Optimization for Machine Learning

Lecture 20: Optimization for Neural networks — II

6.881: MIT

Suvrit Sra
Massachusetts Institute of Technology

May 06, 2021

Some Aspects of NN Optimization

- Backprop ➠ SGD
- Mini-batches
- Initialization
- Batchnorm
- Gradient clipping
- Adaptive methods
- Momentum
- Layerwise params
- …and more!

Some Aspects of NN Optimization

- Backprop ➠ SGD
- Mini-batches
- Initialization
- Batchnorm
- Gradient clipping
- Adaptive methods
- Momentum
- Layerwise params
- …and more!

All while keeping validation / test error performance in mind

Clipping, Adaptivity, Momentum Saddle Points, etc.,...

Gradient Clipping

("hack" for dealing with gradient explosion)

Gradient Clipping

("hack" for dealing with gradient explosion)

- Clipped GD can converge arbitrarily faster than fixed-step GD (for differentiable but non C_L^1 functions)

Clipped GD and Normalized GD

Clipped GD and Normalized GD

Clipped GD

$$x_{k+1} = x_k - \min\left(\eta, \frac{a\eta}{\|\nabla f(x_k)\|}\right)\nabla f(x_k)$$

Clipped GD and Normalized GD

Clipped GD

$$x_{k+1} = x_k - \min\left(\eta, \frac{a\eta}{\|\nabla f(x_k)\|}\right)\nabla f(x_k)$$

Normalized GD

$$x_{k+1} = x_k - \frac{\eta}{\|\nabla f(x_k)\| + b}\nabla f(x_k)$$

Clipped GD and Normalized GD

Clipped GD

$$x_{k+1} = x_k - \min\left(\eta, \frac{a\eta}{\|\nabla f(x_k)\|}\right)\nabla f(x_k)$$

Normalized GD

$$x_{k+1} = x_k - \frac{\eta}{\|\nabla f(x_k)\| + b}\nabla f(x_k)$$

Exercise: Show that clipped GD ~ NGD up to a const factor in step size

Clipped GD and Normalized GD

Clipped GD

$$x_{k+1} = x_k - \min\left(\eta, \frac{a\eta}{\|\nabla f(x_k)\|}\right)\nabla f(x_k)$$

Normalized GD

$$x_{k+1} = x_k - \frac{\eta}{\|\nabla f(x_k)\| + b}\nabla f(x_k)$$

Exercise: Show that clipped GD ~ NGD up to a const factor in step size

Clipping helps (and is used more widely) in more **nonsmooth-like** and noisy regimes such as language modeling.

Clipped GD and Normalized GD

Clipped GD

$$x_{k+1} = x_k - \min\left(\eta, \frac{a\eta}{\|\nabla f(x_k)\|}\right)\nabla f(x_k)$$

Normalized GD

$$x_{k+1} = x_k - \frac{\eta}{\|\nabla f(x_k)\| + b}\nabla f(x_k)$$

Exercise: Show that clipped GD ~ NGD up to a const factor in step size

Clipping helps (and is used more widely) in more **nonsmooth-like** and noisy regimes such as language modeling.

WHY GRADIENT CLIPPING ACCELERATES TRAINING: A THEORETICAL JUSTIFICATION FOR ADAPTIVITY

Jingzhao Zhang, Tianxing He, Suvrit Sra & Ali Jadbabaie
Massachusetts Institute of Technology

Clipped GD

L-smoothness $\|\nabla^2 f(x)\| \le L$

Clipped GD

L-smoothness $$\|\nabla^2 f(x)\| \leq L$$

(L,M)-smoothness $$\|\nabla^2 f(x)\| \leq L_0 + L_1 \|\nabla f(x)\|$$

Clipped GD

Relaxed to once differentiable

$$\limsup_{\delta \to \vec{0}} \frac{\|\nabla f(x) - \nabla f(x+\delta)\|}{\|\delta\|} \leq L_1 \|\nabla f(x)\| + L_0.$$

L-smoothness $\|\nabla^2 f(x)\| \leq L$

(L,M)-smoothness $\|\nabla^2 f(x)\| \leq L_0 + L_1 \|\nabla f(x)\|$

Clipped GD

Relaxed to once differentiable

$$\limsup_{\delta \to \vec{0}} \frac{\|\nabla f(x) - \nabla f(x+\delta)\|}{\|\delta\|} \le L_1 \|\nabla f(x)\| + L_0.$$

L-smoothness $\quad \|\nabla^2 f(x)\| \le L$

(L,M)-smoothness $\quad \|\nabla^2 f(x)\| \le L_0 + L_1 \|\nabla f(x)\|$

Clipped GD

Relaxed to once differentiable

L-smoothness $\|\nabla^2 f(x)\| \leq L$

$$\limsup_{\delta \to \vec{0}} \frac{\|\nabla f(x) - \nabla f(x+\delta)\|}{\|\delta\|} \leq L_1 \|\nabla f(x)\| + L_0.$$

(L,M)-smoothness $\|\nabla^2 f(x)\| \leq L_0 + L_1 \|\nabla f(x)\|$

Example: Consider a univariate polynomial of degree ≥ 3

Clipped GD

Relaxed to once differentiable

$$\limsup_{\delta\to\vec{0}} \frac{\|\nabla f(x)-\nabla f(x+\delta)\|}{\|\delta\|} \le L_1\|\nabla f(x)\| + L_0.$$

L-smoothness $\|\nabla^2 f(x)\| \le L$

(L,M)-smoothness $\|\nabla^2 f(x)\| \le L_0 + L_1\|\nabla f(x)\|$

Example: Consider a univariate polynomial of degree ≥ 3

Theorem (informal). GD can be arbitrarily slow to converge to a stationary point for functions satisfying $(L_0, L_1)-$smoothness, whereas Clipped GD converges as $O(1/\epsilon^2)$

Clipped GD

Relaxed to once differentiable

L-smoothness $\|\nabla^2 f(x)\| \leq L$

$$\limsup_{\delta \to \vec{0}} \frac{\|\nabla f(x) - \nabla f(x+\delta)\|}{\|\delta\|} \leq L_1 \|\nabla f(x)\| + L_0.$$

(L,M)-smoothness $\|\nabla^2 f(x)\| \leq L_0 + L_1 \|\nabla f(x)\|$

Example: Consider a univariate polynomial of degree ≥ 3

Theorem (informal). GD can be arbitrarily slow to converge to a stationary point for functions satisfying $(L_0, L_1)-$smoothness, whereas Clipped GD converges as $O(1/\epsilon^2)$

Exercise: Analyze convergence of all other methods under (L_0, L_1)-smoothness for which we previously assumed L-smoothness.

Clipped GD

Clipped GD converges at "usual" speed

Theorem 3. *Let $\mathcal{F}$ denote the class of functions that satisfy Assumptions 1, 2, and 3 in set $\mathcal{S}$ defined in (3). Recall f^* is a global lower bound for function value. With $\eta_c = \frac{1}{10L_0}, \gamma = \min\{\frac{1}{\eta_c}, \frac{1}{10L_1\eta_c}\}$, we can prove that the iteration complexity of clipped GD (Algorithm 5) is upper bounded by*

$$\frac{20L_0(f(x_0) - f^*)}{\epsilon^2} + \frac{20\max\{1, L_1^2\}(f(x_0) - f^*)}{L_0}.$$

GD can be arbitrarily slower

Theorem 4. *Let $\mathcal{F}$ be the class of objectives satisfying Assumptions 1, 2, 3, and 4 with fixed constants $L_0 \geq 1$, $L_1 \geq 1$, $M > 1$. The iteration complexity for the fixed-step gradient descent algorithms parameterized by step size h is at least*

$$\frac{L_1 M(f(x_0) - f^* - 5\epsilon/8)}{8\epsilon^2(\log M + 1)}.$$

Explore: Lower bound for SGD (afaik unknown in this setting)

Momentum, Adaptivity

(Adam, Adam-like methods)

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta m_t$$

https://distill.pub/2017/momentum/

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta m_t$$

https://distill.pub/2017/momentum/

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t - \eta m_t$$

- handle ill-conditioning
- accelerated convg. (eg, convex quadratics)
- works well in practice
- still subject of research esp due to great success in deep learning

https://distill.pub/2017/momentum/

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta m_t$$

Unroll gradient descent

$$\theta_{t+1} = \theta_0 - \eta \sum_{i=1}^{t} \nabla f(\theta_i)$$

- handle ill-conditioning
- accelerated convg. (eg, convex quadratics)
- works well in practice
- still subject of research esp due to great success in deep learning

https://distill.pub/2017/momentum/

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta m_t$$

Unroll gradient descent

$$\theta_{t+1} = \theta_0 - \eta \sum_{i=1}^{t} \nabla f(\theta_i)$$

Unroll GD with momentum

$$\theta_{t+1} = \theta_0 - \eta \sum_{i=1}^{t} \frac{1 - \beta^{t+1-i}}{\beta - 1} \nabla f(\theta_i)$$

- handle ill-conditioning
- accelerated convg. (eg, convex quadratics)
- works well in practice
- still subject of research esp due to great success in deep learning

https://distill.pub/2017/momentum/

Momentum

Gradient Descent with Momentum

$$m_t = \beta m_{t-1} + \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta m_t$$

Unroll gradient descent

$$\theta_{t+1} = \theta_0 - \eta \sum_{i=1}^{t} \nabla f(\theta_i)$$

Unroll GD with momentum

$$\theta_{t+1} = \theta_0 - \eta \sum_{i=1}^{t} \frac{1-\beta^{t+1-i}}{\beta - 1} \nabla f(\theta_i)$$

- handle ill-conditioning
- accelerated convg. (eg, convex quadratics)
- works well in practice
- still subject of research esp due to great success in deep learning

https://distill.pub/2017/momentum/

Adaptive gradients

Adaptive gradients

Scaled Gradient Method

$$\theta_{t+1} = \theta_t - G_t^{-1/2} \nabla f(x_t)$$

Adaptive gradients

Scaled Gradient Method

$$\theta_{t+1} = \theta_t - G_t^{-1/2} \nabla f(x_t)$$

Adagrad $$G_t = \sum_{i=1}^{t} g_i g_i^T$$ (typically just $Diag(G_t)$ used)

Adaptive gradients

Scaled Gradient Method

$$\theta_{t+1} = \theta_t - G_t^{-1/2} \nabla f(x_t)$$

Adagrad $$G_t = \sum_{i=1}^{t} g_i g_i^T$$ (typically just $Diag(G_t)$ used)

Adagrad originally proposed to benefit from sparse data, an assumption not true for neural network training in general. **Con:** Can shrink learning rate too fast.

Adaptive gradients

Scaled Gradient Method

$$\theta_{t+1} = \theta_t - G_t^{-1/2} \nabla f(x_t)$$

Adagrad $$G_t = \sum_{i=1}^{t} g_i g_i^T$$ (typically just $Diag(G_t)$ used)

Adagrad originally proposed to benefit from sparse data, an assumption not true for neural network training in general. **Con:** Can shrink learning rate too fast.

