

Optimization In AI

Class 01: Overview

Instructor: Jianhao Ma

Department of Industrial Engineering, Tsinghua University

Instructor

Jianhao Ma

Assistant professor, Department of Industrial Engineering, Tsinghua University

Research: optimization $\times$ AI

Background

- University of Pennsylvania, stats, postdoc (2025–2026)
- University of Michigan, IOE, Ph.D. (2021–2025)
- Tsinghua University, IE and math, undergraduate (2016–2020)
- Industry: ByteDance AI Lab (2020) and Meta FAIR (2024)

Teaching Assistant

Yuzhen Chen

Third-year Ph.D. student
Department of Industrial Engineering

Research: optimization and decision-making in AI-driven systems (OR/OM in AI).



Office Hours

Instructor: Jianhao Ma

Wednesday, 1–2 pm

TA: Yuzhen Chen

Thursday, 3–4 pm

Survey

- **Do you know the following concepts?**

Gradient, gradient descent, SGD, Adam, Muon, ...

- **Do you know linear algebra and calculus?**

Eigenvalues, SVD, matrix differentiation, Taylor expansion, ...

- **Have you trained models before?**

Linear regression, CNNs, transformers, Llama, ...

Goal

The course is not about

Traditional convex optimization theory:

- Convex analysis, linear programming, gradient descent, duality theory, ...

Practical LLM training:

- Model architecture, data recipe, optimizer, alignment, agents, ...

The course is about

Principles of practical optimization for large-scale AI.

Gradient Descent

Goal: $\min_{\theta \in \mathbb{R}^d} f(\theta)$, with f differentiable.

Initialize θ_0 ; choose learning rates $\eta_t > 0$.

For $t = 0, 1, 2, \dots$:

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

AdamW In PyTorch

```
input :  $\gamma(\text{lr})$ ,  $\beta_1, \beta_2(\text{betas})$ ,  $\theta_0(\text{params})$ ,  $f(\theta)(\text{objective})$ ,  $\epsilon$  (epsilon)
         $\lambda(\text{weight decay})$ , amsgrad, maximize
initialize :  $m_0 \leftarrow 0$  (first moment),  $v_0 \leftarrow 0$  (second moment),  $v_0^{max} \leftarrow 0$ 
```

```
for  $t = 1$  to ... do
    if maximize :
         $g_t \leftarrow -\nabla_{\theta} f_t(\theta_{t-1})$ 
    else
         $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ 
     $\theta_t \leftarrow \theta_{t-1} - \gamma \lambda \theta_{t-1}$ 
     $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
     $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
     $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
    if amsgrad
         $v_t^{max} \leftarrow \max(v_{t-1}^{max}, v_t)$ 
         $\widehat{v}_t \leftarrow v_t^{max} / (1 - \beta_2^t)$ 
    else
         $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
     $\theta_t \leftarrow \theta_t - \gamma \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ 
```

```
return  $\theta_t$ 
```

Three Mechanisms

Two running statistics.

Coordinate-dependent scaling.

Weight decay and corrections.

[PyTorch official documentation](#)

What Does AdamW Add?

$$\theta_{t+1} = (1 - \eta_t \lambda) \theta_t - \eta_t \frac{1}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Weight decay
Shrink parameters

Preconditioning
Rescale coordinates

Momentum
Use gradient history

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

$$\underbrace{\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}}_{\text{Bias correction}}$$

$\epsilon > 0$
Numerical stability

$$t \geq 1, \quad m_0 = v_0 = 0, \quad g_t: \text{sampled gradient.}$$

Syllabus

- 01 Introduction
- 02 Mathematical Foundations
- 03 Gradient Descent
- 04 Stochastic Gradient Descent

Syllabus

- 05** Momentum
- 06** Adaptive Optimizers
- 07** Weight Decay
- 08** Selected Advanced Topics

Guest Speakers

Yushun Zhang (DeepSeek): confirmed.
Additional speaker: to be confirmed.

References

Books and lecture notes

Convex Optimization

Boyd and Vandenberghe

Convex Optimization:

Algorithms and Complexity

Bubeck

Optimization for Machine Learning

Hazan

Numerical Optimization

Nocedal and Wright

Related courses

ELE 522: Large-Scale Optimization
for Data Science

Princeton University, Yuxin Chen

STAT 4830: Numerical Optimization
for Data Science and Machine Learning
University of Pennsylvania, Damek Davis

DSC 243: Advanced Optimization
UC San Diego, Dmitriy Drusvyatskiy

Assignments

Homework (10%)

Two assignments.

