Journal of the Operations Research Society of China (2023) 11:245–275
https://doi.org/10.1007/s40305-023-00453-9

An Overview of Stochastic Quasi-Newton Methods for Large-Scale Machine Learning

Tian-De Guo[1] · **Yan Liu**[2] · **Cong-Ying Han**[1]

Received: 4 May 2022 / Accepted: 15 January 2023 / Published online: 25 February 2023

Abstract
Numerous intriguing optimization problems arise as a result of the advancement of machine learning. The stochastic first-order method is the predominant choice for those problems due to its high efficiency. However, the negative effects of noisy gradient estimates and high nonlinearity of the loss function result in a slow convergence rate. Second-order algorithms have their typical advantages in dealing with highly nonlinear and ill-conditioning problems. This paper provides a review on recent developments in stochastic variants of quasi-Newton methods, which construct the Hessian approximations using only gradient information. We concentrate on BFGS-based methods in stochastic settings and highlight the algorithmic improvements that enable the algorithm to work in various scenarios. Future research on stochastic quasi-Newton methods should focus on enhancing its applicability, lowering the computational and storage costs, and improving the convergence rate.

Keywords Stochastic quasi-Newton methods · BFGS · Large-scale machine learning

Mathematics Subject Classification 90C06 · 90C53 · 65K05 · 90C90

This paper is supported by the National Key R&D Program of China (No. 2021YFA1000403), the National Natural Science Foundation of China (Nos. 11731013, 12101334 and U19B2040), the Natural Science Foundation of Tianjin (No. 21JCQNJC00030) and the Fundamental Research Funds for the Central Universities.

✉ Yan Liu
liuyan23@nankai.edu.cn

Tian-De Guo
tdguo@ucas.ac.cn

Cong-Ying Han
hancy@ucas.ac.cn

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 101408, China

[2] School of Statistics and Data Science, KLMDASR, LEBPS, and LPMC, Nankai University, Tianjin 300071, China

1 Introduction

In essence, artificial intelligence is a process of optimization. The intelligence we are trying to develop almost always comes back to the optimization problem in the end. Whichever traditional machine learning, deep learning or reinforcement learning, their core concepts can be attributed to optimization problems. As a result, optimization theory and algorithms constitute a fundamental component of machine learning and become one of its main pillar. Additionally, the objective function in machine learning always takes on a special form, making it possible to create an optimization algorithm that can effectively handle massive amounts of data.

In machine learning, it is common to encounter optimization problems involving tens of millions of training examples and variables. Consider the following general optimization problem in an expectation form,

$$\min_{w\in\mathbb{R}^d} \; F(w) = E[f(w,\xi)], \tag{1}$$

where $f:\mathbb{R}^d\to\mathbb{R}, \xi\in\mathbb{R}^{d_\xi}$ denotes a random variable with distribution $\mathcal{P}$, and $E[\cdot]$ represents the expectation with respect to ξ. In machine learning, the function $f(\cdot,\xi)$ is implicitly given or the distribution $\mathcal{P}$ is unknown, making it difficult to calculate the function value and gradient. A special case of (1) that arises frequently in machine learning is the empirical risk minimization problem,

$$\min_{w\in\mathbb{R}^d} \; F(w) = \frac{1}{n}\sum_{i=1}^{n} f(w;\xi_i), \tag{2}$$

where the training set is assumed to be a collection of independent and identically distributed (i.i.d.) samples $\xi_i = (x_i, y_i)$ with $i \in \{1,2,\cdots,n\}$ according to $\mathcal{P}$ via certain observations. And n is the number of data sample which is always extremely large.

Gradient descent (GD) is a popular approach to solve (2); it usually employs the following updates as shown in (3). But the computation cost of full gradient is expensive, so it is imperative to employ stochastic approximation (SA) algorithms to solve large-scale optimization problems, which can be traced back to the seminal work by [1]. The predominant methodology in machine learning advocates the use of stochastic gradient descent (SGD) methods [2]. In the k-th iteration, SGD chooses a subset $N_k \subset \{1,2,\cdots,n\}$ randomly and then calculates the stochastic gradient $\nabla F_{N_k}(w_k)$ as shown in (3),

$$w_{k+1} = w_k - \alpha_k g_k, \;\; g_k := \begin{cases} \nabla F(w_k) = \frac{1}{n}\sum\limits_{i=1}^{n} \nabla f(w;\xi_i), & \text{(GD)} \\ \nabla F_{N_k}(w_k) = \frac{1}{|N_k|}\sum\limits_{i\in N_k} \nabla f(w;\xi_i), & \text{(SGD)} \end{cases} \tag{3}$$

where $\alpha_k > 0$ is the k-th step size. $\nabla F_{N_k}(w_k)$ is an unbiased estimate of the gradient of F at w_k, namely $E[\nabla F_{N_k}(w_k)] = \nabla F(w_k)$.

SGD methods are widely used in machine learning [3–6]. There are also several variants, including momentum [7, 8], Nesterov-accelerated gradient [9], adaptive learning-method [10–12], etc. In practice, SGD with momentum is widely used, especially in deep learning [13, 14]. When deterministic risk components predominate in the beginning of the process, Nesterov's acceleration can speed up the rate of convergence of SGD [15, 16]. Adaptive learning methods, such as Adagrad [10], Adadelta [11], Adam [12], compute adaptive learning rates for each parameter, where Adam algorithm outperforms other adaptive learning methods. However, since α_k must decay to zero for convergence, SGD methods suffer from the adverse effect of noisy gradient estimates and slower convergence rate. To address this limitation, methods endowed with variance reduction [17] capabilities have been developed. Stochastic variance reduced methods [18] can converge linearly on strongly convex problems, including SAG [19, 20], SAGA [21], SVRG [22], SARAH [23]. SAG and SAGA need to store the auxiliary vectors for every sample, which amounts to an $\mathcal{O}(nd)$ storage. The SVRG method only requires $\mathcal{O}(d)$ memory and consists of two loops: an outer loop where the reference gradient $\tilde{g}_k = \nabla F(\tilde{w}_k)$ is calculated, and an inner loop where the stochastic gradient steps are updated using $g_t = \nabla F_{N_t}(w_t) - \nabla F_{N_t}(\tilde{w}_k) + \tilde{g}_k$.

SGD methods, which take into account only gradient information, have a number of drawbacks, including relatively slow convergence and sensitivity to hyperparameter settings. Additionally, SGD methods suffer from ill-conditioning problem and often offer limited opportunities for parallelism. Second-order optimization methods have the potential to address these well-known shortcomings by integrating curvature information [24]. As a result, some methods employ second-order derivative information; the update rule is presented as

$$w_{k+1} = w_k - \alpha_k B_k^{-1} \nabla F_{N_k}(w_k) \quad \text{or} \quad w_{k+1} = w_k - \alpha_k H_k \nabla F_{N_k}(w_k), \tag{4}$$

where B_k, H_k represent the Hessian and the inverse Hessian of object function or the corresponding approximate matrix in the first k-th iteration, respectively.

Compared with the first-order methods, the second-order methods have many advantages. The most crucial one is *scale-invariant* [17], without it, poorly scaled parameters will be much harder to optimize. Another advantages of second-order algorithm are that each iteration, whatever based on gradient or curvature, chooses the subsequent iterate by first computing the minimizer of a second-order Taylor series approximation as follows:

$$q_k(w) = F(w_k) + \nabla F(w_k)^\top (w - w_k) + \frac{1}{2}(w - w_k)^\top B(w - w_k). \tag{5}$$

The GD iteration takes $B = \boldsymbol{I}$, while Newton's method takes $B = H_k$ or $B = B_k^{-1}$, and quasi-Newton's method chooses the approximate H_k or B_k which is constructed using only gradient information.

1.1 Stochastic Second-Order Methods in Machine Learning

Numerous stochastic second-order algorithms have been proposed in recent years for finite sum functions, primarily employing random sampling techniques to approximate the true gradient and Hessian. Byrd et al. [25, 26] proposed and analyzed the complexity and sample size of a subsampled Newton-CG algorithm. Erdogdu et al. [27] explored a subsampled Newton algorithm combined with low-rank approximation when the sample size is much larger than the number of parameters, that is $n \gg d$, and analyzed its convergence rate. Utilizing random matrix concentration inequalities, Mooney et al. [28] analyzed the global and local convergence rate of subsampled Newton methods when $n, d \gg 1$. Xu et al. [29] exploited the convergence rate of subsampled Newton algorithm in nonuniform sampling schemes. Ballapragada et al. [30] analyzed the convergence rate of the subsampled Newton algorithm in expectation. Zhang et al. [31] proposed a technique combining variance reduction with subsampling to improve the convergence rate and analyzed the convergence rate when the subproblem is a quadratic function. In [32], Pilanci et al. proposed a new Newton sketch algorithm; they calculated the approximate Newton step by randomly projected Hessian and established its superlinear convergence under certain conditions. The numerical performance of Newton sketch algorithm and subsampled Newton algorithm was compared by Berahas et al. [33].

These methods we mentioned above are of the form

$$A_k p_k = -\nabla F_{N_k}(w_k), \quad w_{k+1} = w_k + \alpha_k p_k, \tag{6}$$

where $A_k \succ 0$ is a stochastic approximation of the Hessian and the conjugate gradient (CG) [34] method is always used by investigators to solve it inexactly. In subsampled Newton method,

$$A_k = \nabla^2 F_{S_H}(w_k) = \frac{1}{|S_H|} \sum_{i \in S_H} \nabla^2 f(w_k; \xi_i), \tag{7}$$

where $S_H \subset \{1, 2, \cdots, n\}$ is chosen at random either uniformly or in a nonuniform manner [29]. While Newton sketch algorithm defines a random sketch matrix $D_k \in \mathbb{R}^{q \times n}$ with the property that $E[(D_k)^\top D_k / q] = \boldsymbol{I}_n$ $(q < n)$ at each iteration and then calculates a square-root decomposition of the Hessian $\nabla^2 F(w_k)$, denoted by $\nabla^2 F(w_k)^{1/2} \in \mathbb{R}^{n \times d}$, and defines the Hessian approximation as $A_k = \left(\left(D_k \nabla^2 F(w_k)^{1/2} \right)^\top D_k \nabla^2 F(w_k)^{1/2} \right)$. Agarwal et al. [35] proposed LiSSA, a new Newton-type algorithm for generalized linear models, with the complexity increasing linearly with the amount of parameters.

In addition to the direct approximation of Hessian, the quasi-Newton approach has also been used in stochastic settings. Bordes et al. [36] proposed an SGD algorithm based on diagonal curvature estimation matrix. Byrd et al. [37] combined SGD and limited memory BFGS (L-BFGS). Stochastic L-BFGS algorithm with variance reduction was proposed by Moritz et al. [38]. Gower et al. [39] propose a stochastic block L-BFGS method with variance reduction and demonstrate its linear convergence rate.

Although the majority of these works focus on smooth objective functions, nonsmooth regularizations like the L_1 norm are frequently taken into account in practice. How to create nonsmooth stochastic second-order algorithms is a crucial challenge.

Deep learning has been shown to be resistant to algorithms with certain optimization errors in real applications. As a result, some sophisticated traditional optimization methods can be simplified to some extent, allowing the algorithm to be applied to the neural network and the training speed to be greatly accelerated. Many second-order methods for neural network, including the generalized Gaussian Newton (GGN) and the Kronecker-factored approximate curvature (K-FAC) method, have been proposed recently.

The Hessian-free (HF) and Gauss–Newton methods are two examples of GGN methods. HF [40] solves a sub-optimization problem using the conjugate gradient (CG) algorithm, which does not require explicitly forming the curvature matrix but instead uses Hessian-vector products. It was demonstrated in [41] that Hessian-free works very well for training deep neural networks (DNNs) and recurrent neural networks (RNN) if it is correctly planned and implemented. In [42], the Gauss–Newton matrix is investigated to approximate the Hessian matrix, and [43] studies a practical block-diagonal approximation to the Gauss–Newton matrix.

