nature machine intelligence

Article

https://doi.org/10.1038/s42256-026-01182-3

Preconditioned inexact stochastic ADMM for deep models

Received: 14 February 2025

Accepted: 3 January 2026

Published online: 20 February 2026

Check for updates

Shenglong Zhou[1], Ouya Wang[2], Ziyan Luo[1], Yongxu Zhu[3] & Geoffrey Ye Li[2]

Deep learning models are usually trained with stochastic gradient descent-based algorithms, but these optimizers face inherent limitations, such as slow convergence and stringent assumptions for convergence. In particular, data heterogeneity arising from distributed settings poses significant challenges to their theoretical and numerical performance. Here we develop an algorithm called PISA (preconditioned inexact stochastic alternating direction method of multipliers). Grounded in rigorous theoretical guarantees, the algorithm converges under the sole assumption of Lipschitz continuity of the gradient on a bounded region, thereby removing the need for other conditions commonly imposed by stochastic methods. This capability enables the proposed algorithm to tackle the challenge of data heterogeneity effectively. Moreover, the algorithmic architecture enables scalable parallel computing and supports various preconditions, such as second-order information, second moment and orthogonalized momentum by Newton–Schulz iterations. Incorporating the last two preconditions in PISA yields two computationally efficient variants: SISA and NSISA. Comprehensive experimental evaluations for training or fine-tuning diverse deep models, including vision models, large language models, reinforcement learning models, generative adversarial networks and recurrent neural networks, demonstrate superior numerical performance of SISA and NSISA compared with various state-of-the-art optimizers.

Deep models have led to extensive applications across a variety of industries. For instance, fine-tuning large language models (LLMs), such as Gemma[1], GPT[2] and Llama[3], with user-specific data can significantly enhance the model performance. Diffusion models are increasingly used to generate personalized content, including images, audio and video, for entertainment. Vision models, such as ResNet[4] and vision transformers[5], are in high demand for tasks, such as image classification, segmentation and object detection on large-scale image datasets. However, the growing complexity and scale of these tasks present considerable challenges for widely used stochastic gradient descent (SGD)-based optimizers due to their inherent limitations. In this section, we review a subset of these methods, sufficient to motivate the development of our proposed approach.

SGD-based learning algorithms

It is known that the plain SGD-based algorithms, such as the original stochastic approximation approaches[6,7], the parallelized SGD[8] for machine learning and the newly developed federated averaging (FedAvg[9,10]) and local SGD (LocalSGD[11]) algorithms for federated learning, are frequently prone to high sensitivity to poor conditioning[12] and slow convergence

[1]School of Mathematics and Statistics, Beijing Jiaotong University, Beijing, China. [2]ITP Laboratory, Department of EEE, Imperial College London, London, UK. [3]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China. e-mail: shlzhou@bjtu.edu.cn; ouya.wang20@imperial.ac.uk

Table 1 | Assumptions imposed by different stochastic algorithms for convergence

Algorithms	Refs.	Class	Data	Convergence rate B	Assumptions
Type I convergence: $F(\mathbf{w}^T)-F^*=B$					
AdaGrad	15	SGD	IID	$O(1/\sqrt{T})$	1, 4, 6
Adam	17	SGD	IID	$O(1/\sqrt{T})$	1, 4, 6
RMSProp	92	SGD	IID	$O(1/\sqrt{T})$	1, 4, 6,
AMSGrad	18	SGD	IID	$O(1/\sqrt{T})$	1, 4, 8
AdaBound	20	SGD	IID	$O(1/\sqrt{T})$	1, 4, 8
FedAvg	10	SGD	IID/Non-IID	$O(1/T)$	2–5
LocalSGD	11	SGD	IID	$O(1/T)$	2–5
SADMM	53	SADMM	IID	$O(\log(T)/T)$	2, 4, 8
ASVRG-ADMM	57	SADMM	IID	$O(1/T^2)$	1, 3, 8, 11
ASVRG-ADMM	57	SADMM	IID	$O(\gamma^T)$	2, 3, 8, 11
PS-ADMM	54	SADMM	IID	$O(1/T)$	2, 3, 9
PISA	Ours	SADMM	IID/Non-IID	$O(\gamma^T)$	12
Type II convergence: $\|\nabla F(\mathbf{w}^T)\|^2=B$					
Adam	93	SGD	IID	$O(\log(T)/\sqrt{T})$	3, 4, 7
Padam	19	SGD	IID	$O(1/\sqrt{T})$	3, 4, 7
PRSGD	21	SGD	IID	$O(1/\sqrt{T})$	3, 4, 5, 7
Lamb	22	SGD	IID	$O(1/\sqrt{T})$	3, 4, 5
Adan	36	SGD	IID	$O(1/\sqrt{T})$	3, 4, 5, 7
FedAvg	94	SGD	IID/Non-IID	$O(1/\sqrt{T})$	3, 4, 5, 7
SPIDER-ADMM	56	SADMM	IID	$O(1/T)$	3, 4

Assumptions: (1) convexity; (2) strong convexity; (3) Lipschitz continuity of the gradient; (4) bounded (stochastic) gradient or the second moment of the (stochastic) gradient; (5) bounded variance; (6) bounded sequence generated by the algorithm; (7) unbiased gradient estimation; (8) compact convex feasible region; (9) Lipschitz continuity of the objective function; (10) Lipschitz subminimization paths; (11) bounded dual variables; (12) Lipschitz continuity of the gradient on a bounded region.

in high-dimensional and non-convex landscapes[13]. To overcome these drawbacks, SGD with momentum and adaptive learning rates has been proposed to enhance robustness and accelerate convergence. The former introduced first-order momentum to suppress the oscillation of SGD[14], and the latter updated the learning rate iteratively based on historical information.

For instance, the adaptive gradient (AdaGrad[15]) interpolated the accumulated second-order moment of gradients to achieve the adaptive learning rate. The root mean-squared propagation (RMSProp[16]) exploited an exponential weighting technique to balance the distant historical information and the knowledge of the current second-order gradients, avoiding premature termination encountered by AdaGrad. The adaptive moment (Adam[17]) combined the first-order moment and adaptive learning rates, thereby exhibiting robustness to hyperparameters. The adaptive method setup-based gradient (AMSGrad[18]) took smaller learning rates than those of Adam by using a maximum value for normalizing the running average of the gradients, thereby fixing the convergence of Adam. Partially Adam (Padam[19]) used a similar strategy to AMSGrad with a difference in the rate power. Adam with dynamic bounds on learning rates (AdaBound[20]) designed a clipping mechanism on Adam-type learning rates by clipping the gradients larger than a threshold to avoid gradient explosion. Parallel restarted SGD (PRSGD[21]) simply benefited from a decayed power learning rate. A layerwise adaptive large batch optimization technique (Lamb[22]) performed per-dimension normalization with respect to the square root of the second moment used in Adam and set large batch sizes suggested by the layerwise adaptive rate scaling method (Lars[23]).

For some other SGD-based algorithms, one can refer to a per-dimension learning rate method based gradient descent (AdaDelta[24]), a variant of Adam based on the infinity norm (Adamax[17]), Adam with decoupled weight decay (AdamW[25]), a structure-aware preconditioning algorithm (Shampoo[26]), momentum orthogonalized by Newton–Schulz iterations (Muon[27]), Shampoo with Adam in the preconditioner's eigenbasis (SOAP[28]), Adam with fewer learning rates (Adam-mini[29]) and those reviewed in a few studies[30–32].

The aforementioned algorithms primarily leveraged the heavy ball acceleration technique to estimate the first-order and second-order moments of the gradient. An alternative approach to accelerating convergence is Nesterov acceleration, which has been theoretically proved to converge faster[33,34]. This has led to the development of several Nesterov-accelerated SGD algorithms. For example, Nesterov-accelerated Adam (NAdam[35]) integrated this technique into Adam to enhance convergence speed. More recently, an adaptive Nesterov momentum algorithm (Adan[36]) adopted Nesterov momentum estimation to refine the estimation of the first-order and second-order moments of the gradient, thereby improving adaptive learning-rate adjustments for faster convergence.

ADMM-based learning algorithms

The alternating direction method of multipliers (ADMM)[37,38] is a promising framework for solving large-scale optimization problems because it decomposes complex problems into smaller and more manageable subproblems. This characteristic makes ADMM particularly well-suited for various challenges in distributed learning. It has demonstrated substantial potential in applications, such as image compressive sensing[39],

Table 2 | Testing accuracy under data heterogeneity with skewed label distributions

Dataset	Partition	Fedavg	Fedprox	Fednova	Scaffold	SISA
MNIST	1-label	53.19 ± 0.21	54.33 ± 0.67	49.00 ± 0.42	47.04 ± 0.02	94.97 ± 0.01
	2-label	90.70 ± 0.02	90.75 ± 0.03	90.40 ± 0.05	87.43 ± 0.08	96.14 ± 0.00
	3-label	95.11 ± 0.01	94.73 ± 0.02	94.37 ± 0.01	93.19 ± 0.03	96.21 ± 0.01
CIFAR-10	1-label	10.02 ± 0.00	10.00 ± 0.00	10.31 ± 0.03	11.41 ± 0.03	30.16 ± 0.00
	2-label	13.76 ± 0.00	13.77 ± 0.00	18.02 ± 0.02	12.73 ± 0.00	30.53 ± 0.00
	3-label	20.06 ± 0.01	20.03 ± 0.01	21.20 ± 0.01	17.00 ± 0.00	30.96 ± 0.00
FMNIST	1-label	48.81 ± 0.34	47.14 ± 0.32	49.08 ± 0.34	43.93 ± 0.20	72.85 ± 0.02
	2-label	69.33 ± 0.01	69.29 ± 0.02	69.10 ± 0.04	65.99 ± 0.06	70.24 ± 0.05
	3-label	70.48 ± 0.02	69.96 ± 0.08	69.70 ± 0.08	63.64 ± 0.15	72.46 ± 0.01
Adult	1-label	76.38[a] ± 0.00	78.30[a] ± 0.03	85.15[a] ± 0.08	80.00[a] ± 0.02	82.21 ± 0.00

[a]Indicates mini-batch training in each local client. Higher values indicate better performance.

federated learning[40–42], reinforcement learning[43] and few-shot learning[44]. It is worth noting that there are many other distributed optimization frameworks, such as dual averaging[45,46], push-sum[47] and push–pull[48]. However, in the sequel, our focus is on the family of distributed algorithms, ADMM, as they are most directly relevant to the design and convergence analysis of our algorithm.

When applied to deep model training, early studies first relaxed optimization models before developing ADMM-based algorithms. Therefore, these methods can be classified as model-driven or model-specific approaches. They targeted the relaxations rather than the original problems, enabling better handling of the challenges associated with highly non-convex optimization landscapes. For instance, an ADMM-based algorithm in ref. 49 addressed a penalized formulation, avoiding pitfalls that hinder gradient-based methods in non-convex settings. Similarly, a deep learning ADMM algorithm[50] and its variant[51] were designed to enhance convergence by addressing penalization models as well. Moreover, the sigmoid-ADMM algorithm[52] used the algorithmic gradient-free property to address saturation issues with sigmoid activation functions.