Two weeks to complete each.

Exams (60%)

Midterm exam: 30%.

Final exam: 30%.

Course project (30%)

Train small language models using optimization algorithms you design.

AI Policy

Homework and course project

You may use AI tools.

Disclose whether and how you used AI.

Disclosed AI use will not reduce your score.

Exams

AI tools are not allowed in exams.

Optimization Problem

$$\min_{x \in \mathbb{R}^d} f(x)$$

x : decision variable.

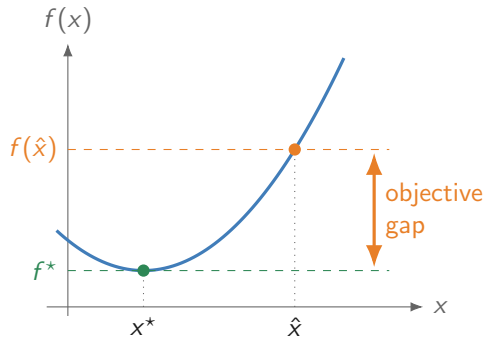
$f(x)$: objective value.

Global Minimizer (assumed to exist)

$$f(x^*) \leq f(x) \quad \forall x \in \mathbb{R}^d.$$

Minimum Value $f^* = f(x^*)$.

Algorithm Output $\hat{x}$.



Target: Objective Gap $0 \leq f(\hat{x}) - f^* \leq \varepsilon$.

Principle 1: Structure Sets The Speed Limit

More Structure $\longrightarrow$ Faster Guarantees

Objective gap $f(\theta) - f^*$ after T exact first-order queries.

Nonsmooth convex

Subgradient methods:

$$O(T^{-1/2})$$

Bounded subgradients and
feasible set.

Principle 1: Structure Sets The Speed Limit

More Structure $\longrightarrow$ Faster Guarantees

Objective gap $f(\theta) - f^*$ after T exact first-order queries.

Nonsmooth convex

Subgradient methods:
 $O(T^{-1/2})$

Bounded subgradients and
feasible set.

Smooth convex

GD: $O(T^{-1})$
Acceleration: $O(T^{-2})$

Lipschitz gradients and a
minimizer.

Principle 1: Structure Sets The Speed Limit

More Structure $\longrightarrow$ Faster Guarantees

Objective gap $f(\theta) - f^*$ after T exact first-order queries.

Nonsmooth convex

Subgradient methods:
 $O(T^{-1/2})$

Bounded subgradients and
feasible set.

Smooth convex

GD: $O(T^{-1})$
Acceleration: $O(T^{-2})$

Lipschitz gradients and a
minimizer.

Smooth + strongly convex

GD: $O(q^T)$, $0 < q < 1$
Linear convergence.
 q depends on conditioning.

Principle 2: No Universal Optimizer

Illustrative Comparison

	Problem 1	Problem 2
Algorithm A	Works well	Fails
Algorithm B	Fails	Works well

Principle 2: No Universal Optimizer

Illustrative Comparison

	Problem 1	Problem 2
Algorithm A	Works well	Fails
Algorithm B	Fails	Works well

What should we inspect?

- Smoothness and curvature.
- Gradient noise.
- Scale and conditioning.
- Compute and memory budget.

Principle 3: Information Has A Cost

For a d -dimensional problem, $\min_{x \in \mathbb{R}^d} f(x)$:

Oracle order	Information	What it tells us	Stored entries
Zeroth order	$f(x) \in \mathbb{R}$	Function value	1

Principle 3: Information Has A Cost

For a d -dimensional problem, $\min_{x \in \mathbb{R}^d} f(x)$:

Oracle order	Information	What it tells us	Stored entries
Zeroth order	$f(x) \in \mathbb{R}$	Function value	1
First order	$\nabla f(x) \in \mathbb{R}^d$	Local slopes	d

Principle 3: Information Has A Cost

For a d -dimensional problem, $\min_{x \in \mathbb{R}^d} f(x)$:

Oracle order	Information	What it tells us	Stored entries
Zeroth order	$f(x) \in \mathbb{R}$	Function value	1
First order	$\nabla f(x) \in \mathbb{R}^d$	Local slopes	d
Second order	$\nabla^2 f(x) \in \mathbb{R}^{d \times d}$	Local curvature	d^2 (dense)

Newton Step: $\nabla^2 f(x) \delta = -\nabla f(x)$.

