AIE6003: Optimization for Machine Learning

Xiaopeng Li
CUHK(SZ)
SAI

April 23, 2026

Outline

Stochastic Gradient Descent with Momentum

Adaptive Methods

Momentum in SGD

Objective:

$$\min_{x\in\mathbb{R}^d} f(x) = \mathbb{E}_\xi[F(x;\xi)] \quad \text{or} \quad \min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Vanilla SGD:

$$x_{k+1} = x_k - \eta_k g_k, \quad \mathbb{E}[g_k \mid x_k] = \nabla f(x_k).$$

Issue:

- ill-conditioning SGD zig-zags in high-curvature directions.
- large step sizes amplify oscillation brought by noise.

Idea: use an exponentially weighted memory of past stochastic gradients.

Note: In the stochastic setting, momentum cannot universally accelerate. It is when there is persistent gradient signal and the noise is not too large.

SGDM & SNAG Algorithm

SGDM: SGD with Polyak's momentum (heavy ball), for $\beta \in (0,1)$,

$$\begin{cases} v_{k+1} = \beta v_k + g_k, \\ x_{k+1} = x_k - \eta v_{k+1}, \end{cases} \iff x_{k+1} = x_k - \eta \sum_{j=0}^{k} \beta^{k-j} g_j.$$

Exercise: It is corresponding to what we learnt in Week 3,

$$x_{k+1} = x_k + \beta(x_k - x_{k-1}) - \eta g_k.$$

SNAG: Stochastic Nesterov Accelerated Gradient, for $\beta \in (0,1)$,

$$\begin{cases} y_k = x_k - \eta\beta v_k, \\ v_{k+1} = \beta v_k + g(y_k, \xi_k), \\ x_{k+1} = x_k - \eta v_{k+1}, \end{cases} \iff x_{k+1} = x_k - \eta \sum_{j=0}^{k} \beta^{k-j} g(y_k, \xi_k).$$

Exercise: It is corresponding to what we learnt in Week 3,

$$x_{k+1} = x_k + \beta(x_k - x_{k-1}) - \eta g(x_k + \beta(x_k - x_{k-1}), \xi_k).$$

Theory

Smooth nonconvex: same order as SGD:

$$\min_{0\le k\le K-1} \mathbb{E}[\|\nabla f(x_k)\|^2] = O(K^{-1/2}).$$

So there is no universal order-level acceleration over SGD. See [Yan et al., 2018, Mai and Johansson, 2020].

Strongly convex:

- the generic asymptotic stochastic order is comparable to SGD [Sebbouh et al., 2021, Liu and Yuan, 2022];
- for structured models, such as quadratics, momentum can have better transient decay [Pan et al., 2023] or accelerated convergence to a low-variance neighborhood [Dang et al., 2024].

Convex: for constant-momentum SGDM, there is no acceleration in general. In fact, the last iterate can be worse than SGD in worst-case convex theory [Li et al., 2022].

A Closer Look

SGDM update:

$$x_{k+1} = x_k - \alpha g_k + \beta(x_k - x_{k-1}), \qquad 0 \le \beta < 1,$$

where $\mathbb{E}[g_k \mid x_k] = \nabla f(x_k)$ and $\mathbb{E}[\|g_k - \nabla f(x_k)\|^2 \mid x_k] \le \sigma^2$.

Assumptions: f is L-smooth, $f \ge f_*$, and $\|\nabla f(x_k)\| \le G$.

Convergence of SGDM

Using step sizes $\alpha = \Theta\left(\frac{1-\beta}{\sqrt{K}}\right)$ with $\alpha \le (1-\beta)/(2L)$, we have

$$\min_{0 \le k \le K-1} \mathbb{E}[\|\nabla f(x_k)\|^2] \le \frac{C(\beta, L, G, \sigma, f(x_0) - f_*)}{\sqrt{K}}.$$

Conclusion: SGDM matches $O(K^{-1/2})$ convergence rate as SGD.

Proof Idea I

Define: $p_k := \dfrac{\beta}{1-\beta}(x_k - x_{k-1})$ and $z_k := x_k + p_k$.

Key relation: direct algebra gives

$$z_{k+1} = z_k - \frac{\alpha}{1-\beta}\, g_k.$$

So momentum is plain SGD on z_k with effective stepsize $\alpha/(1-\beta)$.

Difficulty: g_k is evaluated at x_k, not z_k. So we must control the tracking error $z_k - x_k = p_k$, which evolves as

$$p_{k+1} = \beta p_k - \frac{\alpha\beta}{1-\beta} g_k.$$

Proof Idea II

Using smoothness to obtain one-step progress

$$f(z_{k+1}) \leq f(z_k) - \frac{\alpha}{1-\beta}\langle \nabla f(z_k),\, g_k\rangle + \frac{L\alpha^2}{2(1-\beta)^2}\|g_k\|^2.$$

Taking $\mathbb{E}[\cdot \mid x_k]$ and using $\mathbb{E}[g_k \mid x_k] = \nabla f(x_k)$:

$$\mathbb{E}_k[f(z_{k+1})] \leq f(z_k) - \frac{\alpha}{1-\beta}\langle \nabla f(z_k),\, \nabla f(x_k)\rangle + \frac{L\alpha^2}{2(1-\beta)^2}(\|\nabla f(x_k)\|^2 + \sigma^2).$$

Using $\langle a, b\rangle \geq \frac{1}{2}\|b\|^2 - \frac{1}{2}\|a-b\|^2$ and $\|\nabla f(z_k) - \nabla f(x_k)\| \leq L\|p_k\|$:

$$\mathbb{E}_k[f(z_{k+1})] \leq f(z_k) - \frac{\alpha}{2(1-\beta)}\|\nabla f(x_k)\|^2 + \frac{\alpha L^2}{2(1-\beta)}\|p_k\|^2 + \frac{L\alpha^2}{2(1-\beta)^2}(\|\nabla f(x_k)\|^2 + \sigma^2).$$

Proof Idea III

Tracking error: the recursion $p_{k+1} = \beta p_k - \frac{\alpha\beta}{1-\beta} g_k$ implies

$$\sum_{k=0}^{K-1} \mathbb{E}\|p_k\|^2 \leq \frac{C_1 \alpha^2 \beta^2 K}{(1-\beta)^3}(G^2 + \sigma^2).$$

Telescope: substituting and summing gives

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\|\nabla f(x_k)\|^2 \leq \underbrace{\frac{C_2\,(f(z_0) - f_*)}{\alpha K}}_{\text{initialization}} + \underbrace{\frac{C_3\,\alpha}{(1-\beta)^2}}_{\text{noise + tracking}}.$$

Balance: choosing $\alpha = \Theta(\frac{1-\beta}{\sqrt{K}})$ yields $O(K^{-1/2})$.

