Bolstering Stochastic Gradient Descent with Model Building

Ş. İlker Birbil
University of Amsterdam, 11018 TV Amsterdam, The Netherlands

Özgür Martin
Mimar Sinan Fine Arts University, 34380 Istanbul, Turkey

Gönenç Onay
Coach-Ai GmbH - AI & Analytics, Hilpertstr. 35, D-64295 Darmstadt - Germany

Figen Öztoprak
Artelys Corporation, 60601 Chicago, IL – USA

Abstract: Stochastic gradient descent method and its variants constitute the core optimization algorithms that achieve good convergence rates for solving machine learning problems. These rates are obtained especially when these algorithms are fine-tuned for the application at hand. Although this tuning process can require large computational costs, recent work has shown that these costs can be reduced by line search methods that iteratively adjust the stepsize. We propose an alternative approach to stochastic line search by using a new algorithm based on forward step model building. This model building step incorporates second-order information that allows adjusting not only the stepsize but also the search direction. Noting that deep learning model parameters come in groups (layers of tensors), our method builds its model and calculates a new step for each parameter group. This novel diagonalization approach makes the selected step lengths adaptive. We provide convergence rate analysis, and experimentally show that the proposed algorithm achieves faster convergence and better generalization in well-known test problems. More precisely, SMB requires less tuning, and shows comparable performance to other adaptive methods.

Keywords: model building; second-order information; stochastic gradient descent; convergence analysis

---

Stochastic gradient descent (SGD) is a popular optimization algorithm for machine learning problems which can achieve fast convergence when its stepsize and its scheduling are tuned well for the specific application at hand. This tuning procedure can take up to thousands of CPU/GPU days resulting in big energy costs (Asi and Duchi, 2019).

A number of researchers have studied adaptive strategies for improving the direction and the stepsize choices of the stochastic gradient descent algorithm. Adaptive sample size selection ideas (Byrd et al., 2012; Balles et al., 2016; Bollapragada et al., 2018) improve the direction by reducing its variance around the negative gradient of the empirical loss function, while stochastic quasi-Newton algorithms (Byrd et al., 2016; Wang et al., 2017) provide adaptive preconditioning. Recently, several stochastic line search approaches have been proposed. Not surprisingly, some of these work cover sample size selection as a component of the proposed line search algorithms (Balles et al., 2016; Paquette and Scheinberg, 2020).

The Stochastic Model Building (SMB) algorithm proposed in this paper is not designed as a stochastic quasi-Newton algorithm in the sense explained by Bottou et al. (2018). However, it still produces a scaling matrix in the process of generating trial points, and its overall step at each outer iteration can be written in the form of matrix-vector multiplication. Unlike the algorithms proposed by Mokhtari and Ribeiro (2014) and Schraudolph et al. (2007), we have no accumulation of curvature pairs throughout several iterations. Since there is no memory carried from earlier iterations, the scaling matrices in individual past iterations are based only on the data samples employed in those iterations. In other words, the scaling matrix and the incumbent random gradient vector are dependent. That being said, we also provide a version (SMBi), where the matrix and gradient vector in question become independent (see Algorithm 2).

Vaswani et al. (2019) apply a deterministic globalization procedure on mini-batch loss functions. That is, the same sample is used in all function and gradient evaluations needed to apply the line search procedure at a given iteration. However, unlike our case, they employ a standard line search procedure that does not alter the search direction. They establish convergence guarantees for the empirical loss function under the *interpolation* assumption, which requires each component loss function to have zero gradient at a minimizer of the empirical loss. Mutschler and Zell (2020) assume that the optimal learning rate along the negative batch gradient is a good estimator for the optimal learning rate with respect to the empirical loss along the same direction. They test validity of this assumption empirically on deep neural networks (DNNs). Rather than

making such strong assumptions, we stick to the general theory for stochastic quasi-Newton methods.

Other work follow a different approach to translate deterministic line search procedures into a stochastic setting, and they do not employ fixed samples. In Mahsereci and Hennig (2017), a probabilistic model along the search direction is constructed via techniques from Bayesian optimization. Learning rates are chosen to maximize the expected improvement with respect to this model and the probability of satisfying Wolfe conditions. Paquette and Scheinberg (2020) suggest an algorithm closer to the deterministic counterpart, where the convergence is based on the requirement that the stochastic function and gradient evaluations approximate their true values with a high enough probability.

With our current work, we make the following contributions. We use a model building strategy for adjusting the stepsize and the direction of a stochastic gradient vector. This approach also permits us to work on subsets of parameters. This feature makes our model steps not only adaptive, but also suitable to incorporate into the existing implementations of DNNs. Our method changes the direction of the step as well as its size. This property separates our approach from the backtracking line search algorithms. It also incorporates the most recent curvature information from the current point. This is in contrast with the stochastic quasi-Newton methods which use the information from the previous steps. Capitalizing our discussion on the independence of the sample batches, we also give a convergence analysis for SMB. Finally, we illustrate the computational performance of our method with a set of numerical experiments and compare the results against those obtained with other well-known methods.

1. Stochastic Model Building. We introduce a new stochastic unconstrained optimization algorithm in order to approximately solve problems of the form

$$\min_{x\in\Re^n} \quad f(x) = \mathbb{E}[F(x,\xi)], \tag{1}$$

where $F : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and possibly nonconvex, $\xi \in \mathbb{R}^d$ denotes a random variable, and $\mathbb{E}[.]$ stands for the expectation taken with respect to ξ. We assume the existence of a stochastic first-order oracle which outputs a stochastic gradient $g(x,\xi)$ of f for a given x. A common approach to tackle (1) is to solve the empirical risk problem

$$\min_{x\in\Re^n} \quad f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{2}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ is the loss function corresponding to the ith data sample, and N denotes the data sample size which can be very large in modern applications.

As an alternative approach to line search for SGD, we propose a stochastic model building strategy inspired by the work of Öztoprak and Birbil (2018). Unlike core SGD methods, our approach aims at including a curvature information that adjusts not only the stepsize but also the search direction. Öztoprak and Birbil (2018) consider only the deterministic setting and they apply the model building strategy repetitively until a sufficient decent is achieved. In our stochastic setting, however, we have observed experimentally that using multiple model steps does not benefit much to the performance, and its cost to the runtime can be extremely high in deep learning problems. Therefore, if the sufficient decent is not achieved by the stochastic gradient step, then we construct only one model to adjust the size and the direction of the step.