Principle 3: Information Has A Cost

For a d -dimensional problem, $\min_{x \in \mathbb{R}^d} f(x)$:

Oracle order	Information	What it tells us	Stored entries
Zeroth order	$f(x) \in \mathbb{R}$	Function value	1
First order	$\nabla f(x) \in \mathbb{R}^d$	Local slopes	d
Second order	$\nabla^2 f(x) \in \mathbb{R}^{d \times d}$	Local curvature	d^2 (dense)

Newton Step: $\nabla^2 f(x) \delta = -\nabla f(x)$.

Total Work = Iterations $\times$ Work Per Iteration

A Real Optimization Problem: LLM Pre-Training

$$f(\theta) = \mathbb{E}_{z \sim \mathcal{D}_{\text{train}}} [\ell(\theta; z)] \quad z: \text{a training example}$$

Landscape

Enormous dimension.
Highly nonconvex.

Information

Minibatch gradients only.
Noisy estimates.

Budget

GPU compute.
GPU memory.
Distributed communication.

DeepSeek-V4.1-Flash: report excerpts (p. 22)

configuration, DeepSeek-V4.1-Flash comprises 552B backbone parameters, with 8B activated per token during prefill and 16B during decode.

45T tokens of multimodal data with no instability. We keep the batch size fixed at 100.6 million tokens throughout training. The learning rate is linearly warmed up over the first 2000 steps

DeepSeek-AI, "DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression," 2026.

A Real Optimization Problem: LLM Pre-Training

$$f(\theta) = \mathbb{E}_{z \sim \mathcal{D}_{\text{train}}} [\ell(\theta; z)] \quad z: \text{a training example}$$

Landscape

Enormous dimension.
Highly nonconvex.

Information

Minibatch gradients only.
Noisy estimates.

Budget

GPU compute.
GPU memory.
Distributed communication.

DeepSeek-V4.1-Flash: report excerpts (p. 22)

configuration, DeepSeek-V4.1-Flash comprises 552B backbone parameters, with 8B activated per token during prefill and 16B during decode.

45T tokens of multimodal data with no instability. We keep the batch size fixed at 100.6 million tokens throughout training. The learning rate is linearly warmed up over the first 2000 steps

Geometry, gradient noise, and systems cost all matter.

DeepSeek-AI, "DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression," 2026.

An LLM Optimizer In Practice

DeepSeek-V4.1-Flash Technical Report, Section 2.5

Basic Configurations. We retain AdamW (Loshchilov and Hutter, 2019) for normalization-layer weights and other non-matrix parameters, including biases and scaling factors. We use Muon (Jordan et al., 2024) for the weight matrices of linear transformations in the language-model backbone, the Engram projection layers, and the vision-language projector. We use head-wise Muon for Query and Key weights. We apply decoupled weight decay and Nesterov momentum to Muon (Nesterov, 1983; Liu et al., 2025); normalization-layer weights are also subject to weight decay, whereas biases and scaling factors are not. The Sinkhorn-balanced update also uses Nesterov momentum but does not apply weight decay. During pre-training, we keep the vision encoder frozen until the learning-rate decay stage, while its final normalization layer and the vision-language projector remain trainable. At the onset of learning-rate decay, we unfreeze the vision encoder and optimize it jointly with the LLM with a smaller learning rate.

DeepSeek-AI, "DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression," 2026, p. 15.

