Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training

Abstract

Given the massive cost of language model pre-training, a non-trivial improvement of the optimization algorithm would lead to a material reduction on the time and cost of training. Adam and its variants have been state-of-the-art for years, and more sophisticated second-order (Hessian-based) optimizers often incur too much per-step overhead. In this paper, we propose Sophia, **S**econd-**o**rder Cli**p**ped Stoc**h**astic Opt**i**miz**a**tion, a simple scalable second-order optimizer that uses a light-weight estimate of the diagonal Hessian as the pre-conditioner. The update is the moving average of the gradients divided by the moving average of the estimated Hessian, followed by element-wise clipping. The clipping controls the worst-case update size and tames the negative impact of non-convexity and rapid change of Hessian along the trajectory. Sophia only estimates the diagonal Hessian every handful of iterations, which has negligible average per-step time and memory overhead. On language modeling with GPT-2 models of sizes ranging from 125M to 770M, Sophia achieves a 2x speed-up compared with Adam in the number of steps, total compute, and wall-clock time.

1. Introduction

Language models (LLMs) have gained phenomenal capabilities as their scale grows (Radford et al., 2019; Kaplan et al., 2020; Brown et al., 2020; Zhang et al., 2022; Touvron et al., 2023; OpenAI, 2023). However, pre-training LLMs is incredibly time-consuming due to the massive datasets and model sizes—hundreds of thousands of updates to the model parameters are required. For example, PaLM was trained for two months on 6144 TPUs, which costed 10 million dollars (Chowdhery et al., 2022).

Adam (Kingma & Ba, 2014) (or its variants (Loshchilov & Hutter, 2017; Shazeer & Stern, 2018; You et al., 2019)) is the dominantly used optimizer for training LLMs. Designing faster optimizers for LLMs is challenging. First, the benefit of the first-order (gradient-based) pre-conditioner in Adam is not yet well understood (Liu et al., 2020; Zhang et al., 2020; Kunstner et al., 2023). Second, the choice of pre-conditioners is constrained because we can only afford light-weight options whose overhead can be offset by the speed-up in the number of iterations. For example, the block-diagonal Hessian pre-conditioner in K-FAC is prohibitively expensive for LLMs (Martens & Grosse, 2015; Grosse & Martens, 2016; Ba et al., 2017; Martens et al., 2018). On the other hand, Chen et al. (2023) automatically search among the light-weight gradient-based pre-conditioners and identify Lion, which is substantially faster than Adam on vision Transformers and diffusion models but achieves limited speed-up on LLMs (Chen et al., 2023).

This paper introduces Sophia, **S**econd-**o**rder Cli**p**ped Stoc**h**ast**i**c Optimiz**a**tion, a light-weight second-order optimizer that uses an inexpensive stochastic estimate of the diagonal of the Hessian as a pre-conditioner and a clipping mechanism to control the worst-case update size. On pre-training language models such as GPT-2, Sophia achieves the same validation pre-training loss with 50% fewer number of steps than Adam. Because Sophia maintains almost the memory and average time per step, the speedup also translates to 50% less total compute and 50% less wall-clock time (See Figure 1 (a)&(b).). Moreover, the scaling law based on model size from 125M to 770M is in favor of Sophia over Adam—the gap between Sophia and Adam with 100K steps increases as the model size increases (Figure 1 (c)). In particular, Sophia on a 540M-parameter model with 100K steps gives the same validation loss as Adam on a 770M-parameter model with 100K steps. Note that the latter model needs 40% more training time and 40% more inference cost.

Concretely, Sophia estimates the diagonal entries of the Hessian of the loss using a mini-batch of examples every k step (with $k = 10$ in our implementation). We consider two options for diagonal Hessian estimators: (a) an unbiased estimator that uses a Hessian-vector product with the same run-time as a mini-batch gradient up to a constant factor, and (b) a biased estimator that uses one mini-batch gradient calculated with resampled labels. Both the two estimators only introduce 5% overheads per step (in average). At every step, Sophia updates the parameter with an exponential moving average (EMA) of the gradient divided by

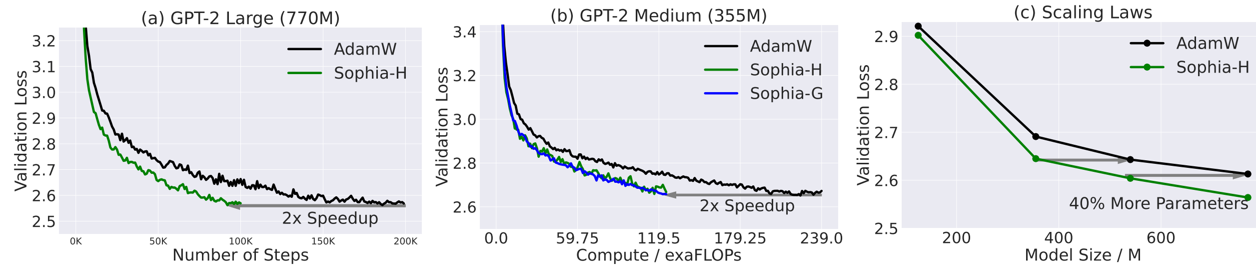

Figure 1. Sophia achieves a 2x speedup over AdamW in GPT-2 pre-training on OpenWebText. (a), (b) Comparison of the number of steps needed to achieve the same level of validation loss on (a) GPT-2-large (770M) and (b) GPT-2-medium (355M). Across all model sizes, Sophia achieves a 2x speedup over AdamW. (c) Validation losses of models with different sizes pre-trained for 100K steps. The gap between Sophia-H and AdamW gets larger as models size grows. Notably, using Sophia-H on a 540M-parameter model results in the same loss as using AdamW on a 770M-parameter model. See Section 3 for details and more results.

Algorithm 1 Hutchinson(θ)

Input: parameter θ.
Compute mini-batch loss $L(\theta)$.
Draw u from $\mathcal{N}(0, \mathrm{I}_d)$.
return $u \odot \nabla(\langle \nabla L(\theta), u \rangle)$.

Algorithm 2 Gauss-Newton-Bartlett(θ)

Input: parameter θ.
Draw a mini-batch of input $\{x_b\}_{b=1}^{B}$.
Compute logits on the mini-batch: $\{f(\theta, x_b)\}_{b=1}^{B}$.
Sample $\hat{y}_b \sim \mathrm{softmax}(f(\theta, x_b)), \forall b \in [B]$.
Calculate $\hat{g} = \nabla(1/B \sum \ell(f(\theta, x_b), \hat{y}_b))$.
return $B \cdot \hat{g} \odot \hat{g}$.

Algorithm 3 Sophia

Input: θ_1, learning rate $\{\eta_t\}_{t=1}^{T}$, hyperparameters $\lambda, \beta_1, \beta_2$, ϵ, and estimator choice Estimator $\in$ {Hutchinson,Gauss-Newton-Bartlett}
Set $m_0 = 0$, $v_0 = 0$, $h_{1-k} = 0$
for $t = 1$ **to** T **do**
  Compute minibach loss $L_t(\theta_t)$.
  Compute $g_t = \nabla L_t(\theta_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  if $t \bmod k = 1$ **then**
    Compute $\hat{h}_t =$ Estimator(θ_t).
    $h_t = \beta_2 h_{t-k} + (1 - \beta_2)\hat{h}_t$
  else
    $h_t = h_{t-1}$
  $\theta_t = \theta_t - \eta_t \lambda \theta_t$ (weight decay)
  $\theta_{t+1} = \theta_t - \eta_t \cdot \mathrm{clip}(m_t / \max\{h_t, \epsilon\}, \rho)$

the EMA of the diagonal Hessian estimate, subsequently clipped by a scalar. (All operations are element-wise.) See Algorithm 3 for the pseudo-code.

Additionally, Sophia can be seamlessly integrated into existing training pipelines, without any special requirements on the model architecture or computing infrastructure. With the either of the Hessian estimators, Sophia only require either standard mini-batch gradients, or Hessian-vector products which are supported in auto-differentiation frameworks such as PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018).

2. Method

We will motivate the use of second-order information and clipping in Section A. We present Sophia in detail in Section 2.1, and the pseudo-code in Algorithm 3. We introduce two choices of estimators of diagonal Hessian used in Sophia in Section 2.2.

2.1. Sophia: *S*econd-*o*rder Cli*p*ped Stoc*h*astic Opt*imiza*tion

Adam does not sufficiently adapt to the heterogeneous curvatures. On the other hand, vanilla Newton's method has a pre-conditioner optimal for convex functions, but is vulnerable to negative curvature and rapid change of Hessian. With these insights, we design a new optimizer, Sophia, which is more adaptive to heterogeneous curvatures than Adam, more resistant to non-convexity and rapid change of Hessian than Newton's method, and also uses a low-cost pre-conditioner.

We use θ_t to denote the parameter at time step t. At each step, we sample a mini-batch from the data distribution and calculate the mini-batch loss, denoted by $L_t(\theta_t)$. We denote by g_t the gradient of $L_t(\theta_t)$, i.e. $g_t = \nabla L_t(\theta_t)$. Let m_t be the EMA of gradients, $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$, which is the numerator of the update.

EMA of diagonal Hessian estimates. Sophia uses a diagonal Hessian-based pre-conditioner, which directly adjusts the update size of different parameter dimensions according to their curvatures. We will present two options in detail in Section 2.2 for estimating the diagonal Hessian efficiently. To mitigate the overhead, we only estimate the Hessian every k steps ($k = 10$ in our implementation). At time step t with $t \bmod k = 1$, the estimator returns an estimate $\hat{h}_t$ of the diagonal of the Hessian of the mini-batch loss.

Similar to the gradient of the mini-batch loss function, the estimated diagonal Hessian can also have large noise. Inspired by the EMA of moments of gradients in Adam, we also denoise the diagonal Hessian estimates with EMA across iterations. We update the EMA every k steps, resulting in the following update rule for the diagonal Hessian estimate:

$$h_t = \beta_2 h_{t-k} + (1-\beta_2)\hat{h}_t \ \text{ if } t \bmod k = 1; \text{ else } h_t = h_{t-1}\,.$$

Pre-coordinate clipping. As discussed in Section A, on nonconvex functions, vanilla Newton's method, which uses Hessian as the pre-conditioner, may converge to local maxima instead of local minima. In addition, the inaccuracy of Hessian estimates and the change of Hessian along the trajectory can make the second-order information unreliable. To this end, we only consider the positive entries of the diagonal Hessian and introduce per-coordinate clipping to the update. Let $\text{clip}(z,\rho) = \max\{\min\{z,\rho\}, -\rho\}$ be the clipping function with threshold $\rho > 0$. The update rule can be written as:

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \cdot \text{clip}(m_t / \max\{h_t, \epsilon\}, \rho), \tag{1}$$

where $\epsilon > 0$ is a very small constant to avoid dividing by 0, and all the operations are applied element-wise. We present the pseudo-code of the Sophia in Algorithm 3.

When any entry of h_t is negative, e.g., $h_t[i] < 0$, the corresponding entry in the pre-conditioned gradient $m_t[i]/\max\{h_t[i],\epsilon\} = m_t[i]/\epsilon$ is extremely large and has the same sign as $m_t[i]$, and thus $\eta \cdot \text{clip}(m_t[i]/\max\{h_t[i],\epsilon\},\rho) = \eta\rho \cdot \text{sign}(m_t[i])$, which is the same as stochastic momentum SignSGD. In other words, Sophia uses stochastic momentum SignSGD as a backup when the Hessian is negative (or mistakenly estimated to be negative or very small.) We also note that the clipping mechanism controls the worst-case size of the updates in all parameter dimensions to be at most ρ, which also improves the stability (which could be a severe issue for second-order methods). Moreover, because for many parameter dimensions, the clipping is not activated and the update is automatically adjusted, our worst-case update size $\eta\rho$ can be chosen to be larger than the worst update size η in stochastic momentum SignSGD.

Several previous works (Becker & Le Cun, 1988; Chapelle et al., 2011; Schaul et al., 2013), including the recent work AdaHessian (Yao et al., 2021), use diagonal Hessian as a pre-conditioner in optimizers for training neural networks. However, they use more frequent Hessian estimations, which leads to significant per-step computation overhead (more than two gradient computations), most likely because of the lack of the clipping mechanism that safeguards against inaccurate and changing Hessian. In general, previous second-order optimizers do not achieve a speed-up on large language models in wall-clock time or total compute (Gupta et al., 2018; Yao et al., 2021) (see more discussions in Section C).