Conventional stochastic quasi-Newton methods adjust the gradient direction by a scaling matrix that is constructed by the information from the previous steps. Our model building approach, however, uses the most recent curvature information around the latest iteration. In the popular deep learning model implementations, model parameters come in groups and updates are applied to each parameter group separately. Therefore, we also propose to build a model for each parameter group separately making the step lengths adaptive.

The proposed iterative algorithm SMB works as follows: At step k, given the iterate x_k, we calculate the stochastic function value $f_k = f(x_k,\xi_k)$ and the mini-batch stochastic gradient $g_k = \frac{1}{m_k}\sum_{i=1}^{m_k} g(x_k,\xi_{k,i})$ at x_k, where m_k is the batch size, and $\xi_k = (\xi_{k,1},\ldots,\xi_{k,m_k})$ is the realization of the random vector ξ. Then, we apply the SGD update to calculate the trial step $s_k^t = -\alpha_k g_k$, where $\{\alpha_k\}_k$ is a sequence of learning rates. With this trial step, we also calculate the function and gradient values $f_k^t = f(x_k^t,\xi_k)$ and $g_k^t = g(x_k^t,\xi_k)$ at

$x_k^t = x_k + s_k^t$. Then, we check the stochastic Armijo condition

$$f_k^t \leq f_k - c\ \alpha_k \|g_k\|^2, \tag{3}$$

where $c > 0$ is a hyper-parameter. If the condition is satisfied and we achieve *sufficient decrease*, then we set $x_{k+1} = x_k^t$ as the next step. If the Armijo condition is not satisfied, then we build a quadratic model using the linear models at the points $x_{k,p}$ and $x_{k,p}^t$ for each parameter group p and find the step $s_{k,p}$ to reach its minimum point. Here, $x_{k,p}$ and $x_{k,p}^t$ denote respectively the coordinates of x_k and x_k^t that correspond to the parameter group p. We calculate the next iterate $x_{k+1} = x_k + s_k$, where $s_k = (s_{k,p_1}, \ldots, s_{k,p_n})$ and n is the number of parameter groups, and proceed to the next step with x_{k+1} . This model step, if needed, requires extra mini-batch function and gradient evaluations (forward and backward pass in deep neural networks).

For each parameter group p, the quadratic model is built by combining the linear models at $x_{k,p}$ and $x_{k,p}^t$, given by

$$l_{k,p}^0(s) := f_k + g_{k,p}^\top s \quad \text{and} \quad l_{k,p}^t(s - s_{k,p}^t) := f_k^t + (g_{k,p}^t)^\top (s - s_{k,p}^t),$$

respectively. Then, the quadratic model becomes

$$m_{k,p}^t(s) := \alpha_{k,p}^0(s) l_{k,p}^0(s) + \alpha_{k,p}^t(s) l_{k,p}^t(s - s_{k,p}^t),$$

where

$$\alpha_{k,p}^0(s) = \frac{(s - s_{k,p}^t)^\top (-s_{k,p}^t)}{(-s_{k,p}^t)^\top (-s_{k,p}^t)} \quad \text{and} \quad \alpha_{k,p}^t(s) = \frac{s^\top s_{k,p}^t}{(s_{k,p}^t)^\top s_{k,p}^t}.$$

The constraint

$$\|s\|^2 + \|s - s_{k,p}^t\|^2 \leq \|s_{k,p}^t\|^2,$$

is also imposed so that the minimum is attained in the region bounded by $x_{k,p}$ and $x_{k,p}^t$. This constraint acts like a trust region. Figure 1 shows the steps of this construction.

In this work, we solve a relaxation of this constrained model as explained in (Öztoprak and Birbil, 2018, Section 2.2). The minimum value of the relaxed model is attained at the point $x_{k,p} + s_{k,p}$ with

$$s_{k,p} = c_{g,p}(\delta) g_{k,p} + c_{y,p}(\delta) y_{k,p} + c_{s,p}(\delta) s_{k,p}^t, \tag{4}$$

where $y_{k,p} := g_{k,p}^t - g_{k,p}$. Here, the coefficients are given as

$$c_{g,p}(\delta) = -\frac{\|s_{k,p}^t\|^2}{2\delta}, \quad c_{y,p}(\delta) = -\frac{\|s_{k,p}^t\|^2}{2\delta\theta}[-(y_{k,p}^\top s_{k,p}^t + 2\delta)(s_{k,p}^t)^\top g_{k,p} + \|s_{k,p}^t\|^2 y_{k,p}^\top g_{k,p}],$$

$$c_{s,p}(\delta) = -\frac{\|s_{k,p}^t\|^2}{2\delta\theta}[-(y_{k,p}^\top s_{k,p}^t + 2\delta) y_{k,p}^\top g_{k,p} + \|y_{k,p}\|^2 (s_{k,p}^t)^\top g_{k,p}],$$

with

$$\theta = \left(y_{k,p}^\top s_{k,p}^t + 2\delta\right)^2 - \|s_{k,p}^t\|^2 \|y_{k,p}\|^2 \text{ and } \delta = \frac{1}{2}\left(\|s_{k,p}^t\| \left(\|y_{k,p}\| + \frac{1}{\eta}\|g_{k,p}\|\right) - y_{k,p}^\top s_{k,p}^t\right), \tag{5}$$

where $0 < \eta < 1$ is a constant which controls the size of $s_{k,p}$ by imposing the condition $\|s_{k,p}\| \leq \eta \|s_{k,p}^t\|$. Then, the adaptive model step becomes $s_k = (s_{k,p_1}, \ldots, s_{k,p_n})$. We note that our construction in terms of different parameter groups lends itself to constructing a different model for each parameter subspace.

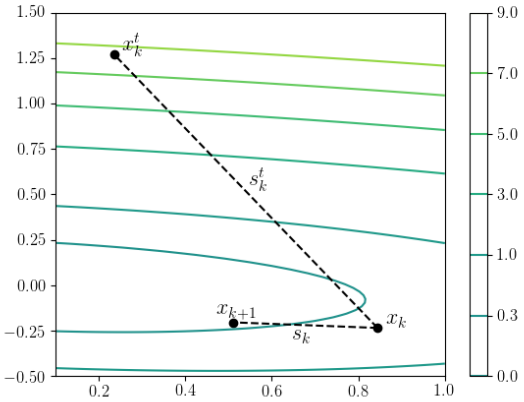


Figure 1: An iteration of SMB on a simple quadratic function. For simplicity we assume that there is only one parameter group, and hence, we drop the subscript p . The algorithm first computes the trial point x_k^t by taking the stochastic gradient step s_k^t. If this point is not acceptable, then it builds a model using the information at x_k and x_k^t, and computes the next iterate $x_{k+1} = x_k + s_k$. Note that s_k not only have a smaller length compared to the trial step s_k^t, but it also lies along a direction decreasing the function value.

