Enhance Curvature Information by Structured Stochastic Quasi-Newton Methods

Minghan Yang[1,2], Dong Xu[1,2], Hongyu Chen[1], Zaiwen Wen[2,3,4], Mengyun Chen[5]
[1] School of Mathematical Sciences, Peking University, China
[2] Beijing International Center for Mathematical Research, Peking University, China
[3] Center for Data Science, Peking University, China
[4] National Engineering Laboratory for Big Data Analysis and Applications, Peking University, China
[5] Huawei Technologies Co. Ltd, China
{yangminghan, taroxd, hongyuchen, wenzw}@pku.edu.cn ,chenmengyun1@huawei.com

Abstract

In this paper, we consider stochastic second-order methods for minimizing a finite summation of nonconvex functions. One important key is to find an ingenious but cheap scheme to incorporate local curvature information. Since the true Hessian matrix is often a combination of a cheap part and an expensive part, we propose a structured stochastic quasi-Newton method by using partial Hessian information as much as possible. By further exploiting either the low-rank structure or the kronecker-product properties of the quasi-Newton approximations, the computation of the quasi-Newton direction is affordable. Global convergence to stationary point and local superlinear convergence rate are established under some mild assumptions. Numerical results on logistic regression, deep autoencoder networks and deep convolutional neural networks show that our proposed method is quite competitive to the state-of-the-art methods.

1. Introduction

Consider the large-scale finite-sum optimization problem:

$$\min_{\theta\in\mathbb{R}^n} \Psi(\theta) = \frac{1}{N}\sum_{i=1}^{N} \psi_i(\theta), \tag{1}$$

where θ is the decision variable, ψ_i is the component function and N is the number of functions. For many cases, ψ_i is related to the data point (x_i, y_i), i.e.,

$$\psi_i(\theta) = \ell\left(f(x_i,\theta), y_i\right), \tag{2}$$

where $f(x_i,\theta) \in \mathbb{R}^m$ is the output of x_i through certain model and ℓ is the loss function to measure the prediction error. The numbers n and N can be very huge. For example, the number of parameters n is 175 billion in GPT-3 [5]. Problem (1) widely arises in many applications such as deep learning [13, 23, 34] and statistical learning [16, 37].

The first-order type methods are dominant approaches for the problem (1). The classical stochastic gradient descent method (SGD) [31] falls into this type and extensions of SGD have been widely studied in [1, 20, 27] for better practical performance and theoretical properties.

Recently, stochastic second-order methods have gained increasing attention. Since the computation and inversion of the Hessian matrix in the large-scale applications is costly, various strategies to approximate the Hessian have been developed in [6, 26, 30, 32, 40, 41]. In deep learning, stochastic second-order methods are expected to address the scalability issue with large batch size. The Hessian-free method [25] uses the conjugate-gradient (CG) method to obtain a descent direction by utilizing Hessian-vector products. The Gauss-Newton matrix is investigated to approximate the Hessian matrix in [35] and a practical block-diagonal approximation to the Gauss-Newton matrix is studied in [4]. The so-called KFAC method developed in [15] utilizes the kronecker-factored approximation to Fisher information matrix (FIM). Its efficiency has been demonstrated in large-scale distributed parallel computing [29]. A Newton method for convolutional neural networks (CNN) is investigated in [38].

In this paper, we consider a structured quasi-Newton (QN) framework to enhance curvature information. This idea has been studied in a few second-order methods. For example, on the nonlinear least squares problems with large residual, approaches that compensate the Gauss-Newton matrix by a quasi-Newton approximation to the complicate part of the Hessian matrix are often much better [28, 36, 44]. This

concept has been further verified in optimization problems with orthogonality constraints in [19]. In this paper, we develop the structured quasi-Newton method in the stochastic setting for problem (1).

1.1. Contribution

Our main contributions are as follows.

(1) The structures where the Hessian matrix is a summation of a cheap part and an expensive part are exploited for machine learning problems. The concept of structured quasi-Newton is extended to stochastic setting.

(2) A general structured stochastic quasi-Newton framework is proposed for the large-scale finite-sum problem (1). We formulate stochastic secant conditions based on the partial Hessian matrix, then various quasi-Newton matrices can be constructed. By further exploiting either the low-rank structure or the kronecker-product properties of the quasi-Newton approximations, the computation of the refined direction is affordable.

(3) Global convergence is established if the step sizes are chosen properly and the stochastic errors satisfy certain summability conditions. A local superlinear convergence rate is also guaranteed for the structured stochastic quasi-Newton method if the sample size is sufficiently large.

2. Structures of the Hessian Matrices

In this part, we assume that the Hessian matrix $\nabla^2\Psi(\theta)$ can be divided into two different parts:

$$\nabla^2\Psi(\theta) = H(\theta) + \Pi(\theta), \tag{3}$$

where the part $H(\theta)$ is relatively cheap and accessible while the other part $\Pi(\theta)$ is expensive or even not computable.

Specifically, there are a few possible situations. **(1) The dimension n is so high** that an explicit storage of $\nabla^2\Psi(\theta)$ is prohibitive. Hence, it is favorable to utilize an low-rank approximation to the Hessian matrix [33] or just use the Hessian information implicitly [25]. **(2) The number of data points N is tremendous.** Even if each component $\nabla^2\psi_i(\theta)$ is cheap, assembling all parts becomes a non-negligible task when the size N is huge. **(3) The derivatives are complicated.** In certain cases, the explicit expression of the Hessian can not be derived or computed easily.

In many applications, the reasons listed above are mixed. We next explain a few typical scenarios of (3). The goal is to explore $\Pi(\theta)$ for better performance at a relatively low computational cost.

2.1. Hessian Matrices for General Problem (1)

The subsampling procedure only selects a small fraction of the data in certain ways for the update. Given a subset $\mathcal{S}_H \subseteq \{1, 2, \ldots, N\}$, the subsampled Hessian matrix is defined as $\nabla^2_{\mathcal{S}_H}\Psi(\theta) := \frac{1}{|\mathcal{S}_H|}\sum_{i\in\mathcal{S}_H}\nabla^2\psi_i(\theta)$. It is common to choose the subset $\mathcal{S}_H$ by uniform random sampling. In this case, if $|\mathcal{S}_H|$ is small, let

$$H(\theta) = \nabla^2_{\mathcal{S}_H}\Psi(\theta), \tag{4}$$

and we have $\Pi(\theta) = \nabla^2\Psi(\theta) - \nabla^2_{\mathcal{S}_H}\Psi(\theta)$ correspondingly.

