On the Local Hessian in Back-propagation

Huishuai Zhang  
Microsoft Research Asia  
Beijing, 100080  
huzhang@microsoft.com

Wei Chen  
Microsoft Research Asia  
Beijing, 100080  
wche@microsoft.com

Tie-Yan Liu  
Microsoft Research Asia  
Beijing, 100080  
tyliu@microsoft.com

Abstract

Back-propagation (BP) is the foundation for successfully training deep neural networks. However, BP sometimes has difficulties in propagating a learning signal deep enough effectively, e.g., the vanishing gradient phenomenon. Meanwhile, BP often works well when combining with “designing tricks” like orthogonal initialization, batch normalization, and skip connection. There is no clear understanding on what is essential to the efficiency of BP. In this paper, we take one step towards clarifying this problem. We view BP as a solution of back-matching propagation which minimizes a sequence of back-matching losses each corresponding to one block of the network. We study the Hessian of the local back-matching loss (local Hessian) and connect it to the efficiency of BP. It turns out that those designing tricks facilitate BP by improving the spectrum of local Hessian. In addition, we can utilize the local Hessian to balance the training pace of each block and design new training algorithms. Based on a scalar approximation of local Hessian, we propose a scale-amended SGD algorithm. We apply it to train neural networks with batch normalization, and achieve favorable results over vanilla SGD. This corroborates the importance of local Hessian from another side.

1 Introduction

Deep neural networks have been advancing the state-of-the-art performance over a number of tasks in artificial intelligence, from speech recognition [Hinton et al., 2012], computer vision [He et al., 2016a] to natural language understanding [Hochreiter and Schmidhuber, 1997]. These problems are typically formulated as minimizing non-convex objectives parameterized by the neural network models. Typically, the models are trained with stochastic gradient descent (SGD) or its variants and the gradient information is computed through back-propagation (BP) [Rumelhart et al., 1986].

It is known that BP sometimes has difficulties in propagating a learning signal deep enough effectively, e.g., the vanishing/exploding gradient phenomenon [Hochreiter, 1991; Hochreiter et al., 2001]. Recent designing tricks, such as orthogonal initialization [Saxe et al., 2014], batch normalization [Ioffe and Szegedy, 2015] and skip connection [He et al., 2016a], improve the performance of deep neural networks on almost all tasks, which are interpreted to be able to alleviate the vanishing gradient to some extent. However, a recent work [Orhan and Pitkow, 2018] shows that a network with non-orthogonal skip connection always underperforms a network with orthogonal (identity is a special case) skip connection and neither network has vanishing gradient as back-propagating through layers. This suggests that vanishing gradient is not the core reason for a network being good or not. We ask that if vanishing gradient is a superficial reason, what is essential to the efficiency of BP?

In this paper, we consider this question from the optimization’s perspective and give an answer: the Hessian of the local back-matching loss is responsible for the difficulty of training deep nets with BP. Specifically, we start from a penalized loss formulation, which takes the intermediate feature outputs as variables of the optimization and the penalty is to enforce the coordination (architecture) connection [Carreira-Perpinan and Wang, 2014]. Minimizing the penalized loss following backward

order leads to a back-matching propagation procedure which involves minimizing a sequence of back-matching losses. Each back-matching loss penalizes the mismatch between a target signal from the upper block and the output of the current block, which is determined by the parameters and the inputs of the block. We show that BP is equivalent to minimizing each back-matching loss with one-step gradient update. This is to say, BP is a solution of the back-matching propagation procedure.

However, in general, the one-step gradient update may not be a good solution to minimize the back-matching loss contingent on the Hessian, as is well known that bad-conditioned Hessian can have enormous adversarial impact on the convergence of the first-order methods [Ben-Tal and Nemirovski, 2001]. Loosely speaking, if the local Hessian is badly-conditioned, the one-step gradient update does not minimize the back-matching loss sufficiently and the target signal distorts gradually when backward through layers.

We mathematically derive the formula of local Hessian and show that the designing tricks including batch normalization and skip connection can drive local Hessian towards a good-conditioned matrix to some extent. This explains why practical designing tricks can stabilize the backward process. In particular, by analyzing the local Hessian of residual block, we can answer the questions about skip connection in [Orhan and Pitkow, 2018] via local Hessian.