We summarize the steps of SMB in Algorithm 1. Line 5 shows the trial point, which is obtained with the standard stochastic gradient step. If this step satisfies the stochastic Armijo condition, then we proceed with the next iteration (line 8). Otherwise, we continue with bulding the models for each parameter group (lines 10- 12), and move to the next iteration with the model building step in line 13.

Algorithm 1: SMB: Stochastic Model Building

1 **Input:** $x_1 \in \mathbb{R}^n$, stepsizes $\{\alpha_k\}_{k=1}^T$, mini-batch sizes $\{m_k\}_{k=1}^T$,$c > 0$, and α_{max} satisfying (8)
2 **for** $k = 1, \ldots, T$ **do**
3 $\quad f_k = f(x_k, \xi_k)$, $g_k = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i})$;
4 $\quad s_k^t = -\alpha_k g_k$;
5 $\quad x_k^t = x_k + s_k^t$;
6 $\quad f_k^t = f(x_k^t, \xi_k)$, $g_k^t = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k^t, \xi_{k,i})$;
7 $\quad$ **if** $f_k^t \leq f_k - c\ \alpha_k \|g_k\|^2$ **then**
8 $\quad\quad x_{k+1} = x_k^t$;
9 $\quad$ **else**
10 $\quad\quad$ **for** $p = 1, \ldots, r$ **do**
11 $\quad\quad\quad y_{k,p} = g_{k,p}^t - g_{k,p}$;
12 $\quad\quad\quad s_{k,p} = c_{g,p}(\delta) g_{k,p} + c_{y,p}(\delta) y_{k,p} + c_{s,p}(\delta) s_{k,p}^t$;
13 $\quad\quad x_{k+1} = x_k + s_k$ with $s_k = (s_{k,p_1}, \ldots, s_{k,p_r})$;

An example run. It is not hard to see that the steps produced by Algorithm 1 always lie in the span of the two stochastic gradients, g_k and g_k^t. In particular, when a model step is computed in line 11, we have

$$s_k = w_1 g_k + w_2 g_k^t.$$

for

$$w_1 = c_g(\delta) - c_y(\delta) - c_s(\delta)\alpha, \quad \text{and} \quad w_2 = c_y(\delta).$$

Therefore, it is interesting to observe how the values of w_1 and w_2 evolve during the course of an SMB run, and how the resulting performance compares to taking SGD steps with various stepsizes. For this purpose, we

investigate the steps of SMB for one epoch on the MNIST dataset with a batch size of 128 (see Section 3 for details of the experimental setting).

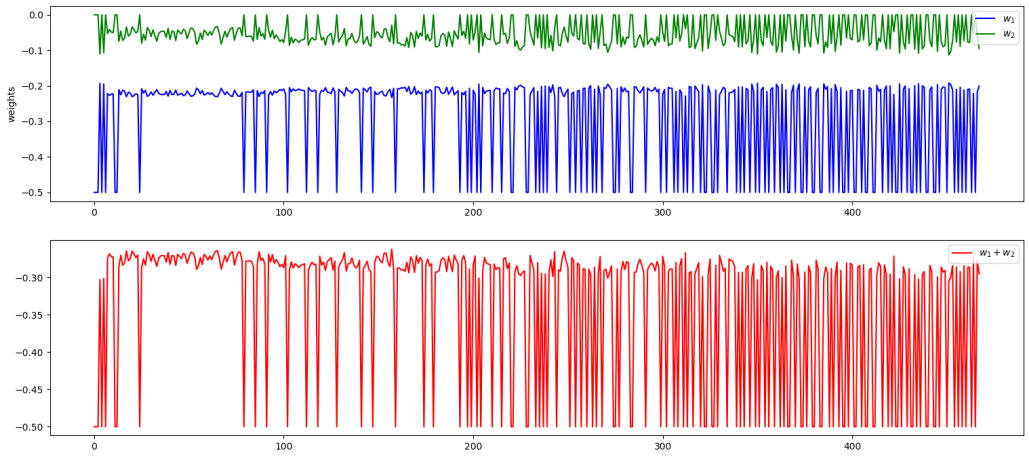


Figure 2: The coefficients of g_k and g_k^t during a single-epoch run of SMB on the MNIST data with $\alpha = 0.5$. Model steps are taken quite often, but not at all iterations. The sum of the two coefficients vary in [-0.5,-0.25].

We provide in Figure 2 the values of w_1 and w_2 for SMB with $\alpha = 0.5$ over the 468 steps taken in an epoch. Note that the computations of g_k^t in line 6 of Algorithm 1 may spend a significant portion of the evaluation budget, if model steps are taken very often. Figure 2 shows that SMB algorithm indeed takes a lot of model steps in this run. To account for the extra gradient evaluations in computing the model steps, we run SGD with a constant learning rate of α on the same problem for two epochs rather than one. Table 1 presents a summary of the resulting training error and testing accuracy values. We observe that the performance of SMB is significantly more stable for different α values, thanks to the adaptive steplength (and the modified search direction) provided by SMB. SGD *can* achieve performance values comparable to or even better than SMB, but only for the *right* values of α.

In Figure 2, it is interesting to see that the values of w_2 are relatively small. We also realize that if we run SGD with a learning rate close to the average $(w_1 + w_2)$ value, it has an inferior performance. For the SMB run with $\alpha = 0.5$, for instance, the average $(w_1 + w_2)$ value is close to -0.3. This can be contrasted with the resulting performance of SGD with $\alpha = 0.3$ in Table 1. These observations suggest that g_k^t contributes to altering the search direction as intended, rather than acting as an *additional stochastic gradient step*.

	$\alpha = 1.0$		$\alpha = 0.5$		$\alpha = 0.3$		$\alpha = 0.1$		$\alpha = 0.05$	
	SGD	SMB	SGD	SMB	SGD	SMB	SGD	SMB	SGD	SMB
Training loss	2.3033	0.3402	2.2947	0.1770	0.7435	0.1889	0.1594	0.3379	0.2410	0.3131
Test accuracy	0.1135	0.8949	0.1137	0.9460	0.7685	0.9422	0.9513	0.8993	0.9298	0.9162

Table 1: Performance on the MNIST data; SMB is run for one epoch, and SGD is run for two epochs.