The Kronecker-factored approximate curvature (K-FAC) [44, 45] is an effective method for simulating natural gradient descent (NGD), with a high-quality approximation of the Fisher information matrix. NGD [46–48] is an information geometry-based second-order optimization method. It employs the Fisher information matrix (FIM) as the curvature, which provides the overall perspective of the loss function and converges faster than the first-order method. Given the cross-entropy loss function $F(w) = E[-\log p(y|x, w)]$, where x, y are the input and label, $p(y|x, w)$ represents the density function of a predictive distribution $\mathcal{P}_{y|x}$. The FIM is formulated as $F = E[\nabla_w \log p(y|x, w) \nabla_w \log p(y|x, w)^\top]$. Consider a deep neural network with ℓ layers and denote the outputs of the i-th layer as b_i, the inputs of the i-th as a_{i-1} and a weight matrix of the i-th layer as W_i. The precise computation performed at each layer: $b_i = W_i a_{i-1}$, $a_i = \phi_i(b_i)$, where ϕ is an element-wise nonlinear function. Define $w = \left[\text{vec}\,(W_1)^\top \text{vec}\,(W_2)^\top \cdots \text{vec}\,(W_\ell)^\top\right]^\top$, $\mathcal{D}v = -\frac{\mathrm{d}\log p(y|x,w)}{\mathrm{d}v}$ and $g_i = \mathcal{D}b_i$. In the first step, K-FAC approximates FIM into block matrix:

$$
\begin{aligned}
F &= E[\mathcal{D}w \mathcal{D}w^\top] \\
&\approx \text{diag}\,(F_1, F_2, \cdots, F_\ell) \\
&= \text{diag}\left(E\left[\text{vec}(\mathcal{D}W_1)\,\text{vec}(\mathcal{D}W_1)^\top\right], \cdots, E\left[\text{vec}(\mathcal{D}W_\ell), \text{vec}(\mathcal{D}W_\ell)^\top\right]\right).
\end{aligned}
$$

Noting that $\mathcal{D}W_i = g_i a_{i-1}$, that is $\text{vec}(\mathcal{D}W_i) = \text{vec}(g_i a_{i-1}) = a_{i-1} \otimes g_i$, then each block of FIM can be rewritten as

$$
\begin{aligned}
F_i &= E\left[\text{vec}(\mathcal{D}W_i)\,\text{vec}(\mathcal{D}W_i)^\top\right] = E\left[a_{i-1}a_{i-1}^\top \otimes g_i g_i^\top\right] \\
&\approx E\left[a_{i-1}a_{i-1}^\top\right] \otimes E\left[g_i g_i^\top\right] = A_{i-1} \otimes G_i.
\end{aligned}
$$

Since $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ for any matrices A and B, we can compute the block-diagonal FIM easily as

$$F_i^{-1} = (A_{i-1} \otimes G_i)^{-1} = A_{i-1}^{-1} \otimes G_i^{-1}.$$

K-FAC was first proposed for MLPs [44] and then extended to CNNs [45]. An eigenvalue-corrected Kronecker factorization (EKFAC), which can approximate FIM significantly better than K-FAC, was developed by George et al. [49]. Shampoo [50] extends adaptive learning rate methods by using a block-diagonal Kronecker-factored preconditioning matrix. However, in fact, the exact strategies used by the present approximation scheme for the inverse of FIM still demand a lot of processing resources. Recently, Chen et al. [51] proposed a novel Trace-based Hardware-driven layer-ORiented natural gradient descent computation method (THOR), in which the update interval is gradually increased and the matrix trace is used to determine which blocks of FIM need to be updated. The K-FAC algorithm simplifies the natural gradient method and obtains a Kronecker product approximation. In practical implementation, it uses block-diagonal or block-tridiagonal approximation to obtain the second-order optimization approach for neural network, which significantly speeds up neural network training.

Stochastic second-order methods make judicious use of curvature information and have proven to be effective for a variety of machine learning tasks [24]. Existing stochastic second-order methods can be classified as follows: stochastic Newton methods [25–30, 33, 52–55]; stochastic quasi-Newton methods (SQNM) [37, 52, 56–61]; generalized Gauss–Newton (GGN) methods [40, 41, 43, 62–64]; Kronecker-factored approximate curvature (K-FAC) methods [44–51, 65, 66].

2 The Deterministic Quasi-Newton Methods

Numerous efforts have been done to develop quasi-Newton methods that incorporate the curvature information of the objective function without computing second derivatives, including the symmetric rank-1 method (SR1) [67–69], DFP rule of Davidon [70] and Fletcher and Powell [71], the Greenstadt [72] rule, and others. The most well-known one is the BFGS method, which is now a fundamental component of many modern optimization techniques and is named after Broyden, Fletcher, Goldfarb, and Shanno [73–76] who proposed the algorithm independently at nearly the same time. The BFGS update has a number of exceptional qualities, including "self-correcting" qualities [69]. As a result, BFGS updating is currently regarded as the most effective of all quasi-Newton updating formulas.

2.1 The BFGS and L-BFGS

The BFGS method iteratively updates an estimate of the inverse Hessian, $H_k = B_k^{-1}$. When the curvature condition is met, the secant equation (8) always has a solution

H_{k+1},

$$w_{k+1} - w_k = H_{k+1} \left(\nabla F(w_{k+1}) - \nabla F(w_k) \right). \tag{8}$$

We require H_{k+1} to be, in certain ways, the closest to the current matrix H_k in order to determine H_{k+1} uniquely. In other words, we solve the problem below

$$\begin{aligned} &\min_{H} \|H - H_k\| \\ &\text{s.t.} \quad H = H^\top, \quad Hy_k = s_k, \end{aligned} \tag{9}$$

where $s_k = w_{k+1} - w_k$, $y_k = \nabla F(w_{k+1}) - \nabla F(w_k)$. Different matrix norms can be used in (9), with the weighted Frobenius norm; we can derive the unique solution,

$$H_{k+1} = \left(\boldsymbol{I} - \rho_k s_k y_k^\top \right) H_k \left(\boldsymbol{I} - \rho_k y_k s_k^\top \right) + \rho_k s_k s_k^\top, \tag{10}$$

that is

$$H_{k+1} = V_k^\top H_k V_k + \rho_k s_k s_k^\top, \tag{11}$$

where $\rho_k = \frac{1}{y_k^\top s_k}$ and $V_k = \boldsymbol{I} - \rho_k y_k s_k^\top$, applying the Sherman–Morrison–Woodbury formula [77] to (10), we obtain

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}. \tag{12}$$

Nocedal and Liu [69, 78, 79] proposed the idea of only using the most recent iterates and gradients in forming the inverse Hessian approximation to handle large-scale optimization problems. By storing only a small number of vectors of length d which implicitly represent the approximations rather than fully dense $d \times d$ approximations, these methods retain straightforward and compact approximations of Hessian matrices.

Algorithm 1 Deterministic BFGS Method

```
initial parameter w_0, inverse Hessian approximation H_0,ε > 0;
k = 0;
while ‖∇F(w_k)‖ > ε do;
    Compute search direction p_k ← −H_k ∇F(w_k);
    Set w_{k+1} = w_k + α_k p_k where α_k is computed from a line search procedure;
    Define s_k = w_{k+1} − w_k and y_k = ∇F(w_{k+1}) − ∇F(w_k);
    H_{k+1} = (I − ρ_k s_k y_k^⊤) H_k (I − ρ_k y_k s_k^⊤) + ρ_k s_k s_k^⊤
    k = k + 1;
end while
```

By saving a certain amount (m) of the vector pair $\{s_i, y_i\}$ used in formulas (10) and (11), they implicitly store a modified version of H_k. A sequence of inner products and vector summations involving ∇F_k and the pairs $\{s_i, y_i\}$ for $i = k - m, \cdots, k - 1$ can be used to obtain the product $H_k \nabla F_k$. The L-BFGS approximation H_k satisfies the

following formula when choosing an initial Hessian approximation H_k^0:

$$\begin{aligned} H_k = & \left(V_{k-1}^\top \cdots V_{k-m}^\top\right) H_k^0 \left(V_{k-m} \cdots V_{k-1}\right) \\ & + \rho_{k-m} \left(V_{k-1}^\top \cdots V_{k-m+1}^\top\right) s_{k-m} s_{k-m}^\top \left(V_{k-m+1} \cdots V_{k-1}\right) \\ & + \rho_{k-m+1} \left(V_{k-1}^\top \cdots V_{k-m+2}^\top\right) s_{k-m+1} s_{k-m+1}^\top \left(V_{k-m+2} \cdots V_{k-1}\right) \\ & + \cdots \\ & + \rho_{k-1} s_{k-1} s_{k-1}^\top. \end{aligned} \tag{13}$$

The oldest vector pair in the collection of pairs $\{s_i, y_i\}$ is replaced by the new pair $\{s_k, y_k\}$ acquired from the current step after the new iteration has been computed. The limited-memory BFGS algorithm can be stated formally as Algorithm 2 with the two-loop recursion [69].

Algorithm 2 Deterministic L-BFGS

Choose starting point w_0, integer $m > 0$;
$k \leftarrow 1$;
while no converge **do**
  Choose H_k^0;
  Compute $p_k \leftarrow -H_k \nabla F_k$ from two-loop recursion;
  Computer $w_{k+1} \leftarrow w_k + \alpha_k p_k$, where α_k is computed from a line search procedure;
  if $k > m$ **then**
    Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage;
  end if
  Compute and save $s_k \leftarrow w_{k+1} - w_k$, $y_k \leftarrow \nabla F_{k+1} - \nabla F_k$;
  $k \leftarrow k + 1$;
end while

3 The Challenges of Quasi-Newton Methods in Stochastic Settings

SGD methods are widely used to solve (2), but they may not be suitable for ill-conditioned and nonconvex problems, which are better treated with second-order information. Although quasi-Newton algorithms have been extensively studied in the deterministic case, can we directly apply them to solve optimization problems in machine learning? When we investigate the extension of a quasi-Newton method from the deterministic setting to the stochastic setting, the answer is not encouraging, there are some difficulties:

(1) Gradient measurements are noisy. The noise in the gradients in the stochastic setup may taint the correction pairs $\{s_i, y_i\}$. Let $g_k = \nabla F(w_k) + e_k$, if e_k is larger than the g_k or if $\|s_i\|$ is too small, then the dominant of "difference of two successive gradients" is just the difference of noise, which indicates that the vector y_i can be very inaccurate.

(2) Line search is highly problematic. In the stochastic context, neither $F(w)$ nor $\nabla F(w)$ is available to us; instead, we only have access to noisy versions of them, and the global validity of the criteria they employ cannot be established from local subsamples. For instance, we may accept a step size α_k in the case where the observed cost has sufficiently decreased, but the true objective function may have increased [80]. However, without line search, there is no guarantee that the curvature condition $s_i y_i > 0$ holds.
(3) The objective function is nonconvex or/and nonsmooth. Several objective functions in machine learning are only convex, not strongly convex. Additionally, nonconvex optimization has fundamental practical value in applications like those resulting from the usage of neural networks. The main obstacle to overcome when creating stochastic quasi-Newton algorithms for nonconvex problems in machine learning is how to preserve the positive definiteness of B_k (or H_k).
(4) The computational cost is prohibitively expensive. To store and update B_k, BFGS requires $\mathcal{O}(d^2)$ space, whereas L-BFGS requires only $\mathcal{O}(md)$ space and time per iteration. However, variables in machine learning are highly dimensional, and there are a tremendous amount of samples.

The purpose of this paper is to provide a review on the stochastic quasi-Newton methods that have been demonstrated in both theory and practice to be effective at counteracting the negative impacts of challenging curvature in stochastic optimization. This paper attempts to demonstrate how to overcome the aforementioned difficulties in order to design more widely applicable stochastic quasi-Newton methods.