Besides interpreting existing practical techniques and providing guidance to design neural network structure, we can also utilize local Hessian to design new algorithms. The general idea is to employ the information of the local Hessian to facilitate the training of neural networks. We propose a scale-amended SGD algorithm to balance the training pace of each block by considering the scaling effect of local Hessian. More specifically, we approximate the local Hessian with a scalar and use the scalar to amend the gradient of each block. Such a scale-amended SGD is built upon the regular BP process, and hence it is easy to implement in current deep learning frameworks [Bastien et al., 2012, Abadi et al., 2016, Paszke et al., 2017, Seide and Agarwal, 2016]. We apply this scale-amended SGD to feed-forward networks with batch normalization and empirically demonstrate that it improves the performance by a considerable margin. This further advocates the key role of the local Hessian in efficient learning of deep neural networks.

1.1 Related Works

The penalty loss formulation is inspired by methods of auxiliary coordinates (MAC) [Carreira-Perpinan and Wang, 2014] and proximal backpropagation [Frerix et al., 2018]. Specifically, Carreira-Perpinan and Wang [2014] applies block coordinate descent to optimize the penalized objective. Frerix et al. [2018] applies proximal gradient when updating W. In contrast, we start from the penalty loss formulation and focus on the local Hessian for each subproblem in the back-matching propagation, and argue that the local Hessian is critical to the efficiency of BP.

Our scale-amended SGD is related to the algorithms tackling the difficulty of BP for training deep nets by incorporating second-order/metric information [Martens, 2010, Amari, 1998, Pascanu and Bengio, 2014, Olivier, 2015, Martens and Grosse, 2015] and the block-diagonal second order algorithms [Lafond et al., 2017, Zhang et al., 2017, Grosse and Martens, 2016]. These second-order algorithms approximate the Hessian/Fisher matrix and are computationally expensive. In contrast, the scale-amended SGD only amends the vanilla SGD with a scalar for each block based on the approximation of local Hessian. The scale-amended SGD is closely related to the layer-wise adaptive learning rate strategy [Singh et al., 2015, You et al., 2017]. However, these two layer-wise learning rate strategies do not have explanation of why the rate is set in that way.

2 BP as a solution of back-matching propagation

In this section, we first introduce the quadratic penalty formulation of the loss of neural network and the back-matching propagation procedure which minimizes a sequence of local back-matching losses following the backward order. Then we connect BP to the one-step gradient update solution of back-matching propagation procedure. The quality of such a solution on minimizing back-matching loss is determined by its local Hessian.

1 Here a block can be composed of one layer or multiple layers.
Suppose the loss of training a neural network is given by\footnote{For simplicity we omit the bias term in the sequel.}

\begin{equation}
J(W; X, y) = \ell(y; F(W, X)),
\end{equation}

where \(\ell(\cdot)\) is the loss function with respect to the training targets \(y\) and the network output, \(F(\cdot, \cdot)\) is the network mapping, \(W\) is the trainable parameter of the network and \(X\) is the input data. Carreira-Perpinan and Wang\cite{carreira-perpinan14} introduces the intermediate output of the network as auxiliary variables and the architecture connection as a quadratic penalty, and proposes to minimize the following quadratic penalty formulation of the loss,

\begin{equation}
Q(W, z; \gamma) = \ell(y, F_0(W_B, z_{B-1})) + \sum_{b=1}^{B-1} \frac{\gamma}{2} \| z_b - F_b(W_b, z_{b-1}) \|^2,
\end{equation}

where \(F_0(\cdot, \cdot)\) is a block mapping, \(W_b, z_{b-1}\) are the trainable parameter and the input of network block \(b\), respectively, for \(b = B, \ldots, 1\) and \(z_0\) is the input data \(X\).

It has been argued in \cite{nocedal-wright} that under mild condition, the solutions of (2) converge to the solution of the original problem (1) as \(\gamma \to \infty\). Carreira-Perpinan and Wang\cite{carreira-perpinan14} minimizes objective (2) via \(z\)-step and \(W\)-step, which is essentially a block coordinate descent algorithm.