Main message: momentum does not improve the generic stochastic nonconvex convergence rate because the gain from the effective stepsize $\alpha/(1-\beta)$ is offset by the tracking-error term $\|p_k\|^2$.

For Practitioners

Parameter Tuning:

- tune η and β jointly;
- using β too close to 1 can slow down SGD and hurt stability;
- in practice, $\beta = 0.9$ is a reasonable starting point, and more schedules can be used later.

Suitable Problems: persistent historical gradient signal

- deep image-classification models;
- recurrent sequence models with long-term dependencies;

Caveat: when gradient directions are not persistent and noise are too large, SGDM is similar or even worse compared to SGD

- Out-of-distribution Vision fine-tuning;
- Few-shot LM fine-tuning.

Numerical Experiments

Three problems:

- strongly convex: ill-conditioned ridge regression ($\kappa \approx 2500$);
- convex: softmax regression with redundant features;
- nonconvex: 2-layer ReLU MLP on digits.

Setup: mini-batch $= 64$, $\beta = 0.9$, 100 epochs, averaged over 3 seeds.

Step size selection:

- for each method, sweep a grid of η up to the edge of divergence;
- pick the η with the lowest final training loss among stable runs;
- we use the notion of effective step sizes.

Effective Step Size

Motivation: SGDM velocity accumulates under a constant gradient g:

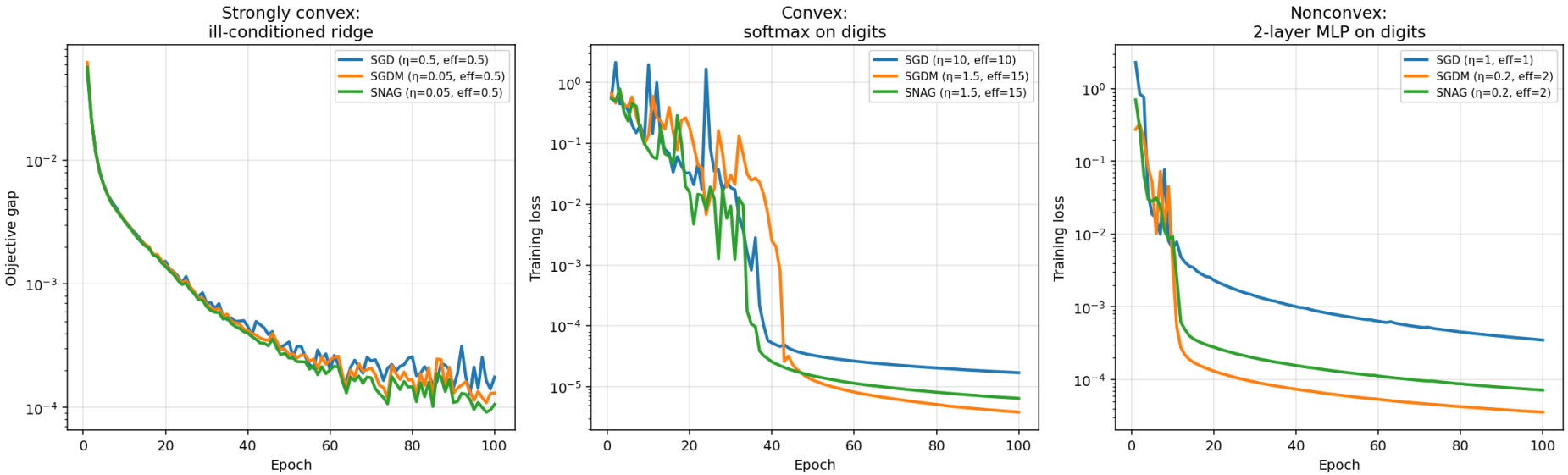
$$v_{k+1} = \beta v_k + g \implies v_\infty = \frac{g}{1-\beta}, \quad \text{so } \Delta x = -\eta\, v_\infty = -\frac{\eta}{1-\beta}\, g.$$

Definition: $\eta_{\text{eff}} := \begin{cases} \eta & \text{SGD,} \\ \eta/(1-\beta) & \text{SGDM \& SNAG.} \end{cases}$

Why it matters:

- It is unfair to compare SGD and SGDM using the same raw step sizes η; otherwise you always find SGDM much better than SGD.
- at the same (small) effective step sizes, all three methods behave nearly identically.
- to see real differences, we must push each method to its own best step size, i.e., largest one that keeps algorithm stable.

Numerical Experiments



Strongly convex: same effective step sizes, and momentum doesn't help. Different from deterministic case.

Convex & nonconvex: momentum tolerates larger effective step sizes before diverging (15 vs. 10 and 2 vs. 1).

Outline

Stochastic Gradient Descent with Momentum

Adaptive Methods

Adagrad

Motivation: during the training, we have huge number of parameters; some has very small or very large magnitude; the range of each gradient component tends to be unbounded.

Adagrad: using second order momentum to normalize gradient:

$$\begin{cases} v_k = \frac{k-1}{k} v_{k-1} + \frac{1}{k} g_k \odot g_k, \\ x_{k+1} = x_k - \eta_k \frac{g_k}{\sqrt{v_k} + \epsilon}, \end{cases}$$

Equivalence: with $\eta_k = \frac{\eta}{\sqrt{k}}$, Adagrad can be rewritten as

$$x_{k+1} = x_k - \eta \frac{g_k}{\sqrt{G_k} + \epsilon}, \quad G_k = \sum_{j=1}^{k} g_j \odot g_j.$$

This algorithm is proposed in [Duchi et al., 2011].

- It outperforms SGDM significantly on language tasks.
- Becomes default choice for NLPers for 5 years.