2.2. Hessian Matrices for Format (2)

By using the chain rule twice, the Hessian matrix of the case (2) can be split as:

$$H(\theta) = \frac{1}{N}\sum_{i=1}^{N} H_i(\theta) = \frac{1}{N}\sum_{i=1}^{N} J_f^i(\theta)\nabla_f^2\ell_i(\theta)(J_f^i(\theta))^\top, \tag{5}$$

$$\Pi(\theta) = \frac{1}{N}\sum_{i=1}^{N} \Pi_i(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m} \nabla_{f_j}\ell_i(\theta)\nabla_\theta^2 f_j^i(\theta), \tag{6}$$

where $J_f^i(\theta) = \nabla_\theta f(x_i,\theta) \in \mathbb{R}^{n\times m}$ and $f_j^i(\theta)$ is the j-th component of $f_i(\theta) := f(x_i,\theta)$. The term $H(\theta)$ here is also called the generalized Gauss-Newton (GGN) matrix, which is a good approximation to the Hessian matrix. It is positive semi-definite (PSD) if the loss function ℓ is convex.

In many problems, it is not easy to compute the Hessian matrix or even the GGN, but for deep learning problems there exists some good estimators in some cases. Assume that the loss function is the negative log probability associated with a distribution, that is, $\ell(f(x_i,\theta),y_i) = -\log p(y_i|x_i,\theta)$. The corresponding FIM is defined as

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{z\sim p(z|x_i,\theta)}\nabla_\theta\ell(f(x_i,\theta),z)\left(\nabla_\theta\ell(f(x_i,\theta),z)\right)^\top.$$

For the square loss and the cross entropy loss function, GGN and FIM are equal. The empirical FIM (EFIM) matrix is also a good choice and is defined as:

$$\textbf{EFIM} := \frac{1}{N}\sum_{i=1}^{N}\nabla\psi_i(\theta)\nabla\psi_i(\theta)^\top.$$

The evaluation of EFIM uses the sample gradients. Therefore, it does not require extra backward passes.

3. Structured Stochastic Quasi-Newton Methods (S2QN)

In this section, we propose a structured stochastic quasi-Newton method to enhance curvature information. First, let us describe a second-order framework for the problem (1).

At the k-th iteration, a quadratic approximation model is constructed as follows:

$$\min_d \; m_k(d) := g_k^\top d + \frac{1}{2} d^\top (B_k + \lambda_k I) d, \tag{7}$$

where g_k and B_k are the estimations of the gradient and the Hessian matrix at θ_k, respectively. A common strategy to choose g_k is the mini-batch gradient $\nabla_{\mathcal{S}_g}\Psi(\theta) = \frac{1}{|\mathcal{S}_g|}\sum_{i\in\mathcal{S}_g}\nabla\psi_i(\theta)$ for a given index set $\mathcal{S}_g \subseteq \{1, 2, \ldots, N\}$. The regularization parameter λ_k is adjusted by the norm of the stochastic gradient. Specifically, for given $r_1 < r_2$ and sequence $\{\alpha_k\}$, we propose

$$\lambda_k = \begin{cases} \frac{2r_1}{\|g_{k-1}\|+r_1}\alpha_k^{-1} & \|g_{k-1}\| < r_1, \\ \frac{2\|g_{k-1}\|}{\|g_{k-1}\|+r_2}\alpha_k^{-1} & \|g_{k-1}\| > r_2, \\ \alpha_k^{-1} & \text{otherwise.} \end{cases} \tag{8}$$

By solving the model (7), we update the parameter θ by

$$\theta_{k+1} = \theta_k + \beta_k d_k, \quad d_k = -(B_k + \lambda_k I)^{-1} g_k,$$

where β_k is the step size. The strategy (8) is similar to (but still different from) the adjustment of the trust region radius in [10]. It can be further viewed as a stochastic Levenberg-Marquardt method for generalized nonlinear least squares problem. The convergence analysis of our method is more straightforward compared to that in [3].

3.1. Structured Stochastic quasi-Newton Matrix

The key concept is to use a quasi-Newton method to compensate the difference between the partial Hessian information and the true Hessian matrix. That is, the matrix B_k is constructed as

$$B_k = H_k + \Lambda_k. \tag{9}$$

Specifically, H_k represents matrix with partial Hessian information which can be either the subsampled Hessian matrix, the GGN/FIM/EFIM, or any other approximation matrices. The matrix Λ_k compensates the difference with the true Hessian matrix. We call H_k the base matrix and Λ_k the refinement matrix. The quasi-Newton method is used to update Λ_k under some constraints. For example, $\Lambda_k = \text{QN}(u_{k-1}, v_{k-1}, \Lambda_{k-1})$ satisfies the secant condition:

$$B_k u_{k-1} = (H_k + \Lambda_k) u_{k-1} = \hat{v}_{k-1}, \tag{10}$$

or equivalently,

$$\Lambda_k u_{k-1} = \hat{v}_{k-1} - H_k u_{k-1} := v_{k-1},$$

where $u_{k-1} = \theta_k - \theta_{k-1}$ and $\hat{v}_{k-1} = \nabla_{\mathcal{S}_g^{k-1}}\Psi(\theta_k) - \nabla_{\mathcal{S}_g^{k-1}}\Psi(\theta_{k-1})$. Other structured conditions can be considered as long as they are reasonable. Since we compute $\hat{v}_{k-1}$ by using two gradients on the same samples in (10) , we can consider an extra-step technique [41] if needed. Note that the pairs $\langle u_k, v_k\rangle$ can be constructed by using other mechanisms, see [2, 7, 14]. These methods can be integrated into our framework naturally.

3.2. Representation of the Inverse Matrix

For an efficient computation of the direction d_k, we update the matrix Λ_k by the limited-memory BFGS (L-BFGS) method [28, 36]. Assume that there are p pairs of vectors:

$$\begin{aligned} U_k &= [u_{k-p}, \ldots, u_{k-1}] \in \mathbb{R}^{n\times p}, \\ V_k &= [v_{k-p}, \ldots, v_{k-1}] \in \mathbb{R}^{n\times p}. \end{aligned} \tag{11}$$

For a given initial matrix Λ_k^0, a compact representation of the L-BFGS matrix is:

$$\Lambda_k = \text{QN}(U_k, V_k) := \text{LBFGS}(U_k, V_k) = \Lambda_k^0 - C_k P_k^{-1} C_k^\top, \tag{12}$$

where

$$\begin{aligned} C_k :=& C_k(U_k, V_k) = \left[\Lambda_k^0 U_k, V_k\right] \in \mathbb{R}^{n\times 2p}, \\ P_k :=& P_k(U_k, V_k) = \begin{bmatrix} U_k^\top \Lambda_k^0 U_k & L_k \\ L_k^\top & -D_k \end{bmatrix} \in \mathbb{R}^{2p\times 2p}, \\ (L_k)_{i,j} :=& (L_k(U_k, V_k))_{i,j}, \\ =& \begin{cases} u_{k-p+i-1}^\top v_{k-p+j-1} & \text{if} \quad i > j, \\ 0 & \text{otherwise,} \end{cases} \\ D_k =& D_k(U_k, V_k) = \text{diag}\left[u_{k-p}^\top v_{k-p}, \ldots, u_{k-1}^\top v_{k-1}\right]. \end{aligned}$$

