system becomes better conditioned and therefore can be solved by a smaller number of CG steps.
V Effectiveness of Combining Two Directions by Solving (68)

In Section 4.3, we discussed a technique from Wang et al. [2015] by combining the direction $d^k$ from the CG procedure and the direction $d^{k-1}$ from the previous direction. Here, we preliminarily investigate the effectiveness of this technique.

We take the approach diag + sync 50% in Section 8.1 and compare the results with/without solving (68).

The results are shown in Figure V.4. From the figures of test accuracy versus the number of iterations, we see that the method of combining two directions after the CG procedure effectively improves the convergence speed. Further, the figures of showing training time are almost the same as those of iterations. This result confirms our analysis in Section III that the extra cost at each iteration is not significant.

References


(a) SensIT Vehicle

(b) Poker

(c) MNIST

(d) Letter
Figure IV.2: A comparison of using LM methods and without using LM methods. Left: testing accuracy versus number of iterations. Right: testing accuracy versus training time.
(a) SensIT Vehicle

(b) poker

(c) MNIST

(d) Letter
Figure V.4: Compare using combination of two directions with directly using the sub-sampled newton direction. Left: testing accuracy versus number of iterations. Right: testing accuracy versus training time.