AdaGrad: Convergence

Adagrad is useful if gradient is sparse: rare but informative features get aggressive updates, while frequent features get dampened.

- frequent/large-gradient coordinates $\implies$ smaller effective step;
- rare/small-gradient coordinates $\implies$ larger effective step.

Theory: for ℓ-smooth f with bounded noise σ^2, [Wang et al., 2023] shows that AdaGrad achieves

$$\min_{1\le k\le T} \mathbb{E}\|\nabla f(x_k)\|^2 = O\left(\frac{\log T}{T} + \frac{\sigma\sqrt{\log T}}{\sqrt{T}}\right),$$

matching SGD's rate, without knowing ℓ or σ a priori.

Practical limitation: the monotonically growing v_k causes the step size to decay to zero, yielding slow convergence in later stages of training.

RMSProp

Motivation: After many iterations, Adagrad stalls even if the model hasn't converged because v_k is monotonically increasing.

RMSProp: replace arithmetic mean with exponential moving average:

$$\begin{cases} v_k = \beta v_{k-1} + (1-\beta) g_k \odot g_k, \\ x_{k+1} = x_k - \eta \frac{g_k}{\sqrt{v_k} + \epsilon}, \end{cases}$$

where $\beta \in [0, 1)$ controls the memory window with length $\approx \frac{1}{1-\beta}$.

Comparison:

- Adagrad: infinite memory, step size can only decrease;
- RMSProp: finite memory, step size can recover when gradient statistics change.

This algorithm is proposed in [Hinton et al., 2012].

Designed for nonconvex deep learning where curvature is non-stationary.

RMSProp: Theory and Insight

How RMSProp helps?

- Consistently large gradients get smaller effective steps;
- Small or infrequent gradients get relatively larger steps;
- unlike AdaGrad, RMSProp only uses recent gradient statistics, so it can track changing regimes.

Theory: for L-smooth nonconvex f,

- [Défossez et al., 2020]: under bounded gradient, $\frac{1}{T}\sum_k \mathbb{E}\|\nabla f(x_k)\|^2 = O(d/T^{1/4})$; same rate for AdaGrad. Note SGD doesn't have the dimension factor d.
- [Shi et al., 2021]: without bounded gradient, RMSProp converges when β is close enough to 1; diverges otherwise. This phase transition explains why default $\beta_2 = 0.999$ works.

Practitioner takeaway: widely used in RNN, reinforcement learning, and other settings with non-stationary training dynamics.

Adam

Motivation: RMSProp adapts per-coordinate step sizes but uses the raw noisy gradient in the numerator. It lacks direction smoothing and suffers from initialization bias when v_k starts at zero.

Adam: combine first-order momentum (direction) + second-order momentum (scale) + bias correction:

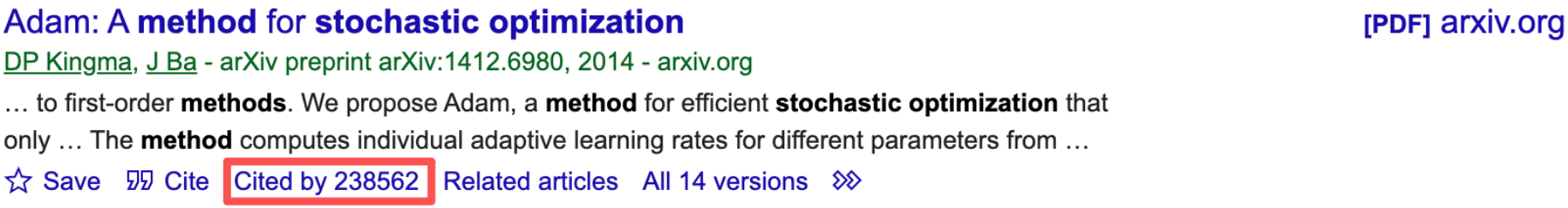
$$\begin{cases} m_k = \beta_1 m_{k-1} + (1-\beta_1)\, g_k, & \hat{m}_k = m_k/(1-\beta_1^k), \\ v_k = \beta_2 v_{k-1} + (1-\beta_2)\, g_k \odot g_k, & \hat{v}_k = v_k/(1-\beta_2^k), \\ x_{k+1} = x_k - \eta\, \hat{m}_k/(\sqrt{\hat{v}_k} + \epsilon). \end{cases}$$

This algorithm is proposed in [Kingma and Ba, 2014].

- Default choice for training LLM, transformers, GANs, and most deep learning models since ∼2015.
- Default setting in PyTorch: $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Adam

Adam is the most popular algorithms in deep learning (230k+ citations)!



[Kingma and Ba, 2014] provides a convergence proof for Adam given $\beta_1 < \sqrt{\beta_2}$, $0 < \beta_1, \beta_2 < 1$.

However, [Reddi et al., 2019] shows there is a mistake in the proof of original Adam paper.

[Reddi et al., 2019] also shows for any $\beta_1 < \sqrt{\beta_2}$, there exists a convex problem s.t. Adam diverges.

The Error

We apply Lemma 9.4 to the above inequality and derive the regret bound by summing across all the dimensions for $i \in 1, ..., d$ in the upper bound of $f_t(\theta_t) - f_t(\theta^*)$ and the sequence of convex functions for $t \in 1, ..., T$:

$$R(T) \leq \sum_{i=1}^{d} \frac{1}{2\alpha_1\beta_1}(\theta_{1,i} - \theta^*_{,i})^2\sqrt{\widehat{v}_{1,i}} + \sum_{i=1}^{d}\sum_{t=2}^{T} \frac{1}{2\beta_1}(\theta_{t,i} - \theta^*_{,i})^2\left(\frac{\sqrt{\widehat{v}_{t,i}}}{\alpha_t} - \frac{\sqrt{\widehat{v}_{t-1,i}}}{\alpha_{t-1}}\right)$$
$$+ \frac{(1-\beta_1)\alpha G_\infty}{\beta_1\sqrt{\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 + \frac{\alpha G_\infty}{\beta_1\sqrt{\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 + \sum_{i=1}^{d}\sum_{t=1}^{T}\frac{1-\beta_{1,t}}{2\alpha_t\beta_{1,t}}(\theta^*_{,i} - \theta_{t,i})^2\sqrt{\widehat{v}_{t,i}}$$