4 Basic Stochastic Quasi-Newton Methods

To solve (1) or (2), a number of stochastic quasi-Newton methods have been presented in recent years. In this section, we introduce the basic SQNM which draws lessons from the ideas of [59].

4.1 Sublinear Convergence

Online BFGS (oBFGS) [52] is a pioneering work in stochastic adaptations of the BFGS method. The oBFGS algorithm is a direct generalization of the BFGS algorithm that employs stochastic gradients rather than full gradients. It took consistent gradient measurements on the same data set N_k, i.e., $\nabla F_{N_k}(w_{k+1}) - \nabla F_{N_k}(w_k)$, which is named "gradient displacements". The update of B_k is then modified by scaling down the last term of the update (10) by a factor $0 < \sigma \leqslant 1$ and selecting the step size without line search. To deal with the small eigenvalues, they add λs_k ($\lambda \geqslant 0$) to y_k (which means modifying the BFGS update to estimate the inverse of $H + \lambda \boldsymbol{I}$), that is

$$y_k = \nabla F_{N_k}(w_{k+1}) - \nabla F_{N_k}(w_k) + \lambda s_k.$$

Algorithm 3 Stochastic BFGS Methods

initial parameter w_0, sequence of step size $\alpha_k > 0$, inverse Hessian approximation H_0;
$k = 0$;
while no converge **do**;
  Compute $p_k \leftarrow -H_k \nabla F_{N_k}(w_k)$;
  Set $w_{k+1} = w_k + \alpha_k p_k$;
  Compute $s_k = w_{k+1} - w_k$;
  Compute y_k by gradient displacements or Hessian action as (17);
  Compute H_{k+1};
  $k = k + 1$;
end while
return w_k.

The online L-BFGS (oL-BFGS) was also proposed in [52] for solving large-scale optimization problems when the $\mathcal{O}(d^2)$ cost of storing and updating H_k would be prohibitively expensive. oL-BFGS reduces the memory requirements as well as the computational cost of each iteration to $\mathcal{O}(md)$. Under the assumption that the objective function is strongly convex and smooth, and the second moment of the stochastic gradient is bounded, oL-BFGS converges to the optimal solution at a rate of $\mathcal{O}(1/k)$ in expectation [81].

To construct the correction pairs, stochastic quasi-Newton (SQN) [37] decouples the computation of the stochastic gradient from the curvature estimate. This method calculates y_k using a Hessian-vector product [25] as (17), which is referred to as "Hessian action" [82]. The results of numerical experiments show that SQN is reliable and effective, although it does not improve convergence rate. In Algorithm 3, we formalize the sublinear convergent stochastic BFGS algorithms [37, 52, 56, 81].

4.2 Linear Convergence

Despite being successful in extending the use of quasi-Newton methods to stochastic settings, the convergence rate of the aforementioned methods is sublinear. This is not better than the SGD convergence rate. The negative impact of noisy gradient estimations results in a slow, sublinear rate of SGD convergence [17]. Both in theory and in practice, stochastic variance reduced methods produce a faster convergence than SGD [18]. A stochastic L-BFGS algorithm [38] that extensively draws on SQN and incorporates variance reduction [22] was proposed. This is the first linearly convergent stochastic quasi-Newton algorithm (Algorithm 4). To calculate the search direction, this algorithm computes a variance-reduced gradient as shown below

$$g_t = \nabla F_{N_t}(w_t) - \nabla F_{N_t}(\tilde{w}) + \tilde{g}_k, \tag{14}$$

where $\tilde{g}_k$ is the full gradient at $\tilde{w}$. Variance-reduced gradient is also used by the VITE algorithm [83], which combines regularized stochastic BFGS (RES) [56]/oBFGS with semi-stochastic gradient descent (S2GD) [84]. Using a coordinate transform framework, Zhao et al. [85] revisited the algorithms and enhanced the results of its convergence rate.

Algorithm 4 Stochastic Quasi-Newton Methods with Variance-Reduced Gradient

initial parameter $\tilde{w}_0$, step size $\alpha > 0$, parameters m, M and C;
$r = 0$;
for $k = 0, 1, 2, \cdots$ **do**
  Compute a full gradient $\tilde{g}_k = \nabla F(\tilde{w}_k)$
  Set $w_0 = \tilde{w}_k$
  for $t = 0, \cdots, m-1$ **do**
    Compute a variance reduced gradient g_t
    Set $w_{t+1} = w_t - \alpha H_r g_t$
    if $t \equiv 0 \; mod \; C$ **then**
      Set $r \leftarrow r + 1$
      $\bar{w}_r = \frac{1}{C} \sum_{j=t-C}^{t-1} w_j$
      Compute $s_r = \bar{w}_r - \bar{w}_{r-1}$ and y_r by Hessian action;
      Compute H_r
    end if
  end for
  Set $\tilde{w}_{k+1} = w_m$
end for

Gao [86] extends the standard BFGS by incorporating information of $\nabla^2 F(w)$ along multiple directions in each update to improve the accuracy of the local Hessian approximation. Gower et al. [39] then presented a new quasi-Newton approach that uses stochastic block BFGS updates [86] in combination with SVRG.

4.3 Superlinear Convergence

The incremental quasi-Newton (IQN) [58] method is the first stochastic quasi-Newton method to achieve superlinear convergence. The information corresponding to the i-th function f_i at step k is referred to as z_i^k, $\nabla f_i(z_i^k)$ and B_i^k (for simplifying notation, let $f_i(w) = f(w; \xi_i)$). Consider the second-order approximation of the objective function $f_i(w)$, which is centered around the current iterate z_i,

$$f_i(w) \approx f_i\left(z_i^k\right) + \nabla f_i\left(z_i^k\right)^\top \left(w - z_i^k\right) + \frac{1}{2}\left(w - z_i^k\right)^\top \nabla^2 f_i\left(z_i^k\right)\left(w - z_i^k\right),$$

then the function $F(w)$ can be approximated with

$$F(w) \approx \frac{1}{n}\sum_{i=1}^{n}\left[f_i(z_i^k) + \nabla f_i(z_i^k)^\top (w - z_i^k) + \frac{1}{2}(w - z_i^k)^\top B_i^k (w - z_i^k)\right]. \quad (15)$$

As a result, the updated iterate w_{k+1} can be defined as the minimizer of the quadratic programming in (15), which is explicitly given by

$$w^{k+1} = \left(\frac{1}{n}\sum_{i=1}^{n} B_i^k\right)^{-1}\left[\frac{1}{n}\sum_{i=1}^{n} B_i^k z_i^k - \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(z_i^k)\right]. \quad (16)$$

It should be noted that the update in (16) demonstrates that the variable w_{k+1} is a function of the stored information of all functions $f_1, \cdots, f_n$. As a result, they only update the local information of one function, chosen in a cyclic manner, in each iteration of the IQN method. Let i_k be the index of the function selected at time k. Then, the update of BFGS can be used to compute the Hessian approximation $B_{i_k}^k$ corresponding to f_{i_k} while leaving all other the same, i.e., $B_i^{k+1} = B_i^k$ for $i \neq i_k$. It is obviously that the update of IQN cannot be implemented at a low computational cost because it necessitates computation of the sums $\frac{1}{n}\sum_{i=1}^n B_i^k$, $\sum_{i=1}^n B_i^k z_i^k$, as well as the inversion $\left(\frac{1}{n}\sum_{i=1}^n B_i^k\right)^{-1}$. This restricts the use of the algorithms, though they discuss an efficient implementation of IQN that costs $\mathcal{O}(d^2)$.

The disadvantage of incremental second-order methods like IQN and stochastic Newton method (SNM) [87] is that they have computational and memory costs per iteration that are greater than or equal to $\mathcal{O}(d^2)$. This is prohibitive in situations where the number of model parameters is large. Chen [88] creates a new stochastic average Newton (SAN) method that is inexpensive to implement when solving regularized generalized linear models, with an iteration cost of the order of $\mathcal{O}(d)$.

5 Practical Considerations

The algorithmic advancements made by various SQNM are outlined in this section, along with a number of implementation-related topics including selecting the step size and how to generate the correction pairs.

5.1 The Correction Pairs

The updating rules for the correction pairs represent the primary distinction between various SQNM. Both oBFGS and RES compute y on the same subset N_k, i.e., $y_k = \nabla F_{N_k}(w_{k+1}) - \nabla F_{N_k}(w_k)$ as opposed to the variation $\nabla F_{N_{k+1}}(w_{k+1}) - \nabla F_{N_k}(w_k)$, even though the former requires twice as many stochastic gradient evaluations and the latter is insufficient in the stochastic scenario to ensure convergence [56, 81].

However, if the size of N_k is too small, the gradient estimates will be extremely noisy, which will have a negative impact on the final Hessian approximation. Berahas et al. [89, 90] proposed to construct the correction pairs by computing gradients based on the overlap of successive batches $O_k = N_k \cap N_{k+1} \neq \varnothing$. The only restriction is that this overlap should not be too small. However, a nonuniform sampling strategy can produce a sample size that is considerably less than what uniform sampling calls for

$$\begin{aligned} \text{gradient displacements} \quad & y_k = \nabla F_{N_k}(w_{k+1}) - \nabla F_{N_k}(w_k), \\ \text{Hessian action} \quad & y_k = \nabla^2 F_{S_H}(\bar{w}_k) s_k. \end{aligned} \tag{17}$$

Another popular way to construct the correction pairs is proposed in SQN [37], which decouples the stochastic gradient and curvature estimate calculation. This approach calculates y_k via a Hessian-vector product [25], named "Hessian action" [82], where s_k is the difference of disjoint average between the $2C$ most recent iter-

ations which means that they compute correction pairs every C iterations, $\bar{w}_k = \frac{1}{C}\sum_{j=t-C}^{t-1} w_j$. And $\nabla^2 F_{S_H}(w_k)$ is a subsampled Hessian defined as (7). We can use the directional derivative [91] to calculate the Hessian-vector product at a low cost without explicitly constructing $\nabla^2 F_{S_H}(w)$. Hessian actions can prevent the potential negative consequences of differencing noisy gradients, but at the cost of more computation because the batch size $|S_H|$ must be selected to be large enough to get useful curvature estimates. In a similar vein, AdaQN [92] computed y by matrix-vector product for training RNNs rather than using the subsampled Hessian but instead used the empirical Fisher information matrix [46, 48]. Hessian action offers a number of interesting qualities. First, since this cost is spread out over C iterations, more computation may be done for the curvature computation. Additionally, using the Hessian action allows for a more robust estimation of curvature, especially when $\|s\|$ is small and the gradients are noisy.

Berahas et al. [93] propose that new curvature pairs be sampled at each iteration. They choose random directions (τ_i) to sample points around the current iteration and establish the iteration displacement $s = \mathbf{r}\tau_i$ (where $\mathbf{r}$ is the sampling radius) before generating y using gradient displacements $\nabla F_N(w) - \nabla F_N(w + \mathbf{r}\tau_i)$ or Hessian action (17). By utilizing parallel/distributed computing settings, this approach can capture more recent and local information to improve the Hessian approximations. Another noteworthy point is that the convergence result for nonconvex problems can be proven because we can force the curvature pairs to meet the curvature requirement $s^\top y > \varepsilon\|s\|^2 > 0$ by eliminating those that do not.

5.2 The Choice of H_0

The initial approximation H_0 has no universally effective magic formula in all cases. Most stochastic BFGS methods choose a small multiple of identity matrix $\boldsymbol{I}$ as H_0, i.e., $\varepsilon\boldsymbol{I}$. Stochastic L-BFGS methods always set $H_k^0 = \gamma_k \boldsymbol{I}$, $\gamma_k = (s_{k-1}^\top y_{k-1})/(y_{k-1}^\top y_{k-1})$ as the standard L-BFGS. Byrd et al. [25] propose to employ a conjugate gradient iteration as subsampled Newton methods to indirectly define H_k^0. According to some works [94, 95], H_k^0 is defined as follows:

$$H_k^0 = \left(\max\left(\underline{\gamma}_k, \min\left(\hat{\gamma}_k, \bar{\gamma}_k\right)\right)\right)\boldsymbol{I},$$

where $0 < \underline{\gamma}_k < \bar{\gamma}_k$ can be constants or iteration-dependent to prevent H_k^0 from becoming nearly singular or nonpositive-definite.