2. Convergence Analysis. The steps of SMB can be considered as a special quasi-Newton update:

$$x_{k+1} = x_k - \alpha_k H_k g_k, \tag{6}$$

where H_k is a symmetric positive definite matrix as an approximation to the inverse Hessian matrix. In Appendix 4, we explain this connection and give an explicit formula for the matrix H_k. We also prove that

there exists $\underline{\kappa}, \overline{\kappa} > 0$ such that for all k, the matrix H_k satisfies

$$\underline{\kappa} I \preceq H_k \preceq \overline{\kappa} I, \tag{7}$$

where for two matrices A and B, $A \preceq B$ means $B - A$ is positive semidefinite. It is important to note that H_k is built with the information collected around x_k, particularly, g_k. Therefore, unlike stochastic quasi-Newton methods, H_k is correlated with g_k, and hence, $\mathbb{E}_{\xi_k}[H_k g_k]$ is very difficult to analyze. Unfortunately, this difficulty prevents us from using the general framework given by Wang et al. (2017).

To overcome this difficulty and carry on with the convergence analysis, we modify Algorithm 1 such that H_k is calculated with a new independent mini batch, and therefore, it is independent of g_k. By doing so, we still build a model using the information around x_k. Assuming that g_k is an unbiased estimator of ∇f, we conclude that $\mathbb{E}_{\xi_k}[H_k g_k] = H_k \nabla f$. In the rest of this section, we provide a convergence analysis for this modified algorithm which we will call as SMBi (i for independent batch). The steps of SMBi are given in Algorithm 2. As Step 11 shows, we obtain the model building step with a new random batch.

Algorithm 2: SMBi: H_k with an independent batch

1 **Input:** $x_1 \in \mathbb{R}^n$, stepsizes $\{\alpha_k\}_{k=1}^T$, mini-batch sizes $\{m_k\}_{k=1}^T$,$c > 0$, and α_{max} satisfying (8)
2 **for** $k = 1, \ldots, T$ **do**
3 $f_k = f(x_k, \xi_k)$, $g_k = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i})$;
4 $s_k^t = -\alpha_k g_k$;
5 $x_k^t = x_k + s_k^t$;
6 $f_k^t = f(x_k^t, \xi_k)$, $g_k^t = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k^t, \xi_{k,i})$;
7 **if** $f_k^t \leq f_k - c\ \alpha_k \|g_k\|^2$ **then**
8 $x_{k+1} = x_k^t$;
9 **else**
10 **for** $p = 1, \ldots, n$ **do**
11 Choose a new independent random batch ξ_k';
12 $g_k' = \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \xi_{k,i}')$;
13 $(s_k^t)' = -\alpha_k g_k'$, $(x_k^t)' = x_k + (s_k^t)'$;
14 $(g_k^t)' = \frac{1}{m_k} \sum_{i=1}^{m_k} g((x_k^t)', \xi_{k,i}')$, $y_{k,p}' = (g_{k,p}^t)' - g_{k,p}'$;
15 $s_{k,p} = -\alpha_k H_{k,p}' g_k$, where $H_{k,p}'$ is calculated using g_k' and y_k' as defined in Appendix;
16 $x_{k+1} = x_k + s_k$ with $s_k = (s_{k,1}, \ldots, s_{k,n})$;

Assumptions: Before providing the analysis, let us assume that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, lower bounded by f^{low}, and there exists $L > 0$ such that for any $x, y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$. We also assume that ξ_k, $k \geq 1$, are independent samples and for any iteration k, ξ_k is independent of $\{x_j\}_{j=1}^k$, $\mathbb{E}_{\xi_k}[g(x_k, \xi_k)] = \nabla f(x_k)$ and $\mathbb{E}_{\xi_k}[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2] \leq M^2$, for some $M > 0$.

In order to be in line with practical implementations and with our experiments, we first provide an analysis covering the constant stepsize case for (possibly) non-convex objective functions. Below, we denote by $\xi_{[T]} = (\xi_1, \ldots, \xi_T)$ the random samplings in the first T iterations. Let α_{max} be the maximum stepsize that is allowed in the implementation of SMBi with

$$\alpha_{max} \geq \frac{-1 + \sqrt{1 + 16\eta^2}}{4L\eta}. \tag{8}$$

This hyper-parameter of maximum stepsize is needed in the theoretical results. Observe that since $\eta^{-1} > 1$, assuming $L \geq 1$ implies that it suffices to choose $\alpha_{max} \geq 1$ to satisfy (8). The proof of the following convergence result is given in Appendix 4

THEOREM 2.1 *Suppose that our assumptions above hold and $\{x_k\}$ is generated by SMBi as given in Algorithm*

2. Suppose also that $\{\alpha_k\}$ in Algorithm 2 satisfies that $0 < \alpha_k < 2/(L\eta^{-1} + 2L^2\alpha_{max}) \leq \alpha_{max}$ for all k. For given T, let R be a random variable with the probability mass function

$$\mathbb{P}_R(k) := \mathbb{P}\{R = k\} = \frac{\alpha_k/(\eta^{-1} + 2L\alpha_{max}) - \alpha_k^2 L/2}{\sum_{k=1}^{T}(\alpha_k/(\eta^{-1} + 2L\alpha_{max}) - \alpha_k^2 L/2)},$$

for $k = 1, \ldots, T$. Then, we have

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{D_f + (\sigma^2 L/2)\sum_{k=1}^{T}(\alpha_k^2/m_k)}{\sum_{k=1}^{T}(\alpha_k/(\eta^{-1} + 2L\alpha_{max}) - \alpha_k^2 L/2)},$$

where $D_f := f(x_1) - f^{low}$ and the expectation is taken with respect to R and $\xi_{[T]}$. Moreover, if we choose $\alpha_k = 1/(L\eta^{-1} + 2L^2\alpha_{max})$ and $m_k = m$ for all $k = 1, \ldots, T$, then this reduces to

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{2L(\eta^{-1} + 2L\alpha_{max})^2 D_f}{T} + \frac{M^2}{m}.$$

Using this theorem, it is possible to deduce that stochastic first-order oracle complexity of SMB with random output and constant stepsize is $\mathcal{O}(\epsilon^{-2})$ (Wang et al., 2017, Corollary 2.12). In Wang et al. (2017) (Theorem 2.5), it is shown that under our assumptions above and the extra assumption of $0 < \alpha_k \leq \frac{1}{L(\eta^{-1}+2L\alpha_{max})} \leq \alpha_{max}$, if the point sequence $\{x_k\}$ is generated by SMBi method (when H_k is calculated by an independent batch in each step) with batch size $m_k = m$ for all k, then there exists a positive constant M_f such that $\mathbb{E}[f(x_k)] \leq M_f$. Using this observation, the proof of Theorem 2.1, and Theorem 2.8 in (Wang et al., 2017), we can also give the following complexity result when the stepsize sequence is diminishing for non-convex objective functions.