Inspired by the form (2), we study the back-matching propagation procedure (Procedure 1), which minimizes a sequence of local back-matching losses following the backward order. The local back-matching loss for block \(b\) at step \(k\) is denoted by \(\ell_b\)

\begin{equation}
\ell_b(W_b, z_{b-1}) = \begin{cases} 
\ell(y^k; F_B(W_B, z_{B-1})), & \text{for } b = B \\
\frac{1}{2} \| z_b^{k+\frac{1}{2}} - F_b(W_b, z_{b-1}) \|^2, & \text{for } b = B - 1, \ldots, 1,
\end{cases}
\end{equation}

where \(z_b^{k+\frac{1}{2}}\) is computed by (4) repeatedly. We note that \(z^k\) is computed by forward pass given a new \(X^k\) and is not updated by Procedure 1 and \(z_b^{k+\frac{1}{2}}\) is an intermediate variable to store the desired change on the output \(z_0\) which is used to compute \(W_b^{k+1}\). For each subproblem at \(b\), we alternatively optimize over \(W_b\) and \(z_{b-1}\) while fixing the other as in the forward process because jointly optimizing over \(W_b\) and \(z_{b-1}\) is non-convex even if \(F_b\) represents matrix-vector product.

A direct explanation of the back-matching loss is that given the target signal \(z_b^{k+\frac{1}{2}}\) propagated from upper block, which is believed to be the direction of \(z_b^k\) to decrease the loss, the new weight \(W_b^{k+1}\) and the new target signal for lower block \(z_{b-1}^{k+\frac{1}{2}}\) should minimize the matching loss \(\ell_b\).

We are not suggesting Procedure 1 as a new algorithm to train neural network. Actually, Procedure 1 may not be stable in practice if solving each subproblem fully \cite{lee15, wiseman17} because the solution of (4) and (5) may deviate from last updated value too much and jump out of the trust region. Instead, we connect BP to the one-step gradient update solution of (4) and (5) and argue that the conditions of the subproblems (4) and (5) affect the efficiency of BP given the explanation of back matching loss.

**Proposition 1.** If (4) is solved by one-step gradient update with step size \(\mu\) and (5) is solved by one-step gradient update with step size 1, then \(W_b^{k+1}\) produced by the procedure is the same as gradient update of the original problem with step size \(\mu\).
We consider a block of a fully connected layer \( z_b = F_b(W_b, z_{b-1}) = W_b \cdot z_{b-1} \),\(^7\) where \( W_b \) is an \( n_b \times n_{b-1} \) matrix.

Suppose that after the gradient step on \( z_b \) from upper layer, we get an intermediate variable \( z_b^{k+1/2} \). The Hessian of back matching loss \(^5\) with respect to \( z_{b-1} \) and \( w_b \) are

\[
H_z = (W_b^\top)^T W_b^k, \\
H_w = \sum_{j=1}^m z_{b-1,j} (z_{b-1,j}^T)^T,
\]

respectively, where \( m \) is the batch size, \( [j] \) represents the \( j \)-th sample and \( w_b \) is a vector of a row of \( W_b \). Then \( H_{\text{vec}(W_b)} \) is a block diagonal matrix with each block being \( H_w \) where \( \text{vec}(W_b) \) is a long vector stacking the rows of \( W_b \) first.

For \(^5\) the local Hessian with respect to \( z_{b-1} \), we derive the distribution of its eigenvalues. For the convenience of analysis, we assume the elements of \( w_b \) are independently generated from Gaussian distribution with mean \( 0 \) and variance \( \sigma^2 \) and \( n_b, n_{b-1} \to \infty \) and the ratio \( n_b/n_{b-1} \to c \in (0, +\infty) \).

\(^3\)The formula for convolution layer is given in Supplemental C.

\(^7\)The formula for convolution layer is given in Supplemental C.
Then by the Marchenko-Pastur law [Marčenko and Pastur 1967], we have the density of the eigenvalue \( \lambda \) of \( \Sigma \) as follows,

\[
\nu(A) = \begin{cases} 
(1-c)1_{0 < c \leq 1} + \nu_2(A), & \text{if } 0 < c \leq 1, \\
\nu_2(A), & \text{if } c > 1,
\end{cases}
\]

where

\[
d\nu_2(\lambda) = \frac{c}{2\pi \sigma^2} \sqrt{(c_+ - \lambda)(\lambda - c_-)} \frac{1}{\lambda} d\lambda,
\]

with \( c_+ = \sigma^2(1 + \sqrt{c})/c, c_- = \sigma^2(1 - \sqrt{c})/c \).