The initial matrix Λ_k^0 is usually set to be $\gamma_k I$, where γ_k is a positive scalar. In order to ensure the positive definiteness of Λ_k, the pair $\{u_i, v_i\}$ should satisfy $u_i^\top v_i \geq \epsilon_B \|u_i\|\|v_i\|$ with a small constant, say $\epsilon_B = 10^{-8}$. We can consider the damping strategy in [39]. Although H_k may not be positive definite due to the nonconvexity of the functions $\psi_i(\theta)$, the parameter λ_k can be adjusted suitably such that $B_k + \lambda_k I$ has good properties to generate descent directions.

We now show how to compute the inverse of $B_k + \lambda_k I$. Let $\tilde{H}_k = H_k + \Lambda_k^0 + \lambda_k I$. Assume that $\tilde{H}_k$ is invertible, otherwise the regularization parameter λ_k can be adjusted accordingly. By using the Sherman-Morrison-Woodbury (SMW) formula, we obtain

$$\begin{aligned} (B_k + \lambda_k I)^{-1} &= (\tilde{H}_k - C_k P_k^{-1} C_k^\top)^{-1} \\ &= \tilde{H}_k^{-1} + \tilde{H}_k^{-1} C_k T_k^{-1} C_k^\top \tilde{H}_k^{-1}, \end{aligned} \tag{13}$$

where $T_k = P_k - C_k^\top \tilde{H}_k^{-1} C_k$. Note that the main computational cost in (13) is the inversion of $\tilde{H}_k$. Assume that Λ_k^0 is set to be $\gamma_k I$ and that H_k can be easily obtained, then the computation of $\tilde{H}_k = H_k + (\gamma_k + \lambda_k) I$ is also cheap. Meanwhile, since p is usually small, *e.g.*, $1 \sim 5$, the computational cost of (13) can be controlled.

Algorithm 1: Structured Stochastic Quasi-Newton Methods (S2QN)

Initialization: Choose an initial point θ_0. Select the sequence (α_k), (β_k). Set the memory size p.
for $k = 0, 1, \ldots$ **do**
  Choose the random sample sets $\mathcal{S}_g^k, \mathcal{S}_H^k \subset [N]$. Compute $\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)$ and base matrix H_k.
  Compute the refinement matrix $\Lambda_k = \mathrm{QN}(U_k, V_k)$.
  Adjust the regularization parameter λ_k by (8) and compute the direction d_k. Update $\theta_{k+1} = \theta_k + \beta_k d_k$.
  Compute the gradient $\nabla_{\mathcal{S}_g^k}\Psi(\theta_{k+1})$ with the same samples and update the pairs (U_{k+1}, V_{k+1}).

3.3. Explicit Inverse by Low-Rank Structures

In many cases, the base matrix $H_k = Q_k Q_k^\top$ is low-rank, where $Q_k \in \mathbb{R}^{n\times r}, r \ll n$. For example, the subsampled EFIM in the sketchy natural gradient method [42] is low-rank and its rank is related to the sample size. For convenience of notation, we define:

$$\widetilde{\Lambda}_k = \Lambda_k^0 + \lambda_k I,\ \widetilde{P}_k = \begin{bmatrix} P_k^{-1} & 0 \\ 0 & -I \end{bmatrix},\ \widetilde{C}_k = [C_k, Q_k].$$

Using the SMW formula yields:

$$\begin{aligned}(B_k + \lambda_k I)^{-1} &= (\widetilde{\Lambda}_k^0 - \widetilde{C}_k \widetilde{P}_k^{-1} \widetilde{C}_k^\top)^{-1} \\ &= \widetilde{\Lambda}_k^{-1} + \widetilde{\Lambda}_k^{-1}\widetilde{C}_k \hat{T}_k^{-1} \widetilde{C}_k^\top \widetilde{\Lambda}_k^{-1},\end{aligned} \tag{14}$$

where $\hat{T}_k = \widetilde{P}_k - \widetilde{C}_k^\top \widetilde{\Lambda}_k^{-1} \widetilde{C}_k$. The size of $\hat{T}_k$ is $r + 2p$. Therefore, its inverse matrix can be computed fast. Since Λ_k^0 is usually set to be $\gamma_k I$, the computational cost (14) can be much smaller than the cost of inverting the matrix directly.

3.4. Block Approximation

Our structured quasi-Newton method can also be easily extended to the case when $B_k = \mathrm{diag}\{B_k^1, \ldots, B_k^L\}$ is a block diagonal matrix. The case often occurs in practice. For L-layers neural network, to reduce the computational complexity, the curvature matrix is chosen to be a block-diagonal matrix and constructed layer by layer [15, 42].

One requirement is that the j-th block of Λ_k should satisfy the following secant equation: $\Lambda_k^j u_{k-1}^j = \hat{v}_{k-1}^j - H_k^j u_{k-1}^j := v_{k-1}^j$, where u_{k-1}^j and v_{k-1}^j are the sub-vector of u_{k-1} and v_{k-1} corresponding to the j-th block.

4. S2QN For Deep Learning

In this section, we extend the structured stochastic quasi-Newton method to deep learning problems by combining their characteristics. We first prove that the block Hessian matrix for each convolutional layer is a kronecker product of two matrices under some assumptions, then propose a sketchy block BFGS method in this setting.