From the assumption, $\|\theta_t - \theta^*\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$, we have:

$$R(T) \leq \frac{D^2}{2\alpha\beta_1}\sum_{i=1}^{d}\sqrt{T\widehat{v}_{T,i}} + \frac{\alpha(2-\beta_1)G_\infty}{\beta_1\sqrt{\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 + \frac{D_\infty^2}{2\alpha}\sum_{i=1}^{d}\sum_{t=1}^{t}\frac{1-\beta_{1,t}}{\beta_{1,t}}\sqrt{t\widehat{v}_{t,i}}$$
$$\leq \frac{D^2}{2\alpha\beta_1}\sum_{i=1}^{d}\sqrt{T\widehat{v}_{T,i}} + \frac{\alpha(2-\beta_1)G_\infty}{\beta_1\sqrt{\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 + \frac{D_\infty^2 G_\infty\sqrt{\beta_2}}{2\alpha}\sum_{i=1}^{d}\sum_{t=1}^{t}\frac{1-\beta_{1,t}}{\beta_{1,t}}\sqrt{t}$$

Their dream is using $|A_t| \leq C$ to obtain

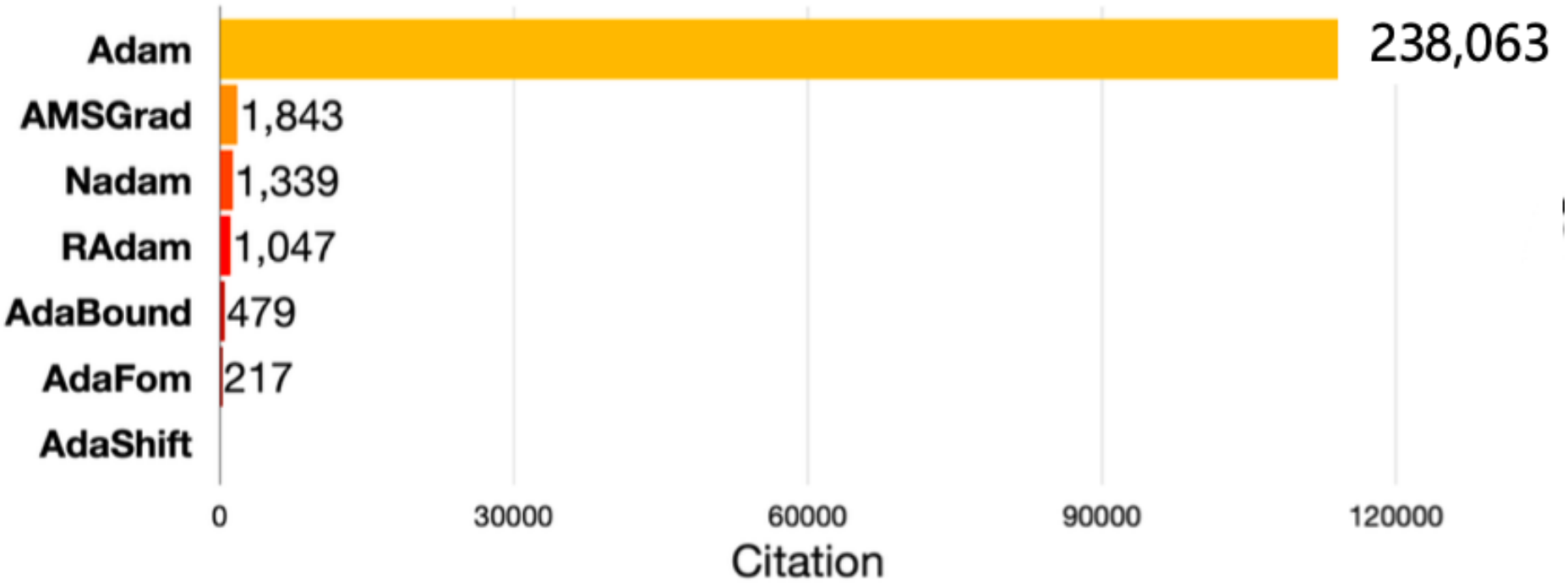
$$A_t(B_t - B_{t-1}) \leq C(B_t - B_{t-1})$$

and then telescoping, which is valid only if $B_t - B_{t-1} \geq 0$. Unfortunately, this is not true for Adam/RMSProp, but true for Adagrad.

Inconsistency with Practice

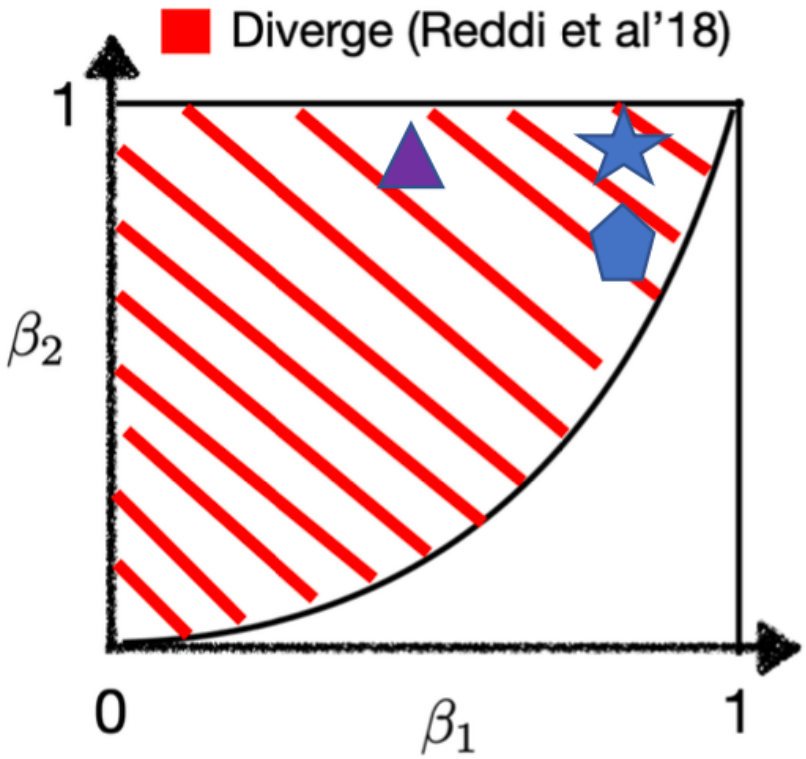
Following this negative result, many new variants are proposed: AMSGrad, AdaBound, AdaFom, AdaShift, Adaptive β_1, β_2.