2.2. Diagonal Hessian Estimators

We introduce two diagonal Hessian estimators, both of which have memory and run-time costs similar to computing a gradient (up to constant factors).

Option 1: Hutchinson's unbiased estimator. For any loss function $\ell(\theta)$ on parameters $\theta \in \mathbb{R}^d$, the Hutchinson's estimator (Hutchinson, 1989; Roosta-Khorasani & Ascher, 2015; Yao et al., 2021) first draws $u \in \mathbb{R}^d$ from the spherical Gaussian distribution $\mathcal{N}(0, \mathrm{I}_d)$, and then outputs $\hat{h} = u \odot (\nabla^2 \ell(\theta) u)$, where $\odot$ denotes the element-wise product, and $\nabla^2\ell(\theta)u$ is the product of the Hessian with the vector u. The Hutchinson's estimator is an unbiased estimator for the diagonal of the Hessian, because

$$\mathbb{E}[\hat{h}] = \text{diag}(\nabla^2 \ell(\theta))\,. \tag{2}$$

The estimator only requires a Hessian-vector product (i.e., $\nabla^2\ell(\theta)u$), which have efficient implementations in PyTorch and JAX, instead of the full Hessian matrix.

Option 2: Gauss-Newton-Bartlett (GNB) estimator. We leverage the structure of the loss to design a biased stochastic estimator for the diagonal Hessian, following Schraudolph (2002); Martens (2020); Wei et al. (2020). Suppose $\ell(\theta,(x,y))$ is a loss function on an example (x,y) of the form $\ell(\theta,(x,y)) = \ell_{\text{ce}}(f(\theta,x),y)$ where ℓ_{ce} is the cross-entropy loss and $f(\theta,x) \in \mathbb{R}^V$ is the logits, and V is the number of items/classes in a multi-class classification problem (e.g., the vocabulary size in LLMs). First, the Hessian of $\ell(\theta,(x,y))$ (w.r.t to variable θ) has the well-known Gauss-Newton decomposition (Ortega & Rheinboldt, 2000; Schraudolph, 2002) (which is a simple consequence of the chain rule),

$$\nabla^2_\theta\, \ell(\theta) = J_\theta f(\theta,x) S J_\theta f(\theta,x)^\top + J_{\theta\theta} f(\theta,x)[q] \tag{3}$$

where $J_\theta f(\theta,x)$ is the Jacobian of f w.r.t to θ viewed as a matrix in $\mathbb{R}^{d\times V}$, $S = \frac{\partial^2 \ell_{\text{ce}}(t,y)}{\partial t^2}\Big|_{t=f(\theta,x)} \in \mathbb{R}^{V\times V}$ is the second-order derivatives of the loss w.r.t to the logits, $q = \frac{\partial \ell_{\text{ce}}(t,y)}{\partial t}\Big|_{t=f(\theta,x)} \in \mathbb{R}^V$ is the first-order derivatives of the loss w.r.t to the logits, and $J_{\theta\theta} f(\theta,x)$ is the second-order derivatives of the multi-variate function $f(\theta,x)$ w.r.t θ, viewed as a linear map from $\mathbb{R}^V$ to $\mathbb{R}^{d\times d}$, where d is the dimension of the parameter θ.

In the context of neural networks, past works have found that the second term $J_{\theta\theta} f(\theta,x)[q]$ in Equation 3 is often relative smaller than the first term $J_\theta f(\theta,x) \cdot S \cdot$

$J_\theta f(\theta,x)^\top$ (Sankar et al., 2021), which is often referred to as the Gauss-Newton matrix (Dennis Jr & Schnabel, 1996; Ortega & Rheinboldt, 2000; Schraudolph, 2002; Chen, 2011) and used as pre-conditioners in second-order optimizers (Botev et al., 2017; Martens, 2020; Gargiani et al., 2020). Following this line of work, we build an unbiased estimator for *the diagonal* of the Gauss-Newton matrix, which is a biased estimator for the diagonal of the Hessian.

We first claim that S only depends $f(\theta,x)$ but not y, even though the loss depends on y.[1] Thus, $S = \frac{\partial^2 \ell_{\text{ce}}(t,\hat{y})}{\partial t^2}\Big|_{t=f(\theta,x)}$ for any $\hat{y}\in\{1,\dots,V\}$, which implies that $S = \mathbb{E}_{\hat{y}\sim p(\theta,x)}\left[\frac{\partial^2 \ell_{\text{ce}}(t,\hat{y})}{\partial t^2}\Big|_{t=f(\theta,x)}\right]$. Because $\ell_{\text{ce}}(t,y)$ is the negative log-probability of the probabilistic model defined by the categorical distribution $\text{Cat}(t)$ with parameter t, by Bartlett's second identity (Bartlett, 1953), we have that $S = \mathbb{E}_{\hat{y}\sim\text{Cat}(t)}\left[\frac{\partial^2 \ell_{\text{ce}}(t,\hat{y})}{\partial t^2}\right] = \mathbb{E}_{\hat{y}\sim\text{Cat}(t)}\left[\frac{\partial \ell_{\text{ce}}(t,\hat{y})}{\partial t}\frac{\partial \ell_{\text{ce}}(t,\hat{y})}{\partial t}^\top\right]$, where the first equality holds for $t=f(\theta,x)$ and the second equality holds for all t by Bartlett's second identity. Therefore, the Gauss-Newton matrix satisfies

$$\begin{aligned}
&J_\theta f(\theta,x)\cdot S\cdot J_\theta f(\theta,x)^\top \\
&= \mathop{\mathbb{E}}_{\hat{y}\sim\text{Cat}(t)}\left[J_\theta f(\theta,x)\frac{\partial \ell_{\text{ce}}(t,\hat{y})}{\partial t}\frac{\partial \ell_{\text{ce}}(t,\hat{y})}{\partial t}^\top J_\theta f(\theta,x)^\top\right] \\
&= \mathop{\mathbb{E}}_{\hat{y}\sim\text{Cat}(t)}\left[\nabla_\theta \ell_{\text{ce}}(f(\theta,x),\hat{y})\nabla_\theta \ell_{\text{ce}}(f(\theta,x),\hat{y})^\top\right], \qquad (4)
\end{aligned}$$

which implies that $\text{diag}(J_\theta f(\theta,x)\cdot S\cdot J_\theta f(\theta,x)^\top) = \mathbb{E}_{\hat{y}\sim\text{Cat}(t)}\left[\nabla_\theta \ell_{\text{ce}}(f(\theta,x),\hat{y})\odot\nabla_\theta \ell_{\text{ce}}(f(\theta,x),\hat{y})\right]$. Hence, the quantity $\ell_{\text{ce}}(f(\theta,x),\hat{y})\odot\nabla_\theta \ell_{\text{ce}}(f(\theta,x),\hat{y})$ is an unbiased estimator of the Gauss-Newton matrix for the Hessian of a one-example loss $\ell(f(\theta,x),y)$.

Mini-batch version. Given a mini-batch of inputs $\{(x_b,y_b)\}_{b=1}^B$. The most natural way to build an estimator for the diagonal of the Gauss-Newton matrix for the Hessian of the mini-batch loss is using

$$\frac{1}{B}\sum_{b=1}^{B}\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)\odot\nabla_\theta\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b), \qquad (5)$$

where $\hat{y}_b$'s are labels sampled from the model on inputs x_b's respectively. However, as noted by Grosse (2022), implementing this estimator is inconvenient under the current auto-differentiation frameworks, where the users only have access to the average gradient over a mini-batch (as opposed to the individual ones). Fortunately, by the Bartlett's first identity (Bartlett, 1953) (which generally holds for the negative log-likelihood loss of any probabilistic model), we have:

$$\forall b,\ \ \mathbb{E}_{\hat{y}_b}\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b) = 0\,. \qquad (6)$$

Let $\widehat{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)$ be the mini-batch loss on the *sampled* labels. Observing that $\hat{y}_b$'s are independent with each other, we have

$$\begin{aligned}
&\mathbb{E}_{\hat{y}_b' s}\left[B\cdot\nabla_\theta\widehat{L}(\theta)\odot\nabla_\theta\widehat{L}(\theta)\right] \qquad (7)\\
=&\mathbb{E}_{\hat{y}_b' s}\left[\frac{1}{B}\sum_{b=1}^{B}\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)\odot\sum_{b=1}^{B}\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)\right]\\
=&\mathbb{E}_{\hat{y}_b' s}\left[\frac{1}{B}\sum_{b=1}^{B}\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)\odot\nabla\ell_{\text{ce}}(f(\theta,x_b),\hat{y}_b)\right].
\end{aligned}$$

The RHS of Equation 7 is the same as the expectation of Equation 5, which, by Equation 4, also equals to the diagonal of the Gauss-Newton matrix for the mini-batch loss. Hence, we use $B\cdot\nabla_\theta\widehat{L}(\theta)\odot\nabla_\theta\widehat{L}(\theta)$ as the estimator.

To the best of our knowledge, this estimator of Gauss-Newton matrix was first used in (Wei et al., 2020). Given the use Bartlett's identities that are central to the estimator, we call it Gauss-Newton-Bartlett (GNB) estimator.

Comparisons of Hessian estimators. The Hutchinson's estimator does not assume any structure of the loss, but requires a Hessian-vector product. The GNB estimator only estimates the Gauss-Newton term but always gives a positive semi-definite (non-negative) diagonal Hessian estimate. The PSDness ensures that the pre-conditioned update is always a descent direction (Dennis Jr & Schnabel, 1996). The GNB estimator can also be easily extended to the negative log-likelihood loss of any exponential family distribution, and be adapted to estimating the trace of the Gauss-Newton matrix as in Wei et al. (2020) or efficiently implementing the product of Gauss-Newton matrix with a vector. The authors suspect the GNB estimator has a smaller variance than the Hutchinson's estimator, but more empirical and theoretical investigation are needed to support the hypothesis.

3. Experiments

We call the algorithm using the Hutchinson's estimator and the GNB estimator Sophia-H and Sophia-G, respectively. We evaluate Sophia on auto-regressive language modeling with GPT-2 (Radford et al., 2019) of model sizes ranging from 125M to 770M.

3.1. Experimental Setup

Language modeling. We train autoregressive models on OpenWebText (Gokaslan & Cohen, 2019). Following the

[1] Denote by $p(\theta,x) = \text{softmax}(f(\theta,x)) \in \mathbb{R}^V$ the probability vector obtained by applying softmax on the logits. Indeed, a simple derivation shows that $S = \text{diagonal}(p(\theta,x)) - p(\theta,x)p(\theta,x)^\top$, where $\text{diagonal}(p(\theta,x))$ is the matrix with the vector $p(\theta,x)$ residing on the diagonal.

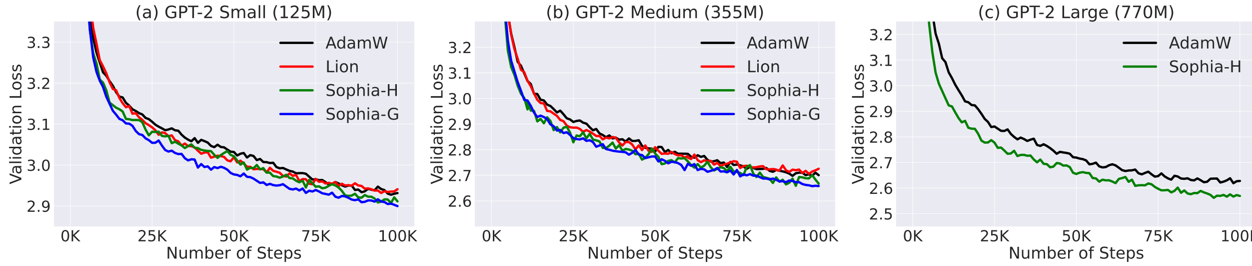

Figure 2. Validation loss on OpenWebText with 100K steps. (a) GPT-2 Small (125M). Adam: 2.921, Lion: 2.924, Sophia-H: 2.901, Sophia-G: 2.895 (b) GPT-2 Medium (355M). Adam: 2.691, Lion: 2.678, Sophia-H: 2.645, Sophia-G: 2.653. (c) GPT-2 Large (770M). Adam: 2.613, Sophia-H: 2.559.

standard protocol of GPT-2 (Radford et al., 2019), we set the context length to 1024. We use decoder-only Transformers (Vaswani et al., 2017) with 125M (small), 355M (medium), and 770M (large) parameters. Detailed model configurations are deferred to Section F.2.

