Stochastic Anderson Mixing for Nonconvex Stochastic Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Anderson mixing (AM) is an acceleration method for fixed-point iterations. Despite its success and wide usage in scientific computing, the convergence theory of AM remains unclear, and its applications to machine learning problems are not well explored. In this paper, by introducing damped projection and adaptive regularization to classical AM, we propose a Stochastic Anderson Mixing (SAM) scheme to solve nonconvex stochastic optimization problems. Under mild assumptions, we establish the convergence theory of SAM, including the almost sure convergence to stationary points and the worst-case iteration complexity. Moreover, the complexity bound can be improved when randomly choosing an iterate as the output. To further accelerate the convergence, we incorporate a variance reduction technique into the proposed SAM. We also propose a preconditioned mixing strategy for SAM which can empirically achieve faster convergence or better generalization ability. Finally, we apply the SAM method to train various neural networks including the vanilla CNN, ResNets, WideResNet, ResNeXt, DenseNet and RNN. Experimental results on image classification and language model demonstrate the advantages of our method.

1 Introduction

Stochastic optimization is important in various areas such as statistics [14] and machine learning [5, 54], power systems [25]. In this paper, we consider the following stochastic optimization problem:

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

where $F : \mathbb{R}^d \times \mathbb{R}^{d_\xi} \to \mathbb{R}$ is continuously differentiable and possibly nonconvex and the random variable $\xi \in \mathbb{R}^{d_\xi}$ may follow an unknown probability distribution. It is assumed that only noisy information about the gradient of f is available through calls to some *stochastic first-order oracle* ($\mathcal{SFO}$). One special case of (1) is the *empirical risk minimization* problem:

$$\min_{x\in\mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{T}\sum_{i=1}^{T} f_{\xi_i}(x), \tag{2}$$

where $f_{\xi_i} : \mathbb{R}^d \to \mathbb{R}$ is the loss function corresponding to the i-th data sample and T denotes the number of data samples. T can be extremely large such that it prohibits the computation of the full gradient ∇f. Thus designing efficient and effective numerical algorithm for solving problem (1) or (2) with rigorous theoretical analysis is a challenging task.

One classical approach for solving (1) is the stochastic gradient descent (SGD) method [45]. It mimics GD method by using noisy gradients and exhibits optimal convergence rate for some strongly convex

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

stochastic problems [10, 48]. Some early related works of SGD in convex optimization can be found in [39, 40, 41]. For nonconvex cases, Ghadimi and Lan [17] propose a randomized stochastic gradient (RSG) method that randomly selects an solution $\bar{x}$ from previous iterates. To ensure $\bar{x}$ satisfying $\mathbb{E}\left[\|\nabla f(\bar{x})\|_2^2\right] \leq \epsilon$, the total number of $\mathcal{SFO}$-calls needed by RSG is $O\left(\epsilon^{-2}\right)$. Spurred by the development of machine learning, adaptive learning rate methods are proposed to accelerate SGD, e.g. Adagrad [11], RMSprop [56] and Adam [29], though the convergence analyses of these methods only apply to convex cases. There are also many works on second-order optimization methods [49, 37, 38]. One notable work is the framework of stochastic quasi-Newton (SQN) method proposed by Wang et al. [60], which covers a class of SQN methods and has theoretical guarantees in nonconvex stochastic optimization. However, these second-order methods usually demand more gradient evaluations in every iteration and less noisy gradient information to achieve actual acceleration [5].

In this paper, we develop a novel second-order method based on Anderson mixing (AM) [2], a sequence acceleration method [6] in scientific computing. AM is widely used to accelerate the slow convergence of nonlinear fixed-point iterations arisen in computational physics and quantum chemistry, e.g., the Self-Consistent Field (SCF) iteration in electronic structure calculations [15, 9], where the function evaluation is costly. AM is appropriate to high dimensional problem since it extrapolates a new iterate using a limit number of historical iterates. It turns out that AM is closely related to multisecant quasi-Newton methods in nonlinear problems [13, 6] or the generalized minimal residual (GMRES) method [46] in linear problems [59, 42].

Inspired by the great success of AM in accelerating fixed-point iterations, it is natural to ask whether AM can be applied to accelerate nonlinear optimization since the gradient descent (GD) method for searching a saddle point in unconstrained optimization is a fixed-point iteration when using constant stepsize. This idea has been explored in [50, 51], but the proposed Regularized Nonlinear Acceleration (RNA) method is built on the minimal polynomial extrapolation (MPE) approach [6], a sequence transformation method that has subtle difference from AM. Also, their methods rely heavily on the contraction assumption of the fixed-point map and the strong convexity. For AM, although current research has proved linear convergence of AM for fixed-point iterations under some conditions [57, 58, 4], there exists no version of AM that guarantees convergence for nonconvex optimization, let alone stochastic optimization.

In this paper, we develop a stochastic extension of AM. Due to the nonconvexity and noise inside the problems and lack of line search or trust-region, a straightforward migration of AM to nonconvex stochastic optimization is infeasible. As a result, we make several fundamental modifications to AM. We highlight the main contributions of our works as follows:

1. We develop a stochastic version of AM, namely Stochastic Anderson Mixing (SAM), by introducing *damped projection* and *adaptive regularization*. We prove its almost sure convergence to a stationary point and analyze its work complexity. When a randomly chosen iterate x_R is returned as the output of SAM, we prove that the worst-case $\mathcal{SFO}$-calls complexity to guarantee $\mathbb{E}\left[\|\nabla f(x_R)\|_2^2\right] \leq \epsilon$ is $O\left(\epsilon^{-2}\right)$. (See Theorem 1 - 4.)

2. We give a variance reduced extension of SAM by borrowing the stochastic variance reduced gradient (SVRG) [28] technique and analyze its $\mathcal{SFO}$-calls complexity. (See Theorem 5.) We also propose a preconditioned mixing strategy for AM and obtain the preconditioned SAM method which can empirically converge faster or generalize better. (See Section 2.3.)

3. Extensive experiments on training Convolutional Neural Network (CNN), ResNet, WideResNet, ResNeXt, DenseNet, and Recurrent Neural Network (RNN) on different tasks and datasets show the faster convergence or better generalization ability of our method compared with the state-of-the-art methods. (See Section 4.)