Despite their advantages, model-driven ADMM approaches face two critical issues. First, these algorithms exhibit high computational complexity because they require computing full gradients using the entire dataset and performing matrix inversions for weight updates, making them impractical for large-scale tasks. Furthermore, the reliance on the specific structure of optimization models limits their general applicability, as different deep models often necessitate distinct formulations.

To address the aforementioned limitations, data-driven ADMM algorithms have emerged as a promising alternative, aiming to reduce computational burdens and enhance adaptability to diverse tasks. For instance, the deterministic ADMM[41,42], designed for distributed learning problems, achieved high accuracy but low computational efficiency due to the use of full dataset gradients. Stochastic ADMM (SADMM) methods tackled the inefficiency by replacing full gradients with stochastic approximations. Examples such as SADMM[53] and distributed stochastic ADMM (PS-ADMM[54]) aimed to solve subproblems exactly and thus still involved high computational costs.

To further enhance convergence and reduce stochastic gradient variance, advanced techniques of variance reduction and Nesterov acceleration have been incorporated into SADMM. Representatives consist of stochastic average ADMM[55], stochastic path-integrated differential estimator-based ADMM (SPIDER-ADMM[56]) and accelerated stochastic variance reduced gradient-based ADMM (ASVRG-ADMM) for convex[57] and non-convex problems[58]. However, these enhanced methods still relied on access to full gradients for the aim of reducing stochastic gradient variance, posing challenges for large-scale applications.

Contributions

In this work, we propose a data-driven preconditioned inexact SADMM algorithm, termed PISA. It distinguishes itself from previous approaches and aims to reduce computational costs, relax convergence assumptions and enhance numerical performance. The key contributions are threefold.

A general algorithmic structure

The algorithm is based on a preconditioned inexact SADMM, combining simplicity with high generality to handle a wide range of deep learning applications. First, the use of preconditioning matrices allows us to incorporate various forms of useful information, such as first-moment and second-moment (yielding second-moment-based inexact SADMM (SISA), a variant of PISA), second-order information (for example, the Hessian) and orthogonalized momentum by Newton–Schulz iterations (leading to Newton–Schulz-based inexact SADMM (NSISA), another variant of PISA), into the updates, thereby enhancing the performance of the proposed algorithms. Moreover, the proposed algorithmic framework is inherently compatible with parallel computing, making it ideal for large-scale data settings.

Strong convergence theory under a sole assumption

PISA is proven to converge under a single assumption: the Lipschitz continuity of the gradient. Despite relying on stochastic gradients, it avoids many of the assumptions typically required by stochastic algorithms. As highlighted in Table 1, all algorithms, except for PISA and FedAvg[10], have drawn identically and independently distributed (IID) samples to derive stochastic gradients for unbiased gradient estimation. However, real-world data are often heterogeneous (that is, non-IID), a phenomenon commonly referred to as statistical or data heterogeneity[59–62], which poses significant challenges to the convergence of these algorithms for IID datasets.

In addition, the table also presents the algorithmic convexity of two types of convergence. Type I convergence, $F(\mathbf{w}^T) - F^* = B$, refers to a rate B at which objective values $\{F(\mathbf{w}^T)\}$ of generated sequence $\{\mathbf{w}^T\}$ approach F^*, where T is the number of iterations and F^* is the limit of sequence $\{F(\mathbf{w}^T)\}$ or a function value at a stationary point. For instance, $B = O(1/T)$ indicates a sublinear convergence, while $B = O(\gamma^T)$ with $\gamma \in (0,1)$ implies a linear rate. Type II convergence describes how quickly the length, $\{\|\nabla F(\mathbf{w}^T)\|\}$, of gradients diminishes, reflecting the stationarity of the iterates. Therefore, ASVRG-ADMM[57] and PISA achieve the best Type I convergence rate. However, the former imposes several restrictive assumptions, while PISA requires a single assumption (that is, condition 12). We emphasize that condition 12 is closely related to the locally Lipschitz continuity of the gradient, which is a mild assumption. It is weaker than condition 3, and any twice-continuously differentiable function satisfies condition 12. More importantly, we eliminate the

Table 3 | Top 1 accuracy (%) obtained by different algorithms

Algorithms	SISA	SGD-M	AdaBd	Adam	Radam	MSVAG	AdamW	AdaBf	Yogi	LAMB	Nadam
Refs.		6	20	17	95	96	25	65	97	22	35
ResNet-34	95.04	94.65[a]	94.33[b]	94.69[a]	94.20	94.44[b]	94.28[a]	94.11[a]	94.52[a]	94.01[a]	–
VGG-11	91.25	91.51	90.62[b]	88.40[b]	89.30[b]	90.24[b]	89.39[b]	90.07	90.67[b]	87.48	–
DenseNet-121	95.77	95.47	94.58[b]	93.35[b]	94.91[b]	94.81[b]	94.55[b]	94.78	95.15	94.53	–
ResNet-18	70.03	70.23[b]	68.13[b]	63.79[b]	67.62[b]	65.99[b]	67.93[b]	70.08[b]	–	68.46[a]	68.82[a]

[a]Reported in ref. 36. [b]Reported in ref. 65, and the remaining values are derived by our implementation. Higher values indicate better performance. '–' means no results were reported. AdaBd and AdaBf stand for AdaBound and AdaBelief, respectively.

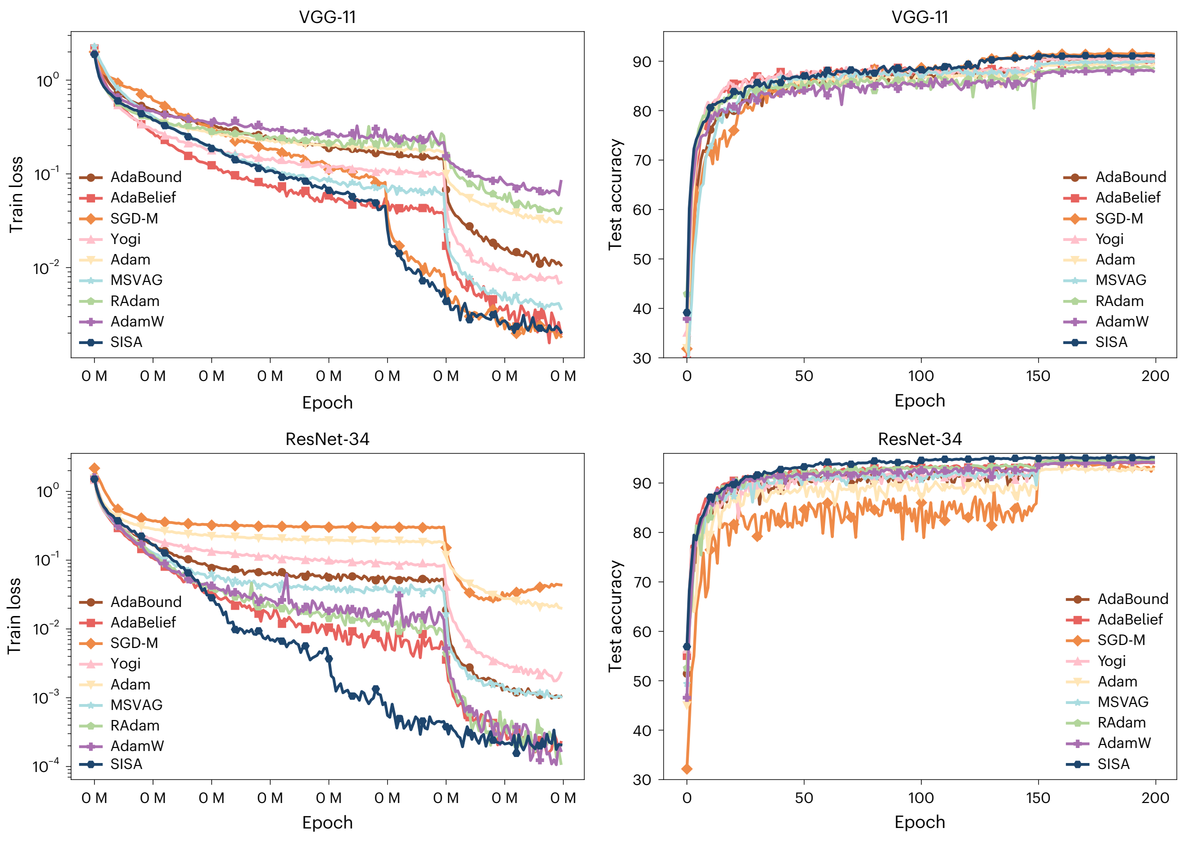

Fig. 1 | Training loss and testing accuracy over epochs for vision models on CIFAR-10. The left panels present the training and the right panels present the testing performances over epochs for nine benchmarked algorithms on two vision models: VGG-11 (top panels) and ResNet-34 (bottom panels).

need for conditions on the boundedness of the stochastic gradient, the second moment of the stochastic gradient and variance. This makes PISA well-suited for addressing the challenges associated with data heterogeneity, an open problem in federated learning[60,62].

High numerical performance for various applications

The effectiveness of PISA and its two variants, SISA and NSISA, is demonstrated through comparisons with many state-of-the-art optimizers using several deep models: vision models, LLMs, reinforcement learning models, generative adversarial networks (GANs) and recurrent neural networks, highlighting its great potential for extensive applications. In particular, numerical experiments on heterogeneous datasets corroborate our claim that PISA effectively addresses this challenge.

Results

This section evaluates the performance of SISA and NSISA in comparison with various established optimizers to process several deep models: vision models, LLMs, reinforcement learning models, recurrent neural networks and GANs. All LLM-targeted experiments are conducted on four NVIDIA H100-80GB graphical processing units, while the remaining experiments are run on a single such graphical processing unit. The source codes are available at https://github.com/Tracy-Wang7/PISA. Due to space limitations, details on hyperparameter setup, effects of key hyperparameters and more numerical evaluations, including experiments for reinforcement learning models and recurrent neural networks, are provided in Supplementary Section 1.

Data heterogeneity

To evaluate the effectiveness of the proposed algorithms in addressing the challenge of data heterogeneity, we design some experiments within a centralized federated learning framework. In this setting, m clients collaboratively learn a shared parameter under a central server, with each client i holding a local dataset $\mathcal{D}_i$. Note that our three algorithms are well-suited for centralized federated learning tasks. As shown in Algorithm 2, clients update their local parameters $(\mathbf{w}_i^{\ell+1}, \boldsymbol{\pi}_i^{\ell+1})$ in parallel, and the server aggregates them to compute global parameter $\mathbf{w}^{\ell+2}$.