Baselines. We mainly compare Sophia with Adam with decoupled weight decay (AdamW) (Loshchilov & Hutter, 2017) which is the dominantly used optimizer on language modeling tasks, and Lion (Chen et al., 2023), which is an first-order adaptive optimizer discovered by symbolic search. All optimizers are well-tuned. The hyperparameters of AdamW on GPT-2 are already well-established in the literature (Radford et al., 2019; Karamcheti et al., 2021). The weight decay is set to 0.1. We use $\beta_1 = 0.9$ and $\beta_2 = 0.95$. For Lion, we use $\beta_1 = 0.95$ and $\beta_2 = 0.98$ following Chen et al. (2023). Although Chen et al. (2023) suggests using 0.1 times the learning rate (LR) of AdamW for vision tasks, we find out the LR should be larger on LMs by a grid search. The LR of Sophia-H is set to the LR of AdamW $/\rho$ (Section F.1).

Implementation. We set batch size to 480, and use cosine LR schedule with the final LR equal to 0.05 times the peak LR, following Rae et al. (2021). We use the standard gradient clipping (by norm) threshold 1.0. We adopt a fixed 2k steps of LR warm-up. For Sophia, we use $\beta_1 = 0.96$, $\beta_2 = 0.99$, $\epsilon =$1e-12 and update diagonal Hessian every 10 steps. For Sophia-H, we use $\rho = 0.01$, and only a subset of 32 examples from the mini-batch to calculate the diagonal Hessian to further reduce overhead. For Sophia-G, we use $\rho = 0.04$, and use a subset of 240 examples from the mini-batch to calculate the diagonal Gauss-Newton. We implement the algorithms in PyTorch (Paszke et al., 2019) and train all the models in bfloat16. The 125M and 355M models are trained on A5000 GPUs, while the 770M models are trained on A100 GPUs.

Evaluation. We pre-train the models with each optimizer for 100K, 200K, or 400K steps to compare the speed. Note that, as is standard, the LR schedule depends on the total pre-specified target number of steps, as shown in Figure 3 (a). This makes the loss curve of the same optimizer different for various total numbers of steps because the LR schedule with fewer total steps decays the LR earlier. For example, the 200K runs in Figure 3 (b) are not the continuation of the 100K runs. We primarily evaluate the models with their log perplexity on OpenWebText.

3.2. Results

Figure 2 illustrates the validation loss curve (token-level log perplexity) on OpenWebText with the same number of steps (100K). Our method consistently achieves better validation loss than AdamW and Lion. As the model size grows, the gap between Sophia and baselines also becomes larger. Sophia-H and Sophia-G both achieve a 0.04 smaller validation loss on the 355M model (Figure 2 (b)). Sophia-H achieves a 0.05 smaller validation loss on the 770M model (Figure 2, (c)), with the same 100K steps. This is a significant improvement since according to scaling laws in this regime (Kaplan et al., 2020) and results in Figure 3, a improvement in loss of 0.05 is equivalent to 2x improvement in terms of number of steps or total compute to achieve the same validation loss.

Sophia is 2x faster in terms of number of steps, total compute and wall-clock time. The improvement in validation loss brought by Sophia can be translated into reduction of number of steps or total compute. In Figure 1 (a)&(b) and Figure 3, we evaluate the optimizers by comparing the number of steps or total compute needed to achieve *the same validation loss level*. As can be observed in Figure 1 (a)&(b), Sophia-H and Sophia-G achieve a 2x speedup compared with AdamW and Lion across different model sizes.

The scaling law is in favor of Sophia-H over AdamW. In Figure 1 (c), we plot the validation loss of models of different sizes pre-trained for 100K steps. The gap between Sophia and AdamW grows as we scale up the models. Moreover, the 540M model trained by Sophia-H has smaller loss than the 770M model trained by AdamW. The 355M model trained by Sophia-H has comparable loss as the 540M model trained by AdamW.

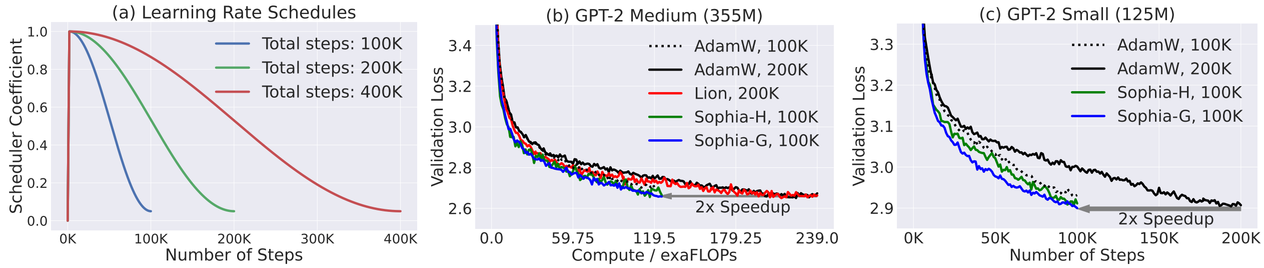

Figure 3. Comparison of numbers of steps to reach the same validation loss on OpenWebText. (a) Learning rate schedules. (b) GPT2-medium (355M). GPT2-large (770M) results are in Figure 1(a). Across all model sizes, Sophia achieve a 2x speedup over AdamW in terms of the number of steps.

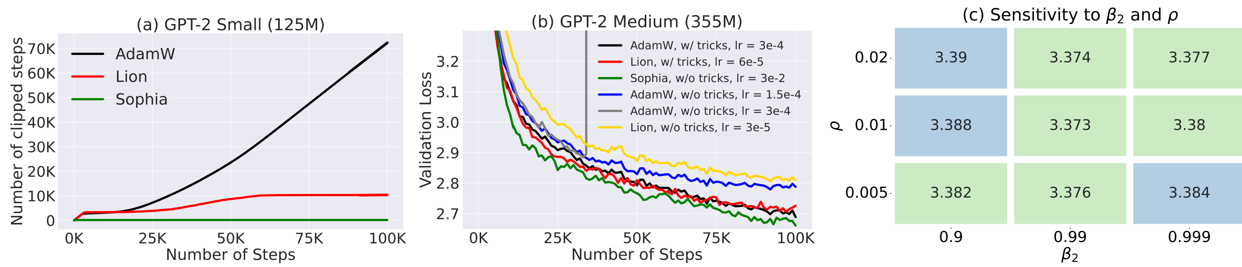

Figure 4. Sophia improves pre-training stability and is insensitive to hyperparameters. (a) With AdamW and Lion, gradient clipping is triggered frequently. With Sophia, gradient clipping rarely happens. (b) AdamW and Lion require the trick of re-parameterizing the attention with a temperature that is the inverse of the layer index (Karamcheti et al., 2021). The plot shows the largest LR that AdamW and Lion without the trick can use to be stable, which is much smaller than with the trick. In contrast, Sophia does not need this trick. (c) Sophia is not sensitive to hyperparameter choice.

Table 1. Wall-clock time and compute.

Algorithm	Model Size	T(step)	T(Hessian)	Compute
AdamW	770M	3.25s	–	2550
Sophia-H	770M	3.40s	0.12s	2708
Sophia-G	770M	3.42s	0.17s	2678
AdamW	355M	1.77s	–	1195
Sophia-H	355M	1.88s	0.09s	1249
Sophia-G	355M	1.86s	0.09s	1255

3.3. Analysis

Comparison of wall-clock time and amount of compute. We compare the total compute (TFLOPs) per step and the wall-clock time on A100 GPUs in Table 1. We report the average time per step (T(step)), the time spent in Hessian computation (T(Hessian)) and the total compute following Chowdhery et al. (2022). Since we calculate the diagonal Hessian estimate with a reduced batch size every 10 steps, the computation of the Hessian accounts for 6% of the total compute, and the overall wall-clock time overhead is less than 5% compared with AdamW. In terms of memory usage, our optimizer has two states, m and h, which results in the same memory cost as AdamW.

Sensitivity to ρ and β_2, and transferability of hyperparameters. On a 30M model, we perform a grid search to test the sensitivity of Sophia-H to hyperparamters (Figure 4 (c)). All combinations have a similar performance, while $\beta_2 = 0.99$ and $\rho = 0.01$ performs the best. The hyperparameter choice is transferable across model sizes.

Training Stability. Sophia-H has better stability in pre-training compared to AdamW and Lion. Gradient clipping (by norm) is an important technique in language model pre-training as it avoids messing up the moment of gradients with one mini-batch gradient computed from rare data (Zhang et al., 2020). In practice, the frequency that gradients clipping is triggered is related to the training stability—if the gradient is frequently clipped, the iterate can be at a very instable state. We compare the proportion of steps where gradient clipping is triggered on GPT-2 small (125M) in Figure 4 (a). Although all methods use the same clipping threshold 1.0, Sophia-H seldomly triggers gradient clipping, while AdamW and Lion trigger gradient clipping in more than 10% of the steps.

A common trick of pre-training deep Transformers is scaling the product of keys and values by the inverse of the layer index as implemented by Mistral (Karamcheti et al., 2021) and Huggingface (Wolf et al., 2020). This stabilizes training and increases the largest possible learning rate. Without this trick, the maximum learning rate of AdamW and Lion on GPT-2 (355M) can only be 1.5e-4, which is much smaller than 3e-4 with the trick (the loss will blow up with 3e-4 without the trick). Moreover, the loss decreases much slower without the trick as shown in Figure 4 (b). In all the experiments, Sophia-H does not require scaling the product of keys and values by the inverse of the layer index.

References

Anil, R., Gupta, V., Koren, T., and Singer, Y. Memory efficient adaptive optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Anil, R., Gupta, V., Koren, T., Regan, K., and Singer, Y. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Ba, J., Grosse, R., and Martens, J. Distributed second-order optimization using kronecker-factored approximations. In *International Conference on Learning Representations*, 2017.

Balles, L. and Hennig, P. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pp. 404–413. PMLR, 2018.

Bartlett, M. Approximate confidence intervals. *Biometrika*, 40(1/2):12–19, 1953.

Becker, S. and Le Cun, Y. Improving the convergence of back-propagation learning with. 1988.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Botev, A., Ritter, H., and Barber, D. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pp. 557–565. PMLR, 2017.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Braun, H. and Riedmiller, M. Rprop: a fast adaptive learning algorithm. In *Proceedings of the International Symposium on Computer and Information Science VII*, 1992.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Broyden, C. G. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

Chapelle, O., Erhan, D., et al. Improved preconditioner for hessian free optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 201. Citeseer, 2011.

Chen, P. Hessian matrix vs. gauss–newton hessian matrix. *SIAM Journal on Numerical Analysis*, 49(4):1417–1435, 2011.

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., et al. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Conn, A. R., Gould, N., and Toint, P. L. Trust-region methods, siam. *MPS, Philadelphia*, 2000.

Crawshaw, M., Liu, M., Orabona, F., Zhang, W., and Zhuang, Z. Robustness to unbounded smoothness of generalized signsgd. *arXiv preprint arXiv:2208.11195*, 2022.

Dennis Jr, J. E. and Schnabel, R. B. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dozat, T. Incorporating nesterov momentum into adam. 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Gargiani, M., Zanelli, A., Diehl, M., and Hutter, F. On the promise of the stochastic generalized gauss-newton method for training dnns. *arXiv preprint arXiv:2006.02409*, 2020.

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31, 2018.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Gokaslan, A. and Cohen, V. Openwebtext corpus, 2019.