Adaptive gradients

Scaled Gradient Method

$$\theta_{t+1} = \theta_t - G_t^{-1/2} \nabla f(x_t)$$

Adagrad $$G_t = \sum_{i=1}^{t} g_i g_i^T$$ (typically just $Diag(G_t)$ used)

Adagrad originally proposed to benefit from sparse data, an assumption not true for neural network training in general. **Con:** Can shrink learning rate too fast.

Idea: Exponential moving averages $\beta \in (0,1)$

$$G_t = (1-\beta) \sum_{i=1}^{t} \beta^{t-i} g_i g_i^T$$

(analogous to what momentum is doing for gradients…)

ADAM

ADAM

Adam's name comes from: Adaptive Moment Estimation

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate $Diag(G_k)$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate ***Diag(G$_k$)***

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate ***Diag(G$_k$)***

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate *Diag*(G_k)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - (G_t^{1/2} + \epsilon I)^{-1} m_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate *Diag(G_k)*

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - (G_t^{1/2} + \epsilon I)^{-1} m_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

bias-correction

$$m_t \leftarrow \frac{m_t}{1-\beta_1^t} \qquad v_t \leftarrow \frac{v_t}{1-\beta_2^t}$$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate *Diag(G_k)*

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - (G_t^{1/2} + \epsilon I)^{-1} m_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

bias-correction

$$m_t \leftarrow \frac{m_t}{1-\beta_1^t} \qquad v_t \leftarrow \frac{v_t}{1-\beta_2^t}$$

ADAM

Adam's name comes from: Adaptive Moment Estimation

1. Use exponential moving averages to estimate gradients (aka momentum)
2. Use exponential moving averages to estimate ***Diag(G$_k$)***

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

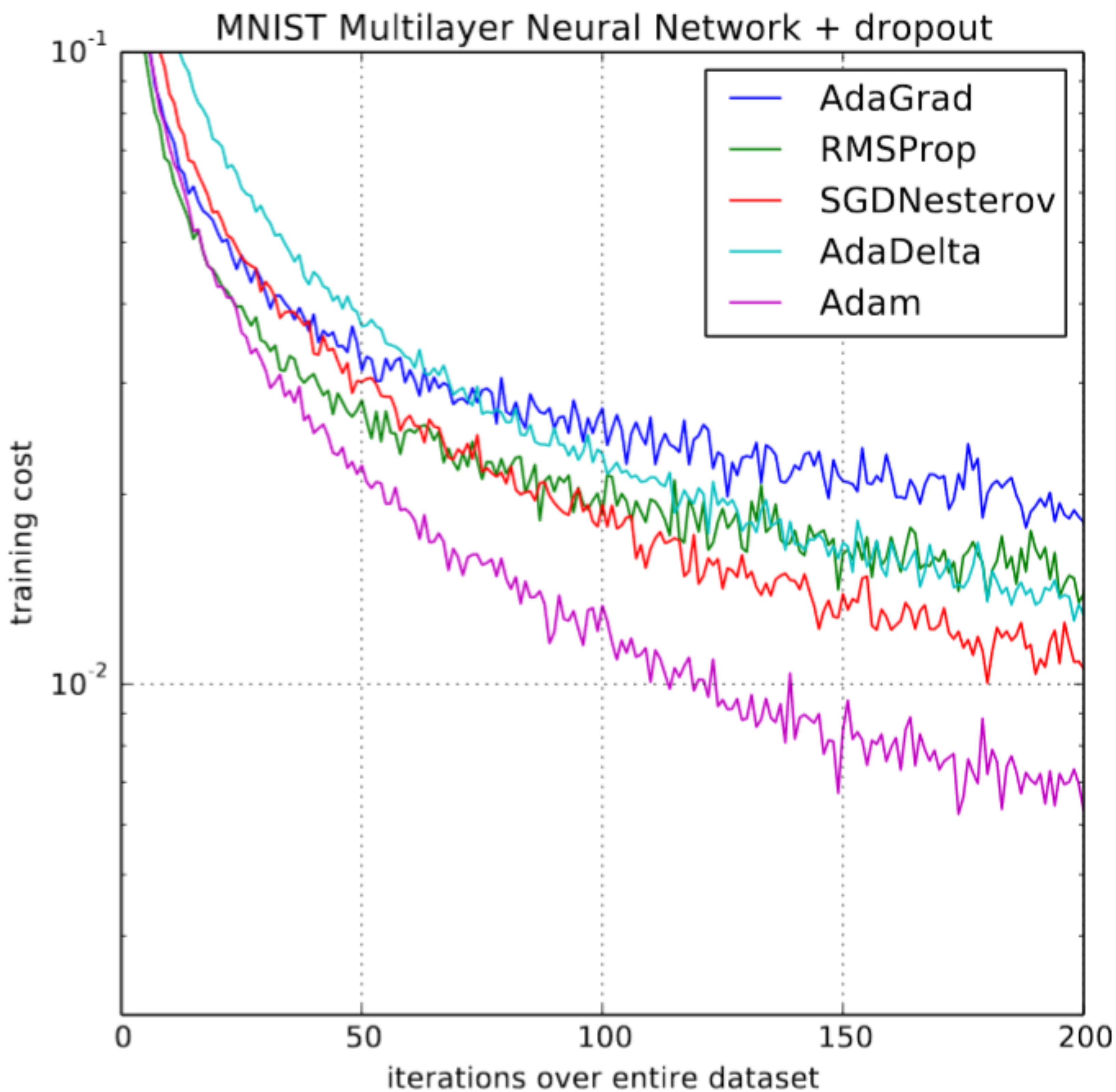
$$\theta_{t+1} = \theta_t - (G_t^{1/2} + \epsilon I)^{-1} m_t$$

$$\boxed{\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t}$$

bias-correction

$$m_t \leftarrow \frac{m_t}{1-\beta_1^t} \qquad v_t \leftarrow \frac{v_t}{1-\beta_2^t}$$

ADAM



From the ADAM paper: https://arxiv.org/abs/1412.6980

Algorithm 2 LAMB https://arxiv.org/abs/1904.00962

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, scaling function ϕ, $\epsilon > 0$

Set $m_0 = 0$, $v_0 = 0$

for $t = 1$ **to** T **do**

  Draw b samples S_t from $\mathbb{P}$.

  Compute $g_t = \frac{1}{|\mathcal{S}_t|} \sum_{s_t \in \mathcal{S}_t} \nabla \ell(x_t, s_t)$.

  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

  $m_t = m_t / (1 - \beta_1^t)$

  $v_t = v_t / (1 - \beta_2^t)$

  Compute ratio $r_t = \frac{m_t}{\sqrt{v_t} + \epsilon}$

  $x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi(\|x_t^{(i)}\|)}{\|r_t^{(i)} + \lambda x_t^{(i)}\|} (r_t^{(i)} + \lambda x_t^{(i)})$

end for

Algorithm 2 LAMB https://arxiv.org/abs/1904.00962

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, scaling function ϕ, $\epsilon > 0$

Set $m_0 = 0$, $v_0 = 0$

for $t = 1$ **to** T **do**

Draw b samples S_t from $\mathbb{P}$.

Compute $g_t = \frac{1}{|\mathcal{S}_t|} \sum_{s_t \in \mathcal{S}_t} \nabla \ell(x_t, s_t)$.

$$\boxed{\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ m_t &= m_t / (1-\beta_1^t) \\ v_t &= v_t / (1-\beta_2^t) \end{aligned}}$$

ADAM

Compute ratio $r_t = \frac{m_t}{\sqrt{v_t} + \epsilon}$

$$x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi(\|x_t^{(i)}\|)}{\|r_t^{(i)} + \lambda x_t^{(i)}\|} (r_t^{(i)} + \lambda x_t^{(i)})$$

end for

Algorithm 2 LAMB https://arxiv.org/abs/1904.00962

Input: $x_1 \in \mathbb{R}^d$, learning rate $\{\eta_t\}_{t=1}^T$, parameters $0 < \beta_1, \beta_2 < 1$, scaling function ϕ, $\epsilon > 0$

Set $m_0 = 0$, $v_0 = 0$

for $t = 1$ **to** T **do**

Draw b samples S_t from $\mathbb{P}$.

Compute $g_t = \frac{1}{|\mathcal{S}_t|} \sum_{s_t \in \mathcal{S}_t} \nabla \ell(x_t, s_t)$.

$$\boxed{\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ m_t &= m_t / (1-\beta_1^t) \\ v_t &= v_t / (1-\beta_2^t) \end{aligned}}$$

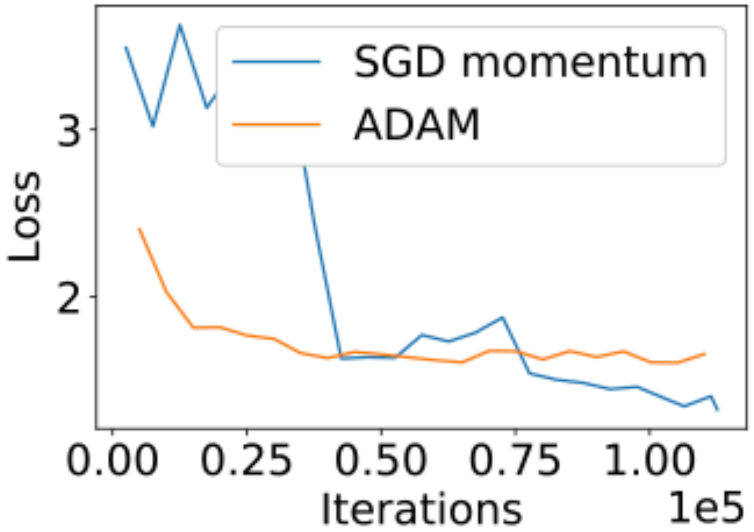
ADAM

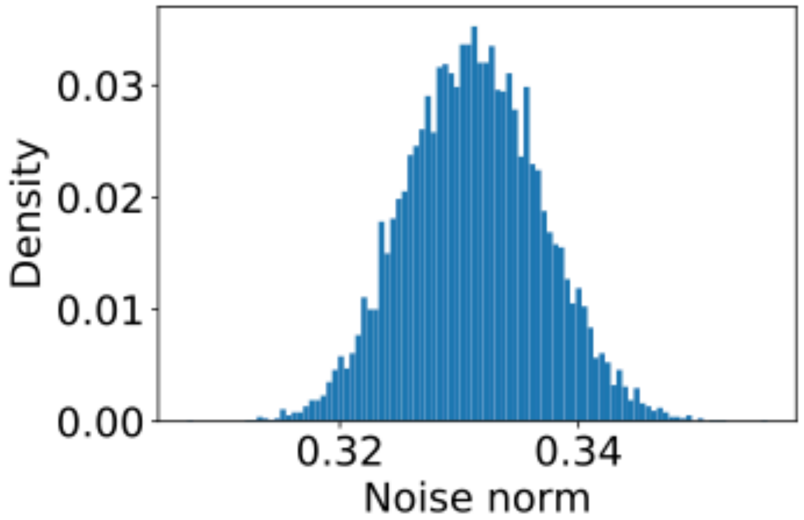
Compute ratio $r_t = \frac{m_t}{\sqrt{v_t} + \epsilon}$