This result affirms that the orthonormal initialization [Mishkin and Matas 2016, Saxe et al. 2014] facilitates backward propagation. If \( W_b \) is an orthonormal matrix, the eigenvalues of \( H_z \) are composed of \( n_b \) 1’s and \( n_{b-1} - n_b \) 0’s if \( n_b < n_{b-1} \). This is the best spectrum of Hessian we can expect for minimizing the back-matching loss \( f_b \).

However, in general, \( W_b \) is not orthonormal and hence \( H_z \) is not identity. The gradient update on \( z_{b-1} \) does not minimize the back-matching loss well. As back propagating to lower blocks, the update \( z_{i+1} = z_i - F_k \) gets far from the direction of minimizing the back-matching loss \( f_b \) for \( t = 1, 2, \ldots, b \). Such discrepancy becomes larger as the condition of \( H_z \) of each block is bad and as the back-propagation goes deep.

For the local Hessian with respect to \( W \), it is hard to control in general. Several recent works [Freix et al. 2018, Ye et al. 2017] suggest using forms involving \( H_W \) to precondition vanilla SGD. We note that Le Cun et al. [1991] has also studied the spectrum of \( H_w \) which gives a theoretical justification for the choice of centered input over biased state variables.

We next study the local Hessian of blocks with skip connection and batch normalization and show that these designing tricks can improve the spectrum of \( H_z \) and \( H_W \) to some extent and hence make the training deep neural networks easier.

### 3.2 Block with skip connection

Skip connection has been empirically demonstrated important to obtain state-of-the-art results [He et al. 2016a, Huang et al. 2017a], while its functionality has various interpretations. Veit et al. [2016] argue that residual network can be seen as an ensemble of shallow nets and avoids vanishing gradient problem by introducing short paths. Jastrzebski et al. [2018] suggest that residual block performs iterative refinement of features for higher layer while lower layers concentrate representation learning behavior. These works focus on the interpretation of how Resnet works. We here try to give an answer on why Resnet works from the optimization perspective. A recent work [Orhan and Pitkow 2018] argues that skip connection eliminates singular points of the Hessian matrix and there are open questions in [Orhan and Pitkow 2018], for which we can give answers by analyzing the local Hessian of residual block.

Suppose that the mapping of residual block is given by

\[
z_b = F_b(W_b, z_{b-1}) = z_{b-1} + \phi_b(W_b, z_{b-1}),
\]

where \( F_b(\cdot) \) is the residual block mapping with parameters \( W_b \) and input \( z_{b-1} \). The Hessians of the back-matching loss \( f_b \) with respect to \( z_{b-1} \) and \( W_b \) are given by

\[
H_z = \left( I + \frac{\partial \phi_b}{\partial z_{b-1}} \right)^T \left( I + \frac{\partial \phi_b}{\partial z_{b-1}} \right) - \frac{\partial}{\partial z_{b-1}} \left( \frac{\partial F_b}{\partial z_{b-1}} \cdot \left( z_{b-1} + \frac{k+1}{2} - z_b \right) \right),
\]

\[
H_W = \left( \frac{\partial F_b}{\partial \text{vec}(W_b)} \right)^T \left( \frac{\partial F_b}{\partial \text{vec}(W_b)} \right) - \frac{\partial}{\partial \text{vec}(W_b)} \left( \frac{\partial F_b}{\partial \text{vec}(W_b)} \cdot \left( z_{b-1} + \frac{k+1}{2} - z_b \right) \right).
\]

We can see that [13] the Hessian of local matching loss for residual block with respect to \( W_b \) is the same as the case without skip connection. Thus we focus on [13] the local Hessian with respect to \( z \). Specifically, we analyze the first part of [13], the Gauss-Newton matrix, which is a good positive semidefinite approximation to the Hessian [Martens 2016, Chen 2011]. Define the condition number of a matrix \( M \) as \( C(M) := \sigma_{\text{max}}(M)/\sigma_{\text{min}}(M) \), where \( \sigma_{\text{max}} \) and \( \sigma_{\text{min}} \) are the largest and smallest non-zero singular values, respectively. The larger the condition number, the worse the problem.
Remark 1. If \( a) \frac{\partial \phi_b}{\partial z_{b-1}} \) is “small” relatively i.e., \( \sigma_{\text{max}} \left( \frac{\partial \phi_b}{\partial z_{b-1}} \right) < 1 - s \) for some constant \( s > 0 \), and \( b) \) \( C \left( \frac{\partial \phi_b}{\partial z_{b-1}} \right) > 1 + \frac{b}{c} \), then
\[
C \left( I + \frac{\partial \phi_b}{\partial z_{b-1}} \right) < C \left( \frac{\partial \phi_b}{\partial z_{b-1}} \right).
\] (15)