The input of a convolutional layer is a set of activations $\{a_{j,t}\}$, where $j \in \{1, 2, \ldots, \mathcal{J}\}$, $t \in \mathcal{T}$ and $\mathcal{T}$ is the set of spatial locations (typically a 2-D grid). We further write the effect area of the convolutional filter by $\Delta = \{-K, \ldots, K\} \times \{-K, \ldots, K\}$ and the learned weight by Θ. We also assume that the padding size equals K and the stride is 1. Assume that the outputs of the convolutional layer have $\mathcal{I}$ channels, then the set of pre-activations $\{s_{i,t}\}$ is as follows (we do not consider the bias for simplicity):

$$s_{i,t} = \sum_{\delta\in\Delta} \Theta_{i,j,\delta} a_{j,t+\delta},$$

where $i \in \{1, 2, \ldots, \mathcal{I}\}$ and $t \in \mathcal{T}$. Therefore, the gradient of the component function with respect to one single sample (x, y) can be computed as: $\mathcal{D}\Theta_{i,j,\delta} = \sum_{t\in\mathcal{T}} a_{j,t+\delta} \mathcal{D}s_{i,t}$, or $\mathcal{D}\widetilde{\Theta} = G(x, y; \widetilde{\Theta}) A(x, \widetilde{\Theta})^\top$ in matrix form where we use the notation $\mathcal{D}(\cdot)$ to represent $\frac{\partial\psi}{\partial(\cdot)}$ and $\widetilde{\Theta}$ is the matrix form of Θ. Differentiating again, we find that the elements of Hessian can be computed as:

$$(\mathcal{D}^2\Theta)_{i,j,\delta;i',j',\delta'} = \sum_{t,t'\in\mathcal{T}} (\mathcal{D}^2 s)_{i,t;i',t'} a_{j,t+\delta} a_{j',t'+\delta'}, \tag{15}$$

where $(D^2\Theta)_{i,j,\delta;i',j',\delta'} := \frac{\partial^2\psi}{(\partial\Theta_{i,j,\delta})(\partial\Theta_{i',j',\delta'})}$ and $(D^2 s)_{i,t;i',t'} := \frac{\partial^2\psi}{(\partial s_{i,t})(\partial s_{i',t'})}$. However, since the size of the matrix is still large, it is not tractable to compute the block Hessian matrix. Instead, analogous to the approximation mechanism that KFAC made in [15], we approximate the Hessian matrix (15) by a product of two smaller matrices. Similar results for fully-connected networks have been studied in [4]. We list some necessary assumptions below.

Assumption 1 *1.1) The activations are independent of the second-order derivatives of the pre-activations.*

1.2) The second-order statistics of pre-activation derivatives at any two spatial locations t and t' depend only on $t' - t$, which means there exists function Γ such that:

$$\mathbb{E}\left[(\mathcal{D}^2 s)_{t,i;t',i'}\right] = \Gamma(i, i', t' - t).$$

Lemma 1 *Suppose that Assumptions 1 are satisfied and there exists function Ω such that*

$$\mathbb{E}\left[a_{j,t} a_{j',t'}\right] = \Omega(j, j', t' - t).$$

If the second-order derivatives of the pre-activations at any two distinct spatial locations are uncorrelated

$$\Gamma(i, i', t - t') = 0, \quad \textit{for} \quad t \neq t',$$

then we have:

$$\mathbb{E}\left[(\mathcal{D}^2\Theta)_{i,j,\delta;i',j',\delta'}\right] = \chi(\delta,\delta')\Omega(j,j',\delta'-\delta)\Gamma(i,i',0).$$

Hence, the Hessian matrix is a kronecker product of two matrices:

$$\mathbb{E}\left[\frac{\partial^2\psi}{\partial^2\,\mathrm{vec}(\theta)}\right] = \mathbf{A}\otimes\mathbf{G}, \tag{16}$$

where $(\mathbf{A})_{j|\Delta|+\delta,j'|\Delta|+\delta'} = \chi(\delta,\delta')\Omega(j,j',\delta'-\delta)$, $(\mathbf{G})_{i,i'} = \Gamma(i,i',0)$, *and* $\chi(,)$ *is a function for indexing the location.*

We omit the proof of the Lemma 1 since it is similar to Theorem 1 in [15] which interested readers can refer to. By now, we have proved that the block Hessian matrix for convolutional layer can be written as a kronecker product of smaller matrices. The approximation is efficient. For example, for some layer in Resnet50 v1.5 with Imagenet-1k dataset, the size of block Hessian matrix is $2,359,296$ while the size of $\mathbf{G}$ and $\mathbf{A}$ are 512 and $4,608$, respectively. In the next, we show that by using the special properties of the block Hessian matrix, we can combine the structured quasi-Newton idea with KFAC approximation naturally.

4.1. Stochastic Structured QN for Kronecker-Factored Approximation

The KFAC method [15] approximates the FIM by a block-diagonal matrix where each block $\mathbf{F}_{\text{KFAC}}$ approximates the block Fisher matrix by a kronecker product of two smaller matrices as follows:

$$\mathbf{F}_{\text{KFAC}} = \hat{\mathbf{A}}\otimes\hat{\mathbf{G}}, \tag{17}$$

where $\hat{\mathbf{A}} = (\frac{1}{|\mathcal{S}|}\sum_{x_i\in\mathcal{S}} A(x_i,\Theta)A(x_i,\Theta)^\top)$ and $\hat{\mathbf{G}} = (\frac{1}{|\mathcal{S}|}\sum_{x_i\in\mathcal{S}}\mathbb{E}_{z\sim p(z|x_i,\theta)}G(x_i,z;\Theta)G(x_i,z;\Theta)^\top)$. However, extra backward passes are required to compute $\hat{\mathbf{G}}$. Notice that (17) has the similar kronecker structure as (16). Hence, we can combine the structured QN idea and use the empirical version of (17) as the base matrix to approximate the Hessian matrix (16). Assume the refinement matrix Λ_k takes the kronecker factorization, then the structured QN matrix is constructed as follows:

$$B_k = \hat{\mathbf{A}}_k\otimes\hat{\mathbf{G}}_k + \Lambda_k \approx \hat{\mathbf{A}}_k\otimes\left(\tilde{\mathbf{G}}_k+\tilde{\Lambda}_k\right), \tag{18}$$

where $\tilde{\mathbf{G}}_k = \frac{1}{|\mathcal{S}^k|}\sum_{(x_i,y_i)\in\mathcal{S}^k} G(x_i,y_i;\Theta_k)G(x_i,y_i;\Theta_k)^\top$.

We propose two different conditions. The first strategy is to let $\tilde{\Lambda}_k$ satisfy the stochastic secant equation:

$$\left(\hat{\mathbf{A}}_k\otimes\left(\tilde{\mathbf{G}}_k+\tilde{\Lambda}_k\right)\right)u_k = v_k,$$

where $u_k = \theta_k-\theta_{k-1}$ and $v_k = \nabla_{\mathcal{S}_g^{k-1}}\Psi(\theta^k) - \nabla_{\mathcal{S}_g^{k-1}}\Psi(\theta^{k-1})$. Equivalently, $\tilde{\Lambda}_k$ is required to satisfy:

$$\tilde{\Lambda}_k\hat{U}_k = \hat{V}_k - \tilde{\mathbf{G}}_k\hat{U}_k\hat{\mathbf{A}}_k, \tag{19}$$

where $\hat{U}_k$ and $\hat{V}_k$ are the matrix format of u_k and v_k, that is, $u_k = \mathrm{vec}(\hat{U}_k)$, $v_k = \mathrm{vec}(\hat{V}_k)$.