An LLM Optimizer In Practice

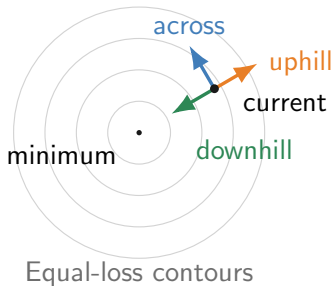
DeepSeek-V4.1-Flash Technical Report, Section 2.5

Basic Configurations. We retain AdamW (Loshchilov and Hutter, 2019) for normalization-layer weights and other non-matrix parameters, including biases and scaling factors. We use Muon (Jordan et al., 2024) for the weight matrices of linear transformations in the language-model backbone, the Engram projection layers, and the vision-language projector. We use head-wise Muon for Query and Key weights. We apply decoupled weight decay and Nesterov momentum to Muon (Nesterov, 1983; Liu et al., 2025); normalization-layer weights are also subject to weight decay, whereas biases and scaling factors are not. The Sinkhorn-balanced update also uses Nesterov momentum but does not apply weight decay. During pre-training, we keep the vision encoder frozen until the learning-rate decay stage, while its final normalization layer and the vision-language projector remain trainable. At the onset of learning-rate decay, we unfreeze the vision encoder and optimize it jointly with the LLM with a smaller learning rate.

One Model, Multiple Update Rules

DeepSeek-AI, "DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression," 2026, p. 15.

Which Way Is Downhill?

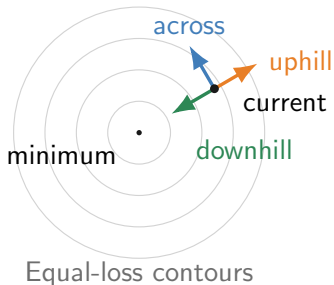


Negative Gradient

$$\delta = -\eta \nabla f(\theta).$$

Steepest local decrease at fixed Euclidean length.

Which Way Is Downhill?



Negative Gradient

$$\delta = -\eta \nabla f(\theta).$$

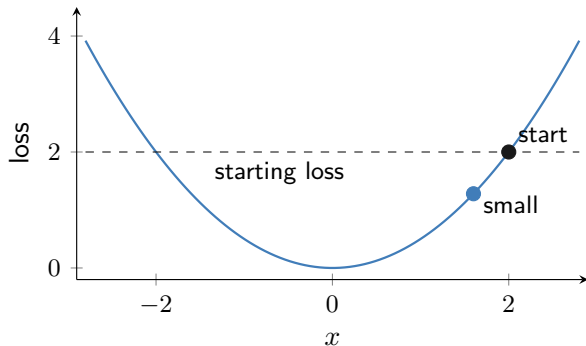
Steepest local decrease at fixed Euclidean length.

First-Order Model (f differentiable)

$$f(\theta + \delta) - f(\theta) \approx \nabla f(\theta)^\top \delta = -\eta \|\nabla f(\theta)\|^2.$$

Descent: $\nabla f(\theta) \neq 0$, sufficiently small $\eta > 0$.

Choosing The Step Size



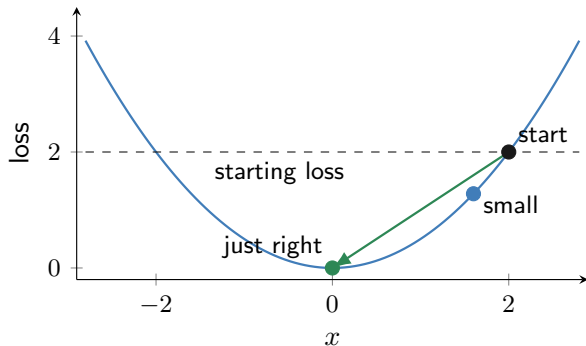
Predict: which step reduces the loss?

$$f(x) = \frac{1}{2}x^2, \quad x = 2.$$

$$x_+ = 2 - 2\eta.$$

Step	x_+	Loss
Too small: $\eta = 0.2$	1.6	1.28

Choosing The Step Size



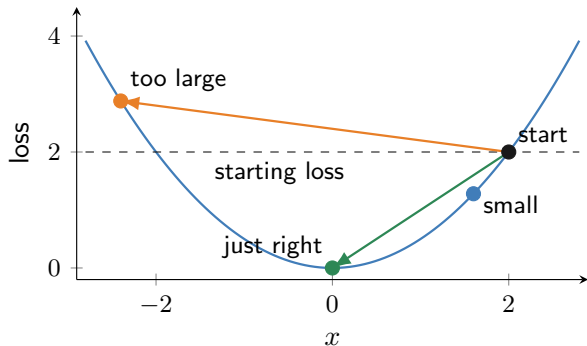
Predict: which step reduces the loss?