2 Methodology

2.1 Anderson Mixing

AM is proposed for acceleration of fixed-point iterations. We assume the fixed-point iteration is $x_{k+1} = g(x_k) \stackrel{\text{def}}{=} x_k + r_k$, where $r_k \stackrel{\text{def}}{=} -\nabla f(x_k)$. Then $g(x_k) - x_k = r_k$. Here, we adopt the description of AM in [13, 59]. Let Δ denote the forward difference operator, say, $\Delta x_k = x_{k+1} - x_k$.

Let X_k and R_k record the most recent $m(m \leq k)$ iterations:

$$X_k = [\Delta x_{k-m}, \Delta x_{k-m+1}, \cdots, \Delta x_{k-1}], R_k = [\Delta r_{k-m}, \Delta r_{k-m+1}, \cdots, \Delta r_{k-1}]. \tag{3}$$

AM can be decoupled into two steps. We call them *the projection step* and *the mixing step*:

$$\bar{x}_k = x_k - X_k \Gamma_k, \quad \text{(Projection step)}, \tag{4a}$$

$$x_{k+1} = \bar{x}_k + \beta_k \bar{r}_k, \quad \text{(Mixing step)}, \tag{4b}$$

where β_k is the mixing parameter, and $\bar{r}_k \stackrel{\text{def}}{=} r_k - R_k \Gamma_k$ is reminiscent of *extragradient* [30]. Γ_k is determined by solving

$$\Gamma_k = \arg\min_{\Gamma \in \mathbb{R}^m} \|r_k - R_k \Gamma\|_2. \tag{5}$$

Combining (4a) and (4b), we obtain the full form of AM [13, 59, 6, 43]:

$$x_{k+1} = x_k + \beta_k r_k - (X_k + \beta_k R_k) \Gamma_k. \tag{6}$$

Remark 1. *To see the rationality of AM, we assume f is twice continuously differentiable. Then a quadratic approximation of f implies $\nabla^2 f(x_k)(x_j - x_{j-1}) \approx \nabla f(x_j) - \nabla f(x_{j-1})$ in a local small region around x_k, so it is reasonable to assume $R_k \approx -\nabla^2 f(x_k) X_k$. Thus we see $\|r_k - R_k \Gamma\|_2 \approx \|r_k + \nabla^2 f(x_k) X_k \Gamma\|_2$. Hence, we can recognize (5) as solving $\nabla^2 f(x_k) p_k = \nabla f(x_k)$ in a least-squares sense, where $p_k = X_k \Gamma_k$. When the quadratic approximation is exact, solving (5) is a minimal residual procedure, thus being verified as a* residual projection *method [47]. From this viewpoint, the fixed-point assumption is unnecessary as long as $R_k \approx -\nabla^2 f(x_k) X_k$. Moreover, let H_k be the solution to a constrained optimization problem [13]:*

$$\min_{H_k} \|H_k - \beta_k I\|_F \text{ subject to } H_k R_k = -X_k, \tag{7}$$

then iterate (6) *is $x_{k+1} = x_k + H_k r_k$, which is indeed a multisecant quasi-Newton method. Note that a key simplification in AM is using differences of historical gradients R_k to approximate $-\nabla^2 f(x_k) X_k$, which reduces the heavy cost to compute Hessian-vector products [8, 19, 26].*

2.2 Stochastic Anderson Mixing

We describe our method Stochastic Anderson Mixing (SAM) in this section. At the k-th iteration, let $S_k \subseteq [T] \stackrel{\text{def}}{=} \{1, 2, \ldots, T\}$ be the sampled mini-batch and the corresponding objective function value is $f_{S_k}(x_k) = \frac{1}{|S_k|} \sum_{i \in S_k} f_{\xi_i}(x_k)$. Then $r_k \stackrel{\text{def}}{=} -\nabla f_{S_k}(x_k)$ and the noisy R_k is defined correspondingly (cf. (3)). Due to the instability and inaccurate estimation of R_k, we stabilize the projection step by proposing *damped projection* and *adaptive regularization* techniques. Algorithm 1 is a sketch of our method. We elaborate the mechanism of this algorithm in the following subsections.

Damped projection From Remark 1, we see the determination of Γ_k in (5) relies on the local quadratic approximation of (2), which can be rather inexact in general nonlinear optimization. To improve the stability, we propose a *damped projection* method for (4a). Let α_k be the damping parameter, we obtain $\bar{x}_k$ via

$$\bar{x}_k = (1 - \alpha_k) x_k + \alpha_k (x_k - X_k \Gamma_k) = x_k - \alpha_k X_k \Gamma_k, \tag{8}$$

Combining (8) and (4b) and noting that $\bar{r}_k = r_k - \alpha_k R_k \Gamma_k$, the new iterate x_{k+1} is given by

$$x_{k+1} = x_k + \beta_k r_k - \alpha_k (X_k + \beta_k R_k) \Gamma_k. \tag{9}$$

It is worth noting that β_k and α_k in (9) behave like stepsize or learning rate in SGD, and the extra term $(\alpha_k X_k + \alpha_k \beta_k R_k) \Gamma_k$ can be viewed as a generalized momentum term.

Adaptive regularization Since R_k may be rank deficient and no safeguard method is used in AM, the least square problem (5) can be unstable. A remedy is to add *regularization* [7] to (5). One well known choice is the Tikhonov regularization introduced in [50, 53], which can be viewed as forcing $\|\Gamma_k\|_2$ to be small [53], leading to a penalty term to (5):

$$\Gamma_k = \arg\min_{\Gamma \in \mathbb{R}^m} \|r_k - R_k \Gamma\|_2^2 + \delta \|\Gamma\|_2^2, \tag{10}$$

Algorithm 1 Stochastic Anderson Mixing (SAM)

Input: $x_0 \in \mathbb{R}^d, m = 10, \alpha_k = 1, \beta_k = 1, \mu = 10^{-8}, max_iter > 0$.
Output: $x \in \mathbb{R}^d$

for $k = 0, 1, \ldots, max_iter$ **do**
  $r_k = -\nabla f_{S_k}(x_k)$
  if $k = 0$ **then**
    $x_{k+1} = x_k + \beta_k r_k$
  else
    $m_k = \min\{m, k\}$
    $X_k = [\Delta x_{k-m_k}, \Delta x_{k-m_k+1}, \cdots, \Delta x_{k-1}]$
    $R_k = [\Delta r_{k-m_k}, \Delta r_{k-m_k+1}, \cdots, \Delta r_{k-1}]$
    Check Condition (15) and use smaller α_k if (15) is violated.
    $x_{k+1} = x_k + \beta_k r_k - (\alpha_k X_k + \alpha_k \beta_k R_k)\left(R_k^{\mathrm{T}} R_k + \delta_k X_k^{\mathrm{T}} X_k\right)^{\dagger} R_k^{\mathrm{T}} r_k$
  end if
  Apply learning rate schedule of α_k, β_k
end for
return x_k

where $\delta \geq 0$ is the penalized constant. The solution of (10) is $\Gamma_k = \left(R_k^{\mathrm{T}} R_k + \delta I\right)^{\dagger} R_k^{\mathrm{T}} r_k$, where "$\dagger$" denotes the Penrose-Moore inverse. We name this regularized variant of AM as RAM.