However, in practice, Adam is still dominant.



Parameter Choice

Why is the divergence not observed with parameter choices in practice?



Most deep learning tasks (e.g. RL, NLP, CV, GAN, etc.):
$\beta_1 = 0.9, \beta_2 = 0.999$

Conditional GAN, DCGAN, etc:
$\beta_1 = 0.5, \beta_2 = 0.999$

LLMs
$\beta_1 = 0.9, \beta_2 = 0.95$

Because [Reddi et al., 2019] first fix β_1, β_2 and then choose the problem.

In practice: hyper-parameters are often problem-dependent, ($\eta = 1/\ell$ in GD). Does Adam converge under fixed problem class?

Adam Can Converge Without Changes

Assuming f_i's are ℓ-smooth and affine variance

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x)-\nabla f(x)\|_2^2 \le D_1\|\nabla f(x)\|_2^2 + D_0.$$

[Zhang et al., 2022] shows the answer is positive.

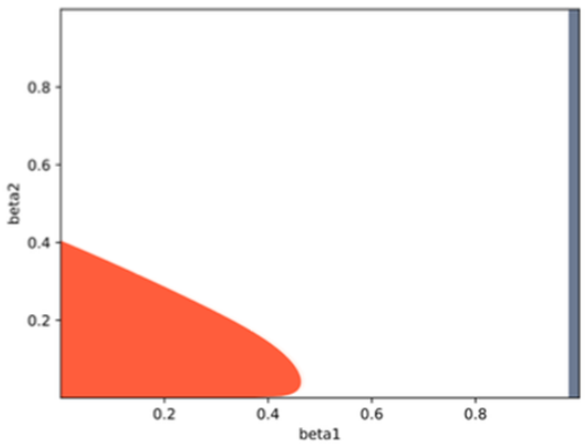
Theory: when $\beta_2 \ge 1 - O(\frac{1-\beta_1^n}{n^{3.5}})$ and $\beta_1 < \sqrt{\beta_2} < 1$, Adam with $\eta_k = \frac{1}{\sqrt{k}}$ converges to the neighborhood of stationary points:

$$\min_{1\le k\le T}\mathbb{E}\|\nabla f(x_k)\|^2 = O\left(\frac{\log T}{\sqrt{T}} + (1-\beta_2)D_0\right).$$

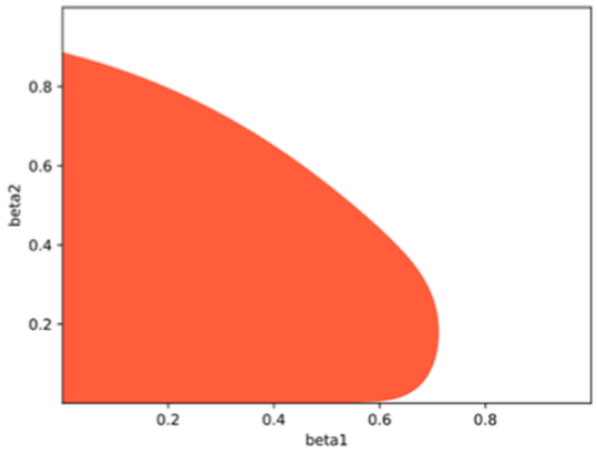
Note that $D_0 = 0$ for over-parameterized models.

Negative Results: $\exists\, f$ so that as long as (β_1, β_2) lies in the red region (depends on n), Adam diverges.

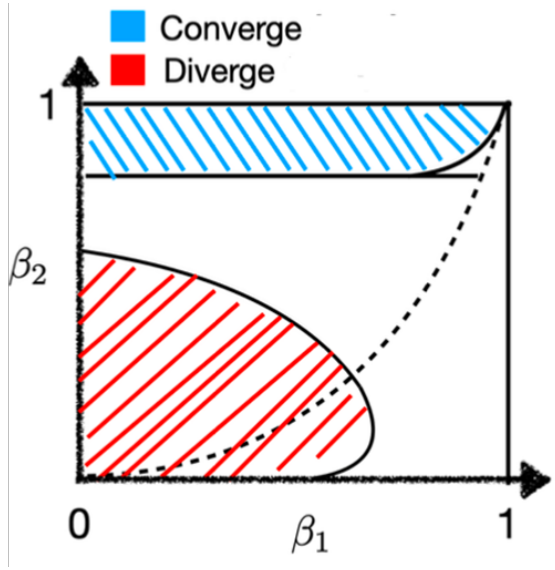
Phase Transition: Converge to Diverge



(b) $n = 10$



(d) $n = 100$



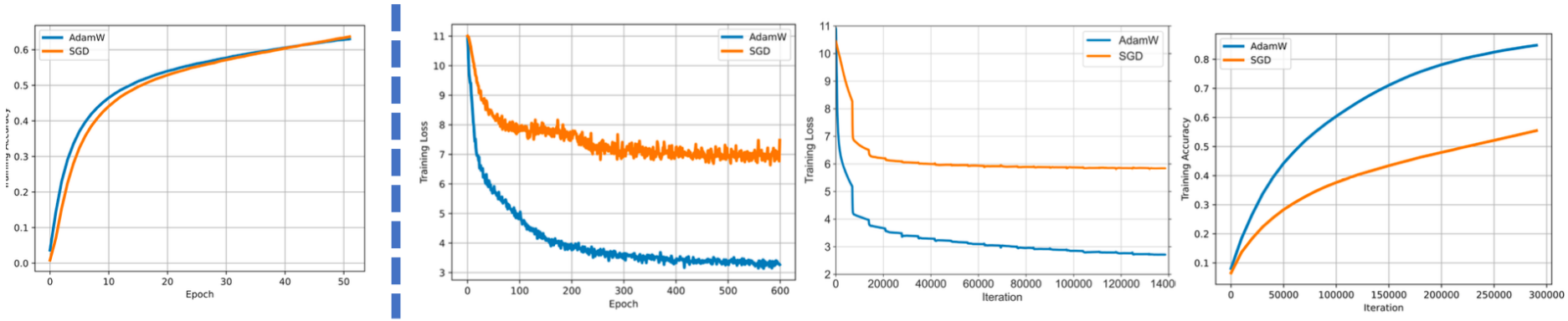
Insight:

- increase β_2 if you have smaller batch size;
- first tune up β_2, then try different β_1 from large to small.