Grosse, R. *Neural Network Training Dynamics*. 2022.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2016.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Izsak, P., Berchansky, M., and Levy, O. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karamcheti, S., Orr, L., Bolton, J., Zhang, T., Goel, K., Narayan, A., Bommasani, R., Narayanan, D., Hashimoto, T., Jurafsky, D., Manning, C. D., Potts, C., Ré, C., and Liang, P. Mistral – a journey towards reproducible language model training. https://crfm.stanford.edu/2021/08/26/mistral.html, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kunstner, F., Chen, J., Lavington, J. W., and Schmidt, M. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. *arXiv preprint arXiv:2304.13960*, 2023.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Mai, V. V. and Johansson, M. Stability and convergence of stochastic gradient clipping: Beyond lipschitz continuity and smoothness. In *International Conference on Machine Learning*, pp. 7325–7335. PMLR, 2021.

Martens, J. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Martens, J., Ba, J., and Johnson, M. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Martens, J. et al. Deep learning via hessian-free optimization. In *ICML*, volume 27, pp. 735–742, 2010.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

OpenAI. Gpt-4 technical report. *arXiv*, 2023.

Ortega, J. M. and Rheinboldt, W. C. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

Pascanu, R. and Bengio, Y. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance pdf.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21: 1–67, 2020.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Roosta-Khorasani, F. and Ascher, U. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

Sagun, L., Bottou, L., and LeCun, Y. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Sankar, A. R., Khasbage, Y., Vigneswaran, R., and Balasubramanian, V. N. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9481–9488, 2021.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates. In *International conference on machine learning*, pp. 343–351. PMLR, 2013.

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

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Wei, C., Kakade, S., and Ma, T. The implicit and explicit regularization effects of dropout. *arXiv preprint arXiv:2002.12915*, 2020.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pp. 581–590. IEEE, 2020.

Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., and Mahoney, M. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S. C., Dvornek, N., Papademetris, X., and Duncan, J. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

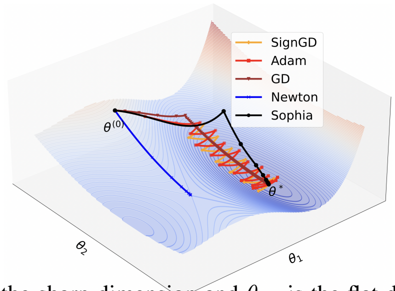

Figure 5. The motivating toy example. $\theta_{[1]}$ is the sharp dimension and $\theta_{[2]}$ is the flat dimension. GD's learning rate is limited by the sharpness in θ_1, and makes slow progress along $\theta_{[2]}$. Adam and SignGD bounce along $\theta_{[1]}$ while making slow progress along $\theta_{[2]}$. Vanilla Newton's method converges to a saddle point. Sophia makes fast progress in both dimensions and converges to the minimum with a few steps.

A. Motivations

Heterogeneous curvatures. The loss functions of modern deep learning problems often have different curvatures across different parameter dimensions (Sagun et al., 2016; Ghorbani et al., 2019; Zhang et al., 2020; Yao et al., 2020). E.g., on a 125M-parameter GPT-2 model, Figure 6 shows that the distribution of positive diagonal entries of the Hessian is dispersed.

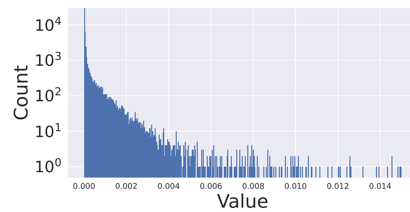

Figure 6. Histogram of positive entries of the diagonal Hessian of a 125M-parameter GPT-2.

We demonstrate the limitations of Adam and GD on heterogeneous landscapes by considering a two-dimensional loss function $L(\theta_{[1]}, \theta_{[2]}) = L_1(\theta_{[1]}) + L_2(\theta_{[2]})$ where L_1 is much sharper than L_2. We plot the loss landscape of $L(\theta_{[1]}, \theta_{[2]})$ in Figure 5.[2] For simplicity, we discuss GD and deterministic versions of Adam. Recall that GD's update in this setting is:

$$\theta_{[1]} \leftarrow \theta_{[1]} - \eta \cdot L_1'(\theta_{[1]}) \text{ and } \theta_{[2]} \leftarrow \theta_{[2]} - \eta \cdot L_2'(\theta_{[2]})\,. \tag{8}$$

A common simplification of Adam that is more amenable to analysis (Balles & Hennig, 2018; Bernstein et al., 2018; Zhuang et al., 2020; Kunstner et al., 2023) is SignGD, which dates back to RProp (Braun & Riedmiller, 1992) that motivated RMSProp (Hinton et al., 2012) and Adam. Observe that without using the EMA (for both the gradient and second moments of the gradient), Adam's update is simplified to $\eta \cdot \nabla L(\theta)/|\nabla L(\theta)| = \eta \cdot \text{sign}(\nabla L(\theta))$ (where all operations are entry-wise), which is called SignGD. Applying the update rule to our setting gives:

$$\theta_{[1]} \leftarrow \theta_{[1]} - \eta \cdot \text{sign}(L_1'(\theta_{[1]})) \text{ and } \theta_{[2]} \leftarrow \theta_{[2]} - \eta \cdot \text{sign}(L_2'(\theta_{[2]}))\,. \tag{9}$$

Limitations of GD and SignGD (Adam). It is well known that the optimal learning rate of GD should be proportional to the inverse of the curvature, that is, the Hessian/second derivative at the local minimum. More precisely, let h_1 and h_2 be the curvatures of L_1 and L_2 at the local minimum (and thus $h_1 \gg h_2$). The optimal learning rate for the update of $\theta_{[1]}$ in equation (8) is $\asymp 1/h_1$, which is much smaller than the optimal learning rate that the update of $\theta_{[2]}$ needs, which is $\asymp 1/h_2$. As a result, the largest shared learning rate can only be $1/h_1$; consequently, the convergence in $\theta_{[2]}$ dimension is slow as demonstrated in the brown curve in Figure 5.

The update size of SignGD is the learning rate η in all dimensions. The same update size translates to less progress in decreasing the loss in the flat direction than in the sharp direction. As observed from the yellow curve in Figure 5, the progress of SignGD in the flat dimension $\theta_{[2]}$ is slow because each step only decreases the loss $L_2(\theta_{[2]})$ slightly. On the other hand, along the direction $\theta_{[1]}$, the iterate quickly travels to the valley in the first three steps and then starts to bounce. To fully converge in the sharp dimension, the learning rate η needs to decay to 0, which will exacerbate the slow convergence in the flat dimension $\theta_{[2]}$. The trajectory of Adam in this example is indeed similar to SignGD, which is also plotted as the red curve in Figure 5.

The behavior of SignGD and Adam above indicates that a more aggressive pre-conditioning is needed—sharp dimensions should have relatively smaller updates than flat dimensions so that the decrease of loss is equalized in all dimensions. As

[2] Concretely, in Figure 5, $L_1(\theta_{[1]}) = 8(\theta_{[1]} - 1)^2(1.3\theta_{[1]}^2 + 2\theta_{[1]} + 1)$ and $L_2(\theta_{[2]}) = 1/2(\theta_{[2]} - 4)^2$.

suggested by well-established literature on second-order optimization (Boyd & Vandenberghe, 2004) for convex functions, the optimal pre-conditioner should be the Hessian, which captures the curvature on each dimension; as in Newton's method, the update is the gradient divided by the Hessian in each dimension:

$$\theta_{[1]} \leftarrow \theta_{[1]} - \eta \cdot L'_1(\theta_{[1]})/h_1 \ \text{ and } \ \theta_{[2]} \leftarrow \theta_{[2]} - \eta \cdot L'_2(\theta_{[2]})/h_2 \,. \tag{10}$$

Limitations of Newton's method. Nevertheless, Newton's method has known limitations as well. For non-convex functions, vanilla Newton's method could converge to a global maximum when the local curvature is negative. In the blue curve of Figure 5, Newton's method quickly converges to a saddle point instead of a local minimum. The curvature might also change rapidly along the trajectory, making the second-order information unreliable. To address these limitations, we propose considering only pre-conditioners that capture positive curvature, and introduce a pre-coordinate clipping mechanism to mitigate the rapid change of Hessian (more detail in Section 2.1). Applying our algorithm on the toy case results in the following update:

$$\theta_{[1]} \leftarrow \theta_{[1]} - \eta \cdot \text{clip}(L'_1(\theta_{[1]})/\max\{h_1,\epsilon\}, \rho) \text{ and } \theta_{[2]} \leftarrow \theta_{[2]} - \eta \cdot \text{clip}(L'_2(\theta_{[2]})/\max\{h_2,\epsilon\}, \rho)\,, \tag{11}$$

where ρ is a constant to control the worst-case update size, ϵ is a very small constant (e.g., 1e-12), which avoids dividing by 0. When the curvature of some dimension is rapidly changing or negative and thus the second-order information is misleading and possibly leads to a huge update before clipping, the clipping mechanism kicks in and the optimizer defaults to SignGD (even though this is sub-optimal for benign situations). Numerous prior methods such as trust region (Conn et al., 2000), backtracking line search (Boyd & Vandenberghe, 2004), and cubic regularization (Nesterov & Polyak, 2006) also tackle the same issue of Newton's method, but the clipping mechanism is much simpler and more efficient.

As shown in the black curve in Fig. 5, the update in equation (11) starts off similarly to SignGD due to the clipping mechanism in the non-convex region, making descent opposed to converging to a local maximum. Then, in the convex valley, it converges to the global minimum with a few steps. Compared with SignGD and Adam, it makes much faster progress in the flat dimension $\theta_{[2]}$ (because the update is bigger in dimension $\theta_{[2]}$), while avoiding boucing in the sharp dimension $\theta_{[1]}$ (because the update is significantly shrunk in the sharp dimension $\theta_{[1]}$).

B. Theoretical Analysis

This section provides runtime bounds for the deterministic version of Sophia that does not depend on the local condition number (the ratio between maximum and minimum curvature at the local minimum) and the worst-case curvature (that is, the smoothness parameter), demonstrating the advantage of Sophia in adapting to heterogeneous curvatures across parameter dimensions.

We start with standard assumptions on the differentiability and uniqueness of the minimizer.

Assumption B.1. *$L : \mathbb{R}^d \to \mathbb{R}$ is a twice continuously differentiable, strictly convex function with θ^* being its minimizer. For convenience, we denote $\lambda_{\min}(\nabla^2 L(\theta^*))$ by μ.*

The following assumptions state that the Hessian has a certain form of continuity—within a neighborhood of size R, the ratio between the Hessians, $\nabla^2 L(\theta')^{-1}\nabla^2 L(\theta)$, is assumed to be bounded by a constant 2.

Assumption B.2. *There exists a constant $R > 0$, such that*

$$\forall \theta, \theta' \in \mathbb{R}^d, \|\theta - \theta'\|_2 \le R \implies \left\|\nabla^2 L(\theta')^{-1}\nabla^2 L(\theta)\right\|_2 \le 2 \tag{12}$$

We analyze the convergence rate of the deterministic version of the Sophia on convex functions,

$$\theta_{t+1} = \theta_t - \eta V_t^\top \text{clip}(V_t(\nabla^2 L(\theta_t))^{-1}\nabla L(\theta_t), \rho), \tag{13}$$

where $\nabla^2 L(\theta_t) = V_t^\top \Sigma_t V_t$ is an eigendecomposition of $\nabla^2 L(\theta_t)$. Here, we use the full Hessian as the pre-conditioner because the diagonal Hessian pre-conditioner cannot always work for general functions which may not have any alignment with the natural coordinate system. Moreover, the matrix V_t transforms $(\nabla^2 L(\theta_t))^{-1}\nabla L(\theta_t)$ into eigenspace and thus the clipping can be done element-wise in the eigenspace. We do not need the max between Hessian and ϵ in the original version of Sophia because the Hessian is always PSD for convex functions. Finally, the matrix $V_t^\top$ transforms the update back to the original coordinate system for the parameter update.