Here, we propose a new regularization, namely *adaptive regularization*, to better suit the stochastic optimization. Since $-X_k \Gamma_k = \bar{x}_k - x_k$ denotes the update from x_k to $\bar{x}_k$, a large magnitude of $\|X_k \Gamma_k\|_2$ tends to make the intermediate step $\bar{x}_k$ overshoot the trust region around x_k. Thus it is more reasonable to force $\|X_k \Gamma_k\|_2$ rather than $\|\Gamma_k\|_2$ to be small. We formulate this idea as

$$\min_{\Gamma} \|r_k - R_k \Gamma\|_2^2 + \delta_k \|X_k \Gamma\|_2^2, \tag{11}$$

where $\delta_k \geq 0$ is a variable determined in each iteration. Explicitly solving (11) leads to

$$\Gamma_k = \left(R_k^{\mathrm{T}} R_k + \delta_k X_k^{\mathrm{T}} X_k\right)^{\dagger} R_k^{\mathrm{T}} r_k, \tag{12}$$

We call AM with this regularization and damped projection as SAM, i.e. the algorithm given in Algorithm 1. The choice of δ_k should reflect the curvature change in the vicinity of x_k, so we give a special choice of δ_k:

$$\delta_k = \max\left\{\frac{c_1 \|r_k\|_2^2}{\|x_k - x_{k-1}\|_2^2 + \epsilon}, c_2 \beta_k^{-2}\right\}, \tag{13}$$

where c_1, c_2 are constants, ϵ is a small constant to prevent the denominator from being zero. Such form of δ_k is reminiscent of AdaDelta [63]. A large $\|r_k\|_2$ indicates a potential dramatic change in landscape, suggesting using a precautious tiny stepsize. The denominator in (13) behaves like *annealing*. In Secant Penalized BFGS [27], this term measures the noise in gradients. In Section 3.2, we will further explain the rationality of (13). We name this new method as AdaSAM.

Positive definiteness. From (9) and (12), the SAM update is $x_{k+1} = x_k + H_k r_k$, where $H_k = \beta_k I - \alpha_k Y_k Z_k^{\dagger} R_k^{\mathrm{T}}$, $Y_k = X_k + \beta_k R_k$, $Z_k = R_k^{\mathrm{T}} R_k + \delta_k X_k^{\mathrm{T}} X_k$. H_k is generally not symmetric. A critical condition for the convergence analysis of SAM is the *positive definiteness* of H_k, i.e.

$$p_k^{\mathrm{T}} H_k p_k \geq \beta_k \mu \|p_k\|_2^2, \quad \forall p_k \in \mathbb{R}^d, \tag{14}$$

where $\mu \in (0, 1)$ is a constant. Next, we give an approach to guarantee it.

Let $\lambda_{min}(\cdot)$ denote the smallest eigenvalue, $\lambda_{max}(\cdot)$ denote the largest eigenvalue. Since $p_k^{\mathrm{T}} H_k p_k = \frac{1}{2} p_k^{\mathrm{T}} (H_k + H_k^{\mathrm{T}}) p_k$, Condition (14) is equivalent to $\lambda_{min}\left(\frac{1}{2}\left(H_k + H_k^{\mathrm{T}}\right)\right) \geq \beta_k \mu$. With some simple algebraic operations, we obtain $\lambda_{min}\left(\frac{1}{2}\left(H_k + H_k^{\mathrm{T}}\right)\right) = \beta_k - \frac{1}{2}\alpha_k \lambda_{max}(Y_k Z_k^{\dagger} R_k^{\mathrm{T}} + R_k Z_k^{\dagger} Y_k^{\mathrm{T}})$. Let $\lambda_k \overset{\text{def}}{=} \lambda_{max}(Y_k Z_k^{\dagger} R_k^{\mathrm{T}} + R_k Z_k^{\dagger} Y_k^{\mathrm{T}})$, then Condition (14) is equivalent to

$$\alpha_k \lambda_k \leq 2\beta_k (1 - \mu). \tag{15}$$

Algorithm 2 Stochastic Anderson Mixing with variance reduction (SAM-VR)

Input: $\tilde{x}_0 \in \mathbb{R}^d$; $\beta_t^k, \alpha_t^k, \delta_t^k$ for $SAM_update(x_t^k, g_t^k)$; Batch size $n \geq 1$.
Output: $x \in \mathbb{R}^d$

for $k = 0, \ldots, N-1$ **do**
  $x_0^k = \tilde{x}_k$
  $\nabla f(\tilde{x}_k) = \frac{1}{T}\sum_{i=1}^{T} \nabla f_{\xi_i}(\tilde{x}_k)$
  for $t = 0, \ldots, q-1$ **do**
    Sample a subset $\mathcal{K} \subseteq [T]$ with $|\mathcal{K}| = n$
    $g_t^k = \nabla f_{\mathcal{K}}(x_t^k) - \nabla f_{\mathcal{K}}(\tilde{x}_k) + \nabla f(\tilde{x}_k)$ where $\nabla f_{\mathcal{K}}(x_t^k) = \frac{1}{|\mathcal{K}|}\sum_{i\in\mathcal{K}} \nabla f_{\xi_i}(x_t^k)$
    $x_{t+1}^k = SAM_update(x_t^k, g_t^k)$
  end for
  Set $\tilde{x}_{k+1} = x_q^k$
end for
return Iterate x chosen uniformly random from $\{\{x_t^k\}_{t=0}^{q-1}\}_{k=0}^{N-1}$