The second strategy is to make $\tilde{\mathbf{G}}_k+\tilde{\Lambda}_k$ closer to $\mathbb{E}[(\mathcal{D}^2 s_k)]$ directly. In this case, $\tilde{\Lambda}_k$ should satisfy

$$\left(\tilde{\mathbf{G}}_k+\tilde{\Lambda}_k\right)\tilde{U}_k = \tilde{V}_k,$$

where $\tilde{U}_k = \mathrm{mat}(s_k - s_{k-1})$, $\tilde{V}_k = \mathrm{mat}(\mathcal{D}s_k - \mathcal{D}s_{k-1})$. The operator mat reshapes the arrays into matrices with correct sizes. Equivalently, the condition is as follows:

$$\tilde{\Lambda}_k\tilde{U}_k = \tilde{V}_k - \tilde{\mathbf{G}}_k\tilde{U}_k. \tag{20}$$

Under the spatial homogeneity assumptions, $\mathbb{E}[\mathcal{D}^2 s]$ is the same for all locations. Therefore, the above condition can be regarded as $|\mathcal{T}|$ secant conditions for all spatial coordinates. The condition (20) can be equivalently formed as $\left(I\otimes(\tilde{\mathbf{G}}_k+\tilde{\Lambda}_k)\right)\mathrm{vec}(\tilde{U}_k) = \mathrm{vec}(\tilde{V}_k)$. Since $\hat{\mathbf{A}}_k$ is an estimator, the two conditions (19) and (20) are actually different. Note that both conditions are in matrix formats, and they can be viewed as modified multi-secant conditions.

Recently, a practical QN method [12] also takes the kronecker factorization (18). The distinction is that they construct two quasi-Newton schemes for the two parts of the kronecker product (16) for the fully-connected layer. On the other hand, our proposed method compensates the base matrix with a quasi-Newton matrix and is also available for the convolutional layer. In the next, we show how to generate the refinement matrix by the sketchy block BFGS method.

4.2. Sketchy Block BFGS Methods

The refinement matrix Λ_{k+1} in (19) and (20) takes the same format: $\Lambda_{k+1}\mathbb{U}_k = \mathbb{V}_k$, where $\mathbb{U}_k$ and $\mathbb{V}_k$ are matrices. We use a block quasi-Newton update as follows:

$$\begin{aligned}\Lambda_{k+1} &= \text{BlockQN}\left(\Lambda_k,\mathbb{U}_k,\mathbb{V}_k,\mathbb{P}_k\right)\\ &= \Lambda_k + \mathbb{V}_k(\mathbb{P}_k)^{-1}\mathbb{V}_k^\top - \Lambda_k\mathbb{U}_k(\mathbb{U}_k^\top\Lambda_k\mathbb{U}_k)^{-1}(\mathbb{U}_k)^\top\Lambda_k.\end{aligned} \tag{21}$$

Let Λ_{k+1} be symmetric and positive definite. However, since $\mathbb{V}_k^\top\mathbb{U}_k$ is not symmetric, $\mathbb{P}_k$ can be chosen as $\frac{1}{2}\left(\mathbb{V}_k^\top\mathbb{U}_k + \mathbb{U}_k^\top\mathbb{V}_k\right)$, $\mathrm{Tr}(\mathbb{V}_k^\top\mathbb{U}_k)$ or $\mathrm{diag}(\mathbb{V}_k^\top\mathbb{U}_k)$. We use the damping strategy to make $\mathbb{P}_k$ positive definite. Specifically, by choosing proper τ_k, we replace $\mathbb{V}_k$ by $\tau_k\mathbb{V}_k + (1-\tau_k)\Lambda_k\mathbb{U}_k$. Note that when $\mathbb{P}_k = \mathbb{V}_k^\top\mathbb{U}_k$, this is indeed the block BFGS method [8] for multi-secant conditions.

Since the size of $(\mathbb{V}_k)^\top\mathbb{U}_k$ and $\mathbb{U}_k^\top\Lambda_k\mathbb{U}_k$ is large, the computation of their inverse matrices is intractable. We use

sketchy techniques to reduce the computational cost. Denote the sketching matrix by Ξ_k and let $\tilde{\mathbb{V}}_k = \Xi_k \mathbb{V}_k, \tilde{\mathbb{U}}_k = \Xi_k \mathbb{U}_k$. Accordingly, the quasi-Newton update is changed to:

$$\Lambda_{k+1} = \text{BlockQN}\left(\Lambda_k, \tilde{\mathbb{U}}_k, \tilde{\mathbb{V}}_k, \tilde{\mathbb{P}}_k\right), \tag{22}$$

where $\mathbb{P}_k$ is $\frac{1}{2}\left(\tilde{\mathbb{V}}_k^\top \tilde{\mathbb{U}}_k + \tilde{\mathbb{U}}_k^\top \tilde{\mathbb{V}}_k\right)$, $\text{Tr}(\tilde{\mathbb{V}}_k^\top \tilde{\mathbb{U}}_k)$ or $\text{diag}(\tilde{\mathbb{V}}_k^\top \tilde{\mathbb{U}}_k)$.

Our method is different from the stochastic block BFGS studied in [14]. They design a special multi-secant condition by sketchy methods to squeeze more curvature information. In addition, $\mathbb{V}_k^\top \mathbb{U}_k$ is designed to be symmetric and positive definite in [14]. However, our secant conditions (19) and (20) are constructed using the special kronecker structure and sketchy techniques are used to reduce the computational complexity.

5. Theoretical Analysis

5.1. Global Convergence

In this part, we give a general analysis for our structured stochastic quasi-Newton method under some mild assumptions. When there is no Λ_k, the method degenerates to the Newton-type method. It is obvious that our analysis fits both cases. Our global analysis is motivated by the strategies used in [10, 39, 41].

According to the stochasticity in the Algorithm 1, we can define a filtration $\mathcal{F}_k$ such that $\theta_k \in \mathcal{F}_{k-1}$, H_k and $\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)$ are $\mathcal{F}_{k-1}$-independent. A few necessary assumptions are listed below.