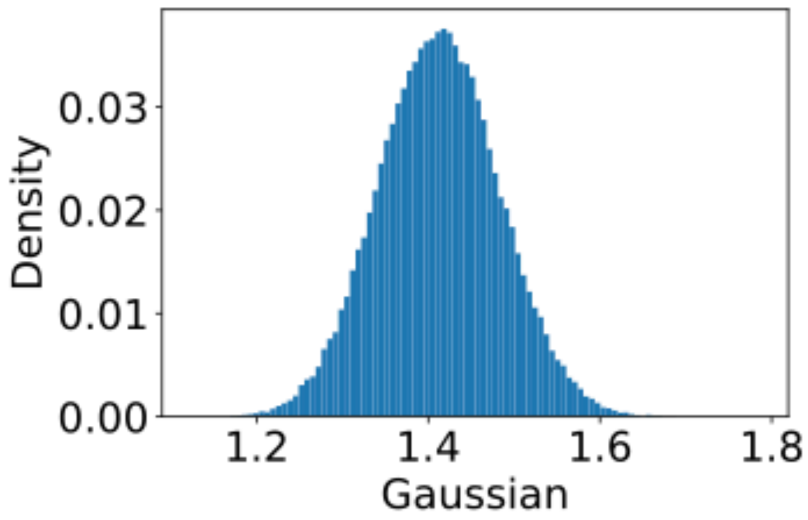
$$\boxed{x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi(\|x_t^{(i)}\|)}{\|r_t^{(i)} + \lambda x_t^{(i)}\|} (r_t^{(i)} + \lambda x_t^{(i)})}$$

end for

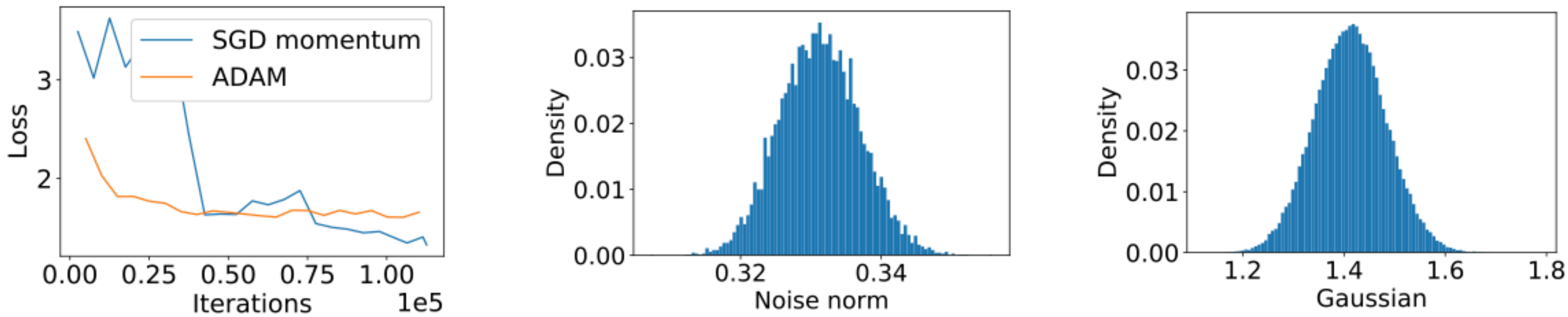
Resnet50 on Imagenet



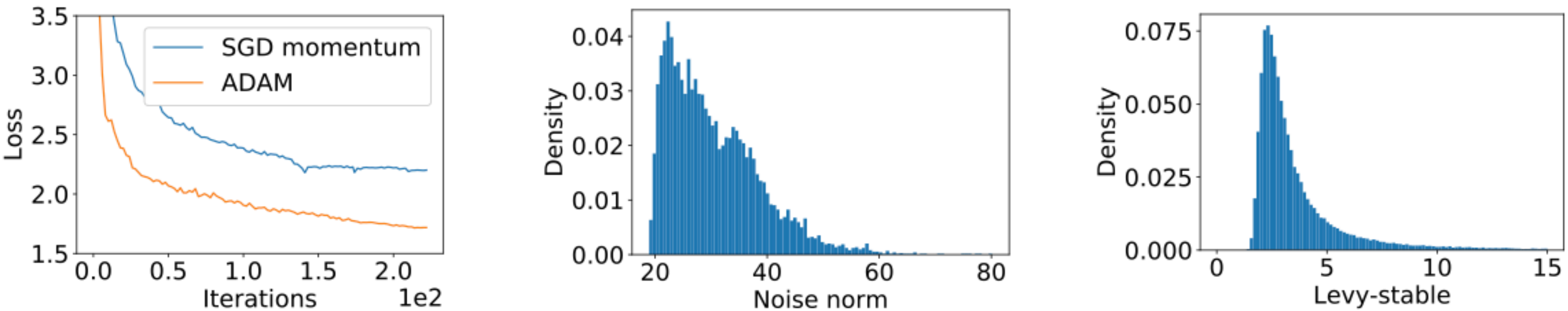




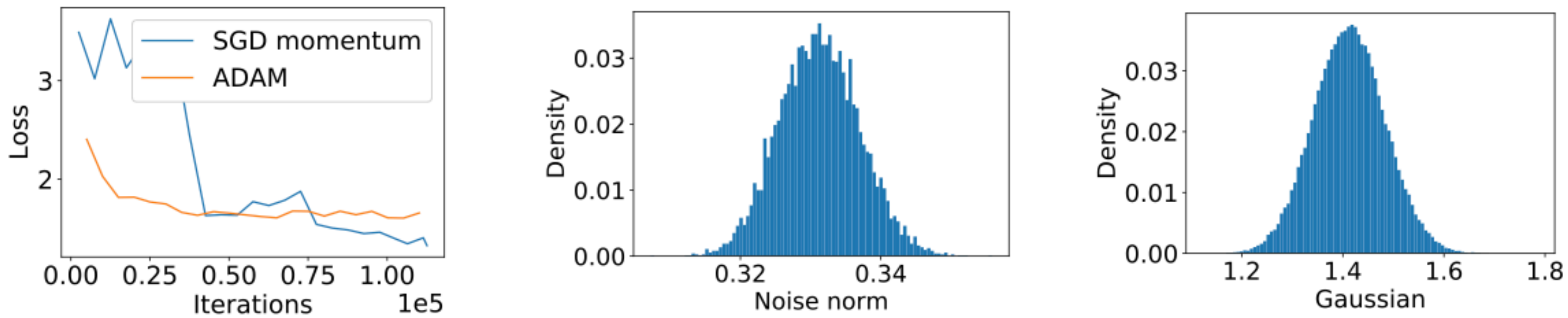
Resnet50 on Imagenet



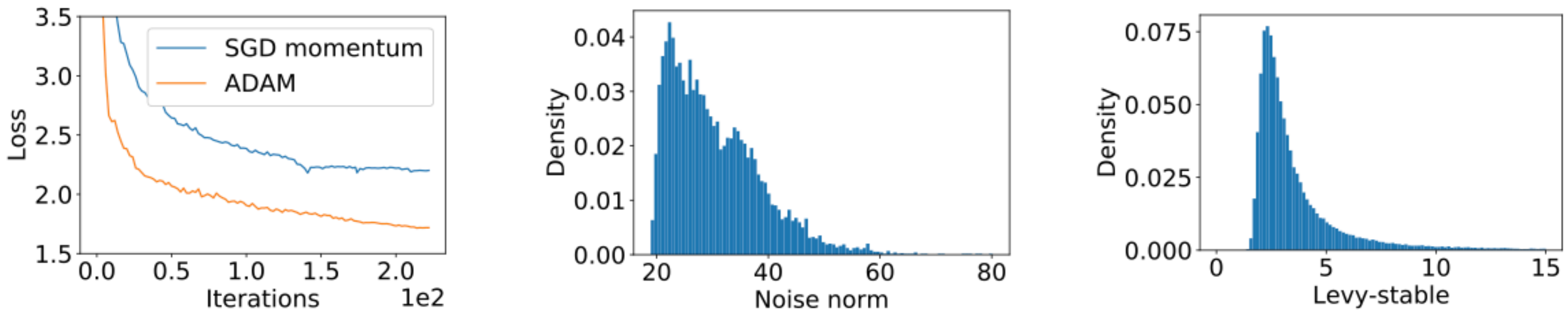
BERT pretraining (Wikipedia+Books data)



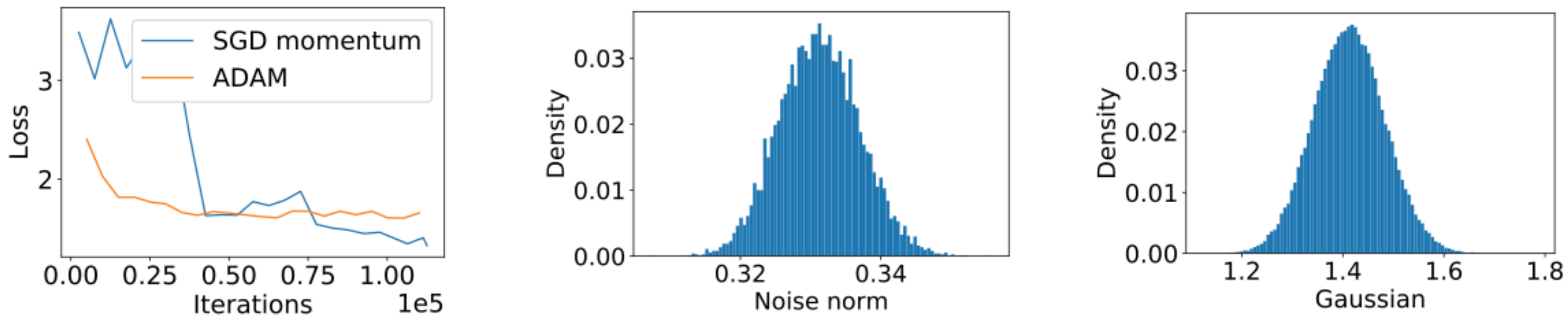
Resnet50 on Imagenet

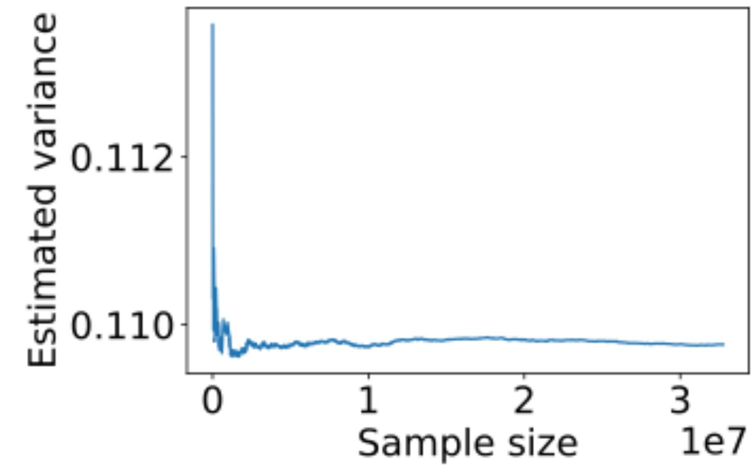


BERT pretraining (Wikipedia+Books data)