To check Condition (15), note that

$$\lambda_k = \lambda_{max}\left(\begin{pmatrix} Y_k & R_k \end{pmatrix} \begin{pmatrix} 0 & Z_k^\dagger \\ Z_k^\dagger & 0 \end{pmatrix} \begin{pmatrix} Y_k^{\mathrm{T}} \\ R_k^{\mathrm{T}} \end{pmatrix} \right) = \lambda_{max}\left(\begin{pmatrix} Y_k^{\mathrm{T}} \\ R_k^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} Y_k & R_k \end{pmatrix} \begin{pmatrix} 0 & Z_k^\dagger \\ Z_k^\dagger & 0 \end{pmatrix} \right). \quad (16)$$

Since $\begin{pmatrix} Y_k^{\mathrm{T}} \\ R_k^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} Y_k & R_k \end{pmatrix}, \begin{pmatrix} 0 & Z_k^\dagger \\ Z_k^\dagger & 0 \end{pmatrix} \in \mathbb{R}^{2m\times 2m}$, and $m \ll d$, λ_k can be computed efficiently, say, using an eigenvalue decomposition algorithm with the time complexity of $O(m^3)$. This cost is negligible compared with those to form $X_k^{\mathrm{T}} X_k, R_k^{\mathrm{T}} R_k$, which need $O(m^2 d)$ flops. After that, to guarantee the positive definiteness, we check if α_k satisfies (15) and use a smaller α_k if necessary.

2.3 Enhancement of Stochastic Anderson Mixing

To further enhance SAM, we introduce two techniques, namely *variance reduction* and *preconditioned mixing*.

Variance reduction. Variance reduction techniques are proved to be effective if a scan over the full dataset is feasible [1, 44]. Similar to SdLBFGS-VR proposed in [60], we also incorporate SVRG to SAM, which we call SAM-VR (Algorithm 2), for solving (2). To simplify the description, we denote one iteration of SAM in Algorithm 1 as $SAM_update(x_k, g_k)$, i.e. one update of x_k given the gradient estimate g_k.

Preconditioned mixing. Motivated by the great success of preconditioning in solving linear systems and eigenvalue computation [18], we present a preconditioned version of SAM. The key modification is the mixing step (4b). We replace the simple mixing $x_{k+1} = \bar{x}_k + \beta_k \bar{r}_k$ with $x_{k+1} = \bar{x}_k + M_k^{-1}\bar{r}_k$ where M_k approximates the Hessian. Combining it with (8) and (12), we obtain

$$x_{k+1} = x_k + \left(M_k^{-1} - \alpha_k \left(X_k + M_k^{-1} R_k \right) \left(R_k^{\mathrm{T}} R_k + \delta_k X_k^{\mathrm{T}} X_k \right)^\dagger R_k^{\mathrm{T}} \right) r_k. \quad (17)$$

Setting $\alpha_k \equiv 1$ and $\delta_k \equiv 0$, (17) reduces to a preconditioned AM update, which can be recast as the solution to the constrained optimization problem: $\min_{H_k} \|H_k - M_k^{-1}\|_F$ subject to $H_k R_k = -X_k$, a direct extension of (7). This preconditioned version of AM is related to quasi-Newton updates [20]. We also point out that the action of M_k^{-1} can be implicitly done via an update of any optimizer at hand, i.e. $x_{k+1} = \bar{x}_k + optim(x_k, -\bar{r}_k)$, where $optim$ updates x_k given the *extragradient* $-\bar{r}_k$. If $R_k = -\nabla^2 f(x_k) X_k$, which is the case in deterministic quadratic optimization, the projection step in preconditioned AM is still a minimal residual procedure.

Remark 2. *Similar to SdLBFGS [60], SAM needs another $2md$ space to store X_k and R_k. The extra main computational cost for SAM compared with SGD is $O(m^2 d) + O(m^3)$, which accounts for the matrix multiplications ($\mathbb{R}^{m\times d} \times \mathbb{R}^{d\times m}$) and matrix decomposition of a small $\mathbb{R}^{m\times m}$ matrix. Since dense matrix multiplication can be ideally parallelized and the cost of gradient evaluations often dominates the computing, the benefit from SAM pays for this extra cost. Besides, we incorporate sanity check of the positive definiteness, alternating iteration and moving average in our implementation and the details are given in the supplementary materials.*

3 Theory

In this section, we give the main results about the convergence and complexity of SAM. All the proofs are left to the Appendix. It is worth noting that since the approximate Hessian H_k of SAM may depends on the data samples $\{\xi_i\}_{i\in S_k}$ of current mini-batch, which violates the assumption AS.4 in [60], the framework of [60] does not apply.

We first give assumptions about the objective function f.

Assumption 1. $f:\mathbb{R}^d\mapsto\mathbb{R}$ *is* continuously differentiable. $f(x)\geq f^{low}>-\infty$ *for any* $x\in\mathbb{R}^d$. ∇f *is globally L-Lipschitz continuous; namely* $\|\nabla f(x)-\nabla f(y)\|_2\leq L\|x-y\|_2$ *for any* $x,y\in\mathbb{R}^d$.

Assumption 2. *For any iteration* k*, the stochastic gradient* $\nabla f_{\xi_k}(x_k)$ *satisfies* $\mathbb{E}_{\xi_k}[\nabla f_{\xi_k}(x_k)]=\nabla f(x_k)$, $\mathbb{E}_{\xi_k}[\|\nabla f_{\xi_k}(x_k)-\nabla f(x_k)\|_2^2]\leq\sigma^2$*, where* $\sigma>0$*, and* $\xi_k, k=0,1,\ldots$ *are independent samples that are independent of* $\{x_j\}_{j=0}^k$.