$$f(x) = \frac{1}{2}x^2, \quad x = 2.$$

$$x_+ = 2 - 2\eta.$$

Step	x_+	Loss
Too small: $\eta = 0.2$	1.6	1.28
Just right: $\eta = 1$	0	0

Choosing The Step Size



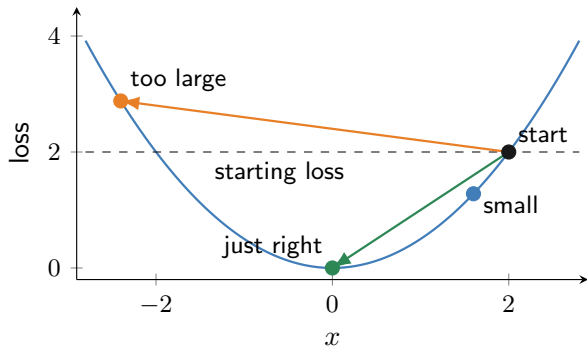
Predict: which step reduces the loss?

$$f(x) = \frac{1}{2}x^2, \quad x = 2.$$

$$x_+ = 2 - 2\eta.$$

Step	x_+	Loss
Too small: $\eta = 0.2$	1.6	1.28
Just right: $\eta = 1$	0	0
Too large: $\eta = 2.2$	-2.4	2.88

Choosing The Step Size



Predict: which step reduces the loss?

$$f(x) = \frac{1}{2}x^2, \quad x = 2.$$

$$x_+ = 2 - 2\eta.$$

Step	x_+	Loss
Too small: $\eta = 0.2$	1.6	1.28
Just right: $\eta = 1$	0	0
Too large: $\eta = 2.2$	-2.4	2.88

A descent direction is only locally reliable.

Why Wait For The Full Gradient?

Finite-Sum Loss $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta).$

Full gradient

n examples per update.

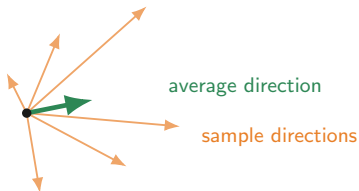


Toy negative gradients at fixed weights.

Why Wait For The Full Gradient?

Finite-Sum Loss $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta).$

One example, one opinion



Toy negative gradients at fixed weights.

Full gradient

n examples per update.

Stochastic gradient

$i_t \mid \theta_t \sim \text{Unif}\{1, \dots, n\}.$

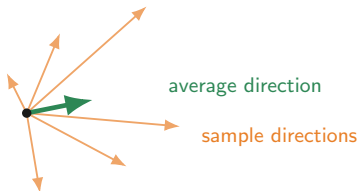
$$g_t = \nabla f_{i_t}(\theta_t).$$

One example per update.

Why Wait For The Full Gradient?

Finite-Sum Loss $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta).$

One example, one opinion



Toy negative gradients at fixed weights.

Full gradient

n examples per update.

Stochastic gradient

$$i_t \mid \theta_t \sim \text{Unif}\{1, \dots, n\}.$$

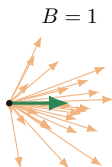
$$g_t = \nabla f_{i_t}(\theta_t).$$

One example per update.

Right on average: $\mathbb{E}[g_t \mid \theta_t] = \nabla f(\theta_t).$ Individual steps can go uphill.

Minibatches: Noise And Cost

$$g_B = \frac{1}{B} \sum_{i \in \text{batch}} \nabla f_i(\theta).$$



Green: full direction Orange: batch estimates

Statistical effect

Lower gradient variance.

Fixed weights, independent identically distributed samples: variance $\propto 1/B$.

Hardware effect

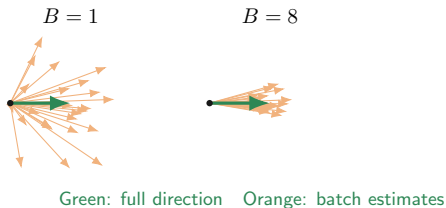
Better GPU utilization
(possible).

Cost

More work and memory per
update.

Minibatches: Noise And Cost

$$g_B = \frac{1}{B} \sum_{i \in \text{batch}} \nabla f_i(\theta).$$



Statistical effect

Lower gradient variance.