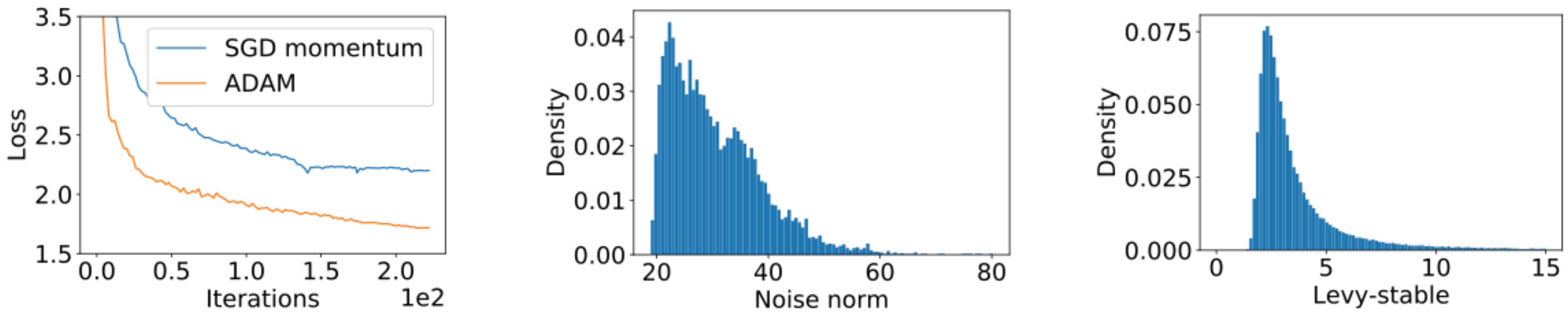


Resnet50 on Imagenet

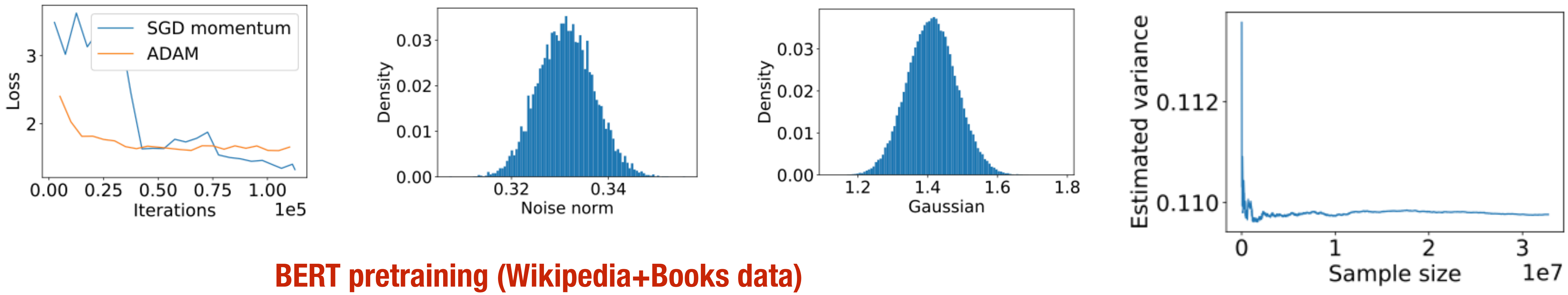




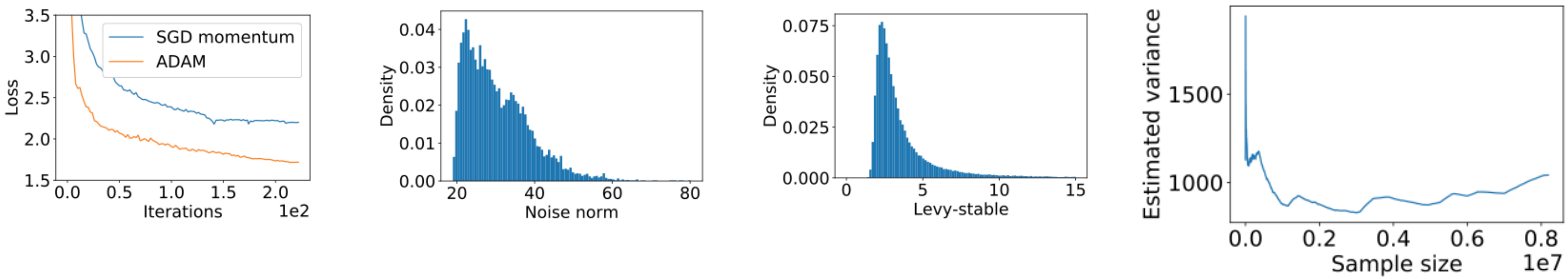
BERT pretraining (Wikipedia+Books data)



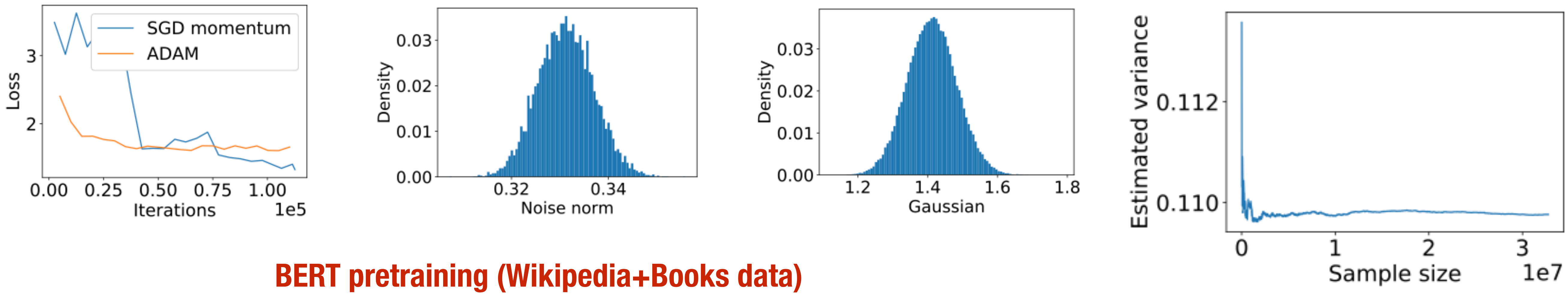
Resnet50 on Imagenet



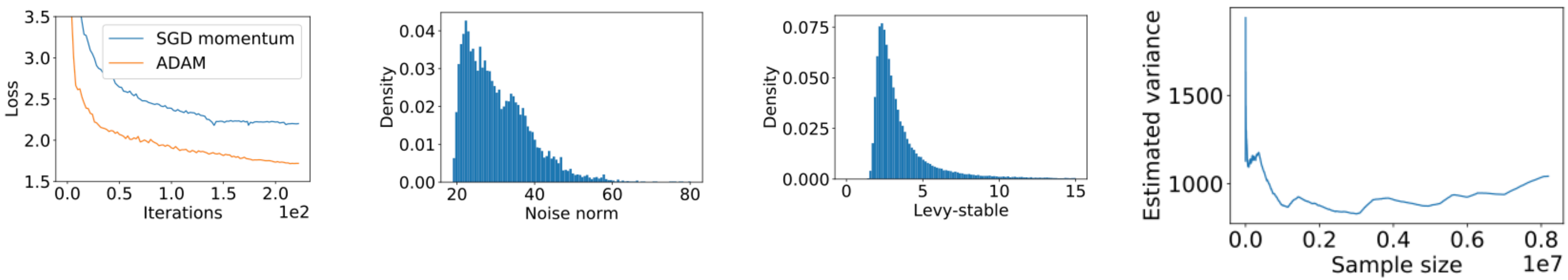
BERT pretraining (Wikipedia+Books data)



Resnet50 on Imagenet



BERT pretraining (Wikipedia+Books data)



The heavy-tailed noise could explain why ADAM works better than SGD?

Convergence rates under heavy-tailed noise

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/A	$\mathcal{O}\left(k^{-1}\right)$	N/A	$\mathcal{O}\left(k^{-\frac{1}{4}}\right)$
GClip	$\mathcal{O}\left(k^{\frac{-(\alpha-1)}{\alpha}}\right)$	$\mathcal{O}\left(k^{-1}\right)$	$\mathcal{O}\left(k^{\frac{-(\alpha-1)}{3\alpha-2}}\right)$	$\mathcal{O}\left(k^{-\frac{1}{4}}\right)$
LowerBound	$\Omega\left(k^{\frac{-(\alpha-1)}{\alpha}}\right)$	$\Omega\left(k^{-1}\right)$	$\Omega\left(k^{\frac{-(\alpha-1)}{3\alpha-2}}\right)$	$\Omega\left(k^{-\frac{1}{4}}\right)$

Error bounds (f-f^* for cvx; $\|\nabla f\|$ for noncvx).

Convergence rates under heavy-tailed noise

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/A	$\mathcal{O}\big(k^{-1}\big)$	N/A	$\mathcal{O}\big(k^{-\frac{1}{4}}\big)$
GClip	$\mathcal{O}\big(k^{\frac{-(\alpha-1)}{\alpha}}\big)$	$\mathcal{O}\big(k^{-1}\big)$	$\mathcal{O}\big(k^{\frac{-(\alpha-1)}{3\alpha-2}}\big)$	$\mathcal{O}\big(k^{-\frac{1}{4}}\big)$
LowerBound	$\Omega\big(k^{\frac{-(\alpha-1)}{\alpha}}\big)$	$\Omega\big(k^{-1}\big)$	$\Omega\big(k^{\frac{-(\alpha-1)}{3\alpha-2}}\big)$	$\Omega\big(k^{-\frac{1}{4}}\big)$

Error bounds (f-f^* for cvx; $\|\nabla f\|$ for noncvx).

$$x_{k+1} = x_k - \eta_k \min\Big\{\frac{\tau_k}{\|g_k\|}, 1\Big\} g_k\,, \;\; \tau_k \in \mathbb{R}_{\geq 0}$$

Convergence rates under heavy-tailed noise

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/A	$\mathcal{O}(k^{-1})$	N/A	$\mathcal{O}(k^{-\frac{1}{4}})$
GClip	$\mathcal{O}(k^{\frac{-(\alpha-1)}{\alpha}})$	$\mathcal{O}(k^{-1})$	$\mathcal{O}(k^{\frac{-(\alpha-1)}{3\alpha-2}})$	$\mathcal{O}(k^{-\frac{1}{4}})$
LowerBound	$\Omega(k^{\frac{-(\alpha-1)}{\alpha}})$	$\Omega(k^{-1})$	$\Omega(k^{\frac{-(\alpha-1)}{3\alpha-2}})$	$\Omega(k^{-\frac{1}{4}})$

Error bounds (f-f^* for cvx; $\|\nabla f\|$ for noncvx).

$$x_{k+1} = x_k - \eta_k \min\{\tfrac{\tau_k}{\|g_k\|}, 1\} g_k\,,\ \tau_k \in \mathbb{R}_{\geq 0} \qquad x_{k+1} = x_k - \eta_k \min\{\tfrac{\tau_k}{|g_k|}, 1\} g_k\,,\ \tau_k \in \mathbb{R}^d_{\geq 0}$$

Convergence rates under heavy-tailed noise

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/A	$\mathcal{O}(k^{-1})$	N/A	$\mathcal{O}(k^{-\frac{1}{4}})$
GClip	$\mathcal{O}(k^{\frac{-(\alpha-1)}{\alpha}})$	$\mathcal{O}(k^{-1})$	$\mathcal{O}(k^{\frac{-(\alpha-1)}{3\alpha-2}})$	$\mathcal{O}(k^{-\frac{1}{4}})$
LowerBound	$\Omega(k^{\frac{-(\alpha-1)}{\alpha}})$	$\Omega(k^{-1})$	$\Omega(k^{\frac{-(\alpha-1)}{3\alpha-2}})$	$\Omega(k^{-\frac{1}{4}})$

Error bounds (f-f for cvx; $\|\nabla f\|$ for noncvx).*

$$x_{k+1} = x_k - \eta_k \min\left\{\frac{\tau_k}{\|g_k\|}, 1\right\} g_k\,,\ \tau_k \in \mathbb{R}_{\geq 0} \qquad x_{k+1} = x_k - \eta_k \min\left\{\frac{\tau_k}{|g_k|}, 1\right\} g_k\,,\ \tau_k \in \mathbb{R}^d_{\geq 0}$$

Check out the details and experiments in

Why are Adaptive Methods Good for Attention Models?