Theorem B.3. *Under Assumption B.1 and Assumption B.2, let $\eta = 1/2, \rho = \frac{R}{2\sqrt{d}}$, the update in Equation 13 reaches a loss at most ϵ in $T \lesssim d \cdot \frac{L(\theta_0)-\min L}{\mu R^2} + \ln \frac{\mu R^2}{32d\epsilon}$ steps.*

The first term in the runtime bound is a burn-in time before reaching a local region, where the error decays exponentially fast so that the runtime bound is logarithmic in $1/\epsilon$ as the second term in the runtime bound shows. We remark that the bound does not depend on the condition number (the ratio between the maximum and minimum eigenvalue of Hessian), as opposed to the typical dependency on the maximum eigenvalue of the Hessian (or the smoothness parameter) in standard analysis of gradient descent in convex optimization (Boyd & Vandenberghe, 2004). Moreover, even on simple quadratic functions, the convergence rate of simplified Adam (SignGD) depends on the condition number (Appendix I.1). This demonstrates the advantage of Sophia in adapting to heterogeneous curvatures across parameter dimensions.

C. Related work

Stochastic Adaptive First-order Optimizers in Deep Learning. The idea of adaptive first-order optimizers dates back to RProp (Braun & Riedmiller, 1992). AdaGrad (Duchi et al., 2011) adapted the learning rate of features by estimated geometry and assign larger learning rate to infrequent features. RMSProp (Hinton et al., 2012) generalized RProp and is capable to work with smaller batch sizes. Adam (Kingma & Ba, 2014) improved RMSProp by introducing a running average of gradients, and has so far become the dominant approach to solve optimization problems in deep learning, especially for training Transformers (Vaswani et al., 2017). Many follow-up works proposed variants of Adam (Dozat, 2016; Shazeer & Stern, 2018; Reddi et al., 2019; Loshchilov & Hutter, 2017; Zhuang et al., 2020; You et al., 2019). Chen et al. (2023) performed a search over adaptive first-order algorithms and discovered Lion, which is a improved version of sign momentum SGD.

Second-order Optimizers in Deep Learning. Second-order optimizers are believed to have the potential to outperform adaptive first-order optimizers. Classical second-order optimization algorithms pre-condition the gradient with curvature information (Broyden, 1970; Nesterov & Polyak, 2006; Conn et al., 2000). Over the years, people have developed numerous ways to adapt these methods to deep learning. To the best of our knowledge, Becker & Le Cun (1988) was the first to use diagonal Hessian as the pre-conditioner. Martens et al. (2010) approximated the Hessian with conjugate gradient. Schaul et al. (2013) automatically tuned learning rate of SGD by considering diagonal Hessian. Pascanu & Bengio (2013) considered Gaussian Newton's approximation of Hessian and Fisher information matrix. Martens & Grosse (2015) and follow-up works (Ba et al., 2017; George et al., 2018; Martens et al., 2018) proposed to approximate the Hessian based on the structure of neural networks. Yao et al. (2021) proposed to use the square root of the EMA of squared Hessian as the pre-conditioner. Despite these progress, the de facto optimization algorithms in modern large models are Adam and its variants. Especially previous second-order optimizers have the following limidations: (1) they have fundamental computational / memory overhead due to frequent Hessian computation, therefore they cannot achieve improvement in wall-clock time (Martens & Grosse, 2015; Gupta et al., 2018) (2) they are difficult to implement and scale up. None of them can work on the scale of GPT-2 (3) they depend heavily on specific model architecture or hardware structures, e.g., Anil et al. (2020) offloads hessian computation to CPUs; George et al. (2018) needs ResNets and very large batch size to approximate the Fisher information matrix.

Gradient Clipping. Global gradient clipping has been a standard practice in pre-training language models (Merity et al., 2017; Radford et al., 2019; Izsak et al., 2021; Zhang et al., 2022). It helps stabilizes training and avoids the effect of rare examples and large gradient noise. Zhang et al. (2019); Mai & Johansson (2021) showed that global gradient clipping is faster than standard SGD when global smoothness does not hold. Zhang et al. (2020); Crawshaw et al. (2022) found out per-coordinate gradient clipping can function as adaptivity. In addition to gradient clipping, Sophia is the first to clip the update in second-order methods to avoid the effect of Hessian's changing along the trajectory and the inaccuracy of Hessian approximation.

Optimization Algorithms in LM Pre-training. Adam (Kingma & Ba, 2014) (with decoupled weight decay (Loshchilov & Hutter, 2017)) has become the dominant approach for language model pre-training (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020; Zhang et al., 2022; Touvron et al., 2023). Different from vision tasks with CNNs (He et al., 2016) where models trained with SGD generalize better than models trained with Adam, Adam outperforms SGD by a huge margin on language modeling tasks with Transformers (Anil et al., 2019; Liu et al., 2020; Kunstner et al., 2023). Raffel et al. (2020); Chowdhery et al. (2022) trained Transformers with AdaFactor (Shazeer & Stern, 2018), which is a low rank version of Adam. You et al. (2019) proposed to make the update of Adam proportional to per-layer paramter norm to stably train LLMs.

D. Conclusion

We introduced Sophia, a scalable second-order optimizer for language model pre-training. Sophia converges in fewer steps than first-order adaptive methods, while maintaining almost the same per-step cost. On language modeling with GPT-2, Sophia achieves a 2x speed-up compared with AdamW in the number of steps, total compute, and wall-clock time.

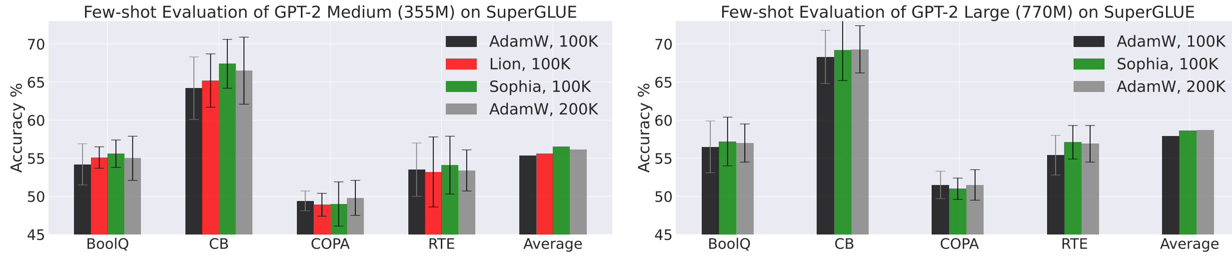

Figure 7. Few-shot evaluation on SuperGLUE. With the same 100K steps, models pre-trained with Sophia outperforms models pre-trained with AdamW and Lion on most tasks. Models pre-trained with Sophia for 100K steps have comparable performance as models pre-trained with AdamW for 200K steps.

E. Additional Experiment Results

Few-shot Evaluation on Downstream Tasks (SuperGLUE). As shown in Figure 7, as expected, the improvement in validation loss transfers to an improvement in downstream task accuracy. With the same number of steps in pre-training, GPT-2 medium and GPT-2 large pre-trained with Sophia have better few-shot accuracy on most subtasks. Also, models pre-trained with Sophia-H have comparable few-shot accuracy as models pre-trained with AdamW for 2x number of steps.

Dynamics of Sophia in training. We measure the ℓ_2 norm of the EMA of the diagonal Hessian h_t, and the proportion of parameters where clipping happens (that is, m_t/h_t is larger than ρ) during pre-training in Figure 8. After the initial stage, the norm of the Hessian steadily grows. The proportion of parameters where clipping happens approaches 60%, which corroborates the importance of per-coordinate clipping in the algorithm.

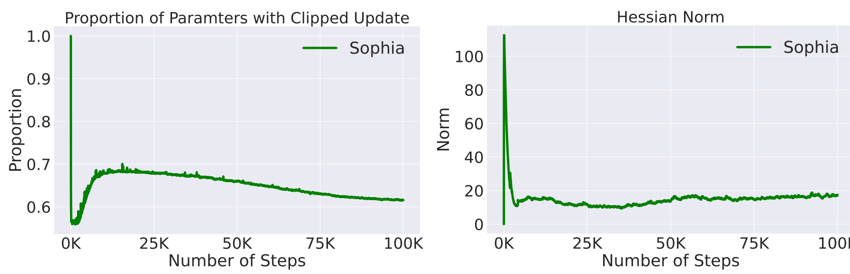

Figure 8. Visualization of training statistics. (a) The proportion of parameters whose update is clipped. (b) ℓ_2 norm of the EMA of Hessian h_t.

Results with different number of steps. Due to space limit, runs with different number of steps and their comparison are provided in Figure 9. Across different total number of steps, Sophia outperforms AdamW and Lion with a large margin as the main experiments we presented in Section 3.2.

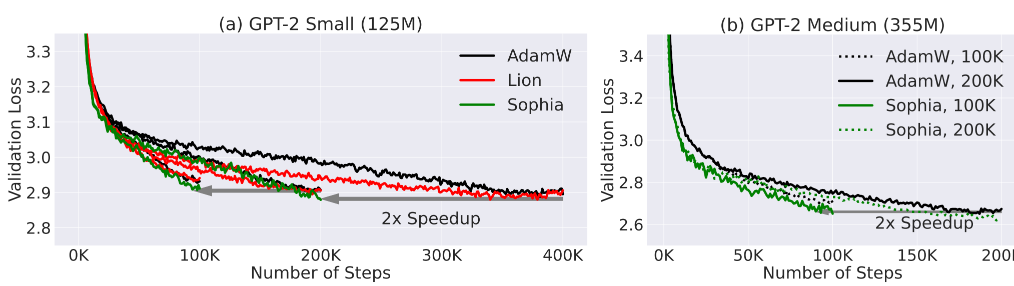

Figure 9. Results of training for different steps.

F. Additional Experiment Details

Table 2. Model Configurations and Peak Learning Rate.

Acronym	Size	d_model	n_head	depth	AdamW lr	Lion lr	Sophia-H lr	Sophia-G lr
–	30M	384	6	6	1e-3	3e-4	1e-1	–
Small	125M	768	12	12	6e-4	1.5e-4	6e-2	1e-5
Medium	355M	1024	16	24	3e-4	6e-5	3e-2	7.5e-6
–	540M	1152	18	30	2.5e-4	–	2.5e-2	–
Large	770M	1280	20	36	2e-4	–	2e-2	–

F.1. Hyperparamter Tuning

The hyperparameters for AdamW on GPT-2 are well-established. Most hyperparameters are used across all model sizes: ϵ =1e-6, $\beta_1 = 0.9$ $\beta_2 = 0.95$, and $\lambda = 0.1$ (weight decay). The gradient clipping (by norm) threshold is set to 1.0. The peak learning rate is different for different model sizes. Generally, larger models use smaller peak learning rate. For Lion, we use $\beta_1 = 0.95$ $\beta_2 = 0.98$ as suggested by Chen et al. (2023). We also set λ to 0.1 and gradient clipping (by norm) threshold to 1.0. However, the peak learning rate for Lion on language models is not established. The suggested 0.1 times peak learning rate of AdamW in vision tasks (Chen et al., 2023) is not optimal for language modeling. We perform a grid search of peak learning rate of Lion and provide the result in Table 2.

We use $\beta_1 = 0.96$, ϵ =1e-12 and $k = 10$ for Sophia-H. We first tune ρ and β_2 with grid search on a 30M model, and directly use ρ and β_2 from the 30M model on larger models. Details of this tuning is provided in Section 3.3. For Sophia-G we use $\rho = 20$ and $\beta_2 = 0.99$. We observe these hyperparameters choice work well across all model sizes. The peak learning rate of Sophia-H is set to 100 times the peak learning rate of AdamW ($1/\rho$ times the peak learning rate of AdamW). The peak learning rate for Sophia-G is also provided in Table 2.

F.2. Model and Implementation Details

We consider three sizes of GPT-2 corresponding to small, medium, and large in Radford et al. (2019). We also introduce a 30M model for efficient hyperparameter grid search and a 540M model for scaling law visualization. We provide the model specifications in Table 2. We use the nanoGPT (`https://github.com/karpathy/nanoGPT/`) code base. Following nanoGPT, we use GELU activations and disable bias and Dropout (Srivastava et al., 2014) during pre-training.

All models are trained on OpenWebText (Gokaslan & Cohen, 2019). The text is tokenized with the GPT-2 tokenizer (Radford et al., 2019). We use the train and validation split from nanoGPT. The training set contains 9B tokens, and the validation set contains 4.4M tokens.