Adam $\gg$ SGDM on Transformer

It is good to have convergence guarantee, but it is not enough. In optimization, convergence is just a basic requirement.

No acceleration in general. Need more problem structure for acceleration.



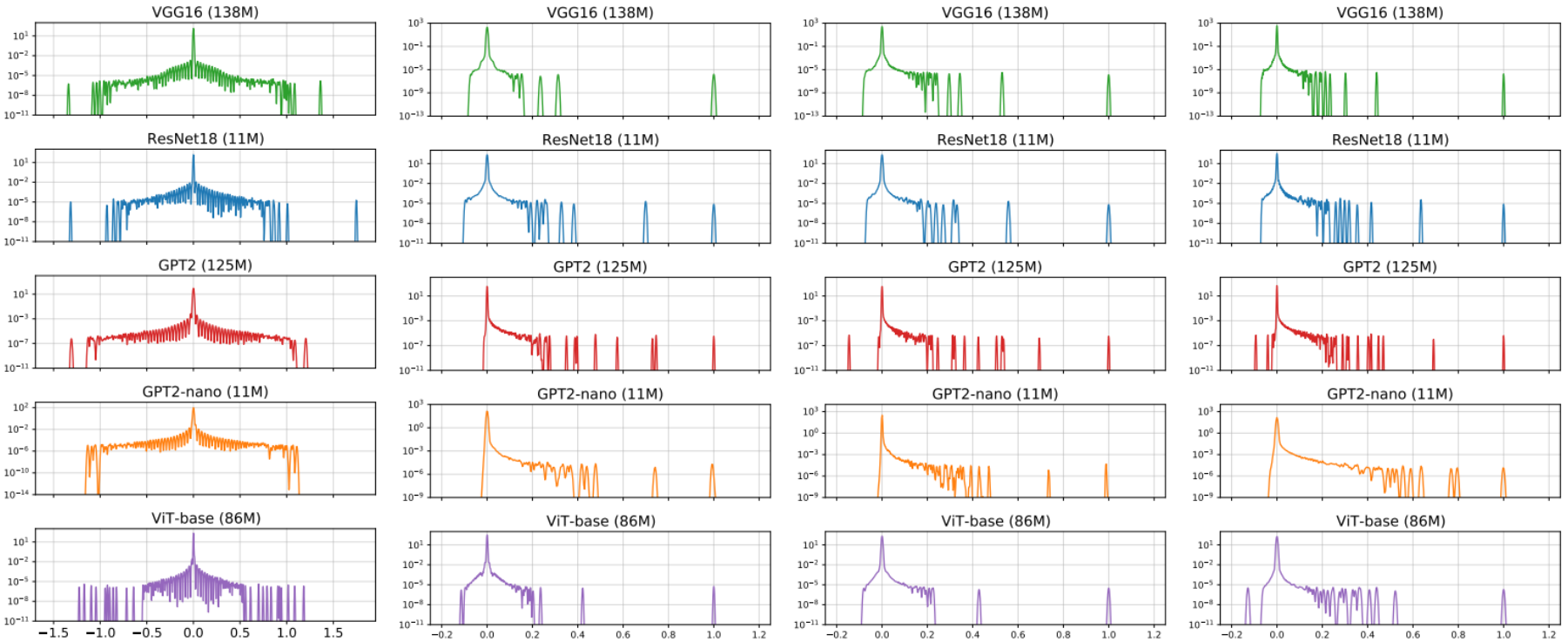
VGG (CNN) GPT2 BERT ViT

Transformers Need Adam, but why? What structure does transformers have while CNN doesn't that make Adam $\gg$ SGDM?

Spectrum of Hessian

Hessian eigenvalues (spectrum) largely decide behavior of gradient methods: ill-conditioning slow down GD.

How to plot Hessian spectrum? Stochastic Lanczos Quadrature (SLQ).