AdaQN [92] was proposed for training RNNs, in which the inverse-Hessian matrix is initialized using accumulated gradient information. It is the same as the scaling matrix used by Adagrad [10] during each iteration

$$\left[H_k^{(0)}\right]_{ii} = \frac{1}{\sqrt{\sum_{j=0}^{k}\left[g_j\right]_i^2 + \varepsilon}}, \forall i = 1, \cdots, d.$$

5.3 Hessian or Inverse Hessian Approximation

As is well known, B_{k+1} or H_{k+1} cannot be determined just by the secant equation. The quasi-Newton methods require the matrix B_{k+1} to be close to the matrix B_k in order to resolve this indeterminacy. Mokhtari et al. [56] proposed that in BFGS, the matrix B_{k+1} is selected as the one that satisfies the secant condition while being closest to B_k in terms of the Gaussian differential entropy,

$$\begin{aligned} B_{k+1} = \underset{B}{\operatorname{argmin}} \ & \operatorname{tr}[B_k^{-1}(B)] - \log\det[B_k^{-1}(B)] - d \\ \text{s.t. } & Bs_k = y_k, \quad B \succeq 0. \end{aligned} \tag{18}$$

In the stochastic setting, they replace B with $B - \delta \boldsymbol{I}$ in (18) to prevent the smallest eigenvalue of B_k to from approaching 0 over time. Furthermore, RES establishes the corrected variation $\tilde{y}_k = y_k - \delta s_k$ to ensure that all eigenvalues of B_{k+1} exceed $\delta > 0$, in other words, if $\tilde{y}_k^\top s_k = (y_k - \delta s_k)^\top s_k$ is positive, then all eigenvalues of B_{k+1} are larger than δ. B_{k+1} can be explicitly given by the expression

$$B_{k+1} = B_k + \frac{\tilde{y}_k \tilde{y}_k^\top}{s_k^\top \tilde{y}_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \delta \boldsymbol{I}.$$

In order to determine H_k, Adrian et al. [96, 97] employ a method similar to standard BFGS (9) and solve the regularized least-square problem as follows:

$$H_k = \arg\min_H \|HY_k - S_k\|_F^2 + \lambda \left\|H - \bar{H}_k\right\|_F^2, \tag{19}$$

where $Y_k \triangleq \left[y_{k-m+1}, \cdots, y_k\right]$, $S_k \triangleq \left[s_{k-m+1}, \cdots, s_k\right]$. The regulator matrix $\bar{H}_k$ acts as a prior on H and can be modified at each iteration k. It was confirmed that the solution to Eq. (19) is given by

$$H_k = \left(\lambda \bar{H}_k + S_k Y_k^\top\right)\left(\lambda \boldsymbol{I} + Y_k Y_k^\top\right)^{-1}.$$

Stocastic block BFGS [39] constructs an update which satisfies a *sketched* version of the equation, namely

$$H_k \nabla^2 F(w_k) D_k = D_k,$$

where $D_k \in \mathbb{R}^{d \times q}$ is a randomly generated matrix, which has relatively few columns ($q \ll d$). Then, the stochastic block BFGS update is defined by the weighted projection

$$\begin{aligned} H_k = \arg \ & \min_{H \in \mathbb{R}^{d \times d}} \|H - H_{k-1}\|_k^2 \\ \text{s.t. } & H \nabla^2 F_{S_H}(w_k) D_k = D_k, H = H^\top, \end{aligned} \tag{20}$$

where $\|H\|_k^2 \triangleq \mathrm{Tr}(H\nabla^2 F_{S_H}(w_k)H^\top \nabla^2 F_{S_H}(w_k))$, and tr denotes the trace. The solution is

$$H_k = D_k \Delta_k D_k^\top + (\boldsymbol{I} - D_k \Delta_k Y_k^\top) H_{k-1} (\boldsymbol{I} - Y_k \Delta_k D_k), \tag{21}$$

where $\Delta_k \triangleq (D_k^\top Y_k)^{-1}$ and $Y_k \triangleq \nabla^2 F_{S_H}(w_k) D_k$, the same solution was given in [98–100] for multiple secant equations. The numerical results show that stochastic bock BFGS is more flexible. Similarly, Ma et al. [101] utilize the weak secant equation [102] to build an adaptive parameter-wise diagonal quasi-Newton for training DNNs. All existing quasi-Newton algorithms, according to Hennig et al. [99], can be reformulated and extended into a probabilistic interpretation. By utilizing this discovery, known as the Gaussian prior Hessian approximation, Wills et al. [103] provide a probabilistic quasi-Newton approach; more details are available in [80, 103].

5.4 The Step Size

The necessity to choose an appropriate step size is still another significant obstacle to the development of stochastic quasi-Newton algorithms. The groundbreaking work of Robbins and Monro [1] made the following conclusion: the step size in stochastic situations should satisfy the conditions

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

It is always necessary to use a "sufficiently small" constant or a diminishing step size, but it might be challenging to choose the appropriate constant step size. Therefore, the most popular option is that

$$\alpha_k = \frac{k_0}{k_0 + k} \alpha_0, \tag{22}$$

where $\alpha_0, k_0 > 0$ are tuning parameters. SQNM, when combined with a variance-reduced gradient, always employs a constant step size. However, we concentrate on adaptive step size and stochastic line search (SLS).

Zhou et al. [104] develop a stochastic adaptive BFGS (SA-BFGS) for *self-concordant function* [105]. They examine whether the Wolfe condition is met for the adaptive step size α_k in the method. Otherwise, SA-BFGS will switch back to taking an SGD step. Since we only have access to a noisy version of the gradient in the stochastic setting, it is difficult to ensure that the update direction is a descent direction. In [106], an available SLS algorithm was proposed, which employs the framework of Gaussian processes and Bayesian optimization. The step length that best satisfies a probabilistic measure combining reduction in the cost function with satisfaction of the Armijo condition is chosen [80].

The key to designing a SLS is ensuring that there is a decrease in the true function value with a high probability when only stochastic approximations can be observed. Based on variance estimates, sample size selection makes the angle between search direction p_k and $\nabla F(w_k)$ is an acute angle with high probability, that

is $E[p_k^\top \nabla F(w_k)] < 0$. Sample size selection is always used in the design of SLS and is critical in both the evaluation of gradients and the incorporation of curvature information [26, 107–110]. A practical inner product quasi-Newton (IPQN) test was proposed in [111] to ensure the stochastic quasi-Newton search direction makes an acute angle with the true quasi-Newton direction in expectation

$$E\left[\left((H_k\nabla F(w_k))^\top\left(H_k g_k\right) - \|H_k\nabla F(w_k)\|^2\right)^2\right] \leqslant \theta^2\|H_k\nabla F(w_k)\|^4,$$

where $\theta > 0$. Progressive Batching L-BFGS [111] performs a backtracking line search that aims to satisfy the Armijo condition

$$F_{N_k}\left(w_k - \alpha_k H_k g_k^{N_k}\right) \leqslant F_{N_k}(w_k) - c\alpha_k (g_k^{N_k})^\top H_k g_k^{N_k}, \tag{23}$$

where $c > 0$. If the condition is not satisfied, set $\alpha_k = \alpha_k/2$.

The line search proposed by Wills et al. [80] is conceptually more similar to this procedure. All existing quasi-Newton algorithms, according to Hennig et al. [99], can be interpreted as maximizing a posteriori estimates. Based on this, Wills et al. [80] proposed a Gaussian prior quasi-Newton approximation and a mechanism to ensure that the search direction is a descent direction in expectation, $E[p_k^\top \nabla F(w_k)] < 0$. Then, they developed a stochastic line search procedure that satisfies an Armijo condition in expectation for early iterations

$$E[\hat{f}(w_k + \alpha p_k) - \hat{f}(w_k) - c\alpha_k g_k^\top p_k] \leqslant 0,$$

where $\hat{f}(w_k)$ is subsampled function. However, this method assumes that the gradient is tainted by additive Gaussian noise, which is inappropriate for some problem classes, particularly when the noise distribution has heavy tails [80].

Xie et al. [112] demonstrated that there is a step size that satisfies the Armijo–Wolfe conditions for both the noisy and true objective functions under the assumption of the errors in function and gradient are bounded for all w. However, in order to guarantee that the condition number of the Hessian approximations is bounded, they need to be aware of the strong convexity parameter, which is typically unknown. A new noise-tolerant quasi-Newton algorithm was proposed by Shi et al. [61], which has no need for exogenous function information. The noise-tolerant L-BFGS is linearly convergent to a neighborhood of the solution determined by the noise level if the objective function F is μ-strongly convex and has L-Lipschitz continuous gradients.

5.5 Diagonal Approximation

Rapid training progress is possible with diagonally scaled stochastic first-order algorithms like Adagrad [10], Adadelta [11], RMSprop, Adam [12]. Numerous SQNM also use a diagonal matrix to approximate the Hessian. Bordes et al. [36] studied SGD with a diagonal rescaling matrix based on the secant condition, named SGD-QN. The approach is highly effective and successful enough to be named the winner of the first

PASCAL Large Scale Learning Challenge [113] because it just requires scalar computation. The corrected SGD-QN algorithm was then proposed in [114] to reap the full benefits of a diagonal scaling strategy. The Hessian is also approximated by Apollo [101] via a diagonal matrix, the elements of which are obtained by the variational method under the constraints of the parameter-wise weak secant equation. In terms of convergence speed and generalization performance, Apollo outperforms SGD and different variants of Adam significantly.

5.6 Regularization

oBFGS may produce divergent sequences when the noise of stochastic gradients is catastrophically amplified in curvature estimation. Regularization is incorporated into RES [56] to address this problem. It redefines B_{k+1} as the solution of the problem as follows:

$$\begin{aligned} B_{k+1} = \underset{B}{\operatorname{argmin}} \operatorname{tr}[B_k^{-1}(B - \delta \boldsymbol{I})] - \log\det[B_k^{-1}(B - \delta \boldsymbol{I})] - d \\ \text{s.t.} \quad B s_k = \tilde{y}_k, \quad B \succeq 0. \end{aligned} \tag{24}$$

The term $B - \delta \boldsymbol{I}$ rather than B in (24) is to preserve the smallest eigenvalue of B_k from 0. The identity bias term $\Gamma \boldsymbol{I}$ is added to the update of RES as follows:

$$w_{k+1} = w_k - \alpha_k (B_k^{-1} + \Gamma \boldsymbol{I}) \nabla F_{N_k}(w_k), \tag{25}$$

which is always used to ensure the positive definiteness of the Hessian approximation. Under the same supposition as oBFGS, the RES method converges to the optimal solution at a rate of $\mathcal{O}(1/k)$ in expectation [56]. To address the merely convex or non-Lipschitz problem, Yousefian et al. [60, 115–118] construct a set of SQNM with regularization terms, in which the regularization parameter is updated iteratively and decays to zero.

5.7 Parallel Implementations

Stochastic quasi-Newton methods are more memory intensive and more expensive (per iteration) than SGD. As a result, it is critical to effectively scale and parallelize SQNM in a distributed system. By avoiding the pricey dot product operations in the two-loop recursion, vector-free L-BFGS [119] significantly increases computing performance while using MapReduce. On the High-Performance Computing Cluster (HPCC) Systems platform, Najafabadi et al. [120] provide a parallelized version of the L-BFGS algorithm. Some of the computing nodes responsible for evaluating the function and gradient in parallel implementations are unable to deliver results on schedule. By performing quasi-Newton update depending on the overlap between succeeding batches, the multi-batch L-BFGS [89, 90] is intended to operate in a distributed setting to address this. They also proposed sampled L-BFGS and sampled L-SR1 methods [93], both of which have enough concurrency to benefit from parallel/distributed com-

puting environments. The sampled L-SR1 approach is then described by Jahani et al. [121] in a scalable distributed implementation that is communication-efficient by applying sketching techniques to reduce the quantity of data communicated at each iteration.

