

CS 486/686

Neural Networks III

Yuntian Deng

Lecture 21

Optimization for deep learning · slides adapted from CMSC 35246 (Shubendu & Risi, U. Chicago)

Search ›

Uncertainty ›

Decisions ›

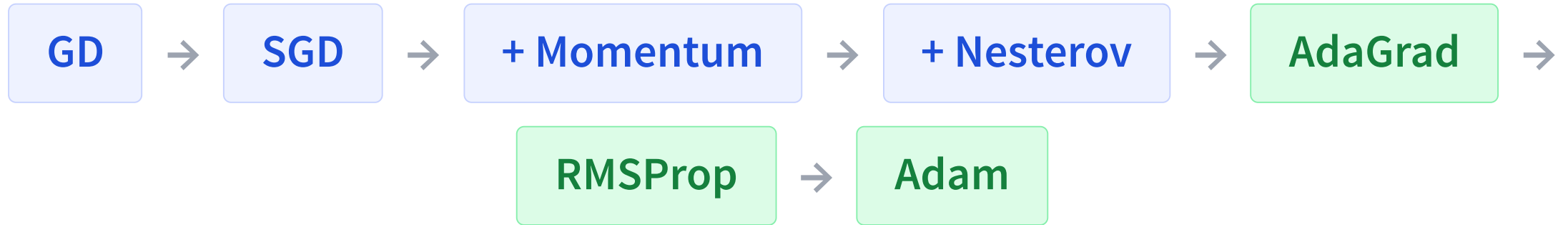
Learning

Learning goals

- **Stochastic gradient descent:** noisy steps, learning-rate schedules, convergence.
- **Momentum** and its **Nesterov** look-ahead variant.
- **Adaptive learning rates:** AdaGrad and RMSProp.
- **Adam:** adaptive moments combining the two ideas.

Roadmap: first-order optimizers

Backprop gives us $\nabla_{\theta} L$. The optimizer decides what to do with it.



- **SGD:** trade accuracy of gradient for speed of updates.
- **Momentum:** add inertia — escape ravines, damp oscillations.
- **Adaptive:** give each parameter its own learning rate.
- **Adam:** combine momentum + adaptive in one optimizer.

SGD recap

Batch GD:

$$\hat{g} \leftarrow \frac{1}{N} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$$

SGD (one example per step):

$$\hat{g} \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$$

In both: $\theta \leftarrow \theta - \varepsilon \hat{g}$.

Decay the learning rate linearly until iteration τ
:

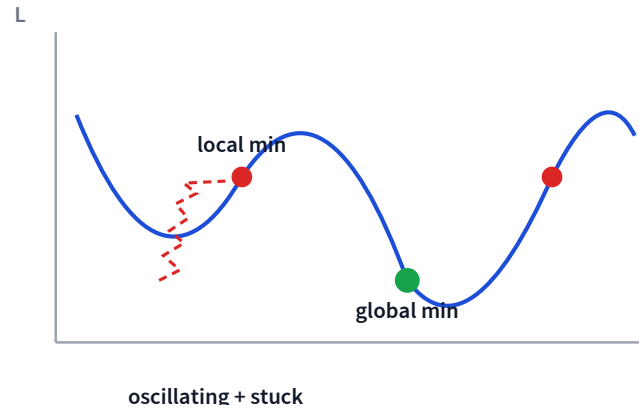
$$\varepsilon_k = (1 - \alpha) \varepsilon_0 + \alpha \varepsilon_{\tau}, \quad \alpha = k/\tau$$

Sufficient for convergence:

$$\sum_k \varepsilon_k = \infty, \quad \sum_k \varepsilon_k^2 < \infty$$

Mini-batch SGD (e.g. 32, 64): cheaper than full-batch, less noisy than per-example, scales to huge datasets.

What goes wrong with plain SGD



Local minima. Non-convex losses have many. SGD shrinks gradients to zero there and stalls.

Noisy oscillations. Each step uses only one example — the descent path zig-zags, slowing convergence.

Both issues motivate the next big idea:
momentum.

Momentum: add inertia

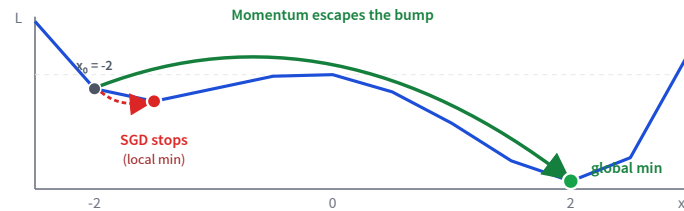
Introduce a **velocity** vector $\mathbf{v}$ — an exponentially weighted average of past gradients:

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \varepsilon \nabla_{\theta} L(\theta), \quad \theta \leftarrow \theta + \mathbf{v}$$

- $\alpha \in [0, 1)$ controls how much past gradients persist. Common: $\alpha = 0.9$.
- **Recent gradients dominate** — older ones decay geometrically.
- **Intuition:** a ball rolling down a hill keeps moving in the average direction, smoothing oscillations and powering through small bumps.