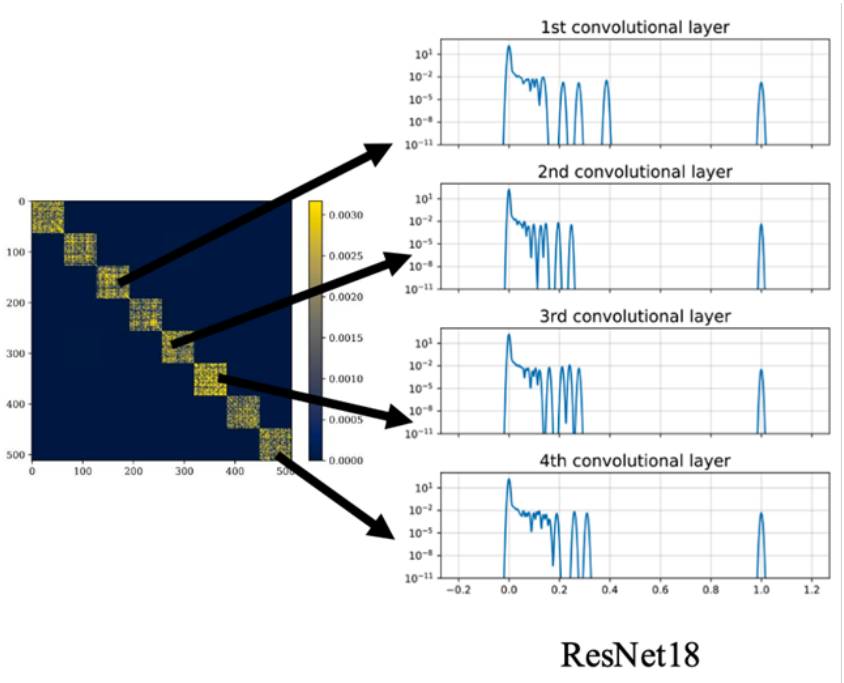


(a) Initialization (b) 25% steps (c) 50% steps (d) 100% steps

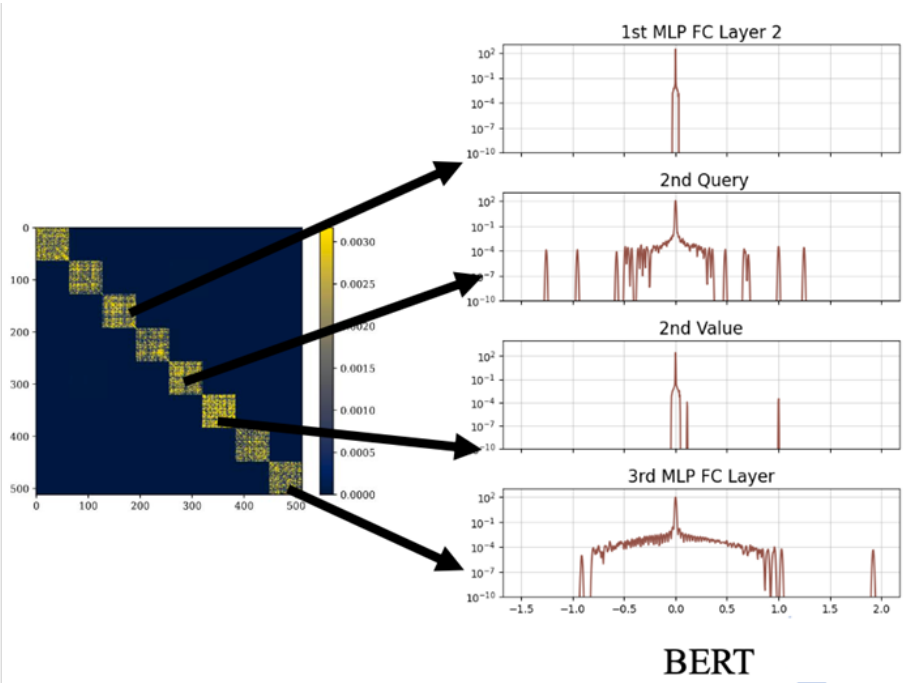
CNN and Transformers: Spectrum looks quite similar!

Heterogeneity makes SGD worse

Major difference of Adam with SGD: its diagonal preconditioner $\sqrt{v_k}$.



ResNet18

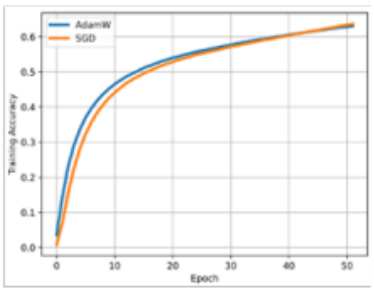


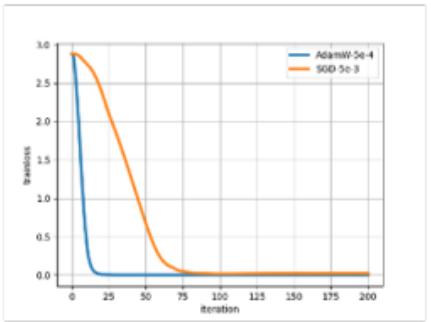
BERT

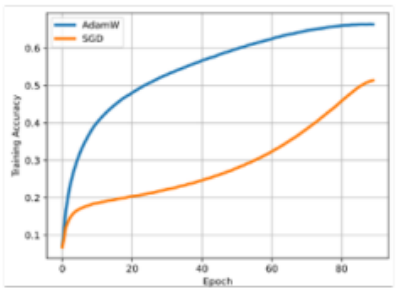
Jensen–Shannon (JS) divergence

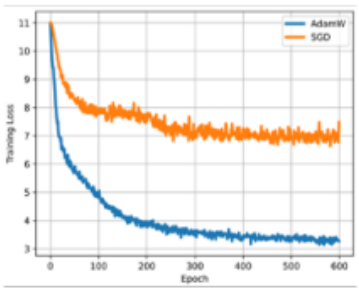
A quantitative way to describe heterogenity is to use JS divergence:

$$J(P\|Q) = (\mathrm{KL}(P\|M) + \mathrm{KL}(Q\|M))/2, \quad M = (P+Q)/2.$$



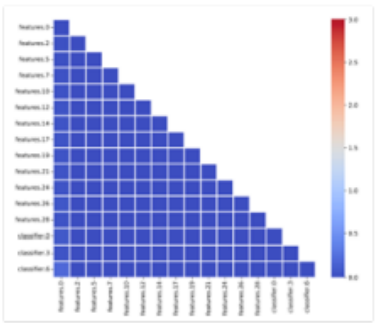




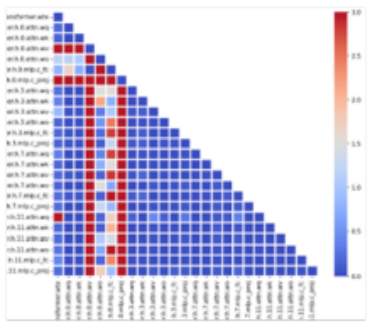


Easy for SGD — Difficult for SGD

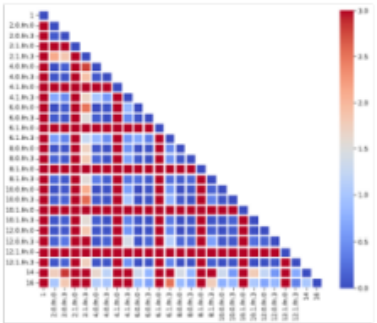
Homogeneity in Hessian — Heterogeneity in Hessian