6 Advanced Algorithms

We discuss extensions to the fundamental SQNM in this section. Some of these modifications make the techniques more general to deal with more challenging situations, like problems that are not smooth and/or strongly convex. Other expansions develop quicker algorithms by using extra algorithmic techniques or problem structure.

6.1 Accelerated Variants

The momentum term [7, 8, 13, 122] and Nesterov's accelerated method [9] can accelerate the convergence of SGD and sometimes provide a distinct improvement in performance for neural network training [123]. The approach for SQNM acceleration is described in several works [124–129]. Nesterov's accelerated quasi-Newton (NAQ) method [124] is a recently proposed BFGS acceleration method. A stochastic variant of the NAQ technique called o(L)NAQ [125] was also devised, and it was found to perform better than the o(L)BFGS method. Then, Yasuda et al. [127] introduced the stochastic variance reduced Nesterov's accelerated quasi-Newton (SVR-NAQ) approach, which is superior to oNAQ since it updates the Hessian matrix with less noise. Chang et al. [126] propose a new sL-BFGS algorithm by importing a proper momentum, which is based on SQN [37] and stochastic L-BFGS with nonuniform sampling [85]. The complexity went from $\mathcal{O}\left((n+\kappa_H\kappa)\,d\log\frac{1}{\varepsilon}\right)$ to $\mathcal{O}\left(\left(n+\kappa_H\sqrt{\kappa}\right)d\log\frac{1}{\varepsilon}\right)$, where κ and κ_H represent the condition number of F and the Hessian approximation, respectively. A faster stochastic L-BFGS was proposed by Gao and Huang [128], and the oracle complexity was improved to $\mathcal{O}\left(\left(n+\frac{\kappa\kappa_H}{1+\sqrt{\kappa\kappa_H/n}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$. For training neural networks, Indrapriyadarsini et al. [129] propose a new limited memory Nesterov's accelerated symmetric rank-1 (L-SR1-N) method. It is demonstrated that the proposed L-SR1-N converges to a stationary point both theoretically and experimentally.

6.2 Relaxing Smoothness

For classical (L)BFGS, the objective function must be smooth since the quasi-Newton direction produced at a nonsmooth point is not always a descent direction. Yu et al. [130] suggested a subgradient (L)BFGS for which global convergence can be restored by determining a descent direction and applying a line search. Yousefian and Nedić [116] modified the update rule as $w_{k+1} = w_k - \alpha_k H_k \nabla F_{N_k}(w_k + z_k)$, where z_k is a uniform random variable drawn from a ball centered at the origin with radius $\mathbf{r} > 0$. They establish the convergence properties of the scheme in the absence

of the Lipschitzian property with this modification and a few acceptable assumptions. Jalilzadeh et al. [60, 117] drew on the Moreau proximal smoothing [131],

$$f^{\eta}(x) \triangleq \min_{u} \left\{ f(u) + \frac{1}{2\eta}\|u - w\|^2 \right\}, \quad \begin{cases} \eta_k := \eta_{k-1}, & \text{if } k \text{ is odd}, \\ \eta_k < \eta_{k-1}, & \text{otherwise}, \end{cases} \tag{26}$$

and they only reduce the smoothing parameter η after an odd number of iterations as the right-hand side of (26). Then, they utilized the η_k-smoothed function F^{η_k} to calculate $y_k = \nabla F_{N_k}^{\eta_k^{\delta}}(w_k) - \nabla F_{N_k}^{\eta_k^{\delta}}(w_{k-1})$, where $0 < \delta \leqslant 1$ is scalar that controls the level of smoothing.

Numerous studies take into account the following optimization problem:

$$\min_{w \in \mathbb{R}^d} \mathcal{F}(w) = F(w) + h(w),$$

where h is a nonsmooth convex function, such as ℓ_1 regularization. Stochastic proximal Newton-type approaches for these kinds of nonsmooth problems are investigated in [132–134].

6.3 Relaxing Strong Convexity

Convex functions are the focus of the majority of theories for stochastic quasi-Newton methods. In the absence of strong convexity, there are few convergence rate statements. For faster convergence and better theory, the objective function in machine learning is always regularized with the term $\frac{1}{2}\mu\|w\|^2$. However, the optimal solution to the $F(w) + \frac{1}{2}\mu\|w\|^2$ is not the optimal solution to the original problem. To deal with the merely convex objective function, Yousefian et al. [60, 115, 117, 118] develop a number of regularized SQN methods. They allow regularization parameters to be updated and decay to zero as iterations progress. Cyclic regularized stochastic BFGS (CR-SQN) [115] generates a sequence $\{w_k\}$ for any $k \geqslant 0$

$$w_{k+1} := w_k - \alpha_k \left(B_k^{-1} + \delta_k \boldsymbol{I} \right) \left(\nabla F_{N_k}(w_k) + \mu_k w_k \right), \tag{27}$$

where $\mu_k > 0$ is the regularization parameter of the gradient mapping, and they only update μ_k if k is even. The corresponding update rule of B_{k+1} is following,

$$B_{k+1} := \begin{cases} B_k - \frac{B_k s_k s_k^{\top} B_k}{s_k^{\top} B_k s_k} + \frac{\tilde{y}_k (\tilde{y}_k)^{\top}}{s_k^{\top} \tilde{y}_k} + \rho \mu_k \boldsymbol{I}, & k \text{ is even}, \\ B_k, & k \text{ is odd}, \end{cases} \tag{28}$$

where $\tilde{y}_k := \nabla F_{N_k}(w_{k+1}) - \nabla F_{N_k}(w_k) + (1-\rho)\mu_k s_k$ for an even k, $0 < \rho < 1$ is the regularization factor. In this way, it is easy to prove that $s_k^{\top} \tilde{y}_k \geqslant (1-\rho)\mu_k \|w_{k+1} - w_k\|^2 > 0$.

Iteratively regularized stochastic limited-memory BFGS (IRS-L-BFGS) updates the parameter as (29)

$$w_{k+1} := w_k - \alpha_k H_k \left(\nabla F_{N_k}(w_k) + \mu_k (w_k - w_0) \right), \tag{29}$$

where the update policy for H_k and the update rule for the regularization parameter μ_k are based on the following general procedure,

$$\begin{cases} H_k := H_{k,m}, & \mu_k := \mu_{k-1}, \quad \text{if } k \text{ is odd}, \\ H_k = H_{k-1}, & \mu_k < \mu_{k-1}, \quad \text{otherwise}. \end{cases} \tag{30}$$

The $\{s_i, y_i\}$ is defined for any odd $k \geqslant 1$,

$$\begin{aligned} s_{\lceil k/2 \rceil} &:= w_k - w_{k-1}, \\ y_{\lceil k/2 \rceil} &:= \nabla F_{N_{k-1}}(w_k) - \nabla F_{N_{k-1}}(w_{k-1}) + \tau \mu_k^{\delta} s_{\lceil k/2 \rceil}, \end{aligned}$$

where $\tau > 0$ and $0 < \delta \leqslant 1$ are parameters to control the level of regularization in the matrix H_k, and the perturbation term $\mu_k^{\delta} s_{\lceil k/2 \rceil} \to 0$, as $k \to \infty$. In terms of the value of the objective function, IRS-L-BFGS shows a convergence rate $\mathcal{O}(k^{-(\frac{1}{3}-\varepsilon)})$, where ε is an arbitrary small positive scalar.

6.4 Nonconvex Problems

The main difficulty in developing quasi-Newton methods for nonconvex problems is that the Hessian of the objective function is not positive definite. In the deterministic scenario, several techniques, including cautious updating [135] and damping [136], have been proposed to establish convergence of the BFGS method in the nonconvex setting. Berahas and Takáč [89, 90] skip the Hessian update if the curvature condition $y_k^{\top} s_k \geqslant \varepsilon \|s_k\|^2$ is not satisfied. Similarly, sampled L-BFGS [93] updates the (inverse) Hessian approximation using only the set of curvature pairs satisfying $y_k^{\top} s_k \geqslant \varepsilon \|s_k\|^2$.

In stochastic optimization, damping is more frequently used. Several works [57, 94, 95] use damping to make sure that H_k or B_k are positive definite. They define

$$\bar{y}_k = \vartheta_k y_k + (1 - \vartheta_k) B_k s_k, \tag{31}$$

where $\vartheta_k = \begin{cases} \frac{c_4 s_k^{\top} B_k s_k}{s_k^{\top} B_k s_k - s_k^{\top} y_k}, & \text{if } s_k^{\top} y_k < (1-c) s_k^{\top} B_k s_k, \\ 1, & \text{otherwise}. \end{cases}$

However, the *self-correcting* properties of BFGS-type updates could be ruined by damping. Self-correcting BFGS (SC-BFGS), which uses a damping procedure intended to maintain these properties, was proposed in [137]. In SC-BFGS, $\bar{y} = \beta_k s_k + (1 - \beta_k)\alpha_k y_k$, where β_k is the smallest value in the interval $[0, 1]$ such that the

following crucial bounds are satisfied:

$$\eta_1 \leqslant \frac{s_k^\top \bar{y}_k}{\|s_k\|_2^2} \text{ and } \frac{\|\bar{y}_k\|_2^2}{s_k^\top \bar{y}_k} \leqslant \eta_2, \quad \eta_1 \in (0, 1) \text{ and } \eta_2 \in (1, \infty).$$

The sequence of inverse Hessian approximations must satisfy this inequality in order to ensure that the global convergence guarantees are met.

Some SQNM approximate the Hessian via a diagonal matrix [101, 138]; these methods always rectify the absolute value of B_k with a convexity hyper-parameter σ to handle nonconvexity, i.e., rectify $(B_k, \sigma) = \max(|B_k|, \sigma)$, where the function rectify$(\cdot, \sigma)$ is similar to the rectified linear unit (ReLU) [139] with a threshold of σ.

6.5 Hybrid Stochastic Second-Order Methods

To improve practical performance, a number of hybrid stochastic second-order algorithms have been proposed. The term "hybrid" refers to the elaborate fusion of these stochastic first- or second-order methods. Incorporating quasi-Newton techniques with SGD methods could shorten the optimization process and eliminate the need to fine-tune optimization hyperparameters.

Symmetric blOckwise truNcated optimIzation Algorithm (SONIA) [140] bridges the gap between first- and second-order methods by computing a search direction in one subspace that incorporates curvature information and performing a scaled gradient descent step in the orthogonal complement.

Yang et al. [141] discovered an intriguing fact: the true Hessian matrix is frequently a combination of a cheap part and a costly part, i.e., $\nabla^2 F(w) = H(w) + \Pi(w)$, where $H(w)$ is relatively cheap and accessible, while $\Pi(w)$ is expensive or even not computable. For example, the subsampled Hessian matrix, the GGN/FIM, or any other approximation matrices are all examples of $\mathrm{H}(w)$, which stands for a matrix with partial Hessian information. We can employ SQNM to update $\Pi(w)$. For training neural network, the Kronecker-factored quasi-Newton method [82, 142] was proposed. They approximate the Hessian by a block-diagonal matrix that is the Kronecker product of two considerably small matrices A_i^{-1} and G_i^{-1}, then they use BFGS update to approximate them with innovative damping and Hessian-action [37, 39, 100] techniques.

7 Discussion and the Future

In this paper, we studied stochastic second-order approaches, particularly stochastic quasi-Newton methods, for large-scale machine learning. We began by briefly outlining the optimization problem in machine learning, and then, we summarized the existing stochastic first-order and second-order methods. The SQNM based on BFGS, which is currently regarded as the most effective quasi-Newton updating formula, was the main topic of discussion in this paper. Although there have been some other intelligent works presented that we do not mention, the majority of the algorithms we cover

are representative. Following that, we briefly discuss some future research directions in stochastic second-order methods.