In the experiment, we focus on heterogeneous datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$. The heterogeneity may arise from label distribution

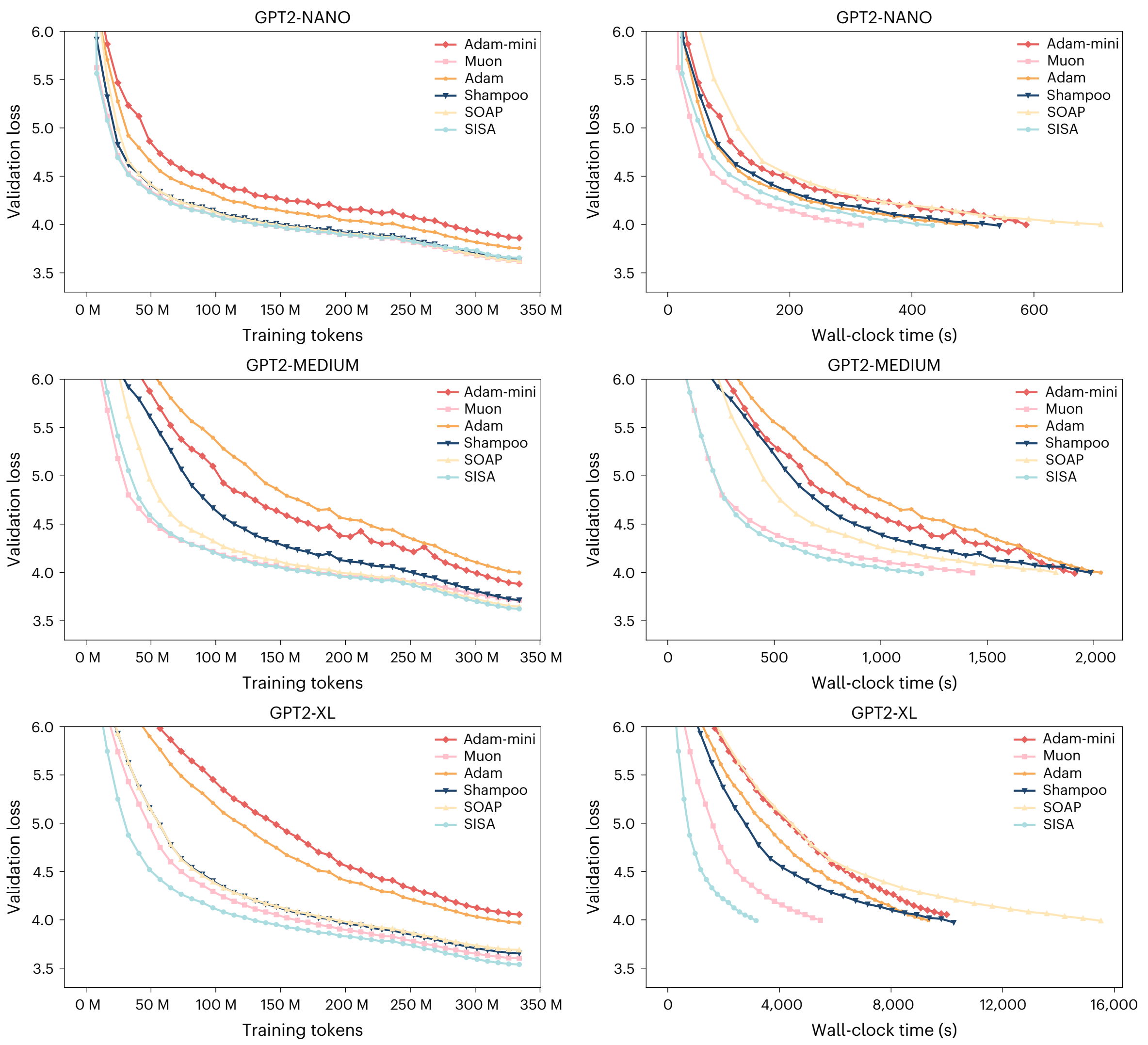

Fig. 2 | Validation loss over tokens and wall-clock time (in seconds) for three GPT2 models. The left three panels present the validation losses over training tokens, and the right three panels present the wall-clock times for six benchmarked algorithms on three GPT2 models: GPT2-NANO (top panels), GPT2-MEDIUM (middle panels) and GPT2-XL (bottom panels).

skew, feature distribution skew or quantity skew[61]. Empirical evidence has shown that the label distribution skew presents a significantly greater challenge to current FL methods compared with the other two types. Therefore, we adopt this skew as the primary setting for evaluating data heterogeneity in our experiments.

To reflect the fact that most benchmark datasets contain 10 image classes, we set $m = 10$. Four datasets, MNIST, CIFAR-10, FMNIST and Adult, are used for the experiment. As shown in Table 2, the configuration 's label' indicates that each client holds data containing s distinct label classes, where $s = 1, 2, 3$. For example, in the '2-label' setting, $\mathcal{D}_1$ may include two labels 1 and 2, while $\mathcal{D}_2$ contains two labels 2 and 3. To ensure better performance, we used a mini-batch training strategy with multiple local updates before each aggregation step for the five baseline algorithms. In contrast, SISA keeps using one local update per aggregation step and still achieves competitive testing accuracy with far fewer local updates. Notably, on the MNIST dataset under the 1-label skew setting, the best accuracy achieved by other algorithms is 54.33%, whereas SISA reaches 94.97%, demonstrating a significant improvement.

Classification by vision models

We perform classification tasks on two well-known datasets, ImageNet[63] and CIFAR-10[64], and evaluate performance based on convergence speed and testing accuracy. Convergence performance is presented in Supplementary Section 1.3. We now focus on testing accuracy, following the experimental setup and hyperparameter configurations from ref. 65. In this experiment, a small batch size (for example, $|\mathcal{D}_i| = 128$ for every $i \in [m]$) is used during training. As SISA updates its hyperparameters frequently, it uses a different learning-rate scheduler from that in ref. 65. For fair comparison, each baseline optimizer is tested with both its default scheduler and the one used by SISA, and the best result is reported. On CIFAR-10, we train three models: VGG-11[66], ResNet-34[67] and DenseNet-121[68]. As shown in Table 3, SISA achieves the highest testing accuracy on ResNet-34 and DenseNet-121, and performs slightly worse than SGD-M on VGG-11. In addition, Fig. 1 illustrates the training loss and testing accuracy over training epochs for VGG-11 and ResNet-34, demonstrating the fast convergence and high accuracy of SISA. For ImageNet, we train a ResNet-18 model and report the testing accuracy in Table 3. SISA outperforms most adaptive methods but is slightly behind SGD-M and AdaBelief.

Training LLMs

In this subsection, we apply SISA and NSISA to train several GPT2 models[69] and compare it with advanced optimizers such as AdamW, Muon, Shampoo, SOAP and Adam-mini that have been proved effective in training LLM. The comparison between SISA and AdamW is provided in Supplementary Section 1.4. We now compare NSISA with the other four optimizers, with the aim of tuning three GPT2 models: GPT2-Nano (125M), GPT2-Medium

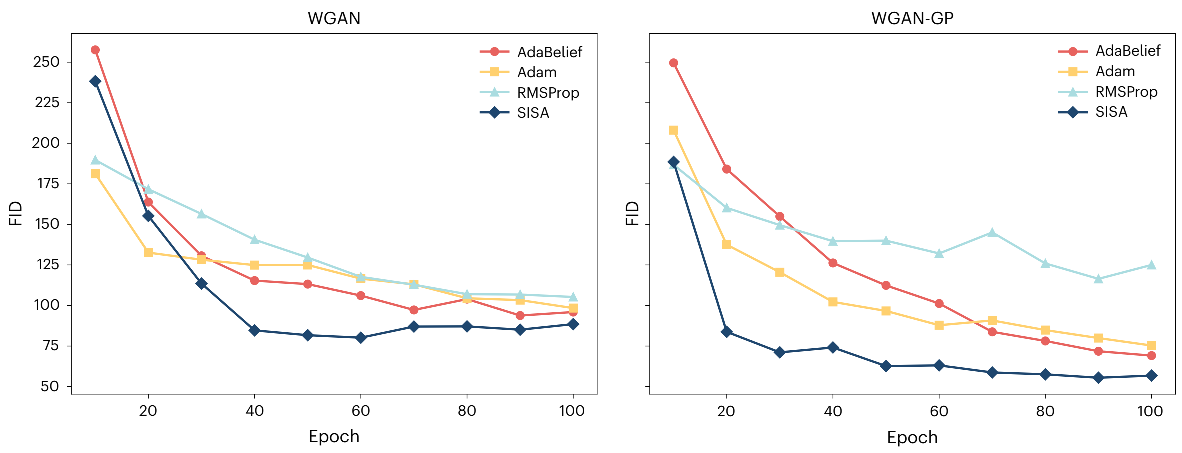

Fig. 3 | Training FID over epochs for GANs on Cifar-10. The left and right panels, respectively, present the training FID over training epochs for four benchmarked algorithms on the WGAN and WGAN_GP architectures.

(330M) and GPT2-XL (1.5B). The first model follows the setup described in ref. 27, while the last two follow the configuration from ref. 29. We use the FineWeb dataset, a large collection of high-quality web text, to fine-tune GPT2 models, enabling them to generate more coherent and contextually relevant web-style content. The hyperparameters of NSISA remain consistent across the experiments for the three GPT2 models.

The comparison of different algorithms in terms of the memory overhead is given in Supplementary Section 1.4. The experimental results are presented in Fig. 2. We evaluate each optimizer's training performance from two perspectives: the number of training tokens consumed and the wall-clock time required. To reduce wall-clock time, we implemented NSISA using parallel computation. From Fig. 2, as the number of GPT2 parameters increases from Nano to Medium to XL, the validation loss gap between NSISA and other baseline optimizers widens. In particular, for GPT2-XL, the last figure demonstrated evident advantage of NSISA in terms of the wall-clock time.

Generation tasks by GANs

GANs involve alternating updates between the generator and discriminator in a minimax game, making training notoriously unstable[70]. To assess optimizer stability, we compare SISA with adaptive methods such as Adam, RMSProp and AdaBelief, which are commonly recommended to enhance both stability and efficiency in training GANs[65,71]. We use two popular architectures, the Wasserstein GAN (WGAN[72]) and WGAN with gradient penalty (WGAN-GP[71]), with a small model and vanilla CNN generator on the CIFAR-10 dataset.

To evaluate performance, we compute the Fréchet inception distance (FID)[73] every 10 training epochs, measuring the distance between 6,400 generated images and 60,000 real images. The FID score is a widely used metric for generative models, assessing both image quality and diversity. The lower FID values indicate better generated images. We follow the experimental setup from ref. 65 and run each optimizer five times independently. The corresponding mean and standard deviation of training FID over epochs are presented in Fig. 3. Clearly, SISA achieves the fastest convergence and the lowest FID. After training, 64,000 fake images are produced to compute the testing FID, and SISA outperforms the other three optimizers, demonstrating its superior stability and effectiveness in training GANs. Detailed testing FID scores can be found in Supplementary Section 1.7.