We also state the diminishing condition about β_k as

$$\sum_{k=0}^{+\infty}\beta_k=+\infty,\sum_{k=0}^{+\infty}\beta_k^2<+\infty. \tag{18}$$

3.1 Convergence and complexity

Theorem 1. *Suppose that Assumptions 1 and 2 hold for* $\{x_k\}$ *generated by SAM with batchsize* $n_k=n$ *for all* k*.* $C>0$ *is a constant. If* β_k *satisfies (18) and* $0<\beta_k\leq\frac{\mu}{4L(1+C^{-1})}$, $\delta_k\geq C\beta_k^{-2}$, $0\leq\alpha_k\leq\min\{1,\beta_k^{\frac{1}{2}}\}$ *and satisfies (15) , then*

$$\liminf_{k\to\infty}\|\nabla f(x_k)\|_2=0 \text{ with probability } 1. \tag{19}$$

Moreover, there exists a positive constant M_f *such that*

$$\mathbb{E}[f(x_k)]\leq M_f\quad\forall k. \tag{20}$$

Theorem 2. *Assume the same assumptions hold as in Theorem 1. If we require that the noisy gradient is bounded, i.e.,*

$$\mathbb{E}_{\xi_k}[\|\nabla f_{\xi_k}(x_k)\|_2^2]\leq M_g, \tag{21}$$

where $M_g>0$ *is a constant, we can obtain a stronger convergence result:*

$$\lim_{k\to\infty}\|\nabla f(x_k)\|=0 \text{ with probability } 1. \tag{22}$$

Now, we give the iteration complexity of SAM.

Theorem 3. *Suppose that Assumptions 1 and 2 hold for* $\{x_k\}$ *generated by SAM with batchsize* $n_k=n$ *for all* k*.* $C>0$ *is a constant.* β_k *is specially chosen as* $\beta_k=\frac{\mu}{4L(1+C^{-1})}(k+1)^{-r}$ *with* $r\in(0.5,1)$*.* $\delta_k\geq C\beta_k^{-2}$, $0\leq\alpha_k\leq\min\{1,\beta_k^{\frac{1}{2}}\}$ *and satisfies (15). Then*

$$\frac{1}{N}\sum_{k=0}^{N-1}\mathbb{E}\|\nabla f(x_k)\|_2^2\leq\frac{16L(1+C^{-1})(M_f-f^{low})}{\mu^2}N^{r-1}+\frac{(1+L^{-1}\mu^{-1})\sigma^2}{(1-r)n}(N^{-r}-N^{-1}), \tag{23}$$

where N *denotes the iteration number. Moreover, for a given* $\epsilon\in(0,1)$*, to guarantee that* $\frac{1}{N}\sum_{k=0}^{N-1}\mathbb{E}\|\nabla f(x_k)\|_2^2<\epsilon$*, the number of iterations* N *needed is at most* $O(\epsilon^{-\frac{1}{1-r}})$.

We analyze the $\mathcal{SFO}$-calls complexity of SAM when the output x_R is randomly selected from previous iterates according to some specially defined probability mass function P_R. We show below that under similar conditions, SAM has the same complexity $O(\epsilon^{-2})$ as RSG [17] and SQN [60].

Theorem 4. *Suppose that Assumptions 1 and 2 hold. Batch size* $n_k=n$ *for* $k=0,\ldots,N-1$*.* $C>0$ *is a constant.* $\beta_k=\frac{\mu}{4L(1+C^{-1})}$*.* $\delta_k\geq C\beta_k^{-2}$, $0\leq\alpha_k\leq\min\{1,\beta_k^{\frac{1}{2}}\}$ *and satisfies (15). Let* R *be a random variable following* $P_R(k)\overset{\text{def}}{=}Prob\{R=k\}=1/N$*, and* $\bar{N}$ *be the total number of* $\mathcal{SFO}$*-calls needed to calculate stochastic gradients* $\nabla f_{S_k}(x_k)$ *in SAM.*

For a given accuracy $\epsilon > 0$, we assume that $\bar{N} \geq \left\{ \frac{C_1^2}{\epsilon^2} + \frac{4C_2}{\epsilon}, \frac{\sigma^2}{L^2 \tilde{D}} \right\}$, where $C_1 = \frac{32 D_f (1+C^{-1})\sigma}{\mu^2 \sqrt{\tilde{D}}} + (L + \mu^{-1})\sigma\sqrt{\tilde{D}}, C_2 = \frac{32 D_f L(1+C^{-1})}{\mu^2}$, where $D_f \overset{\text{def}}{=} f(x_0) - f^{low}$ and $\tilde{D}$ is a problem-independent positive constant. Moreover, we assume that the batch size satisfies $n_k = n := \left\lceil \min\left\{ \bar{N}, \max\left\{ 1, \frac{\sigma}{L}\sqrt{\frac{\bar{N}}{\tilde{D}}} \right\} \right\} \right\rceil$. Then we obtain $\mathbb{E}\left[\|\nabla f(x_R)\|_2^2\right] \leq \epsilon$, where the expectation is taken with respect to R and $\{S_j\}_{j=0}^{N-1}$. In other words, to ensure $\mathbb{E}\left[\|\nabla f(x_R)\|_2^2\right] \leq \epsilon$, the number of $\mathcal{SFO}-$calls is $O(\epsilon^{-2})$.

We analyze the $\mathcal{SFO}$-calls complexity of Algorithm 2.

Theorem 5. *For Algorithm 2, suppose that Assumptions 1 and 2 hold. x_* denotes the minima. Batch size $n_k = n$ for all k. Let $\nu, \mu_0 \in (0,1)$ be two positive constants satisfying $\frac{\mu_0 \mu}{2(1+C^{-1})^{1/2}} - \frac{\mu_0^2(2L+\mu^{-1})(e-1)}{L} - 2\mu_0^2 n - \frac{2\mu_0^3(2L+\mu^{-1})(e-1)n}{L} \geq \nu$, where e is the Euler's number. $C > 0$ is a constant. Set $\beta_t^k = \beta := \frac{\mu_0 n}{L(1+C^{-1})^{1/2} T^{2/3}}$, $\delta_t^k \geq C\beta^{-2}$, $0 \leq \alpha_t^k \leq \min\{1, \beta^{\frac{1}{2}}\}$ and satisfies (15). $q = \left\lfloor \frac{T}{\mu_0 n d_0} \right\rfloor$, where $d_0 = 6 + \frac{2}{L(1+C^{-1})^{1/2}}$. Then $\mathbb{E}\left[\|\nabla f(x)\|_2^2\right] \leq \frac{T^{2/3} L(f(x_0) - f(x_*))}{qNn\nu}$. To ensure $\mathbb{E}\left[\|\nabla f(x)\|_2^2\right] \leq \epsilon$, the total number of $\mathcal{SFO}$-calls is $O(T^{2/3}/\epsilon)$.*