More complex optimization techniques have been used as computing hardware has advanced. The second-order optimization method can speed up the training of deep neural networks, but more effective optimization algorithms cannot be applied directly because they frequently require higher-order function information, such as Hessian, and the acquisition and calculation of this information are not feasible due to the excessive number of parameters. Therefore, the practical application of second-order method for training neural networks is limited by the enormous calculation cost. Deep learning is robust to optimization algorithms; therefore, some advanced classic optimization techniques can be reduced and applied to neural networks to significantly speed up training.

Although no theory exists to clearly analyze or explain the precise effects of optimization models and algorithms on the generalization of neural networks, relevant research is still being conducted in this area, and optimization methods for deep learning have become the mainstream, with many extraordinary algorithms emerging. Theoretical evaluations of algorithm performance currently lag considerably behind the practical application. The efficiency analysis of traditional optimization methods often stops at the analysis of convergence order, while various improvements based on SGD methods are the same convergence order in theory in deep learning. However, even the progress of constant level can significantly increase training efficiency in practice. This demonstrates that traditional theoretical analysis methods cannot fully accommodate machine learning, but there is currently no reasonable additional theoretical analysis standard. In fact, the majority of the present approaches are difficult to theoretically analyze in terms of performance and can only be evaluated through tests with large amounts of data. Although the Adagrad method has been shown to have good convergence and lower regret accumulation, practice shows that Adagrad reduces the learning rate too quickly on many problems, resulting in unsatisfactory training efficiency.

In addition to the significant theoretical challenges, the development of more effective optimization methods like K-FAC shows that there is still a large research space to be discovered regarding the training of deep neural networks. Stochastic second-order algorithms still have numerous limitations in comparison with the well-studied first-order method. Although the current second-order algorithms utilized for deep neural network training are significantly more effective, they have more restrictions than SGD approaches in terms of the types of problems that may be addressed. For instance, the implementation of K-FAC is complicated and heavily dependent on the unique neural network structure, making it more challenging than other approaches. The K-FAC approach can only be employed in the context of traditional convolution networks, such as image recognition, whereas Transformer model, which is frequently used in natural language processing, requires significant modification. Future research will focus heavily on finding ways to get around these restrictions, create second-order algorithms that are applicable to many applications, or enhance the training effect of second-order algorithms.

Under some conditions, gradient descent can converge linearly in the deterministic setting, whereas quasi-Newton methods can converge superlinearly. Typically, SGD

can only achieve the sublinear convergence rate, but variance reduction can enable linear convergence. As shown in Sect. 4, some SQNM achieve sublinear convergence. SQNM with variance-reduced gradient can converge linearly; it does not recover a superlinear convergence rate. Although the IQN can converge superlinearly by using a proper quadratic approximation of individual functions, the memory required by the IQN is $\mathcal{O}(nd^2)$, which is unacceptable for high-dimensional or large-scale problems. As a result, a superlinear convergent stochastic quasi-Newton approach with lower costs is a promising area for further study [59]. Another point worth noting is that most SQNM avoid possible difficulties with BFGS or L-BFGS updating by assuming that the quality of gradient differences is sufficiently controlled. This is something that can be addressed.

Dynamic sample size schemes [26, 107] have served as the foundation for many variance reduction schemes in machine learning [60, 117]. The sample size used to approximate the Hessian strongly influences the performance of the subsampled Newton method. In addition, the construction of gradient displacement y within allowable error bounds also required careful consideration of sample size selection. The basic idea behind dynamic sample size is to increase the size of the training sample used in the gradient and Hessian evaluation by a factor. SGD methods converge linearly for strongly convex problems by utilizing dynamic sample size, and this scheme is used in stochastic second-order algorithms [143, 144]. If the initial sample size is sufficiently large, Jin and Mokhtari [144] demonstrate that the subproblems can be solved superlinearly fast by applying quasi-Newton methods. But only convex environments are covered by their assurances. Another disadvantage is that the sample size needs to be increased geometrically, and the literature offers no direction on how to select the factor (AdaQN choose 2 [144]). As a result, investigating the use of dynamic sample size strategies in quasi-Newton methods is an intriguing line of research that merits further research. In addition, sampling is also a major concern in machine learning stochastic optimization techniques.

The step size is a crucial hyperparameter in the optimization process that has a direct impact on how well it works in practical applications. In deterministic settings, line search can be used to determine the step size. However, stochastic line search is computationally prohibited in machine learning since we only have subsampled data on function value, gradient, and Hessian approximation. How to make sure that the search direction is a descent direction in expectation is the first difficulty in constructing the SLS. Another issue is that we may reject an appropriate α_k if the observed objective function value increases, even if the true cost has decreased sufficiently. Some studies proposed the IPQN test with a dynamic sample size. Another class of SLS is being developed that is based on a probabilistic framework for the Hessian; these methods are derived from the fact that all existing quasi-Newton algorithms can be interpreted as maxima of Gaussian posterior probability distributions [99]. There has always been a need for study in the area of stochastic line search.

In nonconvex settings, an essential challenge in designing quasi-Newton methods is that the eigenvalues of (inverse) Hessian approximations are not uniformly bounded above and away from zero. The positive-definite Hessian approximations were preserved in several studies via damping or regularization. Simultaneously, a more careful calculation method of correction pairs $\{s_i, y_i\}$ is used to keep the gradient noise within

an acceptable range. However, the *self-correcting* property of updates of the BFGS type could be ruined by damping techniques. Moreover, BFGS has some disadvantages when trying to enforce the positive definiteness of the approximated Hessian matrices in a nonconvex setting. As a result, developing a convergent quasi-Newton method for nonconvex settings is a very active direction of research.

Limited memory quasi-Newton methods form the basis of the majority of SQNM. A significant challenge with L-BFGS is that, while memory size m can have a significant effect on performance, it is difficult to predict which memory size will work best. Boggs and Byrd [145] recently proposed that each iteration choose a different memory size adaptively. Additionally, a displacement aggregation strategy [146] is suggested for the curvature pairs stored in an L-BFGS. These adaptive L-BFGS, however, cannot be used directly in stochastic scenarios. Therefore, stochastic L-BFGS or L-SR1 with adaptive memory size is a promising approach.

In the stochastic contexts, the intelligent fusion of second-order methods has already demonstrated its special attractiveness. On some machine learning problems, the structured stochastic quasi-Newton methods (S2QN) [141] framework is fairly competitive with the state-of-the-art approaches. If the sample size is large enough, a local superlinear convergence result is assured. On a number of CNN tasks, a new class of Kronecker-factored quasi-Newton methods (KF-QN-CNN) [142] outperformed state-of-the-art first- and second-order methods. Such methods have numerous obvious advantages, including comparable memory requirements, lower per-iteration time complexity, and less expensive computation. An intriguing study area would be to create a better framework that combines the advantages of stochastic first-order and second-order optimization methods.

Learning to optimize (L2O) has gained popularity in recent years, and it is based on "learning to learn" or "meta learning" [147, 148]. The primary goal of L2O is to develop optimization methods by leveraging neural networks. L2O has begun to show great promise in a few applications and optimization domains. Egidio et al. [149] proposed using the truncated backpropagation through time algorithm to learn the step-size policy for L-BFGS in a constrained domain. On some problems, the suggested step-size strategy is competitive with heuristically tuned optimization procedures. As a result, we may be able to use objective function and gradient information to learn an approximate "quasi-Newton" optimizer for machine learning. However, L2O research is still in its infancy, with numerous open challenges and research opportunities ranging from practice to theory.

Acknowledgements The authors are grateful to the editor and the reviewers for their helpful comments and suggestions, which have improved the presentation of the paper.

Author Contributions Tian-De Guo contributed to conceptualization, methodology, supervision, writing review and editing; Yan Liu contributed to writing original draft; Cong-Ying Han contributed to methodology, supervision, writing review and editing.

Conflict of Interest The authors declare that they have no conflict of interest

References

[1] Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**(3), 400–407 (1951)
[2] Le, Q., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.: On optimization methods for deep learning. In: International Conference on Machine Learning. pp. 265–272. ACM, New York, USA (2011)
[3] Bottou, L.: Stochastic gradient learning in neural networks. In: Proceedings of Neuro-Nımes. **91** (1991)
[4] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
[5] Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Neural Information Processing Systems. vol. **20**. Curran Associates, Inc. (2007). https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf
[6] Zinkevich, M., Weimer, M., Li, L., Smola, A.: Parallelized stochastic gradient descent. In: Neural Information Processing Systems. vol. **23**. Curran Associates, Inc. (2010). https://proceedings.neurips.cc/paper/2010/file/abea47ba24142ed16b7d8fbf2c740e0d-Paper.pdf
[7] Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Netw. **12**(1), 145–151 (1999)
[8] Loizou, N., Richtárik, P.: Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. Comput. Optim. Appl. **77**(3), 653–710 (2020)
[9] Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. Doklady Akademia Nauk USSR **269**, 543–547 (1983)
[10] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(7), 257–269 (2011)
[11] Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv:1212.5701 (2012)
[12] Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
[13] Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning. pp. 1139–1147. PMLR (2013)
[14] Yuan, K., Ying, B., Sayed, A.H.: On the influence of momentum acceleration on online learning. J. Mach. Learn. Res. **17**(1), 6602–6667 (2016)
[15] Lan, G.: An optimal method for stochastic composite optimization. Math. Progr. **133**(1), 365–397 (2012)
[16] Ghadimi, S., Lan, G.: Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. SIAM J. Optim. **22**(4), 1469–1492 (2012)
[17] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)
[18] Gower, R.M., Schmidt, M., Bach, F., Richtarik, P.: Variance-reduced methods for machine learning. Proc. IEEE **108**(11), 1968–1983 (2020)
[19] Schmidt, M., Roux, N.L., Bach, F.: Minimizing finite sums with the stochastic average gradient. Math. Progr. **162**(5), 1–30 (2013)
[20] Roux, N., Schmidt, M., Bach, F.: A stochastic gradient method with an exponential convergence rate for finite training sets. In: Neural Information Processing Systems. vol. **25**. Curran Associates, Inc. (2012). https://proceedings.neurips.cc/paper/2012/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf
[21] Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: a fast incremental gradient method with support for non-strongly convex composite objectives. In: Neural Information Processing Systems. vol. **27**. Curran Associates, Inc. (2014). https://proceedings.neurips.cc/paper/2014/file/ede7e2b6d13a41ddf9f4bdef84fdc737-Paper.pdf