Conclusion

This paper introduces an ADMM-based algorithm that leverages stochastic gradients and solves subproblems inexactly, significantly accelerating computation. The preconditioning framework allows it to incorporate popular second-moment schemes, enhancing training performance. Theoretical guarantees, based solely on the Lipschitz continuity of the gradients, make the algorithm suitable for heterogeneous datasets, effectively addressing an open problem in stochastic optimization for distributed learning. The algorithm demonstrates superior generalization across a wide range of architectures, datasets and tasks, outperforming well-known deep learning optimizers.

Methods

Organization and notation

This section is organized as follows: in the next subsection, we introduce the main model and develop the proposed algorithm, PISA. The section 'Convergence of PISA' provides rigorous proofs of the convergence. The section 'Second moment' specifies the precondition by the second moment to derive a variation of PISA, termed SISA.

Let $[m]:=\{1,2,\ldots,m\}$, where ':=' means 'define'. The cardinality of a set $\mathcal{D}$ is written as $|\mathcal{D}|$. For two vectors $\mathbf{w}$ and $\mathbf{v}$, their inner product is denoted by $\langle\mathbf{w},\mathbf{v}\rangle:=\sum_i w_i v_i$. Let $\|\cdot\|$ be the Euclidean norm for vectors, namely $\|\mathbf{w}\|=\sqrt{\langle\mathbf{w},\mathbf{w}\rangle}$, and the Spectral norm for matrices. A ball with a positive radius r is written as $\mathbb{N}(r):=\{\mathbf{w}:\|\mathbf{w}\|\le r\}$. A symmetric positive semi-definite matrix Q is written as $Q\succcurlyeq 0$. Then $P\succcurlyeq Q$ means that $P-Q\succcurlyeq 0$. Denote the identity matrix by I and let $\mathbf{1}$ be the vector with all entries being 1. We write

$$\Pi=(\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\ldots,\boldsymbol{\pi}_m),\quad W=(\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_m),$$
$$M=(\mathbf{m}_1,\mathbf{m}_2,\ldots,\mathbf{m}_m),\quad \boldsymbol{\sigma}=(\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2,\ldots,\boldsymbol{\sigma}_m).$$

Similar rules are also used for the definitions of Π^ℓ, W^ℓ, M^ℓ and $\boldsymbol{\sigma}^\ell$.

Preconditioned inexact SADMM

We begin this subsection by introducing the mathematical optimization model for general distributed learning. Then we go through the development of the algorithm.

Model description. Suppose we are given a set of data as $\mathcal{D}:=\{\mathbf{x}_t: t=1,2,\ldots,|\mathcal{D}|\}$, where x_t is the tth sample. Let $f(\mathbf{w};\mathbf{x}_t)$ be a function (such as neural networks) parameterized by $\mathbf{w}$ and sampled by $\mathbf{x}_t$. The total loss function on $\mathcal{D}$ is defined by $\sum_{\mathbf{x}_t\in\mathcal{D}}f(\mathbf{w};\mathbf{x}_t)/|\mathcal{D}|$. We then divide data $\mathcal{D}$ into m disjoint batches, namely, $\mathcal{D}=\mathcal{D}_1\cup\mathcal{D}_2\cup\ldots\cup\mathcal{D}_m$ and $\mathcal{D}_i\cap\mathcal{D}_{i'}=\emptyset$ for any two distinct i and i'. Denote

$$H_i(\mathbf{w};\mathcal{D}_i):=\frac{1}{|\mathcal{D}_i|}\sum_{\mathbf{x}_t\in\mathcal{D}_i}f(\mathbf{w};\mathbf{x}_t)\ \text{ and }\ \alpha_i:=\frac{|\mathcal{D}_i|}{|\mathcal{D}|}. \tag{1}$$

Clearly, $\sum_{i=1}^m\alpha_i=1$. Now, we can rewrite the total loss as follows:

$$\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}_t\in\mathcal{D}}f(\mathbf{w};\mathbf{x}_t)=\frac{1}{|\mathcal{D}|}\sum_{i=1}^m\sum_{\mathbf{x}_t\in\mathcal{D}_i}f(\mathbf{w};\mathbf{x}_t)=\sum_{i=1}^m\alpha_iH_i(\mathbf{w};\mathcal{D}_i).$$

The task is to learn an optimal parameter to minimize the following regularized loss function:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \alpha_i H_i(\mathbf{w}; \mathcal{D}_i) + \frac{\mu}{2} \parallel \mathbf{w} \|^2, \tag{2}$$

where $\mu \geq 0$ is a penalty constant and $\|\mathbf{w}\|^2$ is a regularization.

Main model. Throughout the paper, we focus on the following equivalent model of problem (2):

$$F^* := \min_{\mathbf{w}, W} \sum_{i=1}^{m} \alpha_i F_i(\mathbf{w}_i) + \frac{\lambda}{2} \parallel \mathbf{w} \|^2, \quad \text{s.t. } \mathbf{w}_i = \mathbf{w},\ i \in [m], \tag{3}$$

where $\lambda \in [0, \mu]$ and

$$\begin{aligned} F_i(\mathbf{w}) &:= F_i(\mathbf{w}; \mathcal{D}_i) := H_i(\mathbf{w}; \mathcal{D}_i) + \frac{\mu - \lambda}{2} \parallel \mathbf{w} \|^2, \\ F(\mathbf{w}) &:= \sum_{i=1}^{m} \alpha_i F_i(\mathbf{w}), \quad F_\lambda(\mathbf{w}) := F(\mathbf{w}) + \frac{\lambda}{2} \parallel \mathbf{w} \|^2. \end{aligned} \tag{4}$$

In problem (3), m auxiliary variables $\mathbf{w}_i$ are introduced in addition to the global parameter $\mathbf{w}$. We emphasize that problems (2) and (3) are equivalent in terms of their optimal solutions but are expressed in different forms when $\lambda \in [0, \mu)$, and they are identical when $\lambda = \mu$. Throughout this work, we assume that optimal function value F^* is bounded from below, namely, $F^* > -\infty$.

The algorithmic design. When using ADMM to solve problem (3), we need its associated augmented Lagrange function, which is defined as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, W, \Pi; \boldsymbol{\sigma}) &:= \sum_{i=1}^{m} \alpha_i L_i(\mathbf{w}, \mathbf{w}_i, \boldsymbol{\pi}_i; \sigma_i) + \frac{\lambda}{2} \parallel \{\mathbf{w} \|^2, \\ L_i(\mathbf{w}, \mathbf{w}_i, \boldsymbol{\pi}_i; \sigma_i) &:= F_i(\mathbf{w}_i) + \langle \boldsymbol{\pi}_i, \mathbf{w}_i - \mathbf{w} \rangle + \frac{\sigma_i}{2} \parallel \mathbf{w}_i - \mathbf{w} \|^2, \end{aligned} \tag{5}$$

where $\sigma_i > 0$ and $\boldsymbol{\pi}_i$, $i \in [m]$ are the Lagrange multipliers. Based on the above augmented Lagrange function, the conventional ADMM updates each variable in $(\mathbf{w}, W, \Pi)$ iteratively. However, we modified the framework as follows. Given initial point $(\mathbf{w}^0, W^0, \Pi^0; \boldsymbol{\sigma}^0)$, the algorithm performs the following steps iteratively for $\ell = 0, 1, 2, \ldots$:

$$\mathbf{w}^{\ell+1} = \arg\min_{\mathbf{w}} \mathcal{L}\left(\mathbf{w}, W^\ell, \Pi^\ell; \boldsymbol{\sigma}^\ell\right), \tag{6a}$$

$$\mathbf{w}_i^{\ell+1} = \arg\min_{\mathbf{w}_i} L_i(\mathbf{w}^{\ell+1}, \mathbf{w}_i, \boldsymbol{\pi}_i^\ell; \sigma_i^{\ell+1}) + \frac{\rho_i}{2} \langle \mathbf{w}_i - \mathbf{w}^{\ell+1}, Q_i^{\ell+1}(\mathbf{w}_i - \mathbf{w}^{\ell+1}) \rangle, \tag{6b}$$

$$\boldsymbol{\pi}_i^{\ell+1} = \boldsymbol{\pi}_i^\ell + \sigma_i^{\ell+1}(\mathbf{w}_i^{\ell+1} - \mathbf{w}^{\ell+1}), \tag{6c}$$

for each $i \in [m]$, where $\rho_i > 0$, both scalar $\sigma_i^{\ell+1}$ and matrix $Q_i^{\ell+1} \succcurlyeq 0$ will be updated properly. Hereafter, superscripts ℓ and $\ell + 1$ in σ_i^ℓ and $\sigma_i^{\ell+1}$ stand for the iteration number rather than the power. Here $Q_i^{\ell+1}$ is commonly referred to as an (adaptively) preconditioning matrix in preconditioned gradient methods[26,74–76].

Remark 1. *The primary distinction between algorithmic framework (equation (6a–c)) and conventional ADMM lies in the inclusion of a term* $\frac{\rho_i}{2} \langle \mathbf{w}_i - \mathbf{w}^{\ell+1}, Q_i^{\ell+1}(\mathbf{w}_i - \mathbf{w}^{\ell+1}) \rangle$ *. This term enables the incorporation of various forms of useful information, such as second-moment, second-order information (for example, Hessian) and orthogonalized momentum by Newton–Schulz iterations, thereby enhancing the performance of the proposed algorithms; see the section 'Second moment' for more details.*

One can check that subproblem (6a) admits a closed-form solution outlined in equation (8). For subproblem (6b), to accelerate the computational speed, we solve it inexactly by

$$\begin{aligned} \mathbf{w}_i^{\ell+1} &= \arg\min_{\mathbf{w}_i} \langle \boldsymbol{\pi}_i^\ell, \mathbf{w}_i - \mathbf{w}^{\ell+1} \rangle + \frac{\sigma_i^{\ell+1}}{2} \parallel \mathbf{w}_i - \mathbf{w}^{\ell+1} \|^2 \\ &\quad + F_i(\mathbf{w}^{\ell+1}) + \langle \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1}), \mathbf{w}_i - \mathbf{w}^{\ell+1} \rangle \\ &\quad + \frac{\rho_i}{2} \left\langle \mathbf{w}_i - \mathbf{w}^{\ell+1}, Q_i^{\ell+1} (\mathbf{w}_i - \mathbf{w}^{\ell+1}) \right\rangle \\ &= \mathbf{w}^{\ell+1} - \left(\sigma_i^{\ell+1} I + \rho_i Q_i^{\ell+1}\right)^{-1} \left(\boldsymbol{\pi}_i^\ell + \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})\right). \end{aligned} \tag{7}$$