3.2 Convergence of AdaSAM

Now we state the convergence of AdaSAM. From the definition of δ_k (13), $\delta_k \geq c_2 \beta_k^{-2}$, thus fulfilling the condition in previous theorems. We discuss the rationality of the first term, which is somewhat heuristic. Since $r_k^T H_k r_k \geq \beta_k \mu \|r_k\|_2^2$, and $\|H_k r_k\|_2^2 \leq 2\left(\beta_k^2 + \alpha_k^2 \delta_k^{-1}\right)\|r_k\|_2^2 \leq 2\beta_k^2\left(1 + C^{-1}\right)\|r_k\|_2^2$ if $\alpha_k \leq 1, \delta_k \geq C\beta_k^{-2}$ (see the supplement for this inequality), it is sensible to suppose $\|x_k - x_{k-1}\|_2 \approx \|x_{k+1} - x_k\|_2 = \|H_k r_k\|_2 = \beta_k h_k \|r_k\|_2$, where $0 < h_k \leq h < +\infty$ is a number related to H_k. Therefore,

$$\frac{c_1 \|r_k\|_2^2}{\|x_k - x_{k-1}\|_2^2} \approx \frac{c_1 \|r_k\|_2^2}{\beta_k^2 h_k^2 \|r_k\|_2^2} \geq c_1 h^{-2} \beta_k^{-2}, \tag{24}$$

which coincides with the requirement that $\delta_k \geq C\beta_k^{-2}$. This observation together with $\delta_k \geq c_2 \beta_k^{-2}$ ensures AdaSAM converges in nonconvex stochastic optimization under some proper assumptions.

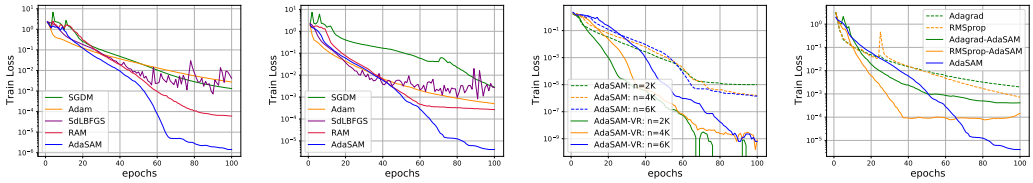
4 Experiments

Our implementation is based on Algorithm 1, with additional details mentioned in Remark 2. The pseudocode is given in the Appendix. To evaluate the effectiveness of our method AdaSAM and its preconditioned variant pAdaSAM, we compared them with several first-order and second-order optimizers for mini-batch training of neural networks, which can be highly nonlinear and stochastic, on different machine learning tasks. The datasets are MNIST [32], CIFAR-10/CIFAR-100 [31] for image classification and Penn TreeBank [36] for language model. For AdaSAM, the individual hyperparameter needs to be tuned is c_1 in (13), others are set as default in Algorithm 1. More Experimental details and hyper-parameter tuning are referred to the Appendix.

Experiments on MNIST. We trained a simple convolutional neural network (CNN) [1] on MNIST, for which we are only concerned about the minimization of the empirical risk (2), i.e. the training loss, with large batch sizes. The training dataset was preprocessed by randomly selecting 12k images from the total 60k images for training. Neither weight-decay nor dropout was used. We compared AdaSAM with SGDM, Adam [29], SdLBFGS [60], and RAM (cf. (10)). The learning rate was tuned and fixed for each optimizer. The historical length for SdLBFGS, RAM and AdaSAM was set as 20. $\delta = 10^{-6}$ for RAM and $c_1 = 10^{-4}$ for AdaSAM.

Figure 1 (a) and (b) show the curves of training loss when training 100 epochs with batch sizes of 6K and 3K, which indicate that AdaSAM can significantly minimize the empirical risk in large mini-batch training. The comparison with RAM verifies the benefit of adaptive regularization. We also notice that there hardly exists any oscillation in AdaSAM during training except for the first few epochs,

[1] Based on the official PyTorch implementation https://github.com/pytorch/examples/blob/master/mnist.



(a) Batchsize=6K (b) Batchsize=3K (c) Variance reduction (d) Preconditioning

Figure 1: Experiments on MNIST. (a) Train Loss using batchsize = 6K; (b) Train Loss using batchsize = 3K; (c) AdaSAM with variance reduction; (d) Preconditioned AdaSAM with batchsize = 3K.

which demonstrates AdaSAM's tolerance to noise. We also tested the effectiveness of variance reduction and preconditioning introduced in Section 2.3. The variance reduced extension of AdaSAM is denoted as AdaSAM-VR and was compared with AdaSAM for different batch sizes. The variants of AdaSAM preconditioned by Adagrad [11] and RMSprop [56] are denoted as Adagrad-AdaSAM and RMSprop-AdaSAM respectively. Though AdaSAM-VR demands more gradient evaluations and the preconditioned variants seem to deteriorate the final training loss, we point out that AdaSAM-VR can achieve lower training loss (10^{-9}) and the preconditioned variants converge faster to an acceptable training loss (e.g. 10^{-3}).

Table 1: Training various neural networks on CIFAR10/CIFAR100. The entry (mean $\pm$ standard deviation) denotes the percentage of the final TOP1 accuracy in test dataset. The bold numbers highlight the best results. WideResNet is abbreviated as WResNet.

Method	CIFAR10						CIFAR100		
	ResNet18	ResNet20	ResNet32	ResNet44	ResNet56	WResNet	ResNet18	ResNeXt	DenseNet
SGDM	94.82±.15	92.03±.16	92.86±.15	93.10±.23	93.47±.28	94.90±.09	77.27±.09	78.41±.54	78.49±.12
Adam	93.03±.07	91.17±.13	92.03±.28	92.28±.62	92.39±.23	92.45±.11	72.41±.17	73.57±.17	70.80±.23
AdaBelief	94.65±.13	91.15±.21	92.15±.17	92.79±.24	93.30±.07	94.46±.13	76.25±.06	78.27±.16	78.83±.15
Lookahead	94.92±.33	92.07±.04	92.86±.15	93.26±.24	93.36±.13	94.90±.15	77.63±.35	78.93±.12	79.37±.16
RNA	93.45±.21	90.73±.12	91.08±.51	91.61±.37	91.23±.14	93.85±.24	75.12±.39	75.88±.40	75.70±.49
RAM	95.10±.05	92.21±.09	93.05±.43	93.42±.13	93.76±.16	95.04±.09	76.19±.12	78.65±.20	78.28±.62
AdaSAM	**95.17±.10**	**92.43±.19**	**93.22±.32**	**93.57±.14**	**93.77±.12**	**95.23±.07**	**78.13±.14**	79.31±.27	**80.09±.52**
AdaSAM-SGD	95.04±.22	92.26±.10	92.92±.28	93.01±.15	93.71±.15	94.99±.19	77.81±.12	**79.47±.44**	79.58±.39
AdaSAM-Adam	93.86±.23	92.27±.29	92.67±.09	92.94±.30	93.22±.12	93.88±.20	74.46±.51	75.34±.20	75.21±.49