This indicates that the condition number of the Gauss-Newton matrix with skip connection is guaranteed to be smaller than that without skip connection given two assumptions. The assumption \( b) \) is generally satisfied for neural network from the spectrum distribution analysis of fully-connected layer in Section 3.1, while the assumption \( a) \) seems a bit strong. We cannot verify assumption \( a) \) analytically because \( \phi_b(\cdot) \) typically involves more than two linear layers, nonlinear activations and batch normalization layers. We leave the empirical study on the spectrum distribution of local Hessian of the residual block for future work.

Interestingly, [Orhan and Pitkow 2018] demonstrate that a network with an orthogonal connection achieves the performance as good as the one with identity skip connection, which can be easily explained from the fact that orthogonal skip connection does not change the condition number of the local Gauss-Newton matrix (the first part of (13)). Furthermore [Orhan and Pitkow 2018] also empirically show that a network with non-orthogonal skip connection always underperforms a network with orthogonal (identity is a special case) skip connection though neither network has vanishing gradient as back-propagating through layers. This can be easily argued from the formula (13) as non-orthogonal skip connection has larger condition number than orthogonal skip connection whose eigenvalues are all 1’s.

### 3.3 Block with batch normalization

Batch normalization (BN) is widely used for accelerating the training of feed-forward neural networks. In this section, we consider adding a BN layer after a fully-connected layer. We fix the affine transformation of BN to be identity for simplicity. If \( z_b^k \) represents one component of \( z_b \) and \( w_b \) a vector of one row of \( W_b \), then the BN layer mapping is given by
\[
z_b^k = \text{BN} \left( \hat{z}_b^k \right) = \frac{\hat{z}_b^k - \mathbb{E}[\hat{z}_b^k]}{\sqrt{\text{Var}[\hat{z}_b^k]}}, \quad \text{where} \quad \hat{z}_b^k = (w_b)^T z_{b-1}.
\] (16)

BP through a fully connected layer with BN is given in [Ioffe and Szegedy 2015] and we provide the form for the back matching loss in Supplemental for completeness. The gradient formula is quite complicated, as the \( \mathbb{E} \) and \( \text{Var} \) involve batch information. To proceed the analysis, we ignore the terms involving \( 1/m \), which does not lose much as the batch size becomes large.

Now we compute the local Hessian of the fully connected layer with BN as follows:

\[
H_z \approx \sum_{i=1}^{n_b} \frac{w_b^k(i) \cdot w_b^l(i)^T}{\text{Var}[\hat{z}_b^k(i)]} = \sum_{i=1}^{n_b} \frac{w_b^k(i) \cdot w_b^l(i)^T}{\text{Var}[w_b^k(i)^T z_{b-1}^k]}, \quad \text{(17)}
\]

\[
H_w \approx \frac{\sum_{j=1}^{m} z_{b-1}^k(j)(z_{b-1}^k[j])^T}{\text{Var}[\hat{z}_b^k(i)]} = \frac{\sum_{j=1}^{m} z_{b-1}^k(j)(z_{b-1}^k[j])^T}{\text{Var}[w_b^k(i)^T z_{b-1}^k]}, \quad \text{(18)}
\]

where \( n_b \) is the number of outputs of layer \( b \), \( w_b^k(i) \) is the vector of the \( i \)-th row of \( W_b^k \), and \( z_{b-1}^k[j] \) represents the input of the block \( b \) of the sample \( j \). We next show how BN facilitates BP for training deep networks.

We first derive the distribution of the eigenvalues of (17) and compare it to (10) (the case without BN). Our assumption on \( w_b \) is the same as the one to derive (11). In contrast to that \( H_z \) being the sum of outer products of Gaussian vectors in Section 3.1, here \( H_z \) is the sum of the outer products of \( w_b/\|w_b\| \)'s which are the unit vectors equally distributed on the sphere. The density of the eigenvalue \( \lambda \) of (17) is of the form [10] with [Marcenko and Pastur 1967],
\[
d
\nu_z(\lambda) = \frac{\sqrt{(c_+ - \lambda)(\lambda - c_-)}}{2\pi\lambda} 1_{[c_-, c_+]} d\lambda,
\] (19)

where \( c_+ = (1 + \sqrt{c})^2, c_- = (1 - \sqrt{c})^2. \)

---

We ignore the terms involving \( 1/m \) again.
Remark 2. Scaling the variance of the block parameter does not affect the spectrum of $H_z$ in (17).