Algorithm 1. PISA.

```
Divide $\mathcal{D}$ into $m$ disjoint batches $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$ and calculate $\alpha_i$ by equation (1).
Initialize $\mathbf{w}^0 = \mathbf{w}_i^0 = \boldsymbol{\pi}_i^0 = 0$, $\gamma_i \in [3/4, 1)$, and $(\sigma_i^0, \eta_i, \rho_i) > 0$ for each $i \in [m]$.
for $\ell = 0, 1, 2, \ldots$ do
    $\mathbf{w}^{\ell+1} = \frac{\sum_{i=1}^{m} \alpha_i (\sigma_i^\ell \mathbf{w}_i^\ell + \boldsymbol{\pi}_i^\ell)}{\sum_{i=1}^{m} \alpha_i \sigma_i^\ell + \lambda}$.   (8)
    for $i = 1, 2, \ldots, m$ do
        Randomly draw a mini-batch $\mathcal{B}_i^{\ell+1} \subseteq D_i$.
        Calculate $\mathbf{g}_i^{\ell+1} = \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})$.   (9)
        Choose $Q_i^{\ell+1}$ to satisfy $\eta_i I \succcurlyeq Q_i^{\ell+1} \succcurlyeq 0$.   (10)
        $\sigma_i^{\ell+1} = \sigma_i^\ell / \gamma_i$.   (11)
        $\mathbf{w}_i^{\ell+1} = \mathbf{w}^{\ell+1} - (\sigma_i^{\ell+1} I + \rho_i Q_i^{\ell+1})^{-1} (\boldsymbol{\pi}_i^\ell + \mathbf{g}_i^{\ell+1})$.   (12)
        $\boldsymbol{\pi}_i^{\ell+1} = \boldsymbol{\pi}_i^\ell + \sigma_i^{\ell+1} (\mathbf{w}_i^{\ell+1} - \mathbf{w}^{\ell+1})$.   (13)
    end
end
```

This update admits three advantages. First, it solves problem (6b) by a closed-form solution, namely, the second equation in (7), reducing the computational complexity. Second, we approximate $F_i(\mathbf{w})$ using its first-order approximation at $\mathbf{w}^{\ell+1}$ rather than $\mathbf{w}_i^\ell$, which facilitates each batch parameter $\mathbf{w}_i^{\ell+1}$ to tend to $\mathbf{w}^{\ell+1}$ quickly, thereby accelerating the overall convergence. Finally, $\nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})$ serves as a stochastic approximation of true gradient $\nabla F_i(\mathbf{w}^{\ell+1}) = \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{D}_i)$, as defined by equation (4), where $\mathcal{B}_i^{\ell+1}$ is a random sample from $\mathcal{D}_i$. By using sub-batch datasets $\{\mathcal{B}_1^{\ell+1}, \ldots, \mathcal{B}_m^{\ell+1}\}$ in every iteration, rather than full data $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$, the computational cost is significantly reduced. Overall, on the basis of these observations, we name our algorithm PISA, as described in Algorithm 1.

Another advantageous property of PISA is its ability to perform parallel computation, which stems from the parallelism used in solving subproblems in ADMM. At each iteration, m nodes (that is, $i = 1, 2, \cdots, m$) update their parameters by equations (9)–(13) in parallel, thereby enabling the processing of large-scale datasets. Moreover, when specifying the preconditioning matrix, $Q_i^{\ell+1}$, as a diagonal matrix (as outlined in the section 'Second moment') and sampling $\mathcal{B}_i^{\ell+1}$ with small batch sizes, each node exhibits significantly low computational complexity, facilitating fast computation.

Convergence of PISA

In this subsection, we aim to establish the convergence property of Algorithm 1. To proceed with that, we first define a critical bound by

$$\varepsilon_i(r) := \sup_{\mathcal{B}_i,\mathcal{B}_i'\subseteq\mathcal{D}_i,\mathbf{w}\in\mathbb{N}(r)} 64\left\|\nabla F_i(\mathbf{w};\mathcal{B}_i)-\nabla F_i(\mathbf{w};\mathcal{B}_i')\right\|^2,\quad \forall i\in[m]. \tag{14}$$

Lemma 1. *$\varepsilon_i(r) < \infty$ for any given $r\in(0,\infty)$ and any $i\in[m]$.*
The proof of Lemma 1 is given in Supplementary Section 2.3. One can observe that $\varepsilon_i(r) = 0$ for any $r > 0$ if we take the full batch data in each step, namely, choosing $\mathcal{B}_i^\ell = (\mathcal{B}_i^\ell)' = \mathcal{D}_i$ for every $i\in[m]$ and all $\ell\ge 1$. However for min-batch dataset $\mathcal{B}_i^\ell\subset\mathcal{D}_i$, this parameter is related to the bound of variance $\mathbb{E}\|\nabla F_i(\mathbf{w};\mathcal{B}_i)-\nabla F_i(\mathbf{w};\mathcal{D}_i)\|^2$, which is commonly assumed to be bounded for any w[10,11,21,36,77]. However, in the subsequent analysis, we can verify that both generated sequences $\{\mathbf{w}^\ell\}$ and $\{\mathbf{w}_i^\ell\}$ fall into a bounded region $\mathbb{N}(\delta)$ for any $i\in[m]$ with δ defined as equation (16), thereby leading to a finitely bounded $\varepsilon_i(\delta)$ naturally, see Lemma 2. In other words, we no longer need to assume the boundedness of the variance, $\mathbb{E}\|\nabla F_i(\mathbf{w};\mathcal{B}_i)-\nabla F_i(\mathbf{w};\mathcal{D}_i)\|^2$ for any w. This assumption is known to be somewhat restrictive, particularly for non-IID or heterogeneous datasets. Therefore, the theorems we establish in the sequel effectively address this critical challenge[60,62]. Therefore, our algorithm demonstrates robust performance in settings with heterogeneous data.

Convergence analysis. To establish convergence, we need the assumption below. It assumes that function f has a Lipschitz continuous gradient on a bounded region, namely, the gradient is locally Lipschitz continuous. This is a relatively mild condition. Functions with (global) Lipschitz continuity and twice-continuously differentiable functions satisfy this condition. It is known that the Lipschitz continuity of the gradient is commonly referred to as L-smoothness. Therefore, our assumption can be regarded as L-smoothness on a bounded region, which is weaker than L-smoothness.

Assumption 1. *For each $t\in[|\mathcal{D}|]$, gradient $\nabla f(\cdot;\mathbf{x}_t)$ is Lipschitz continuous with a constant $c(\mathbf{x}_t) > 0$ on $\mathbb{N}(2\delta)$. Denote $c_i := \max_{\mathbf{x}_t\in\mathcal{D}_i} c(\mathbf{x}_t)$ and $r_i := c_i + \mu - \lambda$ for each $i\in[m]$.*

First, given a constant $\sigma > 0$, we define a set

$$\Omega := \left\{(\mathbf{w},W):\ \sum_{i=1}^m \alpha_i\left(F_i(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{\sigma}{2}\|\mathbf{w}_i-\mathbf{w}\|^2\right) \le F(\mathbf{w}^0) + \frac{1}{1-\gamma}\right\}, \tag{15}$$

where $\gamma := \max_{i\in[m]}\gamma_i$, based on which we further define

$$\delta := \sup_{(\mathbf{w},W)\in\Omega}\{\|\mathbf{w}\|,\|\mathbf{w}_1\|,\|\mathbf{w}_2\|,\ldots,\|\mathbf{w}_m\|\}. \tag{16}$$

This indicates that any point $(\mathbf{w},W)\in\Omega$ satisfies $\{\mathbf{w},\mathbf{w}_1,\ldots,\mathbf{w}_m\}\subseteq\mathbb{N}(\delta)$. Using this δ, we initialize $\boldsymbol{\sigma}^0 := (\sigma_1^0,\sigma_2^0,\cdots,\sigma_m^0)$ by

$$\sigma^0 := \min\{\sigma_1^0,\sigma_2^0,\cdots,\sigma_m^0\} \ge 8\max_{i\in[m]}\{\sigma,\ \rho_i\eta_i,\ r_i,\ \delta^{-2},\ \varepsilon_i(2\delta)\}. \tag{17}$$

It is easy to see that Ω is a bounded set due to F_i being bounded from below. Therefore, δ is bounded and so is $\varepsilon_i(2\delta)$ due to Lemma 1. Hence, σ^0 in equation (17) is a well-defined constant, namely, σ^0 can be set as a finite positive number. For notational simplicity, hereafter, we denote

$$\begin{aligned}
\Delta\mathbf{w}^\ell &:= \mathbf{w}^\ell - \mathbf{w}^{\ell-1}, & \Delta\mathbf{w}_i^\ell &:= \mathbf{w}_i^\ell - \mathbf{w}_i^{\ell-1},\\
\Delta\boldsymbol{\pi}_i^\ell &:= \boldsymbol{\pi}_i^\ell - \boldsymbol{\pi}_i^{\ell-1}, & \Delta\bar{\mathbf{w}}_i^\ell &:= \mathbf{w}_i^\ell - \mathbf{w}^\ell,\\
\Delta\mathbf{g}_i^\ell &:= \mathbf{g}_i^\ell - \nabla F_i(\mathbf{w}^\ell), & \mathcal{L}^\ell &:= \mathcal{L}(\mathbf{w}^\ell, W^\ell,\ ^\ell;\boldsymbol{\sigma}^\ell).
\end{aligned} \tag{18}$$

Our first result shows the descent property of a merit function associated with $\mathcal{L}^\ell$.

Lemma 2. *Let $\{(\mathbf{w}^\ell, W^\ell, \Pi^\ell)\}$ be the sequence generated by Algorithm 1 with $\boldsymbol{\sigma}^0$ chosen as equation (17). Then the following statements are valid under Assumption 1.*

(1) *For any $\ell\ge 0$, sequence $\{\mathbf{w}^\ell,\mathbf{w}_1^\ell,\ldots,\mathbf{w}_m^\ell\}\subseteq\mathbb{N}(\delta)$.*
(2) *For any $\ell\ge 0$,*

$$\widetilde{\mathcal{L}}^\ell - \widetilde{\mathcal{L}}^{\ell+1} \ge \sum_{i=1}^m\alpha_i\left[\frac{\sigma_i^\ell+2\lambda}{4}\left\|\Delta\mathbf{w}^{\ell+1}\right\|^2 + \frac{\sigma_i^\ell}{4}\left\|\Delta\mathbf{w}_i^{\ell+1}\right\|^2\right], \tag{19}$$

where $\widetilde{\mathcal{L}}^\ell$ is defined by

$$\tilde{L}^\ell := L^\ell + \sum_{i=1}^m\alpha_i\left[\frac{8}{\sigma_i^\ell}\|\rho_i Q_i^\ell\Delta\bar{\mathbf{w}}_i^\ell\|^2 + \frac{\gamma_i^\ell}{16(1-\gamma_i)}\right]. \tag{20}$$

The proof of Lemma 2 is given in Supplementary Section 2.6. This lemma is derived from a deterministic perspective. Such a success lies in considering the worst case of bound $\varepsilon_i(2\delta)$ (that is, taking all possible selections of $\{\mathcal{B}_1^\ell,\ldots,\mathcal{B}_m^\ell\}$ into account). On the basis of the above key lemma, the following theorem shows the sequence convergence of the algorithm.