Assumption 2 *2.1) Ψ is continuously differentiable on $\mathbb{R}^n$ and is bounded from below by Ψ_{inf}. The gradient $\nabla\Psi$ is Lipschitz continuous on $\mathbb{R}^n$ with $L_\Psi \geq 1$.*

2.2) For any iteration k, we assume that B_k and $\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)$ are $\mathcal{F}_{k-1}$ -independent and it holds almost surely that

$$\mathbb{E}[\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)|\mathcal{F}_{k-1}] = \nabla\Psi(\theta^k).$$

2.3) It holds almost surely that the variance of stochastic gradient is bounded

$$\mathbb{E}[\|\nabla_{\mathcal{S}_g^k}\Psi(\theta_k) - \nabla\Psi(\theta_k)\|^2|\mathcal{F}_{k-1}] \leq \sigma_k^2.$$

2.4) There exists positive constant h such that for all k,

$$0 \preceq B_k \preceq hI.$$

The above assumptions are common and standard in stochastic QN methods [2, 7, 14, 39, 41]. As shown in Algorithm 1, the base matrix H_k is $\mathcal{F}_{k-1}$-independent of $\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)$ and the refinement matrix $\Lambda_k \in \mathcal{F}_{k-1}$. Hence, $B_k + \lambda_k I$ and $\nabla_{\mathcal{S}_g^k}\Psi(\theta_k)$ are $\mathcal{F}_{k-1}$-independent.

Theorem 1 *Suppose that Assumptions 2 is satisfied, the step sizes $\beta_k \equiv 1$ and the sequence $\{\alpha_k\}_{k=1}^\infty$ satisfy:*

$$\alpha_k \leq \frac{r_1}{4r_2(L_\Psi + h)}. \tag{23}$$

Then, under the conditions: $\sum \alpha_k = \infty$, $\sum \alpha_k \sigma_k^2 < \infty$, it holds for Algorithm 1 almost surely that

$$\lim_{k\to\infty} \nabla\Psi(\theta^k) = 0.$$

The proof is shown in Appendix. In addition to the condition $\sum \alpha_k \sigma_k^2 < \infty$, we can also consider the bounded variance assumption in [39] by adjusting relative conditions. We next show the convergence results for a class of objective function that satisfies the Polyak- Łojasiewicz (PŁ) condition. The PŁ condition can bound the function value by the gradient norm squares and is widely used in algorithm analysis [9, 21, 24]. It is illustrated in [11] that one-hidden neural networks and ResNets with linear activation functions satisfy the PŁ condition.

Assumption 3 ***(PŁ condition)*** *There exists a constant $c \in (0, \infty)$ for all $\theta \in \mathbb{R}^n$, such that*

$$2c(\Psi(\theta) - \Psi_{inf}) \leq \|\nabla\Psi(\theta)\|^2.$$

Theorem 2 *Suppose that Assumptions 2-3 are satisfied and the step sizes $\{\beta_k\}_{k=1}^\infty$ and the sequence $\{\alpha_k\}_{k=1}^\infty$ satisfy:*

$$\alpha_k \equiv \alpha < \min\left\{\frac{r_1}{4r_2(L_\Psi + h)}, \frac{8}{c}\right\}, \quad \beta_k \equiv 1. \tag{24}$$

In addition, assume that the variance of the stochastic gradient is decreased by a geometric speed, i.e., *$\sigma_k^2 \leq M_\sigma \zeta^k$ for some scalar M_σ and $\zeta \in (0, 1)$. It holds for Algorithm 1 almost surely that*

$$\mathbb{E}[\Psi(\theta_k)] - \Psi_{inf} \leq \mu\nu^{k-1},$$

where $\nu = \max\{\zeta, 1 - \frac{1}{16}c\alpha\}$, $\mu = \max\{\Psi(\theta_1) - \Psi_{inf}, \frac{15M_\sigma}{c}\}$.

The proof is shown in Appendix.

5.2. Local Convergence

We analyze the convergence rate of the structured quasi-Newton method in a small local neighborhood of an optimal point. Consider the case where $\nabla^2\Psi(\theta) = H(\theta) + \Pi(\theta)$ in (5)-(6) and the sequence is updated as follows:

$$\theta_{k+1} = \theta_k - B_k^{-1}\nabla\Psi(\theta_k), \tag{25}$$

where $B_k = H_{S_H^k}(\theta) + \Lambda_k$ and $H_{S_H^k} = \frac{1}{|S_H^k|}\sum_{i\in S_H^k} H_i(\theta)$. The refinement matrix Λ_k

is generated by the BFGS method and satisfies : $\Lambda_k u_{k-1} = v_{k-1}$, where $u_{k-1} = \theta_k - \theta_{k-1}$ and $v_{k-1} = \frac{1}{|\mathcal{S}_H^{k-1}|}\sum_{i\in\mathcal{S}_H^{k-1}}\left(J_f^i(\theta_k) - J_f^i(\theta_{k-1})\right)\nabla_f \ell_i(\theta_k)$. A few assumptions are listed below:

Assumption 4 *4.1) The sequence $\{\theta_k\}$ satisfies $\sum_k \|\theta_k - \theta^*\| < \infty$ a.s. for an optimal point θ^* where $\nabla^2\Psi(\theta^*)$ are positive definite and there exists $\widetilde{\lambda} > 0$ such that for $i = 1, \ldots, n$, $\Pi_i(\theta^*) \succeq \widetilde{\lambda} I$.*

4.2) The gradient $\nabla_f \ell_i(\theta)$ is bounded, the Hessian $\nabla_f^2 \ell_i(\theta)$ is bounded and Lipschitz continuous near θ^* with Lipschitz constant L_ℓ, $\forall i = 1, \ldots, N$, i.e., $\|\nabla_f \ell_i(\theta)\| \le \kappa_\ell$, $\|\nabla_f^2 \ell_i(\theta)\| \le \widetilde{\kappa}_\ell$ and $\|\nabla_f^2 \ell_i(\theta_1) - \nabla_f^2 \ell_i(\theta_1)\| \le L_\ell \|\theta_1 - \theta_2\|$, for any θ_1, θ_2 near θ^*.

4.3) The gradient $\nabla f_j^i(\theta)$ is bounded, the Hessian $\nabla^2 f_j^i(\theta)$ is bounded and Lipschitz continuous near θ^* with Lipschitz constant L_f, $\forall i = 1, \ldots, N$ and $\forall j = 1, \ldots, m$, i.e., $\|\nabla f_j^i(\theta)\| \le \kappa_f$, $\|\nabla^2 f_j^i(\theta)\| \le \tilde{\kappa}_f$ and $\|\nabla^2 f_j^i(\theta_1) - \nabla^2 f_j^i(\theta_2)\| \le L_f \|\theta_1 - \theta_2\|$ for any θ_1, θ_2 near θ^*.