Worked example

Minimize $y = 0.3x^4 - 0.1x^3 - 2x^2 - 0.8x$, with gradient $g = 1.2x^3 - 0.3x^2 - 4x - 0.8$. Start at $x_0 = -2$.



SGD (no momentum)

iter	g_i	θ_i
1	-18.2	-1.885
2	-2.36	-1.76
3	-1.2	-1.70
4	-0.78	-1.66
...	...	...
99	0.0002	-1.586

stuck in local min

With momentum ($\alpha = 0.9$)

iter	g_i	v_i	θ_i
1	-18.2	0	-1.885
2	-2.36	-15.1	-1.12
3	1.61	-9.0	-0.67
4	1.39	-9.2	-0.21
...	...	...	...
99	0.0002	0.0003	2.042

reaches global min

Velocity carries the iterate over the local-min barrier near $x \approx -1.6$.

How big is one momentum step?

Plain SGD takes a step of size $\varepsilon \|g\|$. With momentum, recent gradients compound:

$$\text{step} \approx \varepsilon \|g_1\| + \alpha \varepsilon \|g_2\| + \alpha^2 \varepsilon \|g_3\| + \dots \approx \frac{\varepsilon \|\hat{g}\|}{1 - \alpha}$$

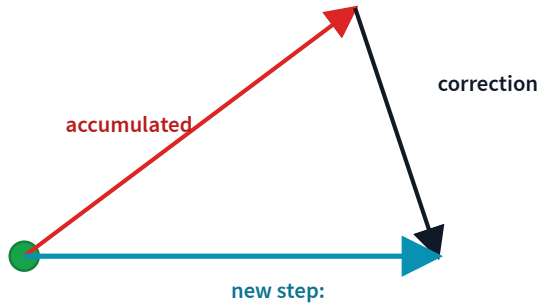
Concretely: $\alpha = 0.9 \Rightarrow$ effective step size is up to **10×** the per-step SGD step in a consistent direction.



In oscillating directions, opposite gradients cancel in $\mathbf{v}$ — momentum naturally damps noise.

Nesterov momentum: look ahead first

Idea: step in the direction of accumulated momentum, then evaluate the gradient there as a *correction*.



Standard momentum:

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \varepsilon \nabla_{\theta} L(\theta)$$

Nesterov momentum:

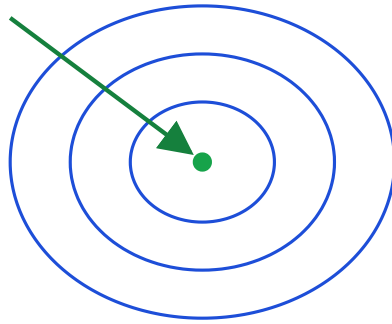
$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \varepsilon \nabla_{\theta} L(\theta + \alpha \mathbf{v})$$

$$\theta \leftarrow \theta + \mathbf{v}$$

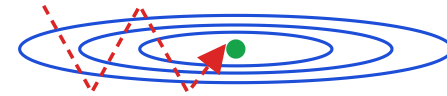
Gradient evaluated at the look-ahead point gives a better local correction.

Why one learning rate isn't enough

Features differ in scale and frequency. A single ε is too small for flat directions, too large for steep ones.



isotropic (easy)



anisotropic (zig-zag)

Give each parameter its **own** learning rate, scaled by its gradient history.

AdaGrad: accumulate squared gradients

Per-parameter learning rate, scaled by the running sum of squared gradients.

AdaGrad

Require: global LR ε , small δ ; init $\mathbf{r} = \mathbf{0}$.

1. Sample example $(x^{(i)}, y^{(i)})$; compute $\hat{\mathbf{g}} \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$.
2. Accumulate: $\mathbf{r} \leftarrow \mathbf{r} + \hat{\mathbf{g}} \odot \hat{\mathbf{g}}$
3. Update: $\Delta\theta \leftarrow -\frac{\varepsilon}{\delta + \sqrt{\mathbf{r}}} \odot \hat{\mathbf{g}}$
4. Apply: $\theta \leftarrow \theta + \Delta\theta$

Catch: $\mathbf{r}$ only grows — the effective LR shrinks to zero, training stalls in non-convex settings.

RMSProp: forget old gradients

Fix AdaGrad's shrink by replacing the running sum with an **exponentially decaying** moving average.

RMSProp

Require: LR ε , decay ρ (typically 0.9), small δ ; init $\mathbf{r} = 0$.

1. Compute gradient: $\hat{g} \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$.
2. Decay+accumulate: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \hat{g} \odot \hat{g}$
3. Update: $\Delta\theta \leftarrow -\frac{\varepsilon}{\delta + \sqrt{\mathbf{r}}} \odot \hat{g}$
4