Theorem 1. *Let $\{(\mathbf{w}^\ell, W^\ell, \Pi^\ell)\}$ be the sequence generated by Algorithm 1 with $\boldsymbol{\sigma}^0$ chosen as equation (17). Then the following statements are valid under* Assumption 1.

(1) *Sequences $\{\mathcal{L}^\ell\}$* and *$\{\widetilde{\mathcal{L}}^\ell\}$ converge and for any $i\in[m]$,*

$$0 = \lim_{\ell\to\infty}\left\|\Delta\mathbf{w}^\ell\right\| = \lim_{\ell\to\infty}\left\|\Delta\mathbf{w}_i^\ell\right\| = \lim_{\ell\to\infty}\left\|\Delta\bar{\mathbf{w}}_i^\ell\right\| = \lim_{\ell\to\infty}(\widetilde{\mathcal{L}}^\ell - \mathcal{L}^\ell). \tag{21}$$

(2) *Sequence $\{(\mathbf{w}^\ell, W^\ell, \mathbb{E}\Pi^\ell)\}$ converges.*

The proof of Theorem 1 is given in Supplementary Section 2.7. To ensure the convergence results, initial value $\boldsymbol{\sigma}^0$ is selected according to equation (17), which involves a hyperparameter δ. If a lower bound $\underline{F}$ of $\min_{\mathbf{w}}\sum_{i=1}^m\alpha_i F_i(\mathbf{w})$ is known, then an upper bound $\bar{\delta}$ for δ can be estimated from equations (15) and (16) by substituting $F_i(\mathbf{w})$ with $\underline{F}$. In practice, particularly in deep learning, many widely used loss functions, such as mean-squared error and cross-entropy, yield non-negative values. This observation allows us to set the lower bound as $\underline{F} = 0$. Once $\bar{\delta}$ is estimated, it can be used in equation (17) to select $\boldsymbol{\sigma}^0$, without affecting the convergence guarantees. However, it is worth emphasizing that equation (17) is a sufficient but not necessary condition. Therefore, in practice, it is not essential to enforce this condition strictly when initializing $\boldsymbol{\sigma}^0$ in numerical experiments.

Complexity analysis. Besides the convergence established above, the algorithm exhibits the following rate of convergence under the same assumption and parameter setup.

Theorem 2. *Let $\{(\mathbf{w}^\ell, W^\ell, \Pi^\ell)\}$ be the sequence generated by Algorithm 1 with $\boldsymbol{\sigma}^0$ chosen as equation (17). Let $\mathbf{w}^\infty$ be the limit of sequence $\{\mathbf{w}^\ell\}$. Then there is a constant $C_1 > 0$ such that*

$$\max\left\{\left\|\mathbf{w}^\ell-\mathbf{w}^\infty\right\|,\left\|\mathbf{w}_i^\ell-\mathbf{w}^\infty\right\|,\left\|\mathbb{E}\boldsymbol{\pi}_i^\ell+\nabla F_i(\mathbf{w}^\infty)\right\|,\ \forall i\in[m]\right\} \le C_1\gamma^\ell \tag{22}$$

and a constant $C_2 > 0$ such that

$$\max\{F_\lambda(\mathbf{w}^\ell),\mathcal{L}^\ell,\widetilde{\mathcal{L}}^\ell\} - F_\lambda(\mathbf{w}^\infty) \le C_2\gamma^\ell. \tag{23}$$

The proof of Theorem 2 is given in Supplementary Section 2.8. This theorem means that sequence $\{(\mathbf{w}^\ell, W^\ell, \mathbb{E}\Pi^\ell)\}$ converges to its limit in a linear rate. To achieve such a result, we only assume Assumption 1 without imposing any other commonly used assumptions, such as those presented in Table 1.

Precondition specification

In this section, we explore the preconditioning matrix, namely, matrix Q_i^ℓ. A simple and computationally efficient choice is to set $Q_i^\ell = I$, which enables fast computation of updating $\mathbf{w}_i^{\ell+1}$ via equation (12). However, this choice is too simple to extract useful information about F_i. Therefore, several alternatives can be adopted to set Q_i^ℓ.

Second-order information. Second-order optimization methods, such as Newton-type and trust region methods, are known to enhance numerical performance by leveraging second-order information, the (generalized) Hessian. For instance, if each function F_i is twice-continuously differentiable, then one can set

$$Q_i^{\ell+1} = \nabla^2 F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1}), \tag{24}$$

where $\nabla^2 F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})$ represents the Hessian of $F_i(\cdot; \mathcal{B}_i^{\ell+1})$ at $\mathbf{w}^{\ell+1}$. With this choice, subproblem (7) becomes closely related to second-order methods, and the update takes the form

$$\mathbf{w}_i^{\ell+1} = \mathbf{w}^{\ell+1} - \left(\sigma_i^{\ell+1} I + \rho_i \nabla^2 F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})\right)^{-1} \left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right).$$

This update corresponds to a Levenberg–Marquardt step[78,79] or a regularized Newton step[80,81] when $\sigma_i^{\ell+1} > 0$, and reduces to the classical Newton step if $\sigma_i^{\ell+1} = 0$. While incorporating the Hessian can improve performance in terms of iteration count and solution quality, it often leads to significantly high computational complexity. To mitigate this, some other effective approaches exploit the second moment derived from historical updates to construct the preconditioning matrices.

Second moment. We note that the second moment to determine an adaptive learning rate enables the improvements of the learning performance of several popular algorithms, such as RMSProp[16] and Adam[17]. Motivated by this, we specify preconditioning matrix by using the second moment as follows:

$$Q_i^{\ell+1} = \mathrm{Diag}\left(\sqrt{\mathbf{m}_i^{\ell+1}}\right), \tag{25}$$

where Diag($\mathbf{m}$) is the diagonal matrix with the diagonal entries formed by $\mathbf{m}$ and $\mathbf{m}_i^{\ell+1}$ can be chosen flexibly as long as it satisfies that $\|\mathbf{m}_i^{\ell+1}\|_\infty \le \eta_i^2$. Here, $\|\mathbf{m}\|_\infty$ is the infinity norm of m. We can set $\mathbf{m}_i^{\ell+1}$ as follows

$$\mathbf{m}_i^{\ell+1} = \min\left\{\widetilde{\mathbf{m}}_i^{\ell+1}, \eta_i^2 \mathbf{1}\right\}, \tag{26}$$

where $\widetilde{\mathbf{m}}_i^{\ell+1}$ can be updated by

$$\begin{aligned}
\text{Scheme I} \;:\; & \widetilde{\mathbf{m}}_i^{\ell+1} = \widetilde{\mathbf{m}}_i^{\ell} + \left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right) \odot \left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right), \\
\text{Scheme II} \;:\; & \widetilde{\mathbf{m}}_i^{\ell+1} = \beta_i \widetilde{\mathbf{m}}_i^{\ell} + (1-\beta_i)\left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right) \odot \left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right), \\
\text{Scheme III} \;:\; & \mathbf{n}_i^{\ell+1} = \beta_i \mathbf{n}_i^{\ell} + (1-\beta_i)\left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right) \odot \left(\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1}\right), \\
& \widetilde{\mathbf{m}}_i^{\ell+1} = \mathbf{n}_i^{\ell+1}/(1-\beta_i^{\ell+1}),
\end{aligned} \tag{27}$$

where $\widetilde{\mathbf{m}}_i^0$ and $\mathbf{n}_i^0$ are given, $\beta_i \in (0,1)$, and β_i^{ℓ} stands for power ℓ of β_i. These three schemes resemble the ones used by AdaGrad[15], RMSProp[16] and Adam[17], respectively. Putting equation (25) into Algorithm 1 gives rise to Algorithm 2. We term it SISA, an abbreviation for the second moment-based inexact SADMM. Compared with PISA in Algorithm 1, SISA admits three advantages.

(1) It is capable of incorporating various schemes of the second moment, which may enhance the numerical performance of SISA significantly.

(2) One can easily check that $\eta_i I \succeq Q_i^{\ell+1} \succeq 0$ for each batch $i \in [m]$ and all $\ell \ge 1$. Therefore, equation (25) enables us to preserve the convergence property as follows.

Theorem 3. *Let* $\{(\mathbf{w}^{\ell}, W^{\ell}, \Pi^{\ell})\}$ *be the sequence generated by Algorithm 2 with* $\boldsymbol{\sigma}^0$ *chosen as equation (17). Then under Assumption 1, all statements in Theorems 1 and 2 are valid.*

(3) Such a choice of $Q_i^{\ell+1}$ enables the fast computation compared with update $\mathbf{w}_i^{\ell+1}$ by equation (7). In fact, since operation $\mathbf{u}/\mathbf{v}$ denotes element-wise division, the complexity of computing equation (28) is $O(p)$, where p is the dimension of $\mathbf{w}_i^{\ell+1}$, whereas the complexity of computing equation (12) is $O(p^3)$.