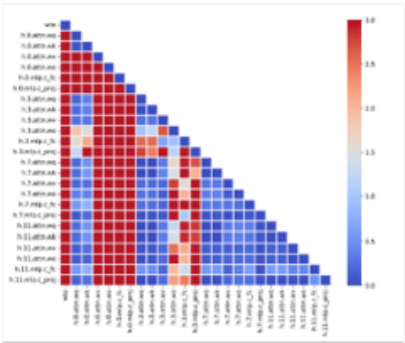
VGG16
$JS^0 = 0.09$



GPT2 (pretrained)
$JS^0 = 18.84$



MLP-mixer
$JS^0 = 34.90$



GPT2
$JS^0 = 83.23$

Case Study: Quadratic Functions

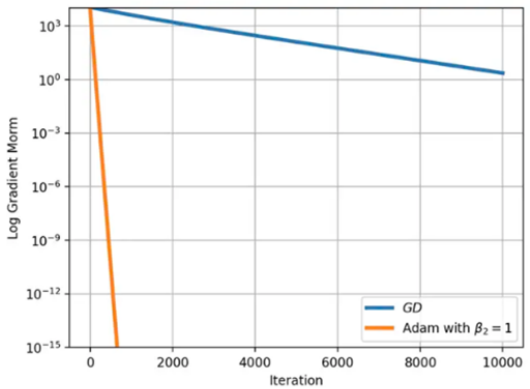
Consider a toy example:

$$\min_w \ \frac{1}{2} w^\top H w, \quad \text{where } H = \text{diag}(H_1, H_2, H_3),\ H_i \in \mathbb{R}^3 \text{ is P.D.}$$

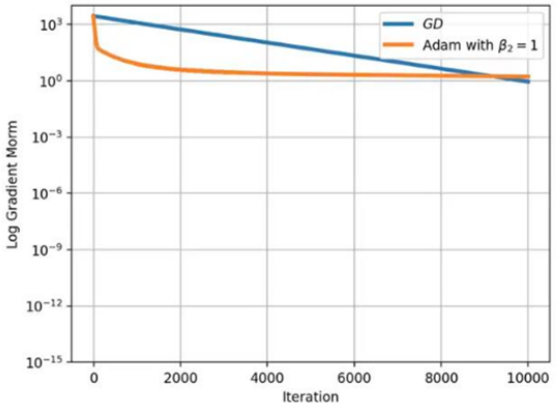
Heterogeneous: Eigenvalues of H_i is: $\{1, 2, 3\}$, $\{99, 100, 101\}$, $\{1998, 1999, 2000\}$.

Homogeneous: Eigenvalues of H_i is: $\{1, 99, 1998\}$, $\{2, 100, 1999\}$, $\{3, 101, 2000\}$.

All eigenvalues are the SAME for the above two cases, but the performance is different, due to homo & heterogeneity.



GD is slower than Adam



GD is similar as Adam

Adam-mini

Hessian structure analysis is not just for explaining things, but also for motivating new design of algorithm.

Adam is NOT effective when H is dense, redundant design on $1/\sqrt{v_t}$.

[Zhang et al., 2024] propose Adam-mini to save memory.

- partition the gradient g into B sub-vectors according to the dense Hessian sub-block g_b, $b = 1, \ldots, B$.
- for each g_b, update with $v_b \in \mathbb{R}$

$$v_b = (1 - \beta_2) \cdot \texttt{mean}(g_b \odot g_b) + \beta_2 \cdot v_b$$

This removes 99.9% dimension of v in Adam, but the final 0.1% is crucial for performance. Overall reduce 50% of the memory.

Adam-mini closely tracks the performance of AdamW, and sometimes even performs better than AdamW.

Wrap-up and Next Week

Week 7: Key takeaways

- SGDM & SNAG, and their convergence rate.
- Effective step sizes.
- Adaptive algorithms including Adagrad, RMSProp & Adam.
- Convergence of Adam and parameter tuning.
- Why Adam performs much better than SGDM?

Thank you for your attendance!

Questions?

Next week: Review (Mar 16) & Midterm (Mar 18)

References I

Dang, A., Babanezhad, R., and Vaswani, S. (2024).
(accelerated) noise-adaptive stochastic heavy-ball momentum.
arXiv preprint arXiv:2401.06738.

Défossez, A., Bottou, L., Bach, F., and Usunier, N. (2020).
A simple convergence proof of adam and adagrad.
arXiv preprint arXiv:2003.02395.

Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research, 12(7).

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

References II

Kingma, D. P. and Ba, J. (2014).

Adam: A method for stochastic optimization.

arXiv preprint arXiv:1412.6980.

Li, X., Liu, M., and Orabona, F. (2022).

On the last iterate convergence of momentum methods.

In *International Conference on Algorithmic Learning Theory*, pages 699–717. PMLR.

Liu, J. and Yuan, Y. (2022).

On almost sure convergence rates of stochastic gradient methods.

In *Conference on Learning Theory*, pages 2963–2983. PMLR.

Mai, V. and Johansson, M. (2020).

Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization.

In *International conference on machine learning*, pages 6630–6639. PMLR.

References III

Pan, R., Liu, Y., Wang, X., and Zhang, T. (2023).

Accelerated convergence of stochastic heavy ball method under anisotropic gradient noise.

arXiv preprint arXiv:2312.14567.

Reddi, S. J., Kale, S., and Kumar, S. (2019).

On the convergence of adam and beyond.

arXiv preprint arXiv:1904.09237.

Sebbouh, O., Gower, R. M., and Defazio, A. (2021).

Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball.

In *Conference on Learning Theory*, pages 3935–3971. PMLR.

Shi, N., Li, D., Hong, M., and Sun, R. (2021).

Rmsprop converges with proper hyper-parameter.

In *International Conference on Learning Representations*.

References IV

Wang, B., Zhang, H., Ma, Z., and Chen, W. (2023).

Convergence of adagrad for non-convex objectives: Simple proofs and relaxed assumptions.

In *The Thirty Sixth Annual Conference on Learning Theory*, pages 161–190. PMLR.

Yan, Y., Yang, T., Li, Z., Lin, Q., and Yang, Y. (2018).

A unified analysis of stochastic momentum methods for deep learning.

arXiv preprint arXiv:1808.10396.

Zhang, Y., Chen, C., Li, Z., Ding, T., Wu, C., Kingma, D. P., Ye, Y., Luo, Z.-Q., and Sun, R. (2024).

Adam-mini: Use fewer learning rates to gain more.

arXiv preprint arXiv:2406.16793.

Zhang, Y., Chen, C., Shi, N., Sun, R., and Luo, Z.-Q. (2022).

Adam can converge without any modification on update rules.

Advances in neural information processing systems, 35:28386–28399.