Fixed weights, independent identically distributed samples: variance $\propto 1/B$.

Hardware effect

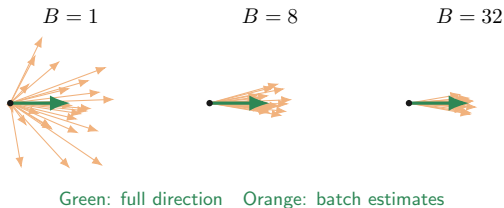
Better GPU utilization
(possible).

Cost

More work and memory per
update.

Minibatches: Noise And Cost

$$g_B = \frac{1}{B} \sum_{i \in \text{batch}} \nabla f_i(\theta).$$



Statistical effect

Lower gradient variance.

Fixed weights, independent identically distributed samples: variance $\propto 1/B$.

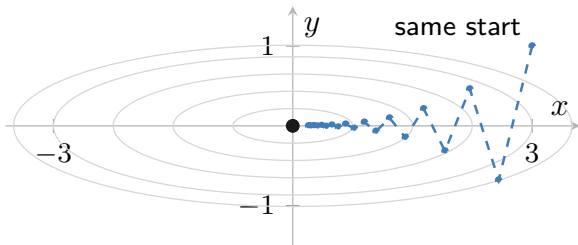
Hardware effect

Better GPU utilization
(possible).

Cost

More work and memory per
update.

Momentum Remembers A Direction



Dashed: GD

Same start; 18 updates on the same loss.

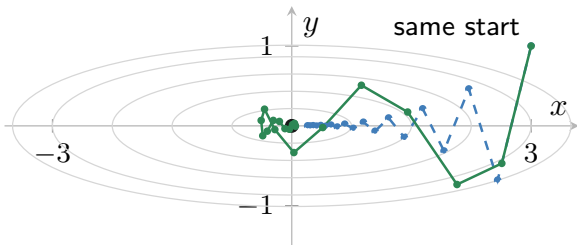
Persistent components accumulate.
Alternating components average down.

$(1, 2), (1, -2), (1, 2), \dots$

Tradeoff: slower turns.

Toy loss: $f(x, y) = (x^2 + 12y^2)/2$. GD: $\eta = 0.14$; momentum: $\eta = 0.35, \beta = 0.65$.

Momentum Remembers A Direction



Dashed: GD Solid: momentum

Same start; 18 updates on the same loss.

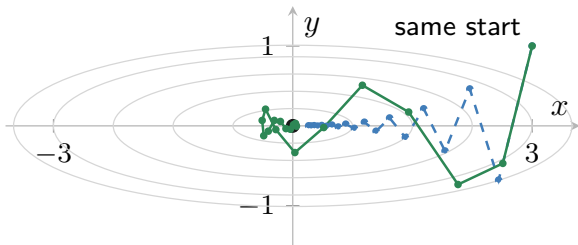
Persistent components accumulate.
Alternating components average down.

$(1, 2), (1, -2), (1, 2), \dots$

Tradeoff: slower turns.

Toy loss: $f(x, y) = (x^2 + 12y^2)/2$. GD: $\eta = 0.14$; momentum: $\eta = 0.35, \beta = 0.65$.

Momentum Remembers A Direction



Dashed: GD Solid: momentum

Same start; 18 updates on the same loss.

Persistent components accumulate.
Alternating components average down.

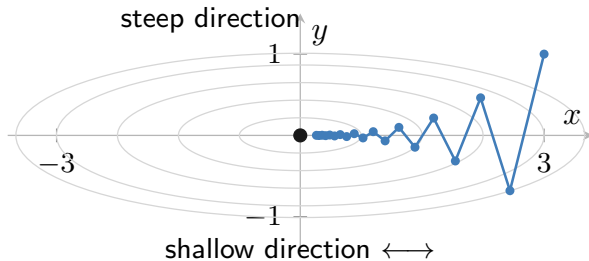
$(1, 2), (1, -2), (1, 2), \dots$

Tradeoff: slower turns.

$$\underbrace{m_t = \beta m_{t-1} + (1 - \beta) \nabla f(\theta_t)}_{\text{remember recent gradients}}, \quad \theta_{t+1} = \theta_t - \eta m_t.$$

Toy loss: $f(x, y) = (x^2 + 12y^2)/2$. GD: $\eta = 0.14$; momentum: $\eta = 0.35, \beta = 0.65$.