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, Suvrit Sra

Convergence rates under heavy-tailed noise

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/			$\mathcal{O}(k^{-\frac{1}{4}})$
GClip	$\mathcal{O}($			$\mathcal{O}(k^{-\frac{1}{4}})$
LowerBound	$\Omega($			$\Omega(k^{-\frac{1}{4}})$

Algorithm 1 ACClip

$x, m_k \leftarrow x_0, 0$
for $k = 1, \cdot, T$ **do**
$\quad m_k \leftarrow \beta_1 m_{k-1} + (1 - \beta_1) g_k$
$\quad \tau_k^\alpha \leftarrow \beta_2 \tau_{k-1}^\alpha + (1 - \beta_2)|g_k|^\alpha$
$\quad \hat{g}_k \leftarrow \min\{\frac{\tau_k}{|m_k|+\epsilon}, 1\} m_k$
$\quad x_k \leftarrow x_{k-1} - \eta_k \hat{g}_k$
end forreturn x_K, where random

$x_{k+1} = x_k - \eta_k \min\{$ ... $\frac{\tau_k}{|g_k|}, 1\} g_k \,, \; \tau_k \in \mathbb{R}^d_{\geq 0}$

Check out the detai

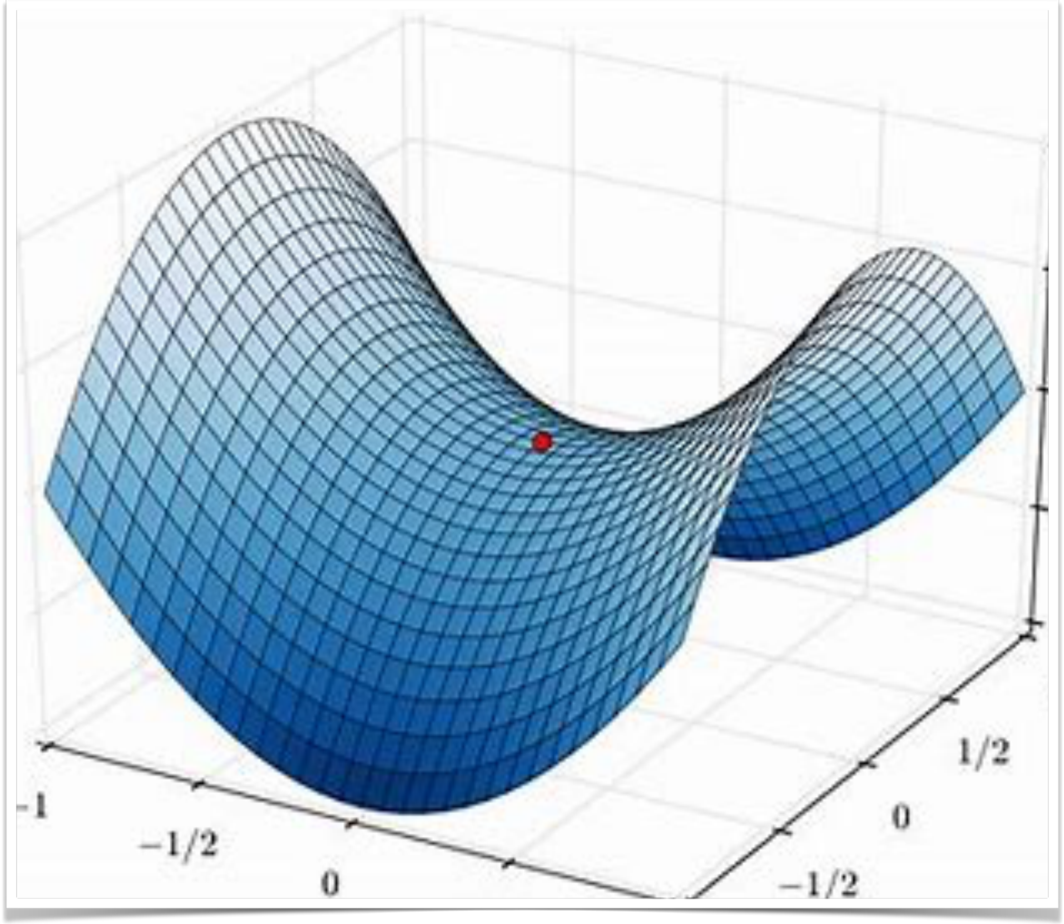
Why are Adaptive Methods Good for Attention Models?