We observed AdamW and Lion does not perform well on 355M and 770M standard transformers. The iterates become unstable when the learning rate is close to the choice of Radford et al. (2019). We introduce scaling attention by the inverse of layer index to address this issue following Karamcheti et al. (2021); Wolf et al. (2020). Note that Sophia does not need this trick as mentioned in Section 3.3.

We use distributed data parallel with gradient accumulation to enable a batch size of 480. All models are trained with bfloat16. The 125M and 355M models are trained on machines with 10 A5000 GPUs, while the 770M models are trained on an AWS p4d.24xlarge instance with 8 A100 GPUs.

F.3. Downtream Evaluation

We perform few-shot evaluation of the models on 4 subtasks of SuperGLUE. We use 2-shot prompting and greedy decoding. The prompt consists of an instruction followed by two examples. The examples are sampled from the train split while we report the accuracy on validation split averaged over 5 selection of exemplars. Prompts for each subtask are illustrated in Figure 10.

The context is a passages containing some information. Given a question about the context, use the information to answer the question with either 'Yes' or 'No'.

Context: 3-way lamp -- The center contact of the bulb typically connects to the medium-power filament, and the ring connects to the low-power filament. Thus, if a 3-way bulb is screwed into a standard light socket that has only a center contact, only the medium-power filament operates. In the case of the 50 W / 100 W / 150 W bulb, putting this bulb in a regular lamp socket will result in it behaving like a normal 100W bulb. Question: do 3-way light bulbs work in any lamp
Answer: Yes

Context: Perfume: The Story of a Murderer (film) -- Perfume: The Story of a Murderer is a 2006 German period psychological crime thriller film directed by Tom Tykwer and starring Ben Whishaw, Alan Rickman, Rachel Hurd-Wood, and Dustin Hoffman. Tykwer, with Johnny Klimek and Reinhold Heil, also composed the music. The screenplay by Tykwer, Andrew Birkin, and Bernd Eichinger is based on Patrick Süskind's 1985 novel Perfume. Set in 18th century France, the film tells the story of Jean-Baptiste Grenouille (Whishaw), an olfactory genius, and his homicidal quest for the perfect scent. Question: is the film perfume based on a true story
Answer: No

BoolQ

Given a premise and a hypothesis, answer whether the hypothesis logically follows from the premise with 'True' or 'False' or 'Neither'.

Context: B: She says that when her husband died oh, that my uncle had said that he would never put her in a rest home. So it's kind of, uh, I don't know. I mean, I don't think my parents would but she is getting pretty bad like she has to have like a little toilet right by her bed and, it's, A: Uh-huh. B: and my mom has to take care of her pretty much so it gets, I don't know. it's a hard decision, but I don't think I would do it to my parents personally. Question: she would do it to her parents
Answer: No

Context: B: No, it was, I didn't like the way it ended. A: I know, well the only reason I know whxy it ended is on Arsenio Hall one night, Christopher Reeves told, that, you know, B: Uh-huh. A: I can't believe they killed them. Question: they killed them
Answer: Yes

CB

Given a premise and a hypothesis, answer whether the hypothesis follows from the premise with 'Yes' or 'No'.

Context: The Bank of Italy, the ultimate arbiter of Italian banking mergers, has been engulfed by scandal since police wire taps revealed Fazio and his wife advised a local banker in a bid for Bank Antonveneta against Dutch bank ABN AMRO.
Question: A local banker bids for Bank Antonveneta.
Answer: Yes

Context: The Statue of Liberty was reopened to the public on July 5 after its extensive refurbishing. 1986 is a common year starting on Wednesday of the Gregorian calendar.
Question: The Statute of Liberty was built in 1986.
Answer: None

RTE

Choose the correct ending for the context.

Choice1: the woman kissed him.
Choice2: the woman made him blush.
Context: The man had lipstick on his cheek because
Answer: Choice1

Choice1: i attended a yoga class.
Choice2: i bought fruits and vegetables.
Context: I made a resolution to eat a healthy diet so
Answer: Choice2

COPA

Figure 10. Prompts for SuperGLUE downstream evaluation.

G. Amount of compute

We train the 125M and 355M models on A5000 GPUs and the 770M models on A100 GPUs. The total amount of compute spent on all experiments is about 6000 hours on A100s and 10000 hours on A5000s. This amounts to 4.38e21 FLOPs.

H. Limitations

Scaling up to larger models and datasets. Although Sophia demonstrates scalability up to 770M models and OpenWebText, and there is no essential constraints from further scaling up, we do not compare with AdamW and Lion on larger models and datasets due to limited resources. We believe Sophia is faster than AdamW and Lion on larger models given the improvement in scaling laws and better pre-training stability.

Holistic downstream evaluation. We evaluate pre-trained checkpoints on 4 SuperGLUE subtasks, which only demonstrates the improvement in downstream performance several datasets. While a holistic evaluation of language models itself is an open research topic, better downstream evaluation is still important. The limitation in downstream evaluation is also due to the limited model size, because language models at this scale do not have enough capabilities such as in-context learning, and mathematical reasoning.

Evaluation on other domains. While this paper focuses on optimizers for large language modeling, a more general optimizer should also be evaluated in other domains such as computer vision, reinforcement learning, and multimodel tasks. Due to the limitation of computation resources, we leave the application to other domains and models to future works.

I. Theoretical Analyses: Details of Section B

Theorem B.3 is a direct combination of the Lemma I.10 (Descent Lemma), Lemma I.9 and Lemma I.11. In the analysis, there will be two phases. In the first phase decrease loss to $\frac{\mu\rho^2}{8}$ in $8\frac{L(\theta(0))-\min L}{\eta\mu\rho^2}$ steps. In the second phase, there will be an exponential decay of error.

Lemma I.1. *Under Assumption B.1, we have that* $L(\theta) \to \infty$ *whenever* $\|\theta\|_2 \to \infty$.

Proof of Lemma I.1. By convexity of L, we have $\forall \theta \in \mathbb{R}^d$ with $\|\theta - \theta^*\|_2 \geq 1$,

$$\frac{1}{\|\theta-\theta^*\|_2} L(\theta) + \frac{\|\theta-\theta^*\|_2 - 1}{\|\theta-\theta^*\|_2} L(\theta^*) \geq L(\theta^* + \frac{\theta-\theta^*}{\|\theta-\theta^*\|_2}) \geq \min_{\|\bar{\theta}\|_2=1} L(\theta^*+\bar{\theta}). \tag{14}$$

Since L is strictly convex, $\Delta \triangleq \min_{\|\bar{\theta}\|_2=1} L(\theta^*+\bar{\theta}) - L(\theta^*) > 0$. Thus we conclude that

$$L(\theta) \geq \|\theta-\theta^*\|_2 \Delta + L(\theta^*) \geq (\|\theta\|_2 - \|\theta^*\|_2)\Delta + L(\theta^*). \tag{15}$$

Therefore when $\|\theta\|_2 \to \infty$, $L(\theta) \to \infty$ as well. $\square$

Note that we don't assume the Hessian of loss is Lipschitz. Assumption B.2 only assumes the Hessian in a neighborhood of constant radius only differs by a constant in the multiplicative sense.

Lemma I.2. *For any $\theta \in \mathbb{R}^d$ satisfying $L(\theta) - \min L \leq \frac{\mu R^2}{4}$, it holds that $\|\theta-\theta^*\|_2 \leq 2\sqrt{\frac{L(\theta)-\min L}{\mu}} \leq R$.*

Proof of Lemma I.2. We will prove by contradiction. Suppose there exists such θ with $L(\theta) \leq \frac{\mu R^2}{4}$ but $\|\theta-\theta^*\|_2 > 2\sqrt{\frac{L(\theta)-\min L}{\mu}}$. We consider $\theta' \triangleq \theta^* + \sqrt{\frac{2L(\theta)}{\mu}} \cdot \frac{\theta-\theta^*}{\|\theta-\theta^*\|_2}$. Since θ' is between θ and θ^* and that L is strictly convex, we know that $L(\theta') < L(\theta)$. However, by Taylor expansion on function $f(t) \triangleq L(\theta^* + t(\theta'-\theta^*))$, we have that

$$f(1) = f(0) + f'(0) + \frac{f''(t)}{2}, \quad \text{for some } t \in [0,1]. \tag{16}$$

Note that $\|\theta'-\theta^*\|_2 \leq \|\theta-\theta^*\|_2 \leq R$, by Assumption B.2 and Assumption B.1, we have $f''(t) = (\theta'-\theta^*)^\top \nabla^2 L(t\theta' + (1-t)\theta^*)(\theta'-\theta^*) \geq \frac{1}{2}(\theta'-\theta^*)^\top \nabla^2 L(\theta^*)(\theta'-\theta^*) \geq \frac{\mu}{2}\|\theta'-\theta^*\|_2^2 = 2(L(\theta) - \min L))$. Also note that $f(1) = L(\theta')$, $f(0) = L(\theta^*)$ and $f'(0) = 0$, we conclude that $L(\theta') - L(\theta^*) \geq L(\theta) - L(\theta^*)$, namely $(\theta') \geq L(\theta)$. Contradiction! $\square$

Lemma I.3. *For any $\theta \in \mathbb{R}^d$ satisfying that $\|\nabla L(\theta)\|_2 \leq \frac{R\mu}{2}$, it holds that $\|\theta-\theta^*\|_2 \leq \frac{2\|\nabla L(\theta)\|}{\mu} \leq R$.*

Proof of Lemma I.3. We will prove by contradiction. We consider function $f(t) \triangleq \left\langle \frac{\theta-\theta^*}{\|\theta-\theta^*\|_2}, \nabla L(\theta^* + t\cdot \frac{\theta-\theta^*}{\|\theta-\theta^*\|_2}) \right\rangle$. Because of the strict convexity of L, f is a strict monotone increasing function. If $\|\theta-\theta^*\| > \frac{2\|\nabla L(\theta)\|}{\mu}$ but $\|\nabla L(\theta)\|_2 \leq \frac{R\mu}{2}$, then we have $f(R) < f(\|\theta-\theta^*\|_2) \leq \|\nabla L(\theta)\|_2$. On the other hand, by Assumption B.2 and Assumption B.1, $f'(t) \geq \frac{\mu}{2}$ for $t \in [0,R]$. Thus $f(R) \geq f(0) + \int_{t=0}^{\frac{2\|\nabla L(\theta)\|}{\mu}} f'(t)\mathrm{d}t = \|\nabla L(\theta)\|$. Contradiction! $\square$

Lemma I.4. *For any $\theta \in \mathbb{R}^d$, the following differential equation has at least one solution on interval $[0,1]$:*

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = -(\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta), \quad \theta(0) = \theta, \tag{17}$$

and the solution satisfies that $\nabla L(\theta(t)) = (1-t)\nabla L(\theta)$ for all $t \in [0,1]$ and $\theta(0) = \theta^*$.