Conditioning And Step Size



Ill-conditioned geometry: a narrow valley.

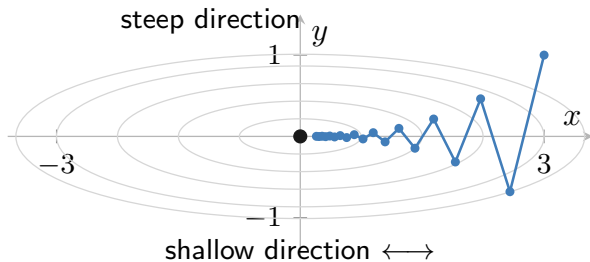
$$f(x, y) = \frac{1}{2}(x^2 + 12y^2).$$

Curvature

Horizontal: 1 Vertical: 12.

Condition number: 12.

Conditioning And Step Size



Ill-conditioned geometry: a narrow valley.

$$f(x, y) = \frac{1}{2}(x^2 + 12y^2).$$

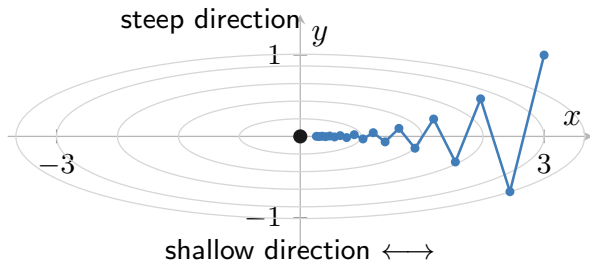
$$\nabla f(x, y) = (x, 12y).$$

Curvature

Horizontal: 1 Vertical: 12.

Condition number: 12.

Conditioning And Step Size



$$f(x, y) = \frac{1}{2}(x^2 + 12y^2).$$

$$\nabla f(x, y) = (x, 12y).$$

Curvature

Horizontal: 1 Vertical: 12.

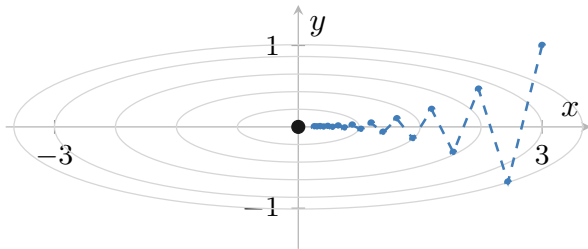
Condition number: 12.

Ill-conditioned geometry: a narrow valley.

Steep Curvature Limits The Step Size.

Here GD uses $\eta = 0.14$: $x_+ = 0.86x$, while $y_+ = -0.68y$.

Preconditioning: Rescale The Gradient



Dashed: one stepsize

$P = \text{diag}(1, 1/12)$.

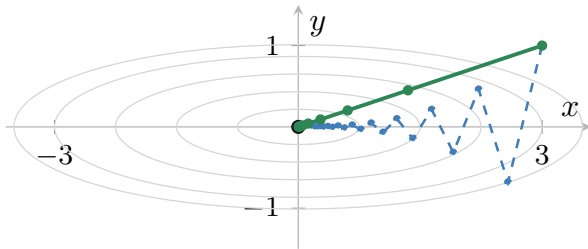
GD: $\eta = 0.14$, 18 steps; rescaled: $\eta = 0.55$, 8 steps.

Rescale before stepping.

$$\begin{pmatrix} x \\ 12y \end{pmatrix} \xrightarrow{P} \begin{pmatrix} x \\ y \end{pmatrix}.$$

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

Preconditioning: Rescale The Gradient



Rescale before stepping.

$$\begin{pmatrix} x \\ 12y \end{pmatrix} \xrightarrow{P} \begin{pmatrix} x \\ y \end{pmatrix}.$$

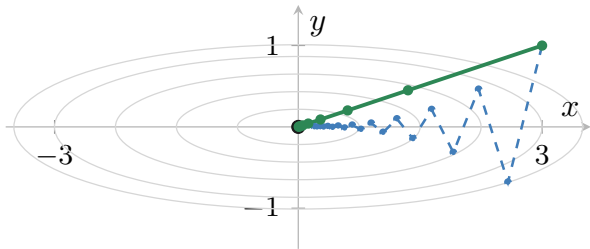
$$\theta_{t+1} = \theta_t - \eta P \nabla f(\theta_t).$$

Dashed: one stepsize Solid: rescaled

$P = \text{diag}(1, 1/12)$.