This is in contrast to [8] where the spectrum is linearly scaled with the variance of weight parameters. Thus BP gains benefit because it acts as one-step gradient update with fixed step size 1 for all blocks.

Another benefit of BN is to improve the condition of $H_w$ if $z_{b-1}$ is the output of a BN.

Remark 3. If $z_{b-1}$ is the output of BN and $w_b$ is independent of $z_{b-1}$, then $\mathbb{E} \text{diag}(H_w) = I/\|w_b\|^2$.

This indicates the problem [3] is well-conditioned and hence large step size is allowed [Ioffe and Szegedy, 2015].

4 Utilize local Hessian: An example

As previous section has shown the importance of the local Hessian, this section discuss how to utilize local Hessian to improve the performance on current deep learning tasks. One direct way of using local Hessian is to design better architecture. The spectrum of local Hessian can be a criteria to determine whether a building block is good or not for BP. One potential usage of local Hessian could be in neural architecture search [Zoph and Le, 2016]. As most of the time in neural architecture search, the local Hessian is to design better architecture. The spectrum of local Hessian can be a criteria to determine whether a building block is good or not for BP.

In this section, we propose a way to employ the information of the local Hessian to facilitate the training of deep neural networks. Ideally, good alternatives to minimize back-matching loss $\ell_b$ are $H_W^{-1} \delta w_b$ and $H_z^{-1} \delta z_{b-1}$, where $\delta w_b$ and $\delta z_{b-1}$ are the gradient computed via BP rule given $z^{k+1} - z^k$. However, $H_W$ and $H_z$ are often indefinite and expensive to compute. We suggest using two scalars $m_{b,W}$ and $m_{b,z}$ to evaluate how $H_W$ and $H_z$ scale the norm of a vector with general position, respectively. Then the back-matching loss can be approximated as

$$
\ell_b \left( W_b, z_{b-1}^k \right) \approx \ell_b \left( W_b, z_{b-1}^k \right) + \left\langle \frac{\partial \ell_b}{\partial W_b} W_b - W_b^k \right\rangle + \frac{1}{2} (W_b - W_b^k)^T H_W (W_b - W_b^k) 
$$

$$
\approx \ell_b \left( W_b, z_{b-1}^k \right) + \left\langle \frac{\partial \ell_b}{\partial W_b} W_b - W_b^k \right\rangle + \frac{1}{2} m_{b,W} \| W_b - W_b^k \|^2, \quad (20)
$$

$$
\ell_b \left( W_b, z_{b-1}^k \right) \approx \ell_b \left( W_b, z_{b-1}^k \right) + \left\langle \frac{\partial \ell_b}{\partial z_{b-1}} z_{b-1} - z_{b-1}^k \right\rangle + \frac{1}{2} m_{b,z} \| z_{b-1} - z_{b-1}^k \|^2, \quad (21)
$$

where the approximation is composed of a second-order Taylor expansion and a scaling effect of local Hessian, and $W_b$ may represent vec$(W_b)$ contingent on the context.

We next propose an algorithm scale-amended SGD to take the effect of $m_{b,W}$ and $m_{b,z}$ into account to balance the training pace of each block. Scale-amended SGD uses $m_{b,W}$ and $m_{b,z}$ to amend the scale of vanilla BP of each block. We set the initial backward factor of the output layer $m = 1$, which indicates that the derivative of the loss with respect to the output of the network is regarded as the desired changes on the output to minimize the loss.

Then following the backward order, if a block has parameter $W_b$ and gradient $\delta W_b$ computed by BP, then we use $\delta' W_b := \delta W_b / m / m_{b,W}$ as the scale-amended gradient to update $W_b$, where $m$ is the backward factor on the output of the block and $m_{b,W}$ is the scalar used to approximate $H_{b,W}$. Then we update the backward factor $m$ for next block via $m \leftarrow m \cdot m_{b,z}$, where $m_{b,z}$ is the scalar used to approximate $H_{b,z}$. This strategy is described in Algorithm 2.