Algorithm 2. SISA

```
Divide $\mathcal{D}$ into $m$ disjoint batches $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$ and calculate $\alpha_i$ by equation (1).
Initialize $\mathbf{w}^0 = \mathbf{w}_i^0 = \boldsymbol{\pi}_i^0 = 0, \gamma_i \in [3/4, 1)$, and $(\sigma_i^0, \eta_i, \rho_i) > 0$ for each $i \in [m]$.
for $\ell = 0, 1, 2, \ldots$ do
    $\mathbf{w}^{\ell+1} = \frac{\sum_{i=1}^m \alpha_i (\sigma_i^\ell \mathbf{w}_i^\ell + \boldsymbol{\pi}_i^\ell)}{\sum_{i=1}^m \alpha_i \sigma_i^\ell + \lambda}.$
    for $i = 1, 2, \ldots, m$ do
        Randomly draw a mini-batch $\mathcal{B}_i^{\ell+1} \subseteq \mathcal{D}_i$.
        Compute $\mathbf{g}_i^{\ell+1} = \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})$.
        Choose $\mathbf{m}_i^{\ell+1}$ to satisfy $\|\mathbf{m}_i^{\ell+1}\|_\infty \le \eta_i^2$.
        $\sigma_i^{\ell+1} = \sigma_i^\ell / \gamma_i$.                                              (28)
        $\mathbf{w}_i^{\ell+1} = \mathbf{w}^{\ell+1} - \frac{\boldsymbol{\pi}_i^\ell + \mathbf{g}_i^{\ell+1}}{\sigma_i^{\ell+1} + \rho_i \sqrt{\mathbf{m}_i^{\ell+1}}}$.
        $\boldsymbol{\pi}_i^{\ell+1} = \boldsymbol{\pi}_i^\ell + \sigma_i^{\ell+1}(\mathbf{w}_i^{\ell+1} - \mathbf{w}^{\ell+1})$.
    end
end
```

Algorithm 3. NSISA

```
Divide $\mathcal{D}$ into $m$ disjoint batches $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$ and calculate $\alpha_i$ by equation (1).
Initialize $\mathbf{w}^0 = \mathbf{w}_i^0 = \boldsymbol{\pi}_i^0 = \mathbf{b}_i^0 = 0, \gamma_i \in [3/4, 1), \epsilon_i \in (0,1)$ and $(\sigma_i^0, \rho_i, \mu_i) > 0$ for each $i \in [m]$.
for $\ell = 0, 1, 2, \ldots$ do
    $\mathbf{w}^{\ell+1} = \frac{\sum_{i=1}^m \alpha_i (\sigma_i^\ell \mathbf{w}_i^\ell + \boldsymbol{\pi}_i^\ell)}{\sum_{i=1}^m \alpha_i \sigma_i^\ell + \lambda}.$
    for $i = 1, 2, \ldots, m$ do
        Randomly draw a mini-batch $\mathcal{B}_i^{\ell+1} \subseteq \mathcal{D}_i$.
        Compute $\mathbf{g}_i^{\ell+1} = \nabla F_i(\mathbf{w}^{\ell+1}; \mathcal{B}_i^{\ell+1})$.
        $\mathbf{b}_i^{\ell+1} = \mu_i \mathbf{b}_i^\ell + \mathbf{g}_i^{\ell+1}$.
        $\mathbf{o}_i^{\ell+1} = \texttt{NewtonSchulz}(\mathbf{b}_i^{\ell+1})$.
        $\sigma_i^{\ell+1} = \sigma_i^\ell / \gamma_i$.                                              (29)
        $\mathbf{w}_i^{\ell+1} = \mathbf{w}^{\ell+1} - \frac{\boldsymbol{\pi}_i^\ell + \mathbf{o}_i^{\ell+1} + \epsilon_i^{\ell+1} \mathbf{v}_i^{\ell+1}}{\sigma_i^{\ell+1} + \rho_i \sqrt{\mathbf{m}_i^{\ell+1}}}$.
        $\boldsymbol{\pi}_i^{\ell+1} = \boldsymbol{\pi}_i^\ell + \sigma_i^{\ell+1}(\mathbf{w}_i^{\ell+1} - \mathbf{w}^{\ell+1})$.
    end
end
```

Orthogonalized momentum by Newton–Schulz iterations. Recently, the authors of ref. 27 proposed an algorithm called Muon, which orthogonalizes momentum using Newton–Schulz iterations. This approach has shown promising results in fine-tuning LLMs, outperforming many established optimizers. The underlying philosophy of Muon can also inform the design of the preconditioning matrix. Specifically, we consider the two-dimensional case, namely, the trainable variable $\mathbf{w}$ is a matrix. Then subproblem (7) in a vector form turns to

$$\mathrm{vec}(\mathbf{w}_i^{\ell+1}) = \mathrm{vec}(\mathbf{w}^{\ell+1}) - (\sigma_i^{\ell+1}I + \rho_i Q_i^{\ell+1})^{-1}(\mathrm{vec}(\boldsymbol{\pi}_i^{\ell}) + \mathrm{vec}(\mathbf{g}_i^{\ell+1})), \quad (30)$$

where vec($\mathbf{w}$) denotes the column-wise vectorization of matrix $\mathbf{w}$. Now, initialize $\mathbf{b}_i^0$ for all $i \in [m]$ and $\mu > 0$, update momentum by $\mathbf{b}_i^{\ell+1} = \mu\mathbf{b}_i^{\ell} + \mathbf{g}_i^{\ell+1}$. Let $\mathbf{b}_i^{\ell+1} = U_i^{\ell+1}\Lambda_i^{\ell+1}(V_i^{\ell+1})$ be the singular value decomposition of $\mathbf{b}_i^{\ell+1}$, where $\Lambda_i^{\ell+1}$ is a diagonal matrix and $U_i^{\ell+1}$ and $V_i^{\ell+1}$ are two orthogonal matrices. Compute $\mathbf{o}_i^{\ell+1} = U_i^{\ell+1}(V_i^{\ell+1})$ and $\mathbf{p}_i^{\ell+1}$ by

$$\mathbf{p}_i^{\ell} = \frac{(\sigma_i^{\ell+1} + \rho_i\sqrt{\mathbf{m}_i^{\ell+1}}) \odot (\boldsymbol{\pi}_i^{\ell} + \mathbf{g}_i^{\ell+1})}{\rho_i(\boldsymbol{\pi}_i^{\ell} + \mathbf{o}_i^{\ell+1} + \epsilon_i^{\ell+1}\mathbf{v}_i^{\ell+1})} - \frac{\sigma_i^{\ell+1}}{\rho_i},$$

where $\mathbf{m}_i^{\ell+1}$ can be the second moment (for example, $\mathbf{m}_i^{\ell+1} = (\boldsymbol{\pi}_i^{\ell} + \mathbf{o}_i^{\ell+1}) \odot (\boldsymbol{\pi}_i^{\ell} + \mathbf{o}_i^{\ell+1})$ is used in the numerical experiment), $\epsilon_i \in (0, 1)$ (here, ϵ_i^{ℓ} stands for power ℓ of ϵ_i) and $\mathbf{v}_i^{\ell+1}$ is a matrix with (k,j)th element computed by $(\mathbf{v}_i^{\ell+1})_{kj} = 1$ if $(\boldsymbol{\pi}_i^{\ell} + \mathbf{o}_i^{\ell+1})_{kj} = 0$ and $(\mathbf{v}_i^{\ell+1})_{kj} = 0$ otherwise. Then we set the preconditioning matrix by

$$Q_i^{\ell+1} = \mathrm{Diag}\ (\mathrm{vec}(\mathbf{p}_i^{\ell+1})).$$

Substituting above choice into equation (30) derives equation (29). The idea of using equation (29) is inspired by ref. 27, where Newton–Schulz orthogonalization[82,83] is used to efficiently approximate $\mathbf{o}_i^{\ell+1}$. Incorporating these steps into Algorithm 1 leads to Algorithm 3, which we refer to as NSISA. The implementation of `NewtonSchulz`($\mathbf{b}$) is provided in ref. 27. Below is the convergence result of NSISA.

Theorem 4. *Let* $\{(\mathbf{w}^{\ell}, W^{\ell}, \Pi^{\ell})\}$ *be the sequence generated by Algorithm* 3 *with* $\boldsymbol{\sigma}^0$ *chosen as equation* (17). *If Assumption* 1 *holds and* $(\mathbf{p}_i^{\ell})_{kj} \in (0, \eta_i)$ *for any* (k, j) *and* $\ell \geq 0$, *all statements in Theorems* 1 *and* 2 *are valid.*

Data availability

The other datasets used in this analysis are publicly available as referenced: ImageNet[84], CIFAR-10[85], MNIST[86], FMNIST[87], Adult[88], Fineweb[89] and Penn Treebank[90].

Code availability

All code is available via GitHub at https://github.com/Tracy-Wang7/PISA and via Code Ocean at https://doi.org/10.24433/CO.3435996.v1 (ref. 91).

References

1. Touvron, H. et al. Llama 2: open foundation and fine-tuned chat models. Preprint at https://arxiv.org/abs/2307.09288 (2023).
2. OpenAI et al. GPT-4 technical report. Preprint at https://arxiv.org/abs/2303.08774 (2024).
3. Gemma Team et al. Gemma: open models based on Gemini research and technology. Preprint at https://arxiv.org/abs/2403.08295 (2024).
4. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778 (IEEE, 2016).
5. Dosovitskiy, A. et al. An image is worth 16 × 16 words: transformers for image recognition at scale. In *International Conference on Learning Representations* (ICLR, 2021).
6. Robbins, H. & Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **22**, 400–407 (1951).
7. Chung, K. L. On a stochastic approximation method. *Ann. Math. Stat.* **25**, 463–483 (1954).
8. Zinkevich, M., Weimer, M., Li, L. & Smola, A. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)* 2595–2603 (Curran Associates, Inc., 2010).
9. McMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. A. Y. Communication-efficient learning of deep networks from decentralized data. In *Proc. 20th International Conference on Artificial Intelligence and Statistics* (eds Singh, A. & Zhu, J.) 1273–1282 (PMLR, 2017).
10. Li, X., Huang, K., Yang, W., Wang, S. & Zhang, Z. On the convergence of FedAvg on non-IID data. In *International Conference on Learning Representations* (ICLR, 2020).
11. Stich, S. U. Local SGD converges fast and communicates little. In *International Conference on Learning Representations* (ICLR, 2019).
12. Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J. & Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. Preprint at arXiv:1802.08760 (2018).
13. Chen, X. et al. Symbolic discovery of optimization algorithms. Preprint at https://arxiv.org/abs/2302.06675 (2023).
14. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Netw.* **12**, 145–151 (1999).
15. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**, 2121–2159 (2011).
16. Tieleman, T. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* **4**, 26–31 (2012).
17. Kingma, D. P. Adam: a method for stochastic optimization. In *International Conference on Learning Representations* (ICLR, 2015).
18. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of Adam and beyond. In *International Conference on Learning Representations* (ICLR, 2018).
19. Chen, J. et al. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *International Joint Conferences on Artificial Intelligence (IJCAI)* 3267–3275 (International Joint Conferences on Artificial Intelligence Organization, 2018).
20. Luo, L., Xiong, Y., Liu, Y. & Sun, X. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations* (ICLR, 2019).
21. Yu, H., Yang, S. & Zhu, S. Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In *Association for the Advancement of Artificial Intelligence (AAAI)* 5693–5700 (AAAI, 2019).
22. You, Y. et al. Large batch optimization for deep learning: training BERT in 76 minutes. In *International Conference on Learning Representations* (ICLR, 2020).
23. You, Y., Gitman, I. & Ginsburg, B. Scaling SGD batch size to 32k for ImageNet training. Preprint at https://arxiv.org/abs/1708.03888v1?2 (2017).
24. Zeiler, M. D. Adadelta: an adaptive learning rate method. Preprint at https://arxiv.org/abs/1212.5701 (2012).
25. Loshchilov, I. Decoupled weight decay regularization. In *International Conference on Learning Representations* (ICLR, 2018).
26. Gupta, V., Koren, T. & Singer, Y. Shampoo: preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)* 1842–1850 (PMLR, 2018).
27. Jordan, K. et al. Muon: an optimizer for hidden layers in neural networks. *Keller Jordan blog* https://kellerjordan.github.io/posts/muon/ (2024).
28. Vyas, N. et al. SOAP: improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations* (ICLR, 2025).
29. Zhang, Y. et al. Adam-mini: use fewer learning rates to gain more. In *International Conference on Learning Representations* (ICLR, 2025).

30. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics* 177–186 (Springer, 2010).
31. Ruder, S. An overview of gradient descent optimization algorithms. Preprint at https://arxiv.org/abs/1609.04747 (2016).
32. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Rev.* **60**, 223–311 (2018).
33. Nesterov, Y. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl akad nauk Sssr* **269**, 543– 547 (1983).
34. Nesterov, Y. On an approach to the construction of optimal methods of minimization of smooth convex functions. *Ekonomika i Mateaticheskie Metody* **24**, 509–517 (1988).
35. Dozat, T. Incorporating Nesterov momentum into Adam. In *International Conference on Learning Representations* 1–4 (ICLR, 2016).
36. Xie, X., Zhou, P., Li, H., Lin, Z. & Yan, S. Adan: adaptive Nesterov momentum algorithm for faster optimizing deep models. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 1–13 (2024).
37. Gabay, D. & Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Comput. Math. Appl.* **2**, 17–40 (1976).
38. Boyd, S. et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**, 1–122 (2011).
39. Yang, Y., Sun, J., Li, H. & Xu, Z. ADMM-CSNet: a deep learning approach for image compressive sensing. *IEEE Trans. Pattern Anal. Mach. Intell.* **42**, 521–538 (2018).
40. Zhang, X., Hong, M., Dhople, S., Yin, W. & Liu, Y. FedPD: a federated learning framework with adaptivity to non-IID data. *IEEE Tran. Signal Process.* **69**, 6055–6070 (2021).
41. Zhou, S. & Li, G. Y. Federated learning via inexact ADMM. *IEEE Trans. Pattern. Anal. Mach. Intell.* **45**, 9699–9708 (2023).
42. Zhou, S. & Li, G. Y. FedGiA: an efficient hybrid algorithm for federated learning. *IEEE Trans. Signal Process.* **71**, 1493–1508 (2023).
43. Xu, K., Zhou, S. & Li, G. Y. Federated reinforcement learning for resource allocation in V2X networks. *IEEE J. Sel. Topics Signal Process.* **24**, 2799–2803 (2024).
44. Wang, O., Zhou, S. & Li, G. Y. Frameworks on few-shot learning with applications in wireless communication. *IEEE Trans. Signal Process.* **73**, 3857–3871 (2025).
45. Duchi, J. C., Agarwal, A. & Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans. Automat. Contr.* **57**, 592–606 (2011).
46. Hosseini, S., Chapman, A. & Mesbahi, M. Online distributed optimization via dual averaging. In *Conference on Decision and Control (CDC)* 1484–1489 (IEEE, 2013).
47. Tsianos, K. I., Lawlor, S. & Rabbat, M. G. Push-sum distributed dual averaging for convex optimization. In *Conference on Decision and Control (CDC)* 5453–5458 (IEEE, 2012).
48. Pu, S., Shi, W., Xu, J. & Nedić, A. Push–pull gradient methods for distributed optimization in networks. *IEEE Trans. Autom. Control.* **66**, 1–16 (2020).
49. Taylor, G. et al. Training neural networks without gradients: a scalable ADMM approach. In *International Conference on Machine Learning (ICML)* 2722–2731 (PMLR, 2016).
50. Wang, J., Yu, F., Chen, X. & Zhao, L. ADMM for efficient deep learning with global convergence. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*. 111–119 (2019).
51. Ebrahimi, Z., Batista, G. & Deghat, M. AA-DLADMM: an accelerated ADMM-based framework for training deep neural networks. Preprint at https://arxiv.org/abs/2401.03619 (2024).
52. Zeng, J., Lin, S.-B., Yao, Y. & Zhou, D.-X. On ADMM in deep learning: convergence and saturation-avoidance. *J. Mach. Learn. Res.* **22**, 9024–9090 (2021).
53. Ouyang, H., He, N., Tran, L. & Gray, A. Stochastic alternating direction method of multipliers. In *International Conference on Machine Learning (ICML)* 80–88 (PMLR, 2013).
54. Ding, J. et al. Stochastic ADMM based distributed machine learning with differential privacy. In *SecureComm* 257–277 (Springer, 2019).
55. Zhong, W. & Kwok, J. Fast stochastic alternating direction method of multipliers. In *International Conference on Machine Learning (ICML)* 46–54 (PMLR, 2014).
56. Huang, F., Chen, S. & Huang, H. Faster stochastic alternating direction method of multipliers for nonconvex optimization. In *International Conference on Machine Learning (ICML)* 2839–2848 (PMLR, 2019).
57. Liu, Y., Shang, F. & Cheng, J. Accelerated variance reduced stochastic ADMM. In *Association for the Advancement of Artificial Intelligence (AAAI)* 2287–2293 (AAAI, 2017).
58. Zeng, Y., Wang, Z., Bai, J. & Shen, X. An accelerated stochastic ADMM for nonconvex and nonsmooth finite-sum optimization. *Automatica* **163**, 111554 (2024).
59. Li, T. et al. Federated optimization in heterogeneous networks. *MLSys* **2**, 429–450 (2020).
60. Kairouz, P. et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **14**, 1–210 (2021).
61. Li, Q., Diao, Y., Chen, Q. & He, B. Federated learning on non-IID data silos: an experimental study. In *International Conference on Data Engineering (ICDE)* 965–978 (IEEE, 2022).
62. Ye, M., Fang, X., Du, B., Yuen, P. C. & Tao, D. Heterogeneous federated learning: state-of-the-art and research challenges. *ACM Comput. Surv.* **56**, 1–44 (2023).
63. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)* 1097–1105 (Curran Associates, Inc., 2012).
64. Krizhevsky, A., Nair, V. & Hinton, G. *CIFAR-10* (University of Toronto, 2010); https://www.cs.toronto.edu/~kriz/cifar.html
65. Zhuang, J. et al. AdaBelief optimizer: adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems (NeurIPS)* **33**, 18795–18806 (Curran Associates, Inc., 2020).
66. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (ICLR, 2014).
67. He, K., Zhang, X., Ren, S. & Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)* 630–645 (Springer, 2016).
68. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 4700–4708 (IEEE, 2017).
69. Radford, A. et al. Language models are unsupervised multitask learners. *OpenAI Blog* **1**, 9 (2019).
70. Goodfellow, I. et al. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)* 2672–2680 (Curran Associates, Inc., 2014).
71. Salimans, T. et al. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NeurIPS)* 2234–2242 (Curran Associates, Inc., 2016).
72. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)* 214–223 (PMLR, 2017).
73. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)* 6626–6637 (Curran Associates, Inc., 2017).

74. Li, X.-L. Preconditioned stochastic gradient descent. *IEEE Trans. Neural Netw. Learn. Syst.* **29**, 1454–1466 (2017).
75. Agarwal, N. et al. Efficient full-matrix adaptive regularization. In *International Conference on Machine Learning (ICML)* 102–110 (PMLR, 2019).
76. Yong, H., Sun, Y. & Zhang, L. A general regret bound of preconditioned gradient method for DNN training. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 7866–7875 (IEEE, 2023).
77. Wang, J. & Joshi, G. Cooperative SGD: a unified framework for the design and analysis of local-update SGD algorithms. *J. Mach. Learn. Res.* **22**, 1–50 (2021).
78. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **2**, 164–168 (1944).
79. Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math* **11**, 431–441 (1963).
80. Li, D.-H., Fukushima, M., Qi, L. & Yamashita, N. Regularized newton methods for convex minimization problems with singular solutions. *Comput. Optim. Appl.* **28**, 131–147 (2004).
81. Polyak, R. A. Regularized Newton method for unconstrained convex optimization. *Math. Program.* **120**, 125–145 (2009).
82. Kovarik, Z. Some iterative methods for improving orthonormality. *SIAM J. Numer. Anal.* **7**, 386–389 (1970).
83. Björck, Å & Bowie, C. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM J. Numer. Anal.* **8**, 358–364 (1971).
84. Deng, J. et al. ImageNet: a large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 248–255 (IEEE, 2009).
85. Krizhevsky, A. & Hinton, G. *Learning Multiple Layers of Features from Tiny Images*. Technical Report TR-2009-0 (University of Toronto, 2009).
86. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (2002).
87. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. Preprint at arXiv:1708.07747 (2017).
88. Chang, C.-C. & Lin, C.-J. Libsvm: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**, 27:1–27:27 (2011).
89. Penedo, G. et al. The fineweb datasets: decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)* **37**, 30811–30849 (Curran Associates, Inc., 2024).
90. Marcus, M., Santorini, B. & Marcinkiewicz, M. A. Building a large annotated corpus of English: the Penn Treebank. *Comput. Linguist.* **19**, 313–330 (1993).
91. Wang, O. Preconditioned inexact stochastic ADMM for deep models (source code). *Code Ocean* https://doi.org/10.24433/CO.3435996.v1 (2025).
92. Mukkamala, M. C. & Hein, M. Variants of RMSPprop and Adagrad with logarithmic regret bounds. In *International Conference on Machine Learning (ICML)* 2545–2553 (PMLR, 2017).
93. Zou, F., Shen, L., Jie, Z., Zhang, W. & Liu, W. A sufficient condition for convergences of Adam and RMSProp. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 11127–11135 (IEEE, 2019).
94. Reddi, S. J. et al. Adaptive federated optimization. In *International Conference on Learning Representations* (ICLR, 2021).
95. Liu, L. et al. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations* (ICLR, 2020).
96. Balles, L. & Hennig, P. Dissecting Adam: the sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning (ICML)* 404–413 (PMLR, 2018).
97. Zaheer, M., Reddi, S., Sachan, D., Kale, S. & Kumar, S. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)* **31**, 9815–9825 (Curran Associates, Inc., 2018).

Acknowledgements

S.Z. is partly supported by the National Key Research and Development Program of China (grant no. 2023YFA1011100), the Fundamental Research Funds for the Central Universities and the Talent Fund of Beijing Jiaotong University. Z.L. acknowledges support from the National Natural Science Foundation of China (grant no. 12271022) and National Key Research and the Development Program of China (grant no. 2024YFA1012901).

Author contributions

S.Z. and O.W. conceived the research. S.Z. developed the algorithms, established all theoretical results, guided the experiments and wrote the paper. O.W. designed and carried out all experiments and prepared the results section. Z.L. checked the theoretical proofs, revised the paper and offered technical insights. Y.Z. and G.Y.L. contributed to discussions and assisted with article preparation.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-026-01182-3.

Correspondence and requests for materials should be addressed to Shenglong Zhou or Ouya Wang.

Peer review information *Nature Machine Intelligence* thanks the anonymous reviewers for their contribution to the peer review of this work.