[22] Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Neural Information Processing Systems. vol. **26**. Curran Associates, Inc. (2013). https://proceedings.neurips.cc/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf
[23] Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: SARAH: a novel method for machine learning problems using stochastic recursive gradient. In: International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. **70**, pp. 2613–2621. PMLR (2017). https://proceedings.mlr.press/v70/nguyen17b.html
[24] Xu, P., Roosta, F., Mahoney, M.W.: Second-order optimization for non-convex machine learning: an empirical study. In: SIAM International Conference on Data Mining. pp. 199–207. SIAM (2020)
[25] Byrd, R.H., Chin, G.M., Neveitt, W., Nocedal, J.: On the use of stochastic hessian information in optimization methods for machine learning. SIAM J. Optim. **21**(3), 977–995 (2011)
[26] Byrd, R.H., Chin, G.M., Nocedal, J., Wu, Y.: Sample size selection in optimization methods for machine learning. Math. Progr. **134**(1), 127–155 (2012)
[27] Erdogdu, M.A., Montanari, A.: Convergence rates of sub-sampled newton methods. In: Neural Information Processing Systems. vol. **28**. Curran Associates, Inc. (2015). https://proceedings.neurips.cc/paper/2015/file/404dcc91b2aeaa7caa47487d1483e48a-Paper.pdf
[28] Roosta-Khorasani, F., Mahoney, M.W.: Sub-sampled Newton methods. Math. Progr. **174**(1), 293–326 (2019)
[29] Xu, P., Yang, J., Roosta, F., Ré, C., Mahoney, M.W.: Sub-sampled newton methods with non-uniform sampling. In: Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016). https://proceedings.neurips.cc/paper/2016/file/55c567fd4395ecef6d936cf77b8d5b2b-Paper.pdf
[30] Bollapragada, R., Byrd, R.H., Nocedal, J.: Exact and inexact subsampled Newton methods for optimization. IMA J. Numer. Anal. **39**(2), 545–578 (2019)
[31] Wang, J., Zhang, T.: Improved optimization of finite sums with minibatch stochastic variance reduced proximal iterations. arXiv:1706.07001 (2017)
[32] Pilanci, M., Wainwright, M.J.: Newton sketch: a near linear-time optimization algorithm with linear-quadratic convergence. SIAM J. Optim. **27**(1), 205–245 (2017)
[33] Berahas, A.S., Bollapragada, R., Nocedal, J.: An investigation of Newton-sketch and subsampled Newton methods. Optim. Methods Softw. **35**(4), 661–680 (2020)
[34] Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
[35] Agarwal, N., Bullins, B., Hazan, E.: Second-order stochastic optimization for machine learning in linear time. J. Mach. Learn. Res. **18**(116), 1–40 (2017)
[36] Bordes, A., Bottou, L., Gallinari, P.: Sgd-qn: careful quasi-newton stochastic gradient descent. J. Mach. Learn. Res. **10**(3), 1737–1754 (2009)
[37] Byrd, R.H., Hansen, S.L., Nocedal, J., Singer, Y.: A stochastic quasi-Newton method for large-scale optimization. SIAM J. Optim. **26**(2), 1008–1031 (2016)
[38] Moritz, P., Nishihara, R., Jordan, M.: A linearly-convergent stochastic L-BFGS algorithm. In: International Conference on Artificial Intelligence and Statistics. pp. 249–258. PMLR (2016)
[39] Gower, R., Goldfarb, D., Richtarik, P.: Stochastic block BFGS: squeezing more curvature out of data. In: International Conference on Machine Learning. pp. 1869–1878. PMLR (2016)
[40] Martens, J., et al.: Deep learning via hessian-free optimization. Int. Conf. Mach. Learn. **27**, 735–742 (2010)
[41] Martens, J., Sutskever, I.: Learning recurrent neural networks with hessian-free optimization. In: International Conference on Machine Learning. pp. 1033–1040. ACM, New York, USA (2011)
[42] Schraudolph, N.N.: Fast curvature matrix-vector products for second-order gradient descent. Neural Comput. **14**(7), 1723–1738 (2002)
[43] Botev, A., Ritter, H., Barber, D.: Practical Gauss-Newton optimisation for deep learning. In: International Conference on Machine Learning. pp. 557–565. PMLR (2017)
[44] Martens, J., Grosse, R.: Optimizing neural networks with Kronecker-factored approximate curvature. In: International Conference on Machine Learning. pp. 2408–2417. PMLR (2015)
[45] Grosse, R., Martens, J.: A Kronecker-factored approximate fisher matrix for convolution layers. In: International Conference on Machine Learning. pp. 573–582. PMLR (2016)
[46] Amari, S.I.: Natural gradient works efficiently in learning. Neural Comput. **10**(2), 251–276 (1998)
[47] Roux, N., Manzagol, P.A., Bengio, Y.: Topmoumoute online natural gradient algorithm. In: Neural Information Processing Systems. vol. **20**. Curran Associates, Inc. (2007). https://proceedings.neurips.cc/paper/2007/file/9f61408e3afb633e50cdf1b20de6f466-Paper.pdf

[48] Martens, J.: New insights and perspectives on the natural gradient method. J. Mach. Learn. Res. **21**(146), 1–76 (2020)
[49] George, T., Laurent, C., Bouthillier, X., Ballas, N., Vincent, P.: Fast approximate natural gradient descent in a kronecker-factored eigenbasis. In: Neural Information Processing Systems. pp. 9573–9583. NIPS'18, Curran Associates Inc., Red Hook, NY, USA (2018)
[50] Gupta, V., Koren, T., Singer, Y.: Shampoo: preconditioned stochastic tensor optimization. In: International Conference on Machine Learning. pp. 1842–1850. PMLR (2018)
[51] Chen, M., Gao, K., Liu, X., Wang, Z., Ni, N., Zhang, Q., Chen, L., Ding, C., Huang, Z., Wang, M., Wang, S., Yu, F., Zhao, X., Xu, D.: THOR, trace-based hardware-driven layer-oriented natural gradient descent computation. AAAI Conf. Artif. Intell. **35**(8), 7046–7054 (2021)
[52] Schraudolph, N.N., Yu, J., Günter, S.: A stochastic quasi-Newton method for online convex optimization. In: Artificial intelligence and statistics. pp. 436–443. PMLR (2007)
[53] Gower, R., Kovalev, D., Lieder, F., Richtarik, P.: RSN: randomized subspace Newton. In: Neural Information Processing Systems. vol. **32**. Curran Associates, Inc. (2019)
[54] Hanzely, F., Doikov, N., Nesterov, Y., Richtarik, P.: Stochastic subspace cubic Newton method. In: International Conference on Machine Learning. pp. 4027–4038. PMLR (2020)
[55] Xu, P., Roosta, F., Mahoney, M.W.: Newton-type methods for non-convex optimization under inexact Hessian information. Math. Progr. **184**(1), 35–70 (2020)
[56] Mokhtari, A., Ribeiro, A.: RES: regularized stochastic BFGS algorithm. IEEE Trans. Signal Process. **62**(23), 6089–6104 (2014)
[57] Wang, X., Ma, S., Goldfarb, D., Liu, W.: Stochastic quasi-Newton methods for nonconvex stochastic optimization. SIAM J. Optim. **27**(2), 927–956 (2017)
[58] Mokhtari, A., Eisen, M., Ribeiro, A.: IQN: an incremental quasi-Newton method with local superlinear convergence rate. SIAM J. Optim. **28**(2), 1670–1698 (2018)
[59] Mokhtari, A., Ribeiro, A.: Stochastic quasi-Newton methods. Proc. IEEE **108**(11), 1906–1922 (2020)
[60] Jalilzadeh, A., Nedić, A., Shanbhag, U.V., Yousefian, F.: A variable sample-size stochastic quasi-newton method for smooth and nonsmooth stochastic convex optimization. Math. Op. Res. **47**(1), 690–719 (2022)
[61] Shi, H.J.M., Xie, Y., Byrd, R., Nocedal, J.: A noise-tolerant quasi-Newton algorithm for unconstrained optimization. SIAM J. Optim. (2022). https://doi.org/10.1137/20M1373190
[62] Martens, J., Sutskever, I.: Training Deep and Recurrent Networks with Hessian-Free Optimization, pp. 479–535. Springer, Berlin (2012)
[63] Cho, M., Dhir, C., Lee, J.: Hessian-free optimization for learning deep multidimensional recurrent neural networks. In: Neural Information Processing Systems. vol. **28**. Curran Associates, Inc. (2015). https://proceedings.neurips.cc/paper/2015/file/a86c450b76fb8c371afead6410d55534-Paper.pdf
[64] Haider, A., Zhang, C., Kreyssig, F.L., Woodland, P.C.: A distributed optimisation framework combining natural gradient with hessian-free for discriminative sequence training. Neural Netw. **143**, 537–549 (2021)
[65] Roux, N.L., Fitzgibbon, A.: A fast natural Newton method. In: International Conference on Machine Learning. pp. 623–630. ICML'10, Omnipress, Madison, WI, USA (2010)
[66] Ba, J., Grosse, R., Martens, J.: Distributed second-order optimization using Kronecker-factored approximations. In: International Conference on Learning Representations (2017)
[67] Khalfan, H.F., Byrd, R.H., Schnabel, R.B.: A theoretical and experimental study of the symmetric rank-one update. SIAM J. Optim. **3**(1), 1–24 (1993)
[68] Conn, A.R., Gould, N.I., Toint, P.L.: Convergence of quasi-newton matrices generated by the symmetric rank one update. Math. Progr. **50**(1), 177–195 (1991)
[69] Jorge, N., Stephen, J.W.: Numerical Optimization. Springer, Berlin (2006)
[70] Davidon, W.C.: Varible metric method for minimization. Tech. rep., Argonne National Lab., Lemont, Ill. (1959). https://doi.org/10.2172/4222000, https://www.osti.gov/biblio/4222000
[71] Fletcher, R., Powell, M.J.: A rapidly convergent descent method for minimization. Comput. J. **6**(2), 163–168 (1963)
[72] Greenstadt, J.: Variations on variable-metric methods. (with discussion). Math. Comput. **24**(109), 1–22 (1970)
[73] Broyden, C.G.: Quasi-newton methods and their application to function minimisation. Math. Comput. **21**(99), 368–381 (1967)
[74] Fletcher, R.: A new approach to variable metric algorithms. Comput. J. **13**(3), 317–322 (1970)

[75] Goldfarb, D.: A family of variable-metric methods derived by variational means. Math. Comput. **24**(109), 23–26 (1970)
[76] Shanno, D.F.: Conditioning of quasi-newton methods for function minimization. Math. Comput. **24**(111), 647–656 (1970)
[77] Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann. Math. Stat. **21**(1), 124–127 (1950)
[78] Nocedal, J.: Updating quasi-newton matrices with limited storage. Math. Comput. **35**(151), 773–782 (1980)
[79] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Progr. **45**(1–3), 503–528 (1989)
[80] Wills, A.G., Schön, T.B.: Stochastic quasi-Newton with line-search regularisation. Automatica **127**, 109503 (2021)
[81] Mokhtari, A., Ribeiro, R.: Global convergence of online limited memory BFGS. J. Mach. Learn. Res. **16**(98), 3151–3181 (2015)
[82] Goldfarb, D., Ren, Y., Bahamou, A.: Practical quasi-newton methods for training deep neural networks. In: Neural Information Processing Systems. vol. **33**, pp. 2386–2396. Curran Associates, Inc. (2020). https://proceedings.neurips.cc/paper/2020/file/192fc044e74dffea144f9ac5dc9f3395-Paper.pdf
[83] Lucchi, A., McWilliams, B., Hofmann, T.: A variance reduced stochastic newton method. arXiv:1503.08316 (2015)
[84] Konečnỳ, J., Richtárik, P.: Semi-stochastic gradient descent methods. arXiv:1312.1666 (2013)
[85] Zhao, R., Haskell, W.B., Tan, V.Y.F.: Stochastic L-BFGS: improved convergence rates and practical acceleration strategies. IEEE Trans. Signal Process. **66**(5), 1155–1169 (2018)
[86] Gao, W., Goldfarb, D.: Block BFGS methods. SIAM J. Optim. **28**(2), 1205–1231 (2018)
[87] Kovalev, D., Mishchenko, K., Richtárik, P.: Stochastic newton and cubic newton methods with simple local linear-quadratic rates. arXiv:1912.01597 (2019)
[88] Chen, J., Yuan, R., Garrigos, G., Gower, R.M.: San: stochastic average newton algorithm for minimizing finite sums. arXiv:2106.10520 (2021)
[89] Berahas, A.S., Nocedal, J., Takac, M.: A multi-batch L-BFGS method for machine learning. In: Neural Information Processing Systems. vol. **29**. Curran Associates, Inc. (2016). https://proceedings.neurips.cc/paper/2016/file/8ebda540cbcc4d7336496819a46a1b68-Paper.pdf
[90] Berahas, A.S., Takáč, M.: A robust multi-batch L-BFGS method for machine learning. Optim. Methods Softw. **35**(1), 191–219 (2020)
[91] Pearlmutter, B.A.: Fast exact multiplication by the hessian. Neural Comput. **6**(1), 147–160 (1994)
[92] Keskar, N.S., Berahas, A.S.: AdaQN: an adaptive quasi-Newton algorithm for training RNNs. In: Machine Learning and Knowledge Discovery in Databases. pp. 1–16. Springer International Publishing, Cham (2016)
[93] Berahas, A.S., Jahani, M., Richtárik, P., Takáč, M.: Quasi-Newton methods for machine learning: forget the past, just sample. Optim. Methods Softw. (2021). https://doi.org/10.1080/10556788.2021.1977806
[94] Chen, H., Wu, H.C., Chan, S.C., Lam, W.H.: A stochastic quasi-Newton method for large-scale nonconvex optimization with applications. IEEE Trans. Neural Netw. Learn. Syst. **31**(11), 4776–4790 (2020)
[95] Lotfi, S., de Ruisselet, T.B., Orban, D., Lodi, A.: Stochastic damped L-BFGS with controlled norm of the Hessian approximation. arXiv:2012.05783 (2020)
[96] Wills, A., Schön, T.: Stochastic quasi-newton with adaptive step lengths for large-scale problems. arXiv:1802.04310 (2018)
[97] Wills, A., Schön, T.B., Jidling, C.: A fast quasi-Newton-type method for large-scale stochastic optimisation. IFAC-PapersOnLine **53**(2), 1249–1254 (2020)
[98] Gower, R.M., Gondzio, J.: Action constrained quasi-newton methods. arXiv:1412.8045 (2014)
[99] Hennig, P.: Probabilistic interpretation of linear solvers. SIAM J. Optim. **25**(1), 234–260 (2015)
[100] Gower, R.M., Richtárik, P.: Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms. SIAM J. Matrix Anal. Appl. **38**(4), 1380–1409 (2017)
[101] Ma, X.: Apollo: an adaptive parameter-wised diagonal quasi-newton method for nonconvex stochastic optimization. In: International Conference on Learning Representations (2020)
[102] Dennis, J.E., Jr., Wolkowicz, H.: Sizing and least-change secant methods. SIAM J. Numer. Anal. **30**(5), 1291–1314 (1993)