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, Suvrit Sra

Escaping Saddle Points

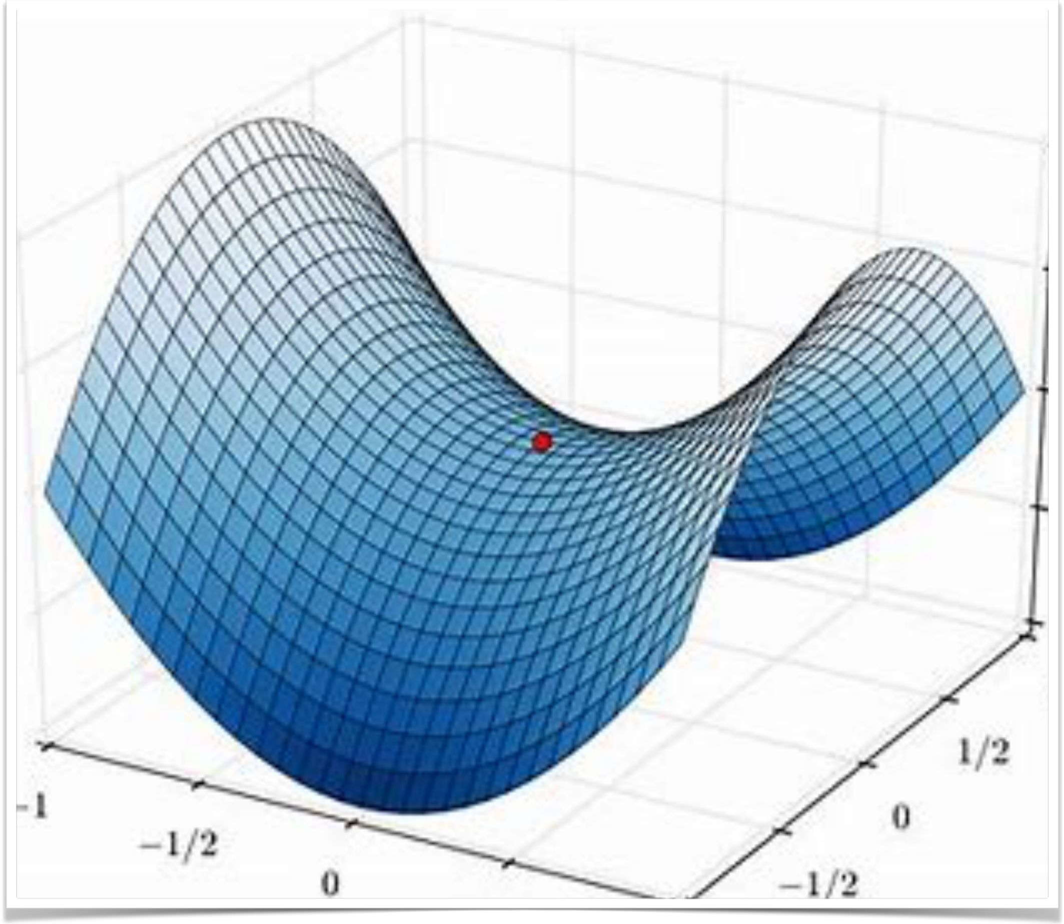
Background

Background



Background

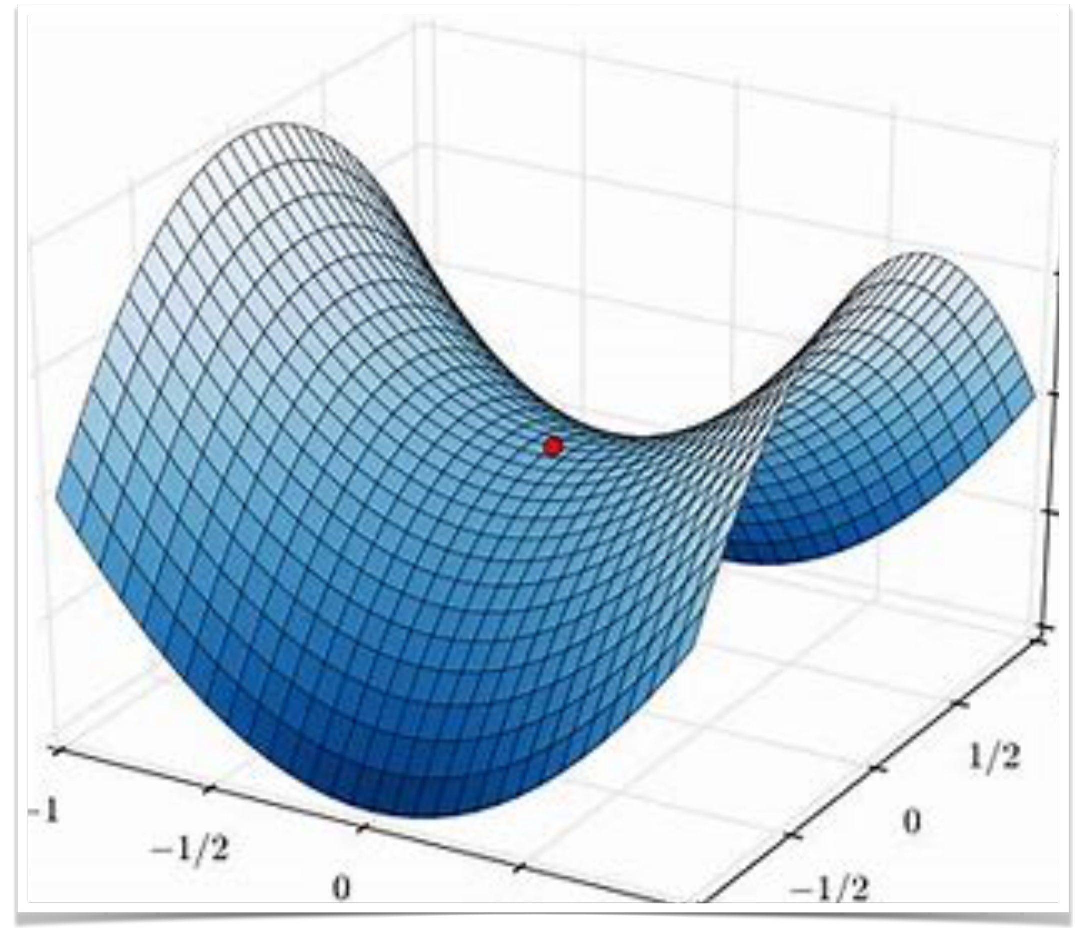
At a stationary point $\nabla f(x) = 0$



Background

At a stationary point $\nabla f(x) = 0$

Local minimum: $\nabla^2 f(x) \succ 0$

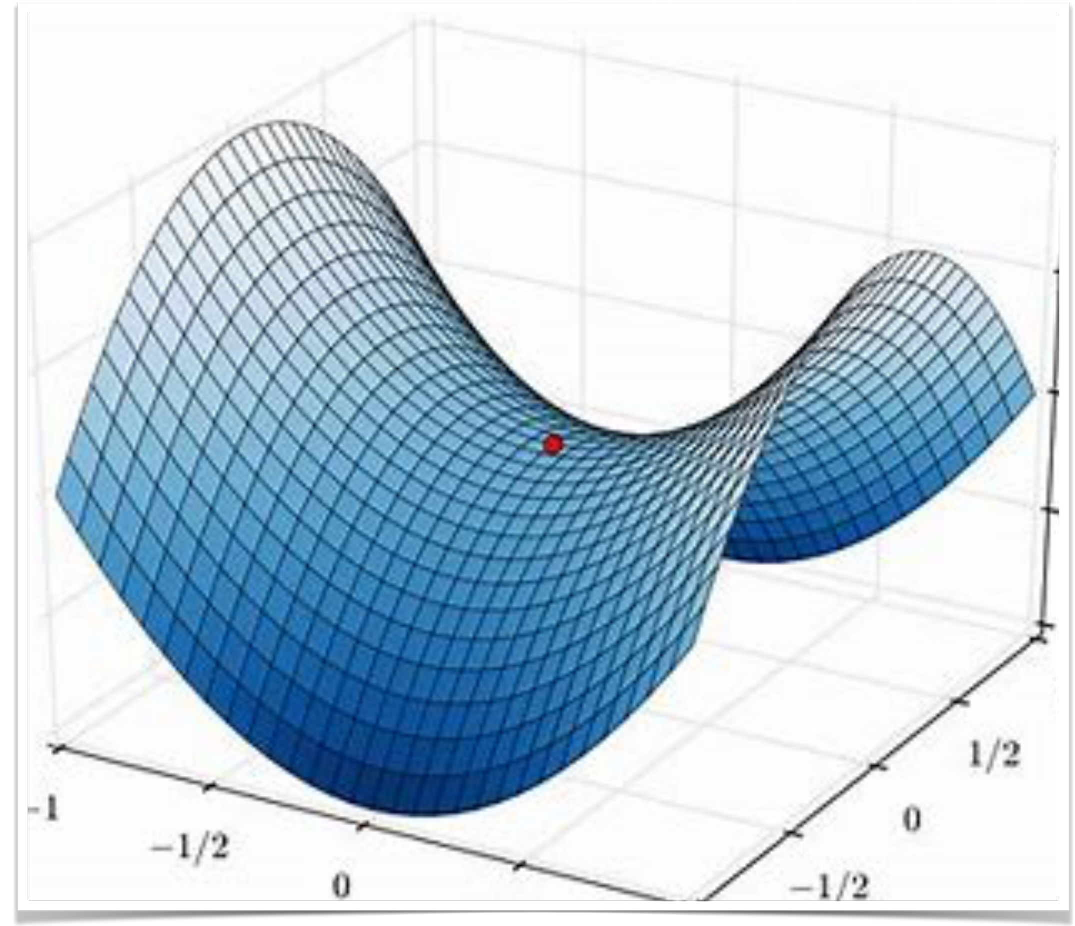


Background

At a stationary point $\nabla f(x) = 0$

Local minimum: $\nabla^2 f(x) \succ 0$

Local minimum: $\nabla^2 f(x) \prec 0$



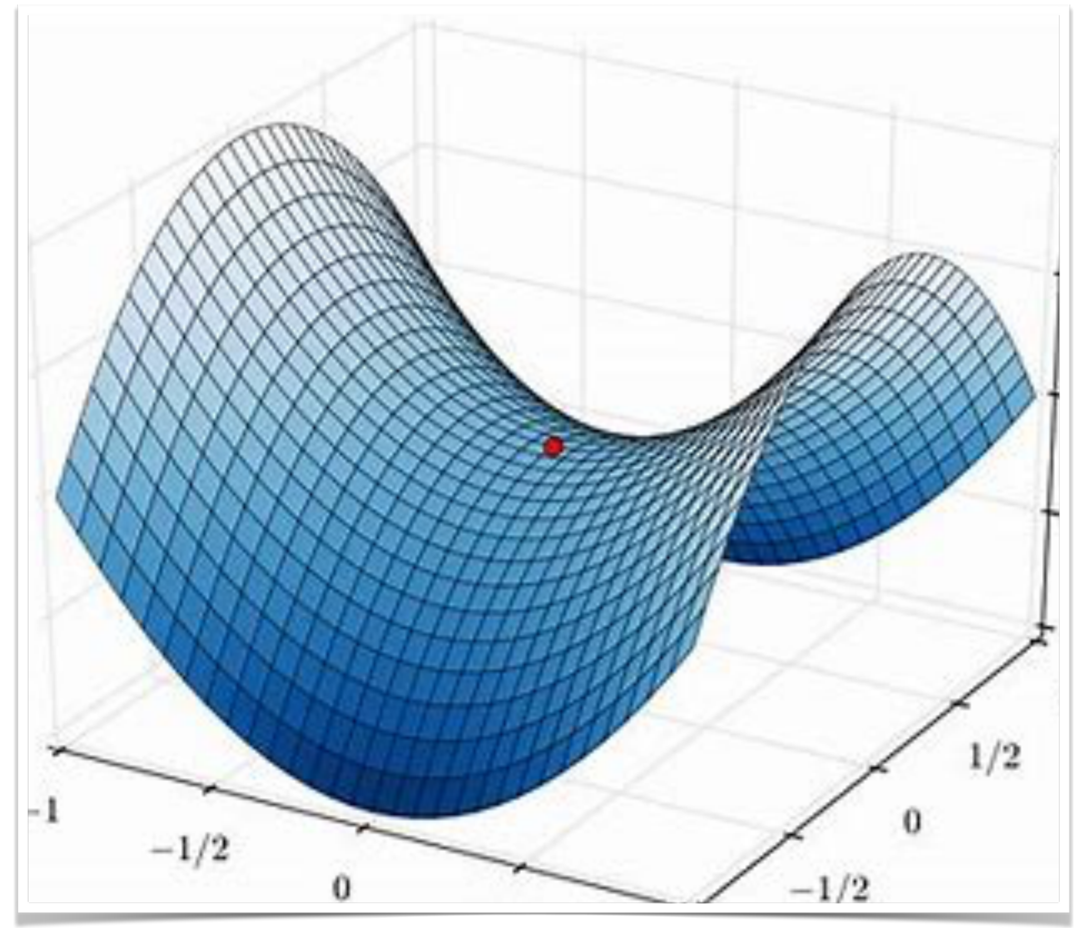
Background

At a stationary point $\nabla f(x) = 0$

Local minimum: $\nabla^2 f(x) \succ 0$

Local minimum: $\nabla^2 f(x) \prec 0$

Almost always stationary point can have an *indefinite* Hessian



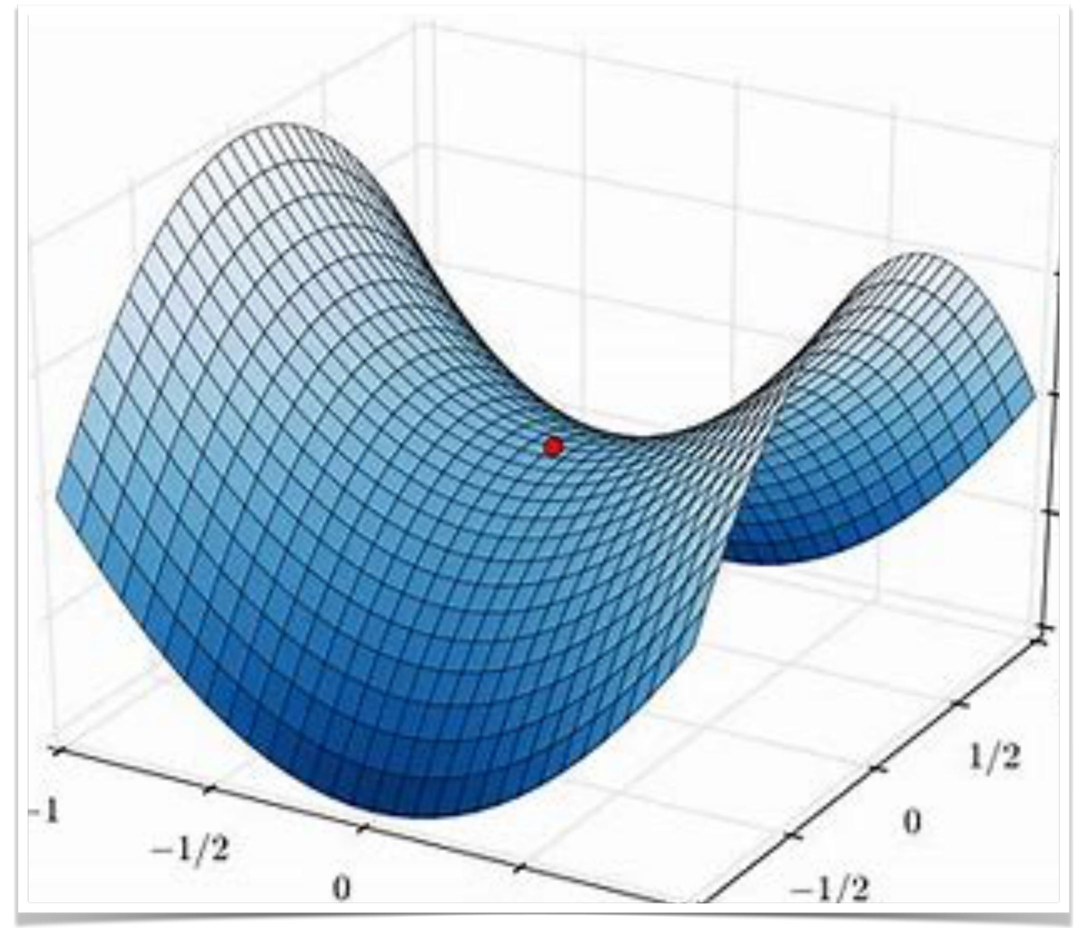
Background

At a stationary point $\nabla f(x) = 0$

Local minimum: $\nabla^2 f(x) \succ 0$

Local minimum: $\nabla^2 f(x) \prec 0$

Almost always stationary point can have an *indefinite* Hessian



Empirically saddle-points have been touted as “major concern”