The above assumptions are common in the structured quasi-Newton method for deterministic nonlinear least squares problems [44]. To establish the fast local convergence results for stochastic methods, the sample size is required to increase superlinearly. These assumptions are similar to these in the subsampled Newton method [32] and stochastic BFGS method [43]. We summarize the results as follows:

Theorem 3 *Suppose that Assumption 4 is satisfied. If the sample size $|S_H^k|$ increases superlinearly, then the sequence $\{\theta_k\}$ generated by (25) converges to θ^* superlinearly almost surely.*

The proof is shown in Appendix. Note that we do not consider the stochastic mini-batch gradient in (25). However, similar results can be established by adding assumptions on the mini-batch gradient and its sample size.

6. Numerical Experiments

In this section, we compare our proposed method with a few standard methods for logistic regression, deep autoencoders and convolutional neural networks. All methods used in the experiments are briefly listed below and their detailed implementation is reported in Appendix. **SGD** is the stochastic gradient method with momentum. **Adam** [22] is an adaptive gradient method. **L-BFGS** [28] is the well-known limited-memory quasi-Newton method. **SSN** [32] is the subsampled Newton method. **KFAC** [15] is a method using the Fisher matrix for deep learning problems. **S4QN** is the variants of S2QN where the base matrix is subsampled Newton matrix. **SKQN-L**, **SKQN-B1** and **SKQN-B2** are the variants of S2QN when the base matrices are the empirical version of KFAC matrices but with different refinement matrix constructions (12), (19) and (20), respectively.

6.1. Logistic Regression

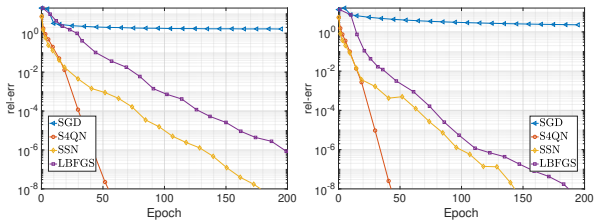
In this part, we consider logistic regression problems for binary classification:

$$\min_{\theta\in\mathbb{R}^n} \Psi(\theta) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-y_i\langle x_i, \theta\rangle) + \mu\|\theta\|^2, \tag{26}$$

where $\{x_i, y_i\} \in \mathbb{R}^n \times \{-1, 1\}$, $i \in [1, 2, \ldots, N]$ correspond to a given dataset. The statistics of the datasets used in our numerical comparisons are listed in Appendix.

We compare S4QN with SGD, SSN and L-BFGS. Let one epoch be a full pass through the dataset and define the relative error as rel-err $:= (\Psi(\theta) - \Psi^*)/\max\{1, \Psi^*\}$, where Ψ^* is the optimal function value. The changes of the relative error with respect to the number of epochs for rcv1 and news20 is shown in Figure 1.

We can observe that S4QN outperforms other methods greatly. It is worth emphasizing that all settings of S4QN are the same as SSN except an additional quasi-Newton matrix. The sample size of the gradient estimation in S4QN and SSN is increasing until the full batch. S4QN exhibits a faster local convergence rate than other methods. These facts illustrate that the structured methods can accelerate the Hessian-based and quasi-Newton methods.

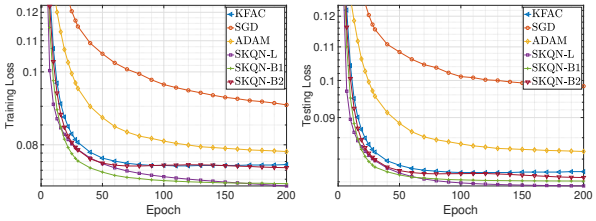


(a) rcv1 (b) news20

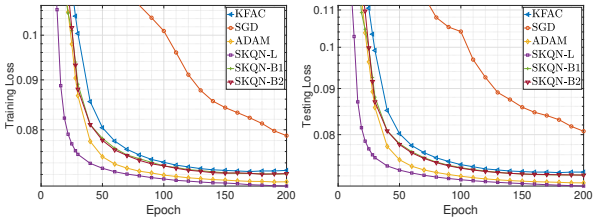
Figure 1: Logistic Regression.

6.2. Deep Autoencoders

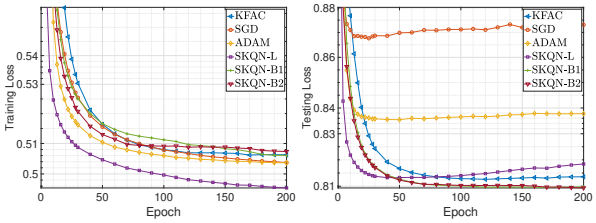
We next consider the deep autoencoder problem [18] on three datasets: "MNIST", "CURVES" and "FACES". The network architecture is D-1000-500-250-30-250-500-1000-D, where D is the dimension of the input data. We use the cross-entropy loss for CURVES and MNIST, and the square error loss for FACES. We compare SKQN-L, SKQN-B1and SKQN-B2 with SGD, ADAM and KFAC. We report the changes of the training loss and testing loss versus epochs in Figure 2.



(a) MNIST: Training Loss (b) MNIST: Testing Loss



(c) CURVES: Training Loss (d) CURVES: Testing Loss



(e) FACES: Training Loss (f) FACES: Testing Loss

Figure 2: Autoencoders.

Compared to the first-order methods SGD and Adam, our structured quasi-Newton methods are better and more stable on all the three datasets. In comparison to KFAC, our proposed methods have improvements in both the training loss and testing loss while the computational cost does not increase much at each iteration. Our results suggest that the structured quasi-Newton method with partial Hessian information indeed accelerates the convergence.

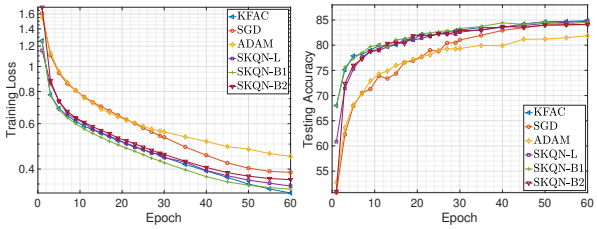
6.3. ConvNet

A 4-layer neural network "ConvNet" is considered in this part: three convolutional layers followed by a fully-connected layer. The detailed network architecture can be found in Appendix. We compare algorithms with "ConvNet" on the CIFAR10 which is a standard dataset used for numerical performance comparison in deep learning.

The changes of the training loss and testing accuracy versus epochs are reported in Figure 3. It is observed that the second-order type method is superior to the Adam and SGD. Our proposed methods have smaller training loss and larger testing accuracy in the end.