THEOREM 2.2 *Let the batch size be m and assume that $\alpha_k = \frac{1}{L(\eta^{-1}+2L\alpha_{max})}k^{-\phi}$ with $\phi \in (0.5, 1)$ for all k. Then $\{x_k\}$ generated by SMBi satisfies*

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}[\|\nabla f(x_k)\|^2 \leq 2L(\eta^{-1} + 2L\alpha_{max})(M_f - f^{low})T^{\phi-1} + \frac{M^2}{(1-\phi)m}(T^{-\phi} - T^{-1})$$

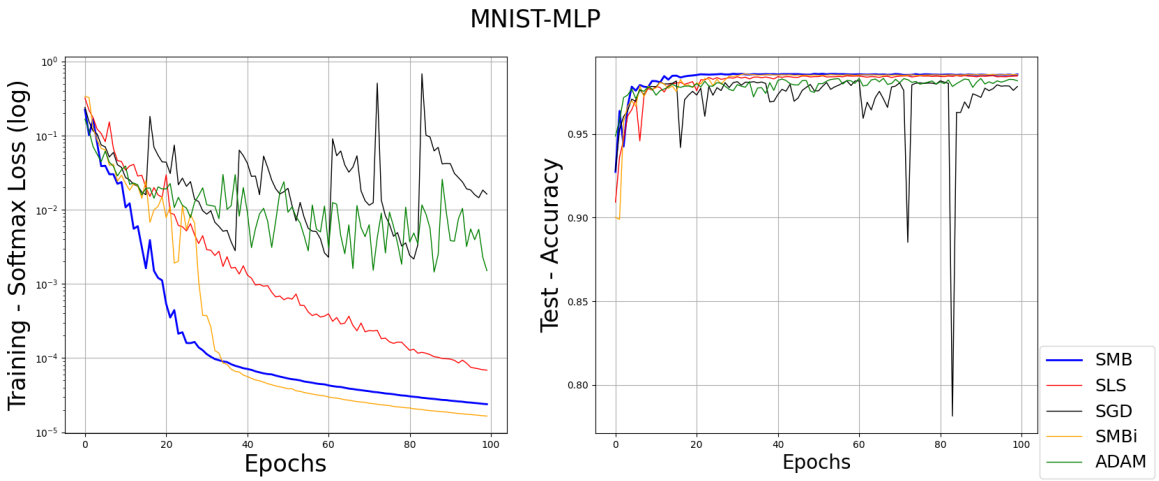
for some $M_f > 0$, where T denotes the iteration number. Moreover, for a given $\epsilon \in (0, 1)$, to guarantee that $\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}[\|\nabla f(x_k)\|^2 < \epsilon$, the number of required iterations T is at most $O\left(\epsilon^{-\frac{1}{1-\phi}}\right)$.

3. Numerical Experiments. In this section, we compare SMB and SMBi against SGD, Adam (Kingma and Ba, 2015), and SLS (SGD+Armijo) (Vaswani et al., 2019). We have chosen SLS, since it is a recent method that uses stochastic line search with backtracking. We have conducted experiments on multi-class classification problems using neural network models*. Our Python package `SMB` along with the scripts to conduct our experiments are available online: https://github.com/sibirbil/SMB

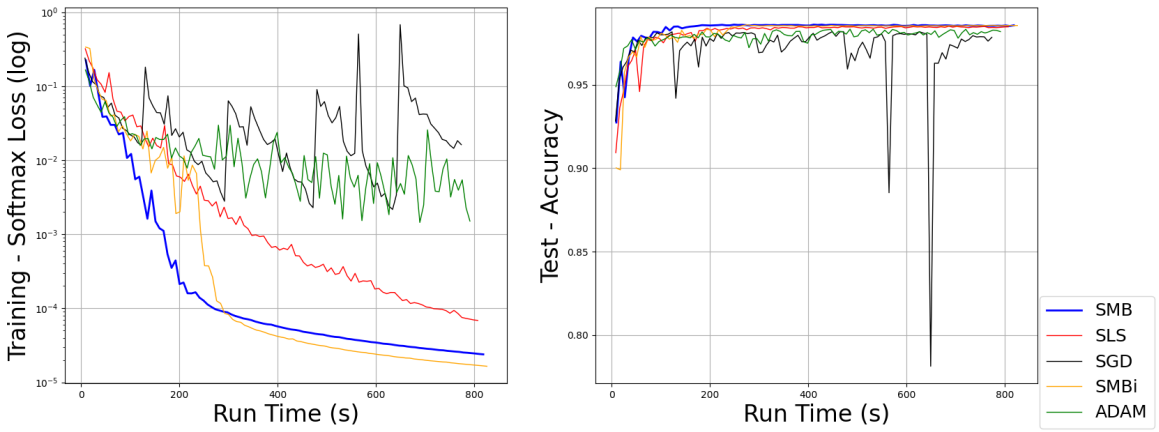
We start our experiments with constant stepsizes for all methods. We should point out that SLS method adjusts the stepsize after each backtracking process and also uses a stepsize reset algorithm between epochs. We refer to this routine as stepsize auto-scheduling. Our numerical experiments show that even without such an auto-scheduling the performances of our methods are on par with SLS. Following the experimental setup in Vaswani et al. (2019), we use the default constant learning rates of 0.5 for SMB and SMBi, 1 for SLS, 0.1 for SGD, and 0.001 for ADAM. For SMB, SMBi, and SLS, we have used the default hyper-parameter value $c = 0.1$ of SLS that appears in the Armijo condition (also recommended by the authors of SLS). Due to the high computational costs of training the neural networks, we report the results of a single run of each method.

MNIST dataset. On the MNIST dataset, we have used the one hidden-layer multi-layer perceptron (MLP) of width 1,000.

*The implementations of the models are taken from https://github.com/IssamLaradji/sls



Training & Test Losses



Training & Test Run Times

Figure 3: Classification on MNIST with an MLP model.