GD: $\eta = 0.14$, 18 steps; rescaled: $\eta = 0.55$, 8 steps.

Preconditioning: Rescale The Gradient



Rescale before stepping.

$$\begin{pmatrix} x \\ 12y \end{pmatrix} \xrightarrow{P} \begin{pmatrix} x \\ y \end{pmatrix}.$$

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

Dashed: one stepsize Solid: rescaled

$P = \text{diag}(1, 1/12)$.

GD: $\eta = 0.14$, 18 steps; rescaled: $\eta = 0.55$, 8 steps.

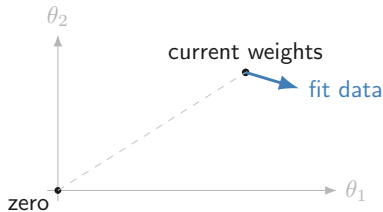
Adam: Squared-Gradient Statistics $P_{jj} \approx 1/(\sqrt{\hat{v}_{t,j}} + \epsilon)$.

Gradient statistics $\neq$ exact curvature.

Weight Decay Pulls Parameters Toward Zero

Parameter Shrinkage

Preference For Smaller Weights

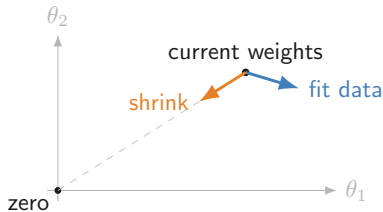


u_t : optimizer direction; $0 < \eta_t \lambda < 1$ gives shrinkage.

Weight Decay Pulls Parameters Toward Zero

Parameter Shrinkage

Preference For Smaller Weights

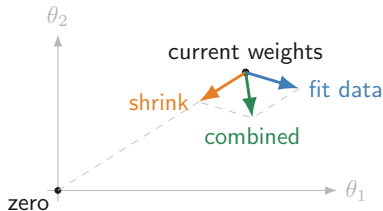


$$\theta_{t+1} = \underbrace{(1 - \eta_t \lambda) \theta_t}_{\text{parameter shrinkage}} - \eta_t u_t.$$

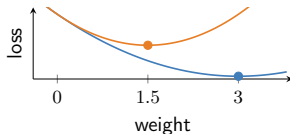
u_t : optimizer direction; $0 < \eta_t \lambda < 1$ gives shrinkage.

Weight Decay Pulls Parameters Toward Zero

Parameter Shrinkage



Preference For Smaller Weights



Data loss With L2 penalty

Plain GD illustration; excessive decay can underfit.

$$\theta_{t+1} = \underbrace{(1 - \eta_t \lambda)}_{\text{parameter shrinkage}} \theta_t - \eta_t u_t.$$

u_t : optimizer direction; $0 < \eta_t \lambda < 1$ gives shrinkage.

AdamW: Decay Separate From Adaptive Scaling.

Putting AdamW Together



$$\theta_{t+1} = (1 - \eta_t \lambda) \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

What would you add?

Expensive full gradients

Alternating directions

Unequal coordinate scales

Preference for smaller weights

Putting AdamW Together



$$\theta_{t+1} = (1 - \eta_t \lambda) \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

What would you add?

Expensive full gradients

Alternating directions

Unequal coordinate scales

Preference for smaller weights

A useful component

→ Sampling / minibatches

Putting AdamW Together



$$\theta_{t+1} = (1 - \eta_t \lambda) \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

What would you add?

Expensive full gradients

Alternating directions

Unequal coordinate scales

Preference for smaller weights

A useful component

→ Sampling / minibatches

→ Momentum

→ Adaptive scaling

Putting AdamW Together



$$\theta_{t+1} = (1 - \eta_t \lambda) \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

What would you add?

Expensive full gradients

Alternating directions

Unequal coordinate scales

Preference for smaller weights

A useful component

→ Sampling / minibatches

→ Momentum

→ Adaptive scaling

→ Weight decay