[103] Wills, A.G., Schön, T.B.: On the construction of probabilistic newton-type algorithms. In: Conference on Decision and Control (CDC). pp. 6499–6504 (2017). https://doi.org/10.1109/CDC.2017.8264638
[104] Zhou, C., Gao, W., Goldfarb, D.: Stochastic adaptive quasi-newton methods for minimizing expected values. In: International Conference on Machine Learning. pp. 4150–4159 (2017)
[105] Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. Springer, Berlin (2003)
[106] Mahsereci, M., Hennig, P.: Probabilistic line searches for stochastic optimization. J. Mach. Learn. Res. **18**, 1–59 (2017)
[107] Bollapragada, R., Byrd, R., Nocedal, J.: Adaptive sampling strategies for stochastic optimization. SIAM J. Optim. **28**(4), 3312–3343 (2018)
[108] Balles, L., Romero, J., Hennig, P.: Coupling adaptive batch sizes with learning rates. In: Uncertainty in Artificial Intelligence. pp. 675–684. Curran Associates, Inc. (2017)
[109] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: training imagenet in 1 hour. arXiv:1706.02677 (2017)
[110] De, S., Yadav, A.K., Jacobs, D.W., Goldstein, T.: Automated inference with adaptive batches. In: International Conference on Artificial Intelligence and Statistics. vol. 54, pp. 1504–1513. PMLR (2017). http://proceedings.mlr.press/v54/de17a.html
[111] Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.J., Tang, P.T.P.: A progressive batching L-BFGS method for machine learning. In: International Conference on Machine Learning. pp. 620–629. PMLR (2018)
[112] Xie, Y., Byrd, R.H., Nocedal, J.: Analysis of the BFGS method with errors. SIAM J. Optim. **30**(1), 182–209 (2020)
[113] Sonnenburg, S., Franc, V., Yom-Tov, E., Sebag, M.: Pascal large scale learning challenge (2008)
[114] Bordes, A., Bottou, L., Gallinari, P., Chang, J., Smith, S.A.: Erratum: SGDQN is less careful than expected. J. Mach. Learn. Res. **11**(77), 2229–2240 (2010)
[115] Yousefian, F., Nedić, A., Shanbhag, U.V.: Stochastic quasi-Newton methods for non-strongly convex problems: convergence and rate analysis. In: Conference on Decision and Control (CDC). pp. 4496–4503. IEEE, Las Vegas, NV, USA (2016). https://doi.org/10.1109/CDC.2016.7798953
[116] Yousefian, F., Nedic, A., Shanbhag, U.V.: A smoothing stochastic quasi-newton method for non-lipschitzian stochastic optimization problems. In: Winter Simulation Conference (WSC). pp. 2291–2302. IEEE, Las Vegas, NV (2017). https://doi.org/10.1109/WSC.2017.8247960
[117] Jalilzadeh, A., Nedić, A., Shanbhag, U.V., Yousefian, F.: A variable sample-size stochastic quasi-newton method for smooth and nonsmooth stochastic convex optimization. In: Conference on Decision and Control (CDC). pp. 4097–4102. IEEE, Institute of Electrical and Electronics Engineers Inc. (2018). https://doi.org/10.1109/CDC.2018.8619209
[118] Yousefian, F., Nedić, A., Shanbhag, U.V.: On stochastic and deterministic quasi-Newton methods for nonstrongly convex optimization: asymptotic convergence and rate analysis. SIAM J. Optim. **30**(2), 1144–1172 (2020)
[119] Chen, W., Wang, Z., Zhou, J.: Large-scale l-bfgs using mapreduce. In: Neural Information Processing Systems. vol. **27**. Curran Associates, Inc. (2014). https://proceedings.neurips.cc/paper/2014/file/e49b8b4053df9505e1f48c3a701c0682-Paper.pdf
[120] Najafabadi, M.M., Khoshgoftaar, T.M., Villanustre, F., Holt, J.: Large-scale distributed L-BFGS. J. Big Data **4**(1), 22 (2017)
[121] Jahani, M., Nazari, M., Rusakov, S., Berahas, A.S., Takáč, M.: Scaling up quasi-newton algorithms: communication efficient distributed SR1. In: Machine Learning, Optimization, and Data Science, pp. 41–54. Springer, Cham (2020)
[122] Liu, Y., Gao, Y., Yin, W.: An improved analysis of stochastic gradient descent with momentum. In: Neural Information Processing Systems. vol. **33**, pp. 18261–18271. Curran Associates Inc. (2020)
[123] Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math. Progr. **156**(1), 59–99 (2016)
[124] Ninomiya, H.: Neural network training based on quasi-Newton method using Nesterov's accelerated gradient. In: IEEE Region 10 Conference (TENCON). pp. 51–54. IEEE, Singapore (2016). https://doi.org/10.1109/TENCON.2016.7847957
[125] Indrapriyadarsini, S., Mahboubi, S., Ninomiya, H., Asai, H.: A stochastic quasi-Newton method with Nesterov's accelerated gradient. In: Machine Learning and Knowledge Discovery in Databases. pp. 743–760. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-46150-8_43

[126] Chang, D., Sun, S., Zhang, C.: An accelerated linearly convergent stochastic L-BFGS algorithm. IEEE Trans. Neural Netw. Learn. Syst. **30**(11), 3338–3346 (2019)
[127] Yasuda, S., Mahboubi, S., Indrapriyadarsini, S., Ninomiya, H., Asai, H.: A stochastic variance reduced Nesterov's accelerated quasi-Newton method. In: IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 1874–1879. IEEE, Boca Raton, FL, USA (2019). https://doi.org/10.1109/ICMLA.2019.00301
[128] Gao, H., Huang, H.: Faster stochastic second order method for large-scale machine learning models. In: SIAM International Conference on Data Mining (SDM). pp. 405–413. Proceedings, Society for Industrial and Applied Mathematics (2021). https://doi.org/10.1137/1.9781611976700.46
[129] Indrapriyadarsini, S., Mahboubi, S., Ninomiya, H., Kamio, T., Asai, H.: Accelerating symmetric rank-1 quasi-newton method with nesterov's gradient for training neural networks. Algorithms **15**(1), 6 (2022)
[130] Yu, J., Vishwanathan, S., Günter, S., Schraudolph, N.N.: A quasi-newton approach to nonsmooth convex optimization problems in machine learning. The J. Mach. Learn. Res. **11**, 1145–1200 (2010)
[131] Beck, A.: First-Order Methods in Optimization. SIAM, New Delhi (2017)
[132] Wang, J., Zhang, T.: Utilizing second order information in minibatch stochastic variance reduced proximal iterations. The J. Mach. Learn. Res. **20**(1), 1578–1633 (2019)
[133] Wang, X., Wang, X., Yuan, Y.X.: Stochastic proximal quasi-Newton methods for non-convex composite optimization. Optim. Methods Softw. **34**(5), 922–948 (2019)
[134] Yang, M., Milzarek, A., Wen, Z., Zhang, T.: A stochastic extra-step quasi-Newton method for nonsmooth nonconvex optimization. Math. Progr. (2021). https://doi.org/10.1007/s10107-021-01629-y
[135] Li, D.H., Fukushima, M.: On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. SIAM J. Optim. **11**(4), 1054–1064 (2001)
[136] Powell, M.J.: Algorithms for nonlinear constraints that use lagrangian functions. Math. Progr. **14**(1), 224–248 (1978)
[137] Curtis, F.: A self-correcting variable-metric algorithm for stochastic optimization. In: International Conference on Machine Learning. pp. 632–641. PMLR (2016)
[138] Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., Mahoney, M.: Adahessian: an adaptive second order optimizer for machine learning. AAAI Conf. Artif. Intell. **35**(12), 10665–10673 (2021)
[139] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning. p. 807-814. ICML'10, Omnipress, Madison, WI, USA (2010)
[140] Jahani, M., Nazari, M., Tappenden, R., Berahas, A., Takac, M.: SONIA: a symmetric blockwise truncated optimization algorithm. In: International Conference on Artificial Intelligence and Statistics. pp. 487–495. PMLR (2021)
[141] Yang, M., Xu, D., Chen, H., Wen, Z., Chen, M.: Enhance curvature information by structured stochastic quasi-Newton methods. In: Computer Vision and Pattern Recognition. pp. 10654–10663 (2021)
[142] Ren, Y., Goldfarb, D.: Kronecker-factored quasi-newton methods for convolutional neural networks. arXiv:2102.06737 (2021)
[143] Mokhtari, A., Daneshmand, H., Lucchi, A., Hofmann, T., Ribeiro, A.: Adaptive Newton method for empirical risk minimization to statistical accuracy. In: Neural Information Processing Systems. vol. **29**. Curran Associates, Inc. (2016)
[144] Jin, Q., Mokhtari, A.: Exploiting local convergence of quasi-Newton methods globally: adaptive sample size approach. In: Neural Information Processing Systems. p. 12 (2021)
[145] Boggs, P.T., Byrd, R.H.: Adaptive, limited-memory BFGS algorithms for unconstrained optimization. SIAM J. Optim. **29**(2), 1282–1299 (2019)
[146] Berahas, A.S., Curtis, F.E., Zhou, B.: Limited-memory BFGS with displacement aggregation. Math. Progr. (2021). https://doi.org/10.1007/s10107-021-01621-6
[147] Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., de Freitas, N.: Learning to learn by gradient descent by gradient descent. In: Neural Information Processing Systems. vol. **29**. Curran Associates, Inc. (2016). https://proceedings.neurips.cc/paper/2016/file/fb87582825f9d28a8d42c5e5e5e8b23d-Paper.pdf
[148] Hochreiter, S., Younger, A.S., Conwell, P.R.: Learning to learn using gradient descent. In: International Conference on Artificial Neural Networks. pp. 87–94. Springer (2001)

[149] Egidio, L.N., Hansson, A., Wahlberg, B.: Learning the step-size policy for the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021). https://doi.org/10.1109/IJCNN52387.2021.9534194