In Figure 3, we see the best performances of all five methods on the MNIST dataset with respect to epochs and run time. Even though SMB and SMBi may calculate an extra function value (forward pass) and a gradient (backward pass), we see in this problem that SMB and SMBi achieve the best performance with respect to the run time as well as the number of epochs. More importantly, the generalization performances of SMB and SMBi are also better than the remaining three methods.

It should be pointed out that, in practice, choosing a new independent batch means the SMBi method can construct a model step in two iteration using two batches. This way the computation cost for each iteration is reduced on average with respect to SMB but the model steps can only be taken in half of the iterations in the epoch. As seen in Figure 3, this does not seem to effect the performance in this problem significantly.

CIFAR10 and CIFAR100 datasets. For the CIFAR10 and CIFAR100 datasets, we have used the standard image-classification architectures ResNet-34 (He et al., 2016) and DenseNet-121 (Huang et al., 2017).

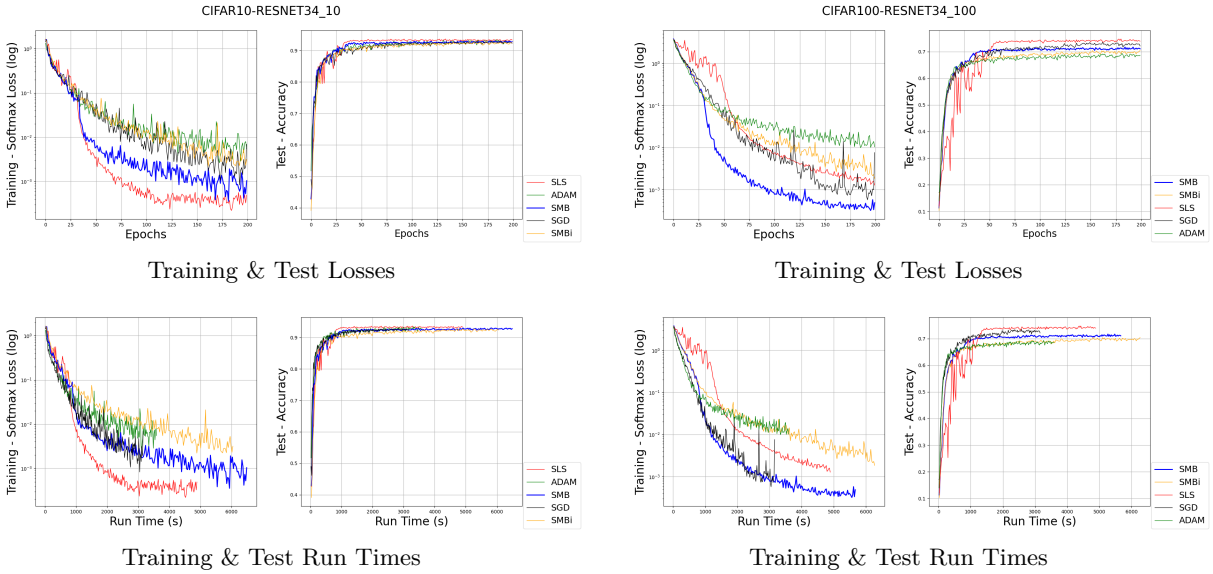


Figure 4: Classification on CIFAR10 (left column) and CIFAR100 (right column) with ResNet-34 model.

In Figure 4, we see that on CIFAR10-Resnet34, SMB performs better than Adam and SGD algorithms. However, its performance is only comparable to SLS. Even though SMB reaches a lower training loss value in CIFAR100-Resnet34, this advantage does not show in test accuracy.

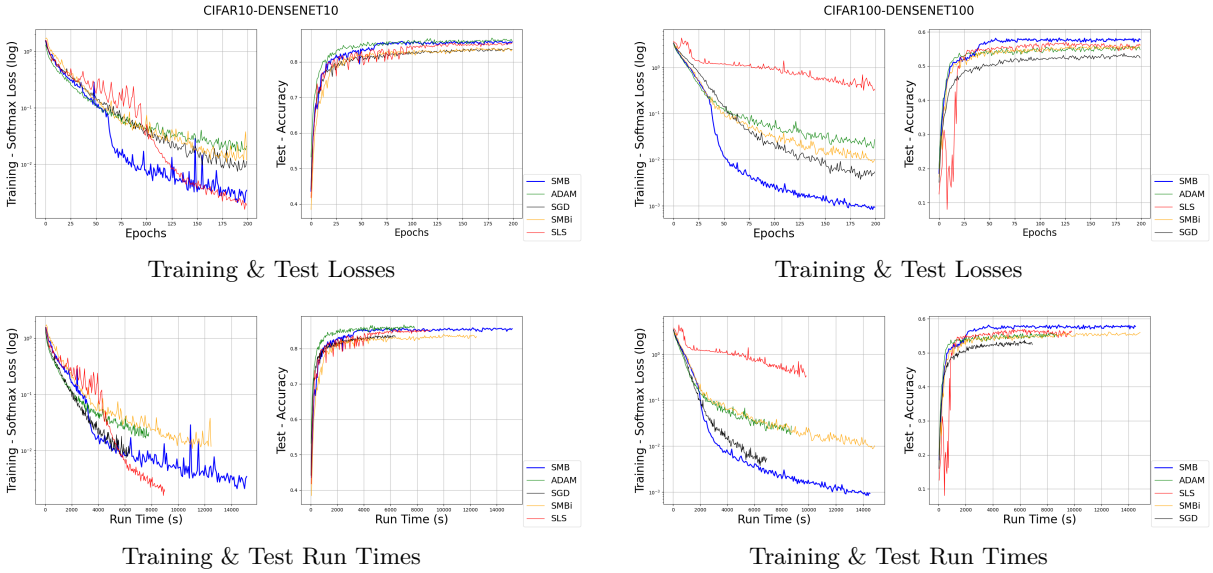


Figure 5: Classification on CIFAR10 (left column) and CIFAR100 (right column) with Densenet121 model.

In Figure 5, we see a comparison of performances of on CIFAR10 and CIFAR100 with DenseNet121. SMB with a constant stepsize outperforms all other optimizers in terms of training error and reaches the best test accuracy on CIFAR100, while showing similar accuracy with ADAM on CIFAR10.