Proof of Lemma I.4. Since $\nabla^2 L$ is continuous and positive definite by Assumption B.1 , $(\nabla^2 L)^{-1}$ is continuous and thus the above ODE (42) has a solution over interval $[0,T)$ for some positive T and we let $T_{\max}$ be the largest positive number such that the solution exists (or $T_{\max} = \infty$). Now we claim $T_{\max} \geq 1$, otherwise $\|\theta(t) - \theta^*\|_2$ must diverge to infinity when $t \to T_{\max}$. However, for any $t \leq 1$, we have

$$\frac{\mathrm{d}\nabla L(\theta(t))}{\mathrm{d}t} = -\nabla L(\theta), \tag{18}$$

which implies that $\nabla L(\theta(t)) = (1-t)\nabla L(\theta)$ for all $t \in [0,1]$. Therefore,

$$\frac{\mathrm{d}L(\theta(t))}{\mathrm{d}t} = -(\nabla L(\theta(t)))^\top (\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta) = (1-t)(\nabla L(\theta))^\top (\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta) \leq 0. \tag{19}$$

Thus $L(\theta(t)) \leq L(\theta(0))$. By Lemma I.1, we know that $\|\theta(t)\|$ remains bounded for all $t \in [0, T_{\max}]$, thus $T_{\max} \geq 1$. Note that $\theta(1)$ has zero gradient, $\theta(1)$ must be θ^*. This completes the proof. $\square$

Lemma I.5. *For any $\theta \in \mathbb{R}^d$ satisfying (1) $L(\theta) - \min L \le \frac{\mu R^2}{16}$ or (2) $\|\nabla L(\theta)\|_2 \le \frac{R\mu}{4}$, it holds that*

$$L(\theta) - \min L \le \nabla L(\theta)^\top (\nabla^2 L(\theta))^{-1} \nabla L(\theta) \le 4(L(\theta) - \min L). \tag{20}$$

Proof of Lemma I.5. Let $\{\theta(t)\}_{t=0}^1$ be the solution of Equation 42. We know that $\nabla L(\theta(t)) = (1-t)\nabla L(\theta)$ for all $t \in [0,1]$ and that $\theta(1) = \theta^*$ by Lemma I.4. For case (1), by Lemma I.2, we know that for any $t \in [0,1]$, $\|\theta(t) - \theta^*\|_2 \le R/2$. For case (2), by Lemma I.3, we know that for any $t \in [0,1]$, $\|\theta(t) - \theta^*\|_2 \le R/2$. Thus in both two cases, $\|\theta(t) - \theta\|_2 = \|\theta(t) - \theta(0)\|_2 =\le \|\theta(t) - \theta^*\| + \|\theta(0) - \theta^*\| \le R$. By Assumption B.2, it holds that

$$2(\nabla^2 L(\theta))^{-1} \succeq (\nabla^2 L(\theta(t)))^{-1} \succeq \frac{1}{2}(\nabla^2 L(\theta))^{-1}. \tag{21}$$

for all $t \in [0,1]$. Therefore, we have that

$$\begin{aligned} L(\theta) - \min L = L(\theta(0)) - L(\theta(1)) &= \int_{t=0}^1 (\nabla L(\theta(t)))^\top (\nabla^2 L(\theta(t)))^{-1} \nabla L(\theta) \\ &= \int_{t=0}^1 (1-t)(\nabla L(\theta))^\top (\nabla^2 L(\theta(t)))^{-1} \nabla L(\theta). \end{aligned} \tag{22}$$

The proof is completed by plugging Equation 21 into Equation 22 and noting that $\int_{t=0}^1 (1-t) = 1/2$. □

Lemma I.6. *For any $\theta \in \mathbb{R}^d$ satisfying (1) $L(\theta) - \min L \le \frac{\mu R^2}{4}$ or (2) $\|\nabla L(\theta)\|_2 \le \frac{R\mu}{2}$, it holds that*

$$L(\theta) - \min L \le \mu^{-1} \|\nabla L(\theta)\|_2^2 \tag{23}$$

Proof of Lemma I.6. The proof of Lemma I.6 is almost the same as that of Lemma I.5 and thus omitted. □

Lemma I.7. *For any $\theta \in \mathbb{R}^d$ satisfying $L(\theta) - \min L \le \frac{\mu R^2}{16}$, it holds that*

$$\left\|(\nabla^2 L(\theta))^{-1} \nabla L(\theta)\right\|_2 \le \sqrt{\frac{8(L(\theta) - \min L)}{\mu}}. \tag{24}$$

Proof of Lemma I.7. By Lemma I.2, we have that $\|\theta - \theta^*\|_2 \le R$. By Assumption B.2, we have $\nabla^2 L(\theta) \succeq \frac{1}{2}\nabla^2 L(\theta^*) \succeq \frac{\mu}{2} I_d$. By Lemma I.5, we have that

$$\begin{aligned} 4(L(\theta) - \min L) \ge& \nabla L(\theta)^\top (\nabla^2 L(\theta))^{-1} \nabla L(\theta) && (25) \\ \ge& \nabla L(\theta)^\top (\nabla^2 L(\theta))^{-1} \nabla^2 L(\theta) (\nabla^2 L(\theta))^{-1} \nabla L(\theta) && (26) \\ \ge& \frac{\mu}{2} \left\|\nabla L(\theta)^\top (\nabla^2 L(\theta))^{-1}\right\|_2^2. && (27) \end{aligned}$$

This completes the proof. □

Lemma I.8. *For any $\theta \in \mathbb{R}^d$ satisfying that $\left\|((\nabla^2 L(\theta))^{-1} \nabla L(\theta)\right\|_2 \le \frac{R}{2}$, it holds that*

$$L(\theta) - \min L \le \nabla L(\theta)^\top (\nabla^2 L(\theta))^{-1} \nabla L(\theta) \le 4(L(\theta) - \min L). \tag{28}$$

Proof of Lemma I.8. Let $\{\theta(t)\}_{t=0}^1$ be the solution of Equation 42 and we claim that for all $t \in [0,1]$, $\|\theta(t) - \theta\|_2 \le R$. Otherwise, let T be the smallest positive number such that $\|\theta(T) - \theta\|_2 = R$. Such T exists because $\|\theta(t) - \theta\|_2$ is continuous in t and $\|\theta(0)-\theta\|_2 = 0$. We have that

$$R = \|\theta(T)-\theta(0)\|_2 \leq \int_{t=0}^{T} \left\|\frac{\mathrm{d}\theta(t)}{\mathrm{d}t}\right\|_2 \mathrm{d}t \tag{29}$$

$$= \int_{t=0}^{T} \left\|((\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta)\right\|_2 \mathrm{d}t \tag{30}$$

$$\leq \int_{t=0}^{T} \left\|(\nabla^2 L(\theta(t)))^{-1}\nabla^2 L(\theta)\right\|_2 \left\|((\nabla^2 L(\theta))^{-1}\nabla L(\theta)\right\|_2 \mathrm{d}t \tag{31}$$

$$\leq 2\int_{t=0}^{T} \left\|((\nabla^2 L(\theta))^{-1}\nabla L(\theta)\right\|_2 \mathrm{d}t \tag{32}$$

$$\leq 2T\frac{R}{2} = RT, \tag{33}$$

which implies $T=1$. Here in Equation 32, we use Assumption B.2. Thus we conclude that for all $t\in[0,1]$, $\|\theta(t)-\theta\|_2 \leq R$. By Assumption B.2, it holds that

$$2(\nabla^2 L(\theta))^{-1} \succeq (\nabla^2 L(\theta(t)))^{-1} \succeq \frac{1}{2}(\nabla^2 L(\theta))^{-1}. \tag{34}$$

Therefore, we have that

$$L(\theta) - \min L = L(\theta(0)) - L(\theta(1)) = \int_{t=0}^{1} (\nabla L(\theta(t)))^\top (\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta)$$
$$= \int_{t=0}^{1} (1-t)(\nabla L(\theta))^\top (\nabla^2 L(\theta(t)))^{-1}\nabla L(\theta). \tag{35}$$

The proof is completed by plugging Equation 34 into Equation 35 and noting that $\int_{t=0}^{1}(1-t) = 1/2$. $\square$

Lemma I.9. *If $\rho \leq \frac{R}{2\sqrt{d}}$, then for any $\Delta \leq \frac{R\rho\mu}{10}$ and any $\theta \in \mathbb{R}^d$ satisfying*

$$\sum_{i=1}^{d} \min\{\rho\left|v_i^\top \nabla L(\theta)\right|, \sigma_i^{-1}\left|v_i^\top \nabla L(\theta)\right|^2\} \leq \Delta, \tag{36}$$

where $\nabla^2 L(\theta) = V^\top \Sigma V$ is the eigendecomposition of $\nabla^2 L(\theta)$, v_i is the ith row of V and $\Sigma = \mathrm{diag}(\sigma_1,\ldots,\sigma_d)$, it holds that

$$L(\theta) - \min L \leq \Delta + \frac{25\Delta^2}{\rho^2\mu} \tag{37}$$

In particular, if we set $\Delta \triangleq \frac{\mu\rho^2}{20}$, we have $L(\theta)-\min L \leq \frac{\mu\rho^2}{8}$.

Proof of Lemma I.9. Let $I_\theta \triangleq \{i\in[d] \mid \left|v_i^\top \nabla L(\theta)\right| \sigma_i^{-1} \leq \rho\}$ be the set of indices where clipping does not happen. Then we have that

$$\sum_{i\in I_\theta} \sigma_i^{-1}\left|v_i^\top \nabla L(\theta)\right|^2 \leq \Delta \tag{38}$$

$$\sum_{i\notin I_\theta} \rho\left|v_i^\top \nabla L(\theta)\right| \leq \Delta \tag{39}$$

Now we consider a new strictly convex loss function in $R^{|I_\theta|}$, which is L restricted on the space of $\{\theta + \sum_{i\in I_\theta} w_{[i]} v_i \mid w\in\mathbb{R}^{|I_\theta|}\}$, that is, $L_\theta(w) = L(\theta + \sum_{i\in I_\theta} w_{[i]} v_i)$. This new loss function L_θ clearly satisfy Assumption B.2 since it is a restriction of L into some subspace of $\mathbb{R}^d$. By Lemma I.1, we know that $\inf_w L_\theta(w)$ can be attained and we denote it by

w^*. By Assumption B.1, we know that L_θ is strictly convex and thus $\nabla^2 L_\theta(w) \succ 0$, which means Assumption B.1 also holds for L_θ.

Next we will apply Lemma I.8 on L_θ at $w=0$. We use $V_{I_\theta} \in \mathbb{R}^{|I|\times d}$ to denote the submatrix of V containing rows in I for any $I \subset [d]$. One can verify by chain rule that $\nabla L_\theta(w) = V_{I_\theta}\nabla L(\theta + V_{I_\theta}^\top w)$ and that $\nabla^2 L_\theta(w) = V_{I_\theta}\nabla^2 L(\theta + V_{I_\theta}^\top w)V_{I_\theta}^\top$. Thus we have that

$$(\nabla^2 L_\theta(0))^{-1}\nabla L_\theta(0) = V_{I_\theta}(\nabla^2 L(\theta))^{-1}\nabla L(\theta). \tag{40}$$

By the definition of I_θ, we know that $\left\|V_{I_\theta}(\nabla^2 L(\theta))^{-1}\nabla L(\theta)\right\|\infty \le \rho$. Thus $\left\|(\nabla^2 L_\theta(0))^{-1}\nabla L_\theta(0)\right\|_2 \le \sqrt{d}\left\|V_{I_\theta}(\nabla^2 L(\theta))^{-1}\nabla L(\theta)\right\|_\infty = \sqrt{d}\cdot\rho \le \frac{R}{2}$. Thus we can apply Lemma I.8 on L_θ at $w=0$ and conclude that

$$L_\theta(0) - L_\theta(w^*) \le \nabla L_\theta(0)^\top(\nabla^2 L_\theta(0))^{-1}\nabla L_\theta(0) = \sum_{i\in I_\theta}\sigma_i^{-1}\left|v_i^\top \nabla L(\theta)\right|^2 \le \Delta \tag{41}$$

Thus $L(\theta) - L(\theta + V_{I_\theta}^\top w^*) = L_\theta(0) - L_\theta(w^*) \le \Delta$.

It remains to show that $L(\theta + V_{I_\theta}^\top w^*) - L(\theta^*) \le \frac{25\Delta^2}{\rho^2\mu}$. To do so, our strategy is to first show that $\left\|\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2$ is small and then to use Lemma I.6. We will use I_θ^c to denote the complement of I_θ in $[d]$ and $V_{I_\theta^c} \in \mathbb{R}^{(d-|I_\theta|)\times d}$ to denote the submatrix of V which contains all the rows that do not belong to I_θ. Note that w^* is the minimizer of L_θ, we know that $V_{I_\theta}\nabla L(\theta + V_{I_\theta}^\top w^*) = 0$ and that $\left\|\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2 = \left\|V_{I_\theta^c}\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2$.