Experiments on CIFAR. For CIFAR-10 and CIFAR-100, both datasets have 50K images for training and 10K images for test. The test accuracy at the final epoch was reported as the evaluation metric. We trained ResNet18/20/32/44/56 [21] and WideResNet16-4 [62] on CIFAR-10, and ResNet18, ResNeXt50 [61] and DenseNet121 [24] on CIFAR-100. The baseline optimizers were SGDM, Adam, AdaBelief [65], Lookahead [64] and RNA [52]. The hyperparameters were kept unchanged across different tests. We trained 160 epochs with batch size of 128 and decayed the learning rate at the 80th and 120th epoch. For AdaSAM/RAM, α_k and β_k were decayed at the 80th and 120th epoch.

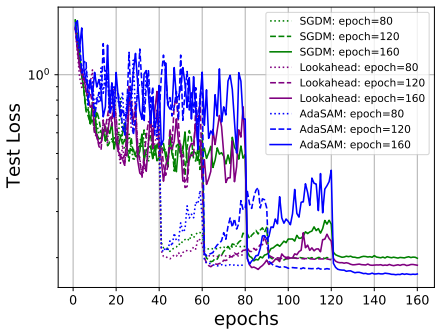
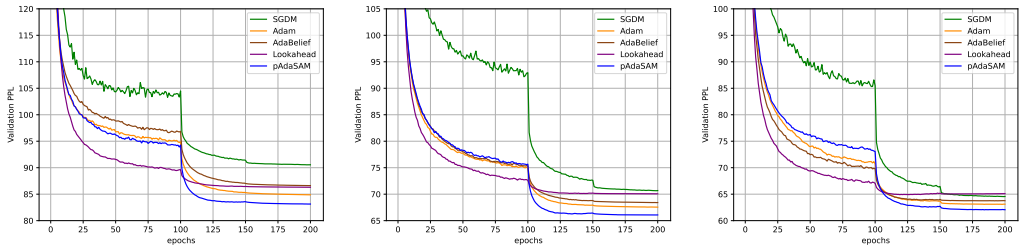


Figure 2: Test loss of training ResNet18 on CIFAR-10.

Table 1 demonstrates the generalization ability of AdaSAM. Compared with SGDM/Lookahead, AdaSAM is built on a noisy quadratic model to extrapolate historical iterates more elaborately, which may explore more information from history. We also conducted tests on training for 120 epochs and 80 epochs. Figure 2 shows AdaSAM can achieve comparable or even lower test loss than SGDM/Lookahead when training with fewer epochs, thus saving large number of iterations. We point out that the slow convergence before the first learning rate decay is attributed to the fact that we use a much larger weight-decay for AdaSAM (1.5×10^{-3} vs. 5×10^{-4} for SGDM) and large learning rate ($\alpha_0 = \beta_0 = 1$) which may slow down training but help generalize [34].

We also explore the scheme of alternating iterations: given an optimizer $optim$, in each cycle, we iterate with $optim$ for $(p-1)$ steps and then apply AdaSAM in the p-th step, the result of which is the starting point of the next cycle. We tested vanilla SGD (momentum = 0) alternated with AdaSAM, and Adam alternated with AdaSAM, which are denoted as AdaSAM-SGD and AdaSAM-Adam respectively. The number of steps of one cycle is 5. Results listed at the bottom of Table 1 show AdaSAM-Adam gives a thorough improvement over Adam. AdaSAM-SGD can even beat AdaSAM on CIFAR-100/ResNeXt50, and exceed the test accuracy of SGDM by 1.06%. Hence alternating iteration can reduce computational overhead while achieving comparable test accuracy.



(a) 1-Layer LSTM (b) 2-Layer LSTM (c) 3-Layer LSTM

Figure 3: Experiments on Penn TreeBank. Validation loss of training 1,2,3-Layer LSTM.

Experiments on Penn TreeBank. We trained LSTM on Penn TreeBank and reported the perplexity on the validation set in Figure 3 and testing set in Table 2, where pAdaSAM denotes the variant of AdaSAM preconditioned by Adam. The experimental setting is the same as that in AdaBelief [65]. In our practice, we find that the vanilla AdaSAM with default hyperparameter setting is not suitable for this task. Nevertheless, a suitable preconditioner (e.g. Adam) can largely improve the behaviour of AdaSAM. Conversely, AdaSAM can also enhance a optimizer when the latter is used as a preconditioner.

Table 2: Test perplexity on Penn TreeBank for 1,2,3-layer LSTM. Lower is better.

Method	1-Layer	2-Layer	3-Layer
SGDM	85.21±.36	67.12±.14	61.56±.14
Adam	80.88±.15	64.54±.18	60.34±.22
AdaBelief	82.41±.46	65.07±.02	60.64±.14
Lookahead	82.01±.07	66.43±.33	61.80±.10
pAdaSAM	**79.34±.09**	**63.18±.22**	**59.47±.08**

5 Related work

AM is well-known in scientific computing [57, 6, 55]. Recently, AM also receives attention in machine learning, such as for accelerating EM algorithm [22], proximal gradient methods [35] and reinforcement learning [16], but there is no general theoretical convergence analysis. AM utilizes historical iterations through the projection and mixing step. Besides, there are also other research works which exploit historical information with tools from machine learning, such as using guided policy search [33] and RNN [3]. However, these methods are much more complicated to implement and apply in practice and the mechanisms underlying these methods are difficult to interpret.