Our last set of experiments are devoted to demonstrating the robustness of SMB. The preliminary results in Figure 6 show that SMB is robust to the choice of the learning rate, especially in deep neural networks. This aspect of SMB needs more attention theoretically and experimentally.

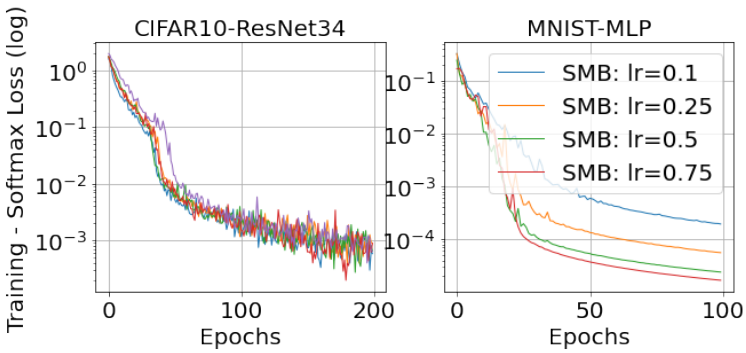


Figure 6: Robustness of SMB under different choices of the learning rate.

4. Conclusion. Stochastic model building (SMB) is a fast alternative to stochastic gradient descent method (SGD). The algorithm provides a model building approach that replaces the one-step backtracking in stochastic line search methods. We have analyzed the convergence properties of a modification of SMB by rewriting its model building step as a quasi-Newton update and constructing the scaling matrix with a new independent batch. Our numerical results have shown that SMB converges fast and its performance is insensitive to the selected stepsize.

In its current state, SMB lacks any internal learning rate adjusting mechanism that could reset the learning rate depending on the progression of the iterations. Our initial experiments show that SMB can greatly benefit from a stepsize auto-scheduling routine. This is a future work that we will consider. Our convergence rate analysis is given for the alternative algorithm SMBi which can perform competitive against other methods, but consistently underperforms the original SMB method. This begs for a convergence analysis for the SMB method.

References

Asi, H. and Duchi, J. C. (2019). The importance of better models in stochastic optimization. *Proceedings of the National Academy of Sciences*, 116(46):22924–22930.

Balles, L., Romero, J., and Hennig, P. (2016). Coupling adaptive batch sizes with learning rates. *arXiv preprint arXiv:1612.05086*.

Bollapragada, R., Byrd, R., and Nocedal, J. (2018). Adaptive sampling strategies for stochastic optimization. *SIAM Journal on Optimization*, 28(4):3312–3343.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155.

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. (2016). A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. CVPR.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. CVPR.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mahsereci, M. and Hennig, P. (2017). Probabilistic line searches for stochastic optimization. *The Journal of Machine Learning Research*, 18(1):4262–4320.

Mokhtari, A. and Ribeiro, A. (2014). Res: Regularized stochastic bfgs algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104.

Mutschler, M. and Zell, A. (2020). Parabolic approximation line search for dnns. *arXiv preprint arXiv:1903.11991*.

Öztoprak, F. and Birbil, Ş. İ. (2018). An alternative globalization strategy for unconstrained optimization. *Optimization*, 67(3):377–392.

Paquette, C. and Scheinberg, K. (2020). A stochastic line search method with expected complexity analysis. *SIAM Journal on Optimization*, 30(1):349–376.

Schraudolph, N. N., Yu, J., and Günter, S. (2007). A stochastic quasi-newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443. PMLR.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. (2019). Painless stochastic gradient: Interpolation, line-search, and convergence rates. *arXiv preprint arXiv:1905.09997*.

Wang, X., Ma, S., Goldfarb, D., and Liu, W. (2017). Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956.

APPENDIX

Proof of Theorem 2.1 First we show that the SMB step for each parameter group p can be expressed as a special quasi-Newton update. For brevity, let us use s_k, s_k^t, g_k, g_k^t, and y_k instead of $s_{k,p}$, $s_{k,p}^t$, $g_{k,p}$, $g_{k,p}^t$, and $y_{k,p}$, respectively. Recalling the definitions of θ and δ given in (5), observe that

$$2\delta = \|s_k^t\|\|y_k\| + \frac{1}{\eta}\|s_k^t\|\|g_k\| - y_k^\top s_k^t = \alpha_k\left(\|g_k\|\|y_k\| + \frac{1}{\eta}\|g_k\|^2 + y_k^\top g_k\right) = \alpha_k\sigma,$$

and

$$\theta = \left(y_k^\top s_k^t + 2\delta\right)^2 - \|s_k^t\|^2\|y_k\|^2 = \alpha_k^2(\sigma - y_k^\top g_k)^2 - \alpha_k^2\|g_k\|^2\|y_k\|^2 = \alpha_k^2(\beta^2 - \|g_k\|^2\|y_k\|^2) = \alpha_k^2\gamma,$$

where

$$\sigma = \|g_k\|\|y_k\| + \frac{1}{\eta}\|g_k\|^2 + y_k^\top g_k,\ \beta = \sigma - y_k^\top g_k,\ \text{and}\ \gamma = (\beta^2 - \|g_k\|^2\|y_k\|^2).$$

Therefore, we have

$$c_g(\delta)g_k = -\frac{\|s_k^t\|^2}{2\delta}g_k = -\frac{\alpha_k^2\|g_k\|^2}{\alpha_k\sigma\gamma}\gamma g_k = -\alpha_k\frac{\|g_k\|^2}{\sigma\gamma}\gamma g_k,$$

$$\begin{aligned} c_y(\delta)y_k &= -\frac{\|s_k^t\|^2}{2\delta\theta}[-(y_k^\top s_k^t + 2\delta)(s_k^t)^\top g_k + \|s_k^t\|^2 y_k^\top g_k]y_k \\ &= -\frac{\|g_k\|^2}{\alpha_k\sigma\gamma}y_k[\alpha_k^2(\sigma - y_k^\top g_k)g_k^\top g_k + \alpha_k^2\|g_k\|^2 y_k^\top g_k] \\ &= -\alpha_k\frac{\|g_k\|^2}{\sigma\gamma}[\beta y_k g_k^\top + \|g_k\|^2 y_k y_k^\top]g_k, \end{aligned}$$

and

$$\begin{aligned} c_s(\delta)s_k^t &= -\frac{\|s_k^t\|^2}{2\delta\theta}[-(y_k^\top s_k^t + 2\delta)y_k^\top g_k + \|y_k\|^2(s_k^t)^\top g_k]s_k^t \\ &= -\frac{\|g_k\|^2}{\alpha_k\sigma\gamma}(-\alpha_k)g_k[-\alpha_k(\sigma - y_k^\top g_k)y_k^\top g_k - \alpha_k\|y_k\|^2 g_k^\top g_k] \\ &= -\alpha_k\frac{\|g_k\|^2}{\sigma\gamma}[\beta g_k y_k^\top + \|y_k\|^2 g_k g_k^\top]g_k. \end{aligned}$$