6.4. ResNet-18

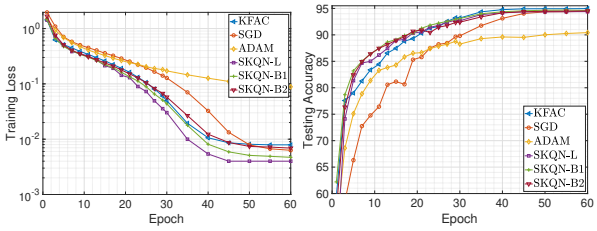
In this part, we consider the "ResNet-18" [17] with the cross-entropy loss on the dataset "CIFAR-10". ResNet is a well-known network and widely used in practice.



(a) Training Loss (b) Testing Accuracy

Figure 3: ConvNet on CIFAR-10.

The changes of the training loss and testing accuracy versus epoch of CIFAR-10 are reported in Figure 4. We can see that the second-order type methods outperform the first-order type methods in terms of both criteria. The training error of our proposed quasi-Newton methods decreases faster than that of KFAC. In terms of the testing accuracy, our proposed methods are better at start and at least comparable with KFAC in the end.



(a) Training Loss (b) Testing Accuracy

Figure 4: ResNet-18 on CIFAR-10.

7. Conclusion

In this paper, a novel S2QN framework is proposed and analyzed for large-scale finite-sum optimization problems. Since the Hessian matrix can be split as a cheap part and an expensive part, we use the structured quasi-Newton method to exploit more curvature information for the expensive part. By further exploiting either the low-rank structure or the kronecker-product properties of the approximations, the computation of the quasi-Newton direction is affordable. We obtain global convergence if the step sizes and the stochastic variances satisfy certain conditions. A local superlinear convergence result is also guaranteed under mind conditions. Our experimental results demonstrate the effectiveness of our structured stochastic quasi-Newton method compared to the state-of-the-art methods. In the future, we will implement our method on MindSpore[1], a unified training and inference framework for device, edge and cloud in Huawei's full-stack, all-scenario AI portfolio.

Acknowledgments M. Yang, D. Xu and Z. Wen are supported in part by Key-Area Research and Development Program of Guangdong Province (No.2019B121204008), the NSFC grants 11831002 and Beijing Academy of Artificial Intelligence.

[1]https://gitee.com/mindspore/

References

[1] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018. 1

[2] Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch l-bfgs method for machine learning. In *Advances in Neural Information Processing Systems*, pages 1055–1063, 2016. 3, 6

[3] E Bergou, Y Diouane, V Kungurtsev, and CW Royer. A stochastic levenberg-marquardt method using random models with application to data assimilation. *arXiv preprint arXiv:1807.02176*, 2018. 3

[4] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565, 2017. 1, 4

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[6] Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011. 1

[7] R. H. Byrd, S. L. Hansen, Jorge Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016. 3, 6

[8] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994. 5

[9] Daqing Chang, Ming Lin, and Changshui Zhang. On the generalization ability of online gradient descent algorithm under the quadratic growth condition. *IEEE transactions on neural networks and learning systems*, 29(10):5008–5019, 2018. 6

[10] Frank E Curtis and Rui Shi. A fully stochastic second-order trust region method. *arXiv preprint arXiv:1911.06920*, 2019. 3, 6

[11] Dylan J Foster, Ayush Sekhari, and Karthik Sridharan. Uniform convergence of gradients for non-convex learning and optimization. In *Advances in Neural Information Processing Systems*, pages 8745–8756, 2018. 6

[12] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. *arXiv preprint arXiv:2006.08877*, 2020. 5

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 1

[14] Robert Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block bfgs: Squeezing more curvature out of data. In *International Conference on Machine Learning*, pages 1869–1878, 2016. 3, 6

[15] Roger Grosse and James Martens. A Kronecker-factored approximate Fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582, 2016. 1, 4, 5, 7

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer-Verlag, New York, 2001. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 8

[18] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 7

[19] Jiang Hu, Bo Jiang, Lin Lin, Zaiwen Wen, and Ya-xiang Yuan. Structured quasi-newton methods for optimization with orthogonality constraints. *SIAM Journal on Scientific Computing*, 41(4):A2239–A2269, 2019. 2

[20] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013. 1

[21] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016. 6

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv:1412.6980*, 2014. 7

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015. 1

[24] Yunwen Lei, Ting Hu, Guiying Li, and Ke Tang. Stochastic gradient descent for nonconvex learning without bounded gradient assumptions. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. 6

[25] James Martens. Deep learning via Hessian free optimization. In *International Conference on Machine Learning*, pages 735–742, 2010. 1, 2

[26] Andre Milzarek, Xiantao Xiao, Shicong Cen, Zaiwen Wen, and Michael Ulbrich. A stochastic semismooth newton method for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 29(4):2916–2948, 2019. 1

[27] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621, 2017. 1

[28] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, New York, 2006. 1, 3, 7

[29] Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12359–12367, 2019. 1

[30] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017. 1

[31] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951. 1

[32] Fred Roosta and Michael W. Mahoney. Sub-sampled newton methods. *Mathematical Programming*, 174:293–326, 2019. 1, 7

[33] Nicolas L Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In *Advances in neural information processing systems*, pages 849–856. 2

[34] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. 1

[35] Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002. 1

[36] Wenyu Sun and Ya-Xiang Yuan. *Optimization theory and methods: nonlinear programming*. Springer US, 2006. 1, 3

[37] Vladimir Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, 2013. 1

[38] Chien-Chih Wang, Kent Loong Tan, and Chih-Jen Lin. Newton methods for convolutional neural networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–30, 2020. 1

[39] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017. 3, 6

[40] Peng Xu, Fred Roosta, and Michael W. Mahoney. Newton-type methods for non-convex optimization under inexact Hessian information. *Mathematical Programming*, May 2019. 1

[41] Minghan Yang, Andre Milzarek, Zaiwen Wen, and Tong Zhang. A stochastic extra-step quasi-newton method for nonsmooth nonconvex optimization. *ArXiv:1910.09373*, 2019. 1, 3, 6

[42] Minghan Yang, Dong Xu, Yongfeng Li, Zaiwen Wen, and Mengyun Chen. Sketchy empirical natural gradient methods for deep learning. *arXiv preprint arXiv:2006.05924*, 2020. 4

[43] Chaoxu Zhou, Wenbo Gao, and Donald Goldfarb. Stochastic adaptive quasi-newton methods for minimizing expected values. In *International Conference on Machine Learning*, pages 4150–4159, 2017. 7

[44] Weijun Zhou and Xiaojun Chen. Global convergence of a new hybrid gauss–newton structured bfgs method for nonlinear least squares problems. *SIAM Journal on optimization*, 20(5):2422–2441, 2010. 1, 7