6 Conclusion

In this paper, we develop an extension of Anderson mixing, namely Stochastic Anderson Mixing, for nonconvex stochastic optimization. By introducing damped projection and adaptive regularization, we establish the convergence theory of our new method. We also analyze its work complexity in terms of $\mathcal{SFO}$-calls and show it can achieve the $O(1/\epsilon^2)$ complexity for an $\epsilon-$accurate solution. We also give a specific form of adaptive regularization. Then we propose two techniques to further enhance our method. One is the variance reduction technique, which can further improve the work complexity of our method theoretically and help achieve lower empirical risk in our experiments. The other one is the preconditioned mixing strategy that directly extends Anderson mixing. Experiments show encouraging results of our method and its enhanced version in terms of convergence rate or generalization ability in training different neural networks in different machine learning tasks. These results confirm the suitability of Anderson mixing for nonconvex stochastic optimization.

References

[1] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

[2] Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3988–3996, 2016.

[4] Wei Bian, Xiaojun Chen, and CT Kelley. Anderson acceleration for a class of nonsmooth fixed-point problems. *SIAM Journal on Scientific Computing*, (0):S1–S20, 2021.

[5] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL `https://doi.org/10.1137/16M1080173`.

[6] Claude Brezinski, Michela Redivo-Zaglia, and Yousef Saad. Shanks sequence transformations and anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.

[7] Claude Brezinski, Stefano Cipolla, Michela Redivo-Zaglia, and Yousef Saad. Shanks and anderson-type acceleration techniques for systems of nonlinear equations. *arXiv preprint arXiv:2007.05716*, 2020.

[8] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[9] Eric Cancès and Claude Le Bris. Can we outperform the diis approach for electronic structure calculations? *International Journal of Quantum Chemistry*, 79(2):82–90, 2000.

[10] Kai Lai Chung. On a stochastic approximation method. *The Annals of Mathematical Statistics*, pages 463–483, 1954.

[11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[12] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[13] Haw-ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

[14] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[15] Alejandro J Garza and Gustavo E Scuseria. Comparison of self-consistent field convergence acceleration techniques. *The Journal of chemical physics*, 137(5):054110, 2012.

[16] Matthieu Geist and Bruno Scherrer. Anderson acceleration for reinforcement learning. In *EWRL 2018-4th European workshop on Reinforcement Learning*, 2018.

[17] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[18] Gene H Golub and Charles F Van Loan. Matrix computations, 4th. *Johns Hopkins*, 2013.

[19] Robert Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik. Rsn: Randomized subspace newton. In *Conference on Neural Information Processing Systems*, 2019.

[20] Robert M Gower and Peter Richtárik. Randomized quasi-newton updates are linearly convergent matrix inversion algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1380–1409, 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Nicholas C Henderson and Ravi Varadhan. Damped anderson acceleration with restarts and monotonicity control for accelerating em and em-like algorithms. *Journal of Computational and Graphical Statistics*, 28(4):834–846, 2019.

[23] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[25] Shaojun Huang, Yuanzhang Sun, and Qiuwei Wu. Stochastic economic dispatch with wind using versatile probability distribution and l-bfgs-b based dual decomposition. *IEEE Transactions on Power Systems*, 33(6):6254–6263, 2018.

[26] Xunpeng Huang, Xianfeng Liang, Zhengyang Liu, Lei Li, Yue Yu, and Yitan Li. Span: A stochastic projected approximate newton method. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1520–1527, 2020.

[27] Brian Irwin and Eldad Haber. Secant penalized bfgs: A noise robust quasi-newton method via penalizing the secant condition. *arXiv preprint arXiv:2010.01275*, 2020.

[28] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Galina M Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[34] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *arXiv preprint arXiv:1907.04595*, 2019.

[35] Vien Mai and Mikael Johansson. Anderson acceleration of proximal gradient methods. In *International Conference on Machine Learning*, pages 6620–6629. PMLR, 2020.

[36] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[37] James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[38] Aryan Mokhtari and Alejandro Ribeiro. Stochastic quasi-newton methods. *Proceedings of the IEEE*, 108(11):1906–1922, 2020.

[39] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4): 1574–1609, 2009.

[40] Boris T Polyak. New stochastic approximation type procedures. *Automat. i Telemekh*, 7(98-107): 2, 1990.

[41] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[42] Florian A Potra and Hans Engler. A characterization of the behavior of the anderson acceleration on linear problems. *linear Algebra and its Applications*, 438(3):1002–1011, 2013.

[43] Phanisri P Pratapa, Phanish Suryanarayana, and John E Pask. Anderson acceleration of the jacobi iterative method: An efficient alternative to krylov methods for large, sparse linear systems. *Journal of Computational Physics*, 306:43–54, 2016.

[44] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[45] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[46] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7 (3):856–869, 1986.

[47] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[48] Jerome Sacks. Asymptotic distribution of stochastic approximation procedures. *The Annals of Mathematical Statistics*, 29(2):373–405, 1958.

[49] Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial intelligence and statistics*, pages 436–443, 2007.

[50] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 712–720, 2016.

[51] Damien Scieur, Francis Bach, and Alexandre d'Aspremont. Nonlinear acceleration of stochastic algorithms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3985–3994, 2017.

[52] Damien Scieur, Edouard Oyallon, Alexandre d'Aspremont, and Francis Bach. Nonlinear acceleration of cnns. In *ICLR Workshop track*, 2018.

[53] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. *Mathematical Programming*, 179(1):47–83, 2020.

[54] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[55] Phanish Suryanarayana, Phanisri P Pratapa, and John E Pask. Alternating anderson–richardson method: An efficient alternative to preconditioned krylov methods for large, sparse linear systems. *Computer Physics Communications*, 234:278–285, 2019.

[56] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31, 2012.

[57] Alex Toth and CT Kelley. Convergence analysis for anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015.

[58] Alex Toth, J Austin Ellis, Tom Evans, Steven Hamilton, CT Kelley, Roger Pawlowski, and Stuart Slattery. Local improvement results for anderson acceleration with inaccurate function evaluations. *SIAM Journal on Scientific Computing*, 39(5):S47–S65, 2017.

[59] Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[60] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[62] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[63] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[64] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9597–9608, 2019.

[65] Juntang Zhuang, Tommy Tang, Sekhar Tatikonda, Nicha Dvornek, Yifan Ding, Xenophon Papademetris, and James S Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *arXiv preprint arXiv:2010.07468*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We reported standard deviation instead. See Table 1,2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The assets are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]