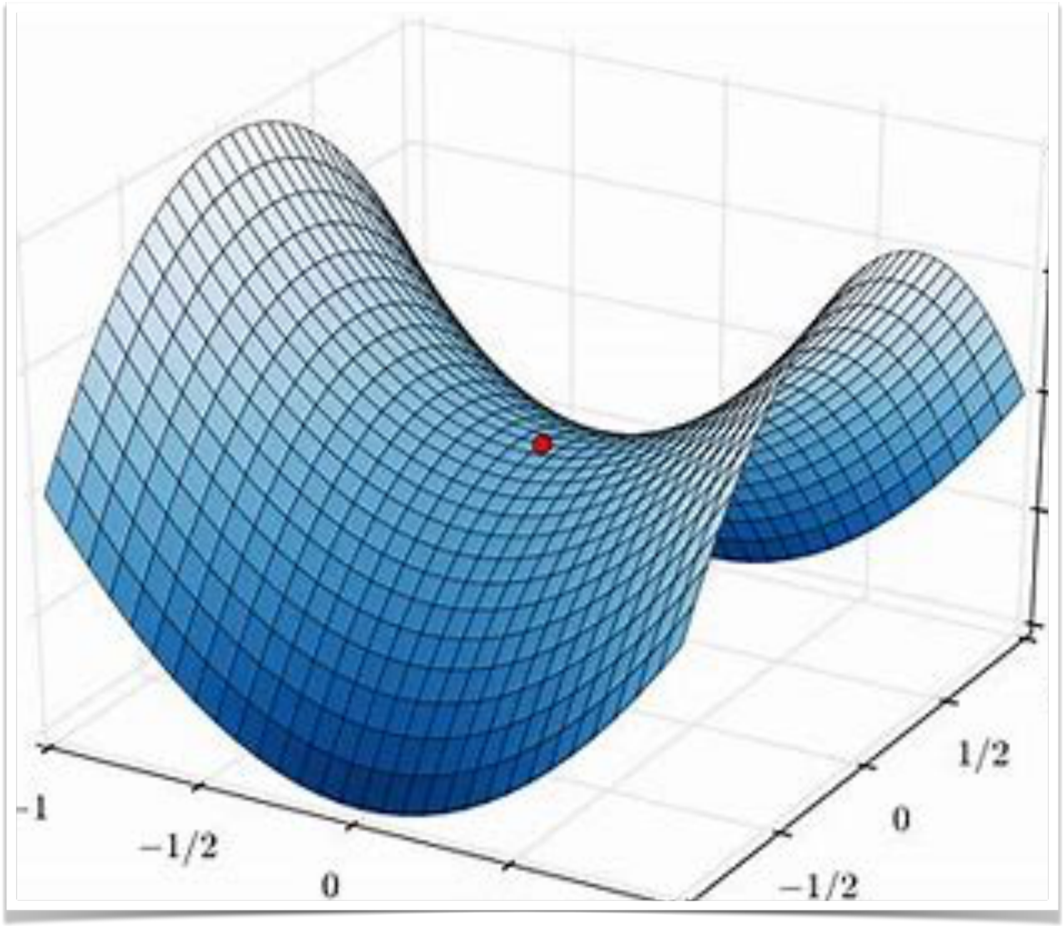
Background

At a stationary point $\nabla f(x) = 0$

Local minimum: $\nabla^2 f(x) \succ 0$

Local minimum: $\nabla^2 f(x) \prec 0$

Almost always stationary point can have an *indefinite* Hessian



Empirically saddle-points have been touted as "major concern"

Explore: Should we worry about saddle points in deep learning? Why?

Gradient descent escapes saddles

First-order methods almost always avoid strict saddle points

Jason D. Lee[1] · Ioannis Panageas[2] · Georgios Piliouras[3] · Max Simchowitz[4] · Michael I. Jordan[4] · Benjamin Recht[4]

Gradient descent escapes saddles

First-order methods almost always avoid strict saddle points

Jason D. Lee[1] · Ioannis Panageas[2] · Georgios Piliouras[3] · Max Simchowitz[4] · Michael I. Jordan[4] · Benjamin Recht[4]

Assumption 1: f is C^2 and L-smooth (i.e., C_L^1)

Assumption 2: only strict saddles, i.e., $\lambda_{\min}(\nabla^2 f(x_s)) < 0$ for stationary x_s

Gradient descent escapes saddles

First-order methods almost always avoid strict saddle points

Jason D. Lee[1] · Ioannis Panageas[2] · Georgios Piliouras[3] · Max Simchowitz[4] · Michael I. Jordan[4] · Benjamin Recht[4]

Assumption 1: f is C^2 and L-smooth (i.e., C_L^1)

Assumption 2: only strict saddles, i.e., $\lambda_{\min}(\nabla^2 f(x_s)) < 0$ for stationary x_s

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

Gradient descent escapes saddles

First-order methods almost always avoid strict saddle points

Jason D. Lee[1] · Ioannis Panageas[2] · Georgios Piliouras[3] · Max Simchowitz[4] · Michael I. Jordan[4] · Benjamin Recht[4]

Assumption 1: f is C^2 and L-smooth (i.e., C^1_L)
Assumption 2: only strict saddles, i.e., $\lambda_{\min}(\nabla^2 f(x_s)) < 0$ for stationary x_s

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

Theorem (informal). Let θ_0 be initialized randomly. Then with $\eta < 1/L$, under Assumptions 1, 2, GD avoids converging to saddle points.

Gradient descent escapes saddles

First-order methods almost always avoid strict saddle points

Jason D. Lee[1] · Ioannis Panageas[2] · Georgios Piliouras[3] · Max Simchowitz[4] · Michael I. Jordan[4] · Benjamin Recht[4]

Assumption 1: f is C^2 and L-smooth (i.e., C^1_L)
Assumption 2: only strict saddles, i.e., $\lambda_{\min}(\nabla^2 f(x_s)) < 0$ for stationary x_s

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

Theorem (informal). Let θ_0 be initialized randomly. Then with $\eta < 1/L$, under Assumptions 1, 2, GD avoids converging to saddle points.

Key idea: show that GD *eventually* escapes any saddle point, by showing that the set of "stable strict saddles" is of measure 0. "Stable" in the sense of: an attracting equilibrium point for a dynamical system.

Escaping saddle points

What about a non-asymptotic result?

Escaping saddle points

What about a non-asymptotic result?

Does randomly initialized GD escape saddle points in polynomial time?

Escaping saddle points

What about a non-asymptotic result?

Does randomly initialized GD escape saddle points in polynomial time?

Bad News!

Escaping saddle points

What about a non-asymptotic result?

Does randomly initialized GD escape saddle points in polynomial time?

Bad News!

Gradient Descent Can Take Exponential Time to Escape Saddle Points

Simon S. Du
Carnegie Mellon University
ssdu@cs.cmu.edu

Chi Jin
University of California, Berkeley
chijin@berkeley.edu

Jason D. Lee
University of Southern California
jasonlee@marshall.usc.edu

Michael I. Jordan
University of California, Berkeley
jordan@cs.berkeley.edu

Barnabás Póczos
Carnegie Mellon University
bapoczos@cs.cmu.edu

Aarti Singh
Carnegie Mellon University
aartisingh@cmu.edu

How to avoid exponential time?

How to avoid exponential time?

Key idea: Run perturbed gradient descent

How to avoid exponential time?

Key idea: Run perturbed gradient descent

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}(r))$$

How to avoid exponential time?

Key idea: Run perturbed gradient descent

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}(r))$$

Theorem 4.1 (Uniform initialization over a unit cube)**.** *Suppose the initialization point is uniformly sampled from* $[-1,1]^d$. *There exists a function* f *defined on* $\mathbb{R}^d$ *that is* B*-bounded,* ℓ*-gradient Lipschitz and* ρ*-Hessian Lipschitz with parameters* B, ℓ, ρ *at most* $\mathrm{poly}(d)$ *such that:*

1. *with probability one, gradient descent with step size* $\eta \le 1/\ell$ *will be* $\Omega(1)$ *distance away from any local minima for any* $T \le e^{\Omega(d)}$.
2. *for any* $\epsilon > 0$*, with probability* $1 - e^{-d}$*, perturbed gradient descent (Algorithm 1) will find a point* x *such that* $\|x - x^*\|_2 \le \epsilon$ *for some local minimum* x^* *in* $\mathrm{poly}(d, \frac{1}{\epsilon})$ *iterations.*

How to avoid exponential time?

Key idea: Run perturbed gradient descent

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}(r))$$

Theorem 4.1 (Uniform initialization over a unit cube). *Suppose the initialization point is uniformly sampled from* $[-1, 1]^d$. *There exists a function* f *defined on* $\mathbb{R}^d$ *that is* B*-bounded,* ℓ*-gradient Lipschitz and* ρ*-Hessian Lipschitz with parameters* B, ℓ, ρ *at most* $\mathrm{poly}(d)$ *such that:*

1. *with probability one, gradient descent with step size* $\eta \leq 1/\ell$ *will be* $\Omega(1)$ *distance away from any local minima for any* $T \leq e^{\Omega(d)}$.
2. *for any* $\epsilon > 0$*, with probability* $1 - e^{-d}$*, perturbed gradient descent (Algorithm 1) will find a point* x *such that* $\|x - x^*\|_2 \leq \epsilon$ *for some local minimum* x^* *in* $\mathrm{poly}(d, \frac{1}{\epsilon})$ *iterations.*

How to avoid exponential time?

Key idea: Run perturbed gradient descent

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}(r))$$

Theorem 4.1 (Uniform initialization over a unit cube). *Suppose the initialization point is uniformly sampled from $[-1,1]^d$. There exists a function f defined on $\mathbb{R}^d$ that is B-bounded, ℓ-gradient Lipschitz and ρ-Hessian Lipschitz with parameters B, ℓ, ρ at most $\mathrm{poly}(d)$ such that:*

1. *with probability one, gradient descent with step size $\eta \leq 1/\ell$ will be $\Omega(1)$ distance away from any local minima for any $T \leq e^{\Omega(d)}$.*
2. *for any $\epsilon > 0$, with probability $1 - e^{-d}$, perturbed gradient descent (Algorithm 1) will find a point x such that $\|x - x^*\|_2 \leq \epsilon$ for some local minimum x^* in $\mathrm{poly}(d, \frac{1}{\epsilon})$ iterations.*

How to avoid exponential time?

Key idea: Run perturbed gradient descent

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}(r))$$

Theorem 4.1 (Uniform initialization over a unit cube). *Suppose the initialization point is uniformly sampled from $[-1,1]^d$. There exists a function f defined on $\mathbb{R}^d$ that is B-bounded, ℓ-gradient Lipschitz and ρ-Hessian Lipschitz with parameters B, ℓ, ρ at most $\mathrm{poly}(d)$ such that:*

1. *with probability one, gradient descent with step size $\eta \leq 1/\ell$ will be $\Omega(1)$ distance away from any local minima for any $T \leq e^{\Omega(d)}$.*
2. *for any $\epsilon > 0$, with probability $1 - e^{-d}$, perturbed gradient descent (Algorithm 1) will find a point x such that $\|x - x^*\|_2 \leq \epsilon$ for some local minimum x^* in $\boxed{\mathrm{poly}(d,} \frac{1}{\epsilon})$ iterations.*

Question: Can this be improved?