4.1 Scale-amended SGD for feed-forward networks with BN

Note that for the feed-forward networks with BN layers, we can obtain a reliable estimation of $m_{b,W}$ and $m_{b,z}$. Specifically, we assume that $W_b$ is row homogeneous [Ba et al., 2016], i.e., they represent the same level of information and are roughly of similar magnitude, and define

$$
\| W_b \|^2_{2,\mu} := \frac{1}{\#\text{row}(W_b)} \sum_{i=1}^{\#\text{row}(W_b)} w_b(i)^T w_b(i),
$$
Algorithm 2 Scale-amended SGD

**Input:** Gradient $\delta W_b$ and scaling factor $m_{b,W}, m_{b,z}$ for $b = 1, ..., B$; Initialize $m = 1$.

for $b = B, ..., 1$ do

\[
\delta W_b \leftarrow \delta W_b / m / m_{b,W}
\]

\[
\frac{m}{m} \cdot m_{b,z}
\]

end for

where $u_{b}(i)$ is the $i$-th row of $W_b$. Under this assumption, the scalars to approximate the local Hessians (17) and (18) of the fully connected layer with BN are computed as follows,

\[
m_{b,z} := \frac{\|W^T_b\|_{2,\mu}^2}{\|W_b\|_{2,\mu}^2}, \quad m_{b,W} := 1/\|W_b\|_{2,\mu}^2.
\]

We next evaluate the scale-amended SGD on training VGG nets [Simonyan and Zisserman, 2015] for image classification tasks with two datasets: CIFAR-10 [Krizhevsky and Hinton, 2009] and CIFAR-100 [Krizhevsky and Hinton, 2009]. We modify the VGG nets by keeping the last fully connected layers and removing the intermediate two fully connected layers and all the biases. Each intermediate layer of the VGG nets concatenates a BN layer right before the activation function and the BN has no trainable parameters.

During training, the images of CIFAR-10 and CIFAR-100 datasets are randomly flipped and rotated for data augmentation. The hyper-parameters for vanilla SGD and our scale-amended SGD are the same including learning rate $\eta = 0.1$ (because the backward factor for linear layer of CIFAR10 is around $10^{-11}$, small learning rate $\eta = 0.005$ works better for CIFAR10 to use scale-amended SGD), momentum 0.9 and weight decay coefficient 0.005. We reduce the learning rate by half once the validation accuracy is on plateau (ReduceLROnPlateau in PyTorch with patience=10), which works well for both vanilla-SGD and scale-amended SGD.

We compare the learning curves between scale-amended SGD and vanilla SGD on training VGG13 for CIFAR10 and CIFAR-100 classification tasks. Two algorithms start from the same initialization and pass the same batches of data. Both algorithms are run 300 epochs. We plot the learning curves in Figure 1. From Figure 1, we can see that the learning curves of our algorithm and SGD have similar trend (we plot curves of multiple runs and their average in Supplemental D). This is because scale-amended SGD only modifies the magnitude of each block gradient as a whole and does not involve any further information (second order information) and hyper-parameters are the same for both algorithms. Scrutinizing more closely, we can see our training loss curve is almost always lower than SGD’s and our test error ends with a considerably lower number. Thus the scale-amended SGD

Figure 1: Comparison of vanilla SGD and scale-amended SGD on training VGG13 for CIFAR10 and CIFAR-100 classification. Hyperparameters are the same: learning rate 0.1 (except for CIFAR10 scale-amended SGD uses 0.005), momentum 0.9, weight decay 0.005.

For scale-amended SGD, we first apply the weight decay and then amend the scale.
achieves favorable result over vanilla SGD on training feed-forward neural network with BN. More extensive experiments can be found in Supplemental D.

5 Conclusion

In this paper we view BP as a solution of back-matching propagation which minimizes a sequence of back-matching losses. By studying the Hessian of the local back-matching loss, we interpret the benefits of practical designing tricks, e.g., batch normalization and skip connection, in a unified way: improving the spectrum of local Hessian. Moreover, we propose scale-amended SGD algorithm by employing the information of local Hessian via a scalar approximation. Scale-amended SGD achieves favorable results over vanilla SGD empirically for training feed-forward networks with BN, which corroborates the importance of local Hessian.

Acknowledgments

The authors would like to thank Prof. Yuejie Chi for helpful discussion.

References