Now we consider the following ODE

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = -(\nabla^2 L_\theta(w(t)))^{-1}\nabla L_\theta(0), \quad w(0) = 0. \tag{42}$$

By Lemma I.4, we know this ODE has solution $w(t)$ over interval $[0,1]$ with $w(1) = w^*$. With the same argument in the proof of Lemma I.8, we know that $\|w(t)\|_2 \le R$ for all $t\in[0,1]$. Thus we have for any $t\in[0,1]$,

$$\begin{aligned}
&\left\|V_{I_\theta^c}\frac{\mathrm{d}\nabla L(\theta + V_{I_\theta}w(t))}{\mathrm{d}t}\right\|_2 && (43)\\
=&\left\|V_{I_\theta^c}\nabla^2 L(\theta + V_{I_\theta}w(t))V_{I_\theta}(\nabla^2 L_\theta(w(t)))^{-1}\nabla L_\theta(0)\right\|_2 && (44)\\
=&\left\|V_{I_\theta^c}\nabla^2 L(\theta + V_{I_\theta}w(t))V_{I_\theta}V_{I_\theta}^\top(\nabla^2 L(\theta + V_{I_\theta}w(t)))^{-1}\nabla L(\theta)\right\|_2 && (45)\\
\le&\left\|V_{I_\theta^c}\sqrt{\nabla^2 L(\theta + V_{I_\theta}w(t))}\right\|_F && (46)\\
&\cdot\left\|\sqrt{\nabla^2 L(\theta + V_{I_\theta}w(t))}V_{I_\theta}V_{I_\theta}^\top(\nabla^2 L(\theta + V_{I_\theta}w(t)))^{-1}\nabla L(\theta)\right\|_2 && (47)
\end{aligned}$$

For the first term (Equation 46), by Assumption B.2, we have that

$$\left\|V_{I_\theta^c}\sqrt{\nabla^2 L(\theta + V_{I_\theta}w(t))}\right\|_F^2 \le 2V_{I_\theta^c}\nabla^2 L(\theta)V_{I_\theta^c} = 2\sum_{i\notin I_\theta}\sigma_i \le 2\sum_{i\notin I_\theta}\frac{v_i^\top\nabla L(\theta)}{\rho} \le \frac{2\Delta}{\rho^2}. \tag{48}$$

For the second term (Equation 47), by Assumption B.2, we have that

$$\begin{aligned}
&\left\|\sqrt{\nabla^2 L(\theta + V_{I_\theta}w(t))}V_{I_\theta}V_{I_\theta}^\top(\nabla^2 L(\theta + V_{I_\theta}w(t)))^{-1}\nabla L(\theta)\right\|_2^2 && (49)\\
\le&8\left\|\sqrt{\nabla^2 L(\theta)}V_{I_\theta}V_{I_\theta}^\top(\nabla^2 L(\theta))^{-1}\nabla L(\theta)\right\|_2^2 && (50)\\
=&8\nabla L(\theta)^\top V_{I_\theta}V_{I_\theta}^\top(\nabla^2 L(\theta))^{-1}V_{I_\theta}V_{I_\theta}^\top\nabla L(\theta) && (51)\\
=&8\sum_{i\in I_\theta}\sigma_i^{-1}\left|v_i^\top\nabla L(\theta)\right|^2 \le 8\Delta. && (52)
\end{aligned}$$

Thus we conclude that $\left\|V_{I_\theta^c}\frac{\mathrm{d}\nabla L(\theta+V_{I_\theta}w(t))}{\mathrm{d}t}\right\|_2 \le \frac{4\Delta}{\rho}$, which implies that

$$\left\|\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2 = \left\|V_{I_\theta^c}\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2 \tag{53}$$

$$= \left\|V_{I_\theta^c}\nabla L(\theta) + \int_{t=0}^{1} V_{I_\theta^c}\frac{\mathrm{d}\nabla L(\theta+V_{I_\theta}w(t))}{\mathrm{d}t}\mathrm{d}t\right\|_2 \tag{54}$$

$$\le \left\|V_{I_\theta^c}\nabla L(\theta)\right\|_2 + \int_{t=0}^{1}\left\|V_{I_\theta^c}\frac{\mathrm{d}\nabla L(\theta+V_{I_\theta}w(t))}{\mathrm{d}t}\right\|_2 \mathrm{d}t \tag{55}$$

$$\le \frac{\Delta}{\rho} + \frac{4\Delta}{\rho} = \frac{5\Delta}{\rho}. \tag{56}$$

Applying Lemma I.6, we have that

$$L(\theta + V_{I_\theta}^\top w^*) - \min L \le \mu^{-1}\left\|\nabla L(\theta + V_{I_\theta}^\top w^*)\right\|_2^2 = \frac{25\Delta^2}{\rho^2\mu}. \tag{57}$$

This completes the proof. □

Lemma I.10 (Descent Lemma)**.** *For any $\eta, \rho > 0$ with $\eta\rho \le R/\sqrt{d}$, $\theta \in \mathbb{R}^d$ and any eigendecomposition of $\nabla^2 L(\theta)$, where $V_t V_t^\top = I_d$, σ_t is diagonal $\nabla^2 L(\theta) = V^\top \Sigma V$, define*

$$\theta_+ \triangleq \theta - \eta V^\top \mathrm{clip}(V(\nabla^2 L(\theta))^{-1}\nabla L(\theta), \rho), \tag{58}$$

it holds that

$$L(\theta_+) - L(\theta) \le -(\eta - \eta^2)\sum_{i=1}^{d}\min\{\rho\left|v_i^\top \nabla L(\theta)\right|, \sigma_i^{-1}\left|v_i^\top\nabla L(\theta)\right|^2\}, \tag{59}$$

where v_i is the ith row of matrix V.

Proof of Lemma I.10. Let $u \triangleq \mathrm{clip}(V(\nabla^2 L(\theta))^{-1}\nabla L(\theta), \rho)$. By the definition of clip operation, we know that $\left\|V^\top u\right\|_2 = \left\|u\right\|_2 \le \sqrt{d}\rho$. Thus we have $\|\theta_+ - \theta\| = \eta\left\|V^\top u\right\|_2 \le \eta\rho\sqrt{d}$. Define $f(t) = L(t\theta_+ + (1-t)\theta)$. By Assumption B.2, we know that $f''(t) \le 2f''(0)$ for all $t \in [0, 1]$ and thus

$$f(1) = f(0) + f'(0) + \int_{s=0}^{1}\int_{t=0}^{s} f''(s)\mathrm{d}s\mathrm{d}t \le f(0) + f'(0) + f''(0). \tag{60}$$

It remains to show that

1. $f'(0) = -\eta\sum_{i=1}^d \min\{\rho\left|v_i^\top\nabla L(\theta)\right|, \sigma_i^{-1}\left|v_i^\top\nabla L(\theta)\right|^2\}$;
2. $f''(0) \le \eta^2\sum_{i=1}^d \min\{\rho\left|v_i^\top\nabla L(\theta)\right|, \sigma_i^{-1}\left|v_i^\top\nabla L(\theta)\right|^2\}$;

First, by chain rule, we have $f'(0) = \left\langle\nabla L(\theta), -\eta V^\top u\right\rangle = \langle V\nabla L(\theta), -\eta u\rangle = -\eta\left\langle V\nabla L(\theta), \mathrm{clip}(\Sigma^{-1}V\nabla L(\theta), \rho)\right\rangle = -\eta\sum_{i=1}^d \min\{\rho\left|v_i^\top\nabla L(\theta)\right|, \sigma_i^{-1}\left|v_i^\top\nabla L(\theta)\right|^2\}$.
Second, again by chain rule, we have $f''(0) = \eta^2\left\langle V^\top u, \nabla^2 L(\theta)V^\top u\right\rangle = \eta^2\langle u, \Sigma u\rangle = \sum_{i=1}^d |u_i|^2\sigma_i$. Note that by definition $|u_i| = \min\{\left|v_i^\top\nabla L(\theta)\right|/\sigma_i, \rho\}$, we have $|u_i|^2\sigma_i \le \min\{\left|v_i^\top\nabla L(\theta)\right|/\sigma_i, \rho\}\cdot\left|v_i^\top\nabla L(\theta)\right|/\sigma_i\cdot\sigma_i = \min\{\left|v_i^\top\nabla L(\theta)\right|^2/\sigma_i, \rho\left|v_i^\top\nabla L(\theta)\right|\}$, which completes the proof. □

Lemma I.11. *If $\eta\rho \le R/\sqrt{d}$ and for some $T \in \mathbb{N}$, $L(\theta_T) - \min L \le \frac{\mu\rho^2}{8}$, then if holds that for all $t \ge T$,*

1. *$\theta_{t+1} = \theta_t - \eta(\nabla^2 L(\theta_t))^{-1}\nabla L(\theta_t)$;*
2. *$L(\theta_t) - \min L \le (1 - \eta(1-\eta))^{t-T}(L(\theta_T) - \min L)$.*

Proof of Lemma I.11. First by Lemma I.10, we have for all $t \geq T$, $(\theta_t) - \min L \leq L(\theta_T) - \min L \leq \frac{\mu\rho^2}{8}$, therefore by Lemma I.7, we have $\left\|(\nabla^2 L(\theta_t))^{-1}\nabla L(\theta_t)\right\|_2 \leq \rho$ for all $t \geq T$, which implies clipping will not happen. This completes the proof of the first claim.

For the second claim, by Lemmas I.5 and I.10, we have that

$$L(\theta_{t+1}) - L(\theta_t) \leq -(\eta - \eta^2)\sum_{i=1}^{d}\sigma_i^{-1}\left|v_i^\top \nabla L(\theta_t)\right|^2 \tag{61}$$

$$= -(\eta - \eta^2)\nabla L(\theta_t)(\nabla^2 L(\theta_t))^{-1}\nabla L(\theta_t) \tag{62}$$

$$\leq -\eta(1-\eta)(L(\theta_t) - \min L), \tag{63}$$

which completes the proof. □

I.1. Lower bound for SignGD on 2-dimensional quadratic loss

Define $L_{\mu,\beta} : \mathbb{R}^2 \to \mathbb{R}$ as a quadratic function with parameter μ, β as $L_{\mu,\beta}(\theta) \triangleq \frac{\mu}{2}\theta_{[1]}^2 + \frac{\beta}{2}\theta_{[2]}^2$. We have the following lower bound, which shows signGD's convergence rate has to depend on the condition number β/μ.

Theorem I.12. *For any $\mu, \beta, \Delta, \epsilon > 0$, suppose there exist a learning rate η and a time T such that for all θ_0 satisfying that $L_{\mu,\beta}(\theta_0) \leq \Delta$, signGD reaches loss at most ϵ at step $T-1$ and T (in the sense that $L_{\mu,\beta}(\theta_T) \leq \epsilon$ and $L_{\mu,\beta}(\theta_{T-1}) \leq \epsilon$). Then, T must satisfy $T \geq \frac{1}{2}(\sqrt{\frac{\Delta}{\epsilon}} - \sqrt{2})\sqrt{\frac{\beta}{\mu}}$.*

Proof of Theorem I.12. We consider two initialization: $\theta_0 = (0, \sqrt{\frac{2\Delta}{\beta}})$ and $\theta_0' = (\sqrt{\frac{2\Delta}{\mu}}, 0)$, and let θ_t and θ_t' be the iterates under the two initializations. For each coordinate $i \in \{1, 2\}$, because $|(\theta_t)_{[i]} - (\theta_{t+1})_{[i]}| = \eta$, we have that $|(\theta_t)_{[i]}| + |(\theta_{t+1})_{[i]}| \geq \eta$. Thus $2\epsilon \geq L_{\mu,\beta}(\theta_T) + L_{\mu,\beta}(\theta_{T-1}) \geq \frac{\beta}{2}((\theta_T)_{[2]}^2 + (\theta_{T-1})_{[2]}^2) \geq \frac{\beta\eta^2}{4}$, which implies $\eta \leq \sqrt{\frac{8\epsilon}{\beta}}$.

The fact that $L_{\mu,\beta}(\theta_T') + L_{\mu,\beta}(\theta_{T-1}') \leq 2\epsilon$ implies $(\theta_T')_{[1]} \leq \sqrt{\frac{4\epsilon}{\mu}}$. Because SignGD can only move each coordinate by η at most, we have $(T-1)\eta \geq \sqrt{2\Delta/\mu} - \sqrt{\frac{4\epsilon}{\mu}}$. Using the fact that $\eta \leq \sqrt{\frac{8\epsilon}{\beta}}$, we have that $2(T-1) \geq (\sqrt{\frac{\Delta}{\epsilon}} - \sqrt{2})\sqrt{\frac{\beta}{\mu}}$, which completes the proof. □