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \rho \|x - y\|$

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \rho \|x - y\|$

Defn: ϵ-second order stationarity: $\|\nabla f(x)\| \leq \epsilon, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \rho \|x - y\|$

Defn: ϵ-second order stationarity: $\|\nabla f(x)\| \leq \epsilon, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$

Perturbed GD

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \quad \xi_t \sim \mathcal{N}(0, \tfrac{r^2}{d}\boldsymbol{I})$$

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le \rho \|x - y\|$

Defn: ϵ-second order stationarity: $\|\nabla f(x)\| \le \epsilon, \quad \lambda_{\min}(\nabla^2 f(x)) \ge -\sqrt{\rho\epsilon}$

Perturbed GD

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \quad \xi_t \sim \mathcal{N}(0, \tfrac{r^2}{d}\boldsymbol{I})$$

Theorem 13. *Let the function $f(\cdot)$ satisfy Assumption A. Then, for any $\epsilon, \delta > 0$, the PGD algorithm (Algorithm 1), with parameters $\eta = \tilde{\Theta}(1/\ell)$ and $r = \tilde{\Theta}(\epsilon)$, will visit an $\epsilon-$second-order stationary point at least once in the following number of iterations, with probability at least $1 - \delta$:*

$$\tilde{\mathcal{O}}\left(\frac{\ell(f(\mathbf{x}_0) - f^\star)}{\epsilon^2}\right),$$

$\boxed{\ell = L \quad (\text{i.e., } f \in C^1_\ell)}$

where $\tilde{\mathcal{O}}$ and $\tilde{\Theta}$ hide polylogarithmic factors in $d, \ell, \rho, 1/\epsilon, 1/\delta$ and $\Delta_f := f(\mathbf{x}_0) - f^\star$.

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \rho \|x - y\|$

Defn: ϵ-second order stationarity: $\|\nabla f(x)\| \leq \epsilon, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$

Perturbed GD

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \quad \xi_t \sim \mathcal{N}(0, \tfrac{r^2}{d}\boldsymbol{I})$$

Theorem 13. *Let the function $f(\cdot)$ satisfy Assumption A. Then, for any $\epsilon, \delta > 0$, the PGD algorithm (Algorithm 1), with parameters $\eta = \tilde{\Theta}(1/\ell)$ and $r = \tilde{\Theta}(\epsilon)$, will visit an $\epsilon-$second-order stationary point at least once in the following number of iterations, with probability at least $1 - \delta$:*

$$\tilde{\mathcal{O}}\left(\frac{\ell(f(\mathbf{x}_0) - f^\star)}{\epsilon^2}\right),$$

$\boxed{\ell = L \quad (\text{i.e., } f \in C^1_\ell)}$

where $\tilde{\mathcal{O}}$ and $\tilde{\Theta}$ hide polylogarithmic factors in $d, \ell, \rho, 1/\epsilon, 1/\delta$ and $\Delta_f := f(\mathbf{x}_0) - f^\star$.

Second-order stationary points

Assumption: Lipschitz Hessian $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \rho\|x - y\|$

Defn: ϵ-second order stationarity: $\|\nabla f(x)\| \leq \epsilon, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$

Perturbed GD

$$\theta_{t+1} = \theta_t - \eta[\nabla f(\theta_t) + \xi_t], \quad \xi_t \sim \mathcal{N}(0, \frac{r^2}{d}\boldsymbol{I})$$

Theorem 13. *Let the function $f(\cdot)$ satisfy Assumption A. Then, for any $\epsilon, \delta > 0$, the PGD algorithm (Algorithm 1), with parameters $\eta = \tilde{\Theta}(1/\ell)$ and $r = \tilde{\Theta}(\epsilon)$, will visit an $\epsilon-$second-order stationary point at least once in the following number of iterations, with probability at least $1 - \delta$:*

$$\tilde{O}\left(\frac{\ell(f(\mathbf{x}_0) - f^\star)}{\epsilon^2}\right),$$

$$\boxed{\ell = L \quad (\text{i.e., } f \in C^1_\ell)}$$

where $\tilde{O}$ and $\tilde{\Theta}$ hide polylogarithmic factors in $d, \ell, \rho, 1/\epsilon, 1/\delta$ and $\Delta_f := f(\mathbf{x}_0) - f^\star$.

Reference: Jin, Netrapalli, Ge, Kakade, Jordan. *"On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points"*. arXiv:1902.04811

Perturbed SGD

Perturbed SGD

Assuming stochastic gradients are also Lipschitz, and assuming sub-gaussian tails for the stochastic noise, a related result is also shown by the same authors to hold for Perturbed SGD (i.e., $O(\epsilon^{-4})$ iterations)

Perturbed SGD

Assuming stochastic gradients are also Lipschitz, and assuming sub-gaussian tails for the stochastic noise, a related result is also shown by the same authors to hold for Perturbed SGD (i.e., $O(\epsilon^{-4})$ iterations)

Without the Lipschitz assumption, the authors show that PSGD finds an ϵ-2nd order stationary point in $O(d\epsilon^{-4})$ iterations (extra 'd' factor)

Perturbed SGD

Assuming stochastic gradients are also Lipschitz, and assuming sub-gaussian tails for the stochastic noise, a related result is also shown by the same authors to hold for Perturbed SGD (i.e., $O(\epsilon^{-4})$ iterations)

Without the Lipschitz assumption, the authors show that PSGD finds an ϵ-2nd order stationary point in $O(d\epsilon^{-4})$ iterations (extra 'd' factor)

Recommended reading:

On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points

Chi Jin
University of California, Berkeley
chijin@cs.berkeley.edu

Praneeth Netrapalli
Microsoft Research, India
praneeth@microsoft.com

Rong Ge
Duke University
rongge@cs.duke.edu

Sham M. Kakade
University of Washington, Seattle
sham@cs.washington.edu

Michael I. Jordan

Other methods

Other methods

A variety of other related results exist for 2nd order stationary points:

Other methods

A variety of other related results exist for 2nd order stationary points:

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only
- Normalized GD escapes saddle points too: *requires perturbation*

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only
- Normalized GD escapes saddle points too: *requires perturbation*
- Momentum based methods with perturbation escape faster than PGD

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only
- Normalized GD escapes saddle points too: *requires perturbation*
- Momentum based methods with perturbation escape faster than PGD

- Stochastic methods using Hessian-vector oracle: $O(\epsilon^{-3.5})$

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only
- Normalized GD escapes saddle points too: *requires perturbation*
- Momentum based methods with perturbation escape faster than PGD

- Stochastic methods using Hessian-vector oracle: $O(\epsilon^{-3.5})$
- SPIDER (variance reduced SGD) yields $O(\epsilon^{-3})$

Other methods

A variety of other related results exist for 2nd order stationary points:

- Using 2nd order information to escape stationary points faster ($O(\epsilon^{-1.5})$) for non-stochastic settings (trust regions, cubic regularization), or $O(\epsilon^{-1.75})$ using Hessian-vector products only
- Normalized GD escapes saddle points too: *requires perturbation*
- Momentum based methods with perturbation escape faster than PGD

- Stochastic methods using Hessian-vector oracle: $O(\epsilon^{-3.5})$
- SPIDER (variance reduced SGD) yields $O(\epsilon^{-3})$
- SGD with averaging and some tricks $O(\epsilon^{-3.5})$

Other methods

Escaping Saddle Points with Adaptive Gradient Methods

Matthew Staib*
MIT EECS
mstaib@mit.edu

Sashank Reddi
Google Research, New York
sashank@google.com

Satyen Kale
Google Research, New York
satyenkale@google.com

Sanjiv Kumar
Google Research, New York
sanjivk@google.com

Suvrit Sra
MIT EECS
suvrit@mit.edu

ADAM-like methods can escape saddle points faster than SGD

Other methods

Escaping Saddle Points with Adaptive Gradient Methods

Matthew Staib* MIT EECS mstaib@mit.edu

Sashank Reddi Google Research, New York sashank@google.com

Satyen Kale Google Research, New York satyenkale@google.com

Sanjiv Kumar Google Research, New York sanjivk@google.com

Suvrit Sra MIT EECS suvrit@mit.edu

ADAM-like methods can escape saddle points faster than SGD

However, iteration bound of the type $O(d^4\epsilon^{-5})$ for escaping saddles. SGD has a similar rate, but the "constants" for ADAM-like method can be much better.

Other methods

Escaping Saddle Points with Adaptive Gradient Methods

Matthew Staib*
MIT EECS
mstaib@mit.edu

Sashank Reddi
Google Research, New York
sashank@google.com

Satyen Kale
Google Research, New York
satyenkale@google.com

Sanjiv Kumar
Google Research, New York
sanjivk@google.com

Suvrit Sra
MIT EECS
suvrit@mit.edu

ADAM-like methods can escape saddle points faster than SGD

However, iteration bound of the type $O(d^4\epsilon^{-5})$ for escaping saddles. SGD has a similar rate, but the "constants" for ADAM-like method can be much better.

Key question: Can SGD/ADAM escape fast without perturbation?

Other methods

Escaping Saddle Points with Adaptive Gradient Methods

Matthew Staib*
MIT EECS
mstaib@mit.edu

Sashank Reddi
Google Research, New York
sashank@google.com

Satyen Kale
Google Research, New York
satyenkale@google.com

Sanjiv Kumar
Google Research, New York
sanjivk@google.com

Suvrit Sra
MIT EECS
suvrit@mit.edu

ADAM-like methods can escape saddle points faster than SGD

However, iteration bound of the type $O(d^4\epsilon^{-5})$ for escaping saddles. SGD has a similar rate, but the "constants" for ADAM-like method can be much better.

Key question: Can SGD/ADAM escape fast without perturbation?

Sharp Analysis for Nonconvex SGD Escaping from Saddle Points

Cong Fang * Zhouchen Lin † Tong Zhang ‡

Other methods

Escaping Saddle Points with Adaptive Gradient Methods

Matthew Staib*
MIT EECS
mstaib@mit.edu

Sashank Reddi
Google Research, New York
sashank@google.com

Satyen Kale
Google Research, New York
satyenkale@google.com

Sanjiv Kumar
Google Research, New York
sanjivk@google.com

Suvrit Sra
MIT EECS
suvrit@mit.edu

ADAM-like methods can escape saddle points faster than SGD

However, iteration bound of the type $O(d^4\epsilon^{-5})$ for escaping saddles. SGD has a similar rate, but the "constants" for ADAM-like method can be much better.

Key question: Can SGD/ADAM escape fast without perturbation?

Sharp Analysis for Nonconvex SGD Escaping from Saddle Points

Cong Fang * Zhouchen Lin † Tong Zhang ‡

This paper claims so, but by assuming "dispersive noise" on SGD. Clean results missing!