Now, it is easy to see that

$$\begin{aligned} s_k &= c_g(\delta)g_k + c_y(\delta)y_k + c_s(\delta)s_k^t \\ &= -\alpha_k\frac{\|g_k\|^2}{\sigma\gamma}\left[\gamma I + \beta y_k g_k^\top + \|g_k\|^2 y_k y_k^\top + \beta g_k y_k^\top + \|y_k\|^2 g_k g_k^\top\right]g_k. \end{aligned}$$

Thus, for each parameter group p, we define

$$H_{k,p} = \frac{\|g_{k,p}\|^2}{\sigma_p\gamma_p}\left[\gamma_p I + \beta_p y_{k,p}g_{k,p}^\top + \|g_{k,p}\|^2 y_{k,p}y_{k,p}^\top + \beta_p g_{k,p}y_{k,p}^\top + \|y_{k,p}\|^2 g_{k,p}g_{k,p}^\top\right], \tag{9}$$

where

$$\sigma_p = \|g_{k,p}\|\|y_{k,p}\| + \frac{1}{\eta}\|g_{k,p}\|^2 + y_{k,p}^\top g_{k,p},\ \beta_p = \sigma_p - y_{k,p}^\top g_{k,p},\ \text{and}\ \gamma_p = (\beta_p^2 - \|g_{k,p}\|^2\|y_{k,p}\|^2).$$

Now, assuming that we have the parameter groups $\{p_1, \ldots, p_n\}$, the SMB steps can be expressed as a quasi-Newton update given by

$$x_{k+1} = x_k - \alpha_k H_k g_k,$$

where

$$H_k = \begin{cases} I, & \text{if the Armijo condition is satisfied;} \\ \operatorname{diag}(H_{k,p_1}, \ldots, H_{k,p_n}), & \text{otherwise.} \end{cases}$$

Here, I denotes the identity matrix, and $\operatorname{diag}(H_{k,p_1}, \ldots, H_{k,p_n})$ denotes the block diagonal matrix with the blocks $H_{k,p_1}, \ldots, H_{k,p_n}$.

We next show that the eigenvalues of the matrices H_k, $k \geq 1$, are bounded from above and below uniformly which is, of course, obvious when $H_k = I$. Using the Sherman-Morrison formula twice, one can see that for each parameter group p, the matrix $H_{k,p}$ is indeed the inverse of the positive semidefinite matrix

$$B_{k,p} = \frac{1}{\|g_{k,p}\|^2}(\sigma_p I - g_{k,p}y_{k,p}^\top - y_{k,p}g_{k,p}^\top),$$

and hence, it is also positive semidefinite. Therefore, it is enough to show the boundedness of the eigenvalues of $B_{k,p}$ uniformly on k and p.

Since $g_{k,p}y_{k,p}^\top + y_{k,p}g_{k,p}^\top$ is a rank two matrix, $\sigma_p/\|g_{k,p}\|^2$ is an eigenvalue of $B_{k,p}$ with multiplicity $n-2$. The remaining extreme eigenvalues are

$$\lambda_{max}(B_{k,p}) = \frac{1}{\|g_{k,p}\|^2}(\sigma_p + \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}) \quad \text{and} \quad \lambda_{min}(B_{k,p}) = \frac{1}{\|g_{k,p}\|^2}(\sigma_p - \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}),$$

with the corresponding eigenvectors $\|y_{k,p}\|g_{k,p} + \|g_{k,p}\|y_{k,p}$ and $\|y_{k,p}\|g_{k,p} - \|g_{k,p}\|y_{k,p}$, respectively.

Observe that,

$$\begin{aligned} \lambda_{min}(B_{k,p}) &= \frac{\sigma_p - \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}}{\|g_{k,p}\|^2} \\ &= \frac{\|g_{k,p}\|\|y_{k,p}\| + \eta^{-1}\|g_{k,p}\|^2 + y_{k,p}^\top g_{k,p} - \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}}{\|g_{k,p}\|^2} \\ &= \frac{\eta^{-1}\|g_{k,p}\|^2}{\|g_{k,p}\|^2} = \frac{1}{\eta} > 1. \end{aligned}$$

Thus, the smallest eigenvalue $B_{k,p}$ is bounded away from zero uniformly on k and p.

Now, by our assumption of Lipschitz continuity of the gradients, for any $x, y \in \mathbb{R}^n$ and ξ_k, we have

$$\|g(x, \xi_k) - g(y, \xi_k)\| \leq L\|x - y\|.$$

Thus, observing that $\|y_{k,p}\| = \|g_{k,p}^t - g_{k,p}\| \leq L\|x_{k,p}^t - x_{k,p}\| \leq \alpha_k L\|g_{k,p}\|$, we have

$$\begin{aligned} \lambda_{max}(B_{k,p}) &= \frac{\sigma_p + \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}}{\|g_{k,p}\|^2} \\ &= \frac{\|g_{k,p}\|\|y_{k,p}\| + \eta^{-1}\|g_{k,p}\|^2 + y_{k,p}^\top g_{k,p} + \|g_{k,p}\|\|y_{k,p}\| - y_{k,p}^\top g_{k,p}}{\|g_{k,p}\|^2} \\ &= \frac{2\|g_{k,p}\|\|y_{k,p}\| + \eta^{-1}\|g_{k,p}\|^2}{\|g_{k,p}\|^2} \leq 2L\alpha_k + \frac{1}{\eta} \leq 2L\alpha_{max} + \eta^{-1}. \end{aligned}$$

This implies that the eigenvalues of $H_{k,p} = B_{k,p}^{-1}$ are bounded below by $1/(\eta^{-1} + 2L\alpha_{max})$ and bounded above by 1 uniformly on k and p. This result, together with our assumptions, shows that steps of the SMBi algorithm satisfy the conditions of Theorem 2.10 in (Wang et al., 2017) with $\underline{\kappa} = 1/(\eta^{-1} + 2L\alpha_{max})$ and $\overline{\kappa} = 1$ and Theorem 2.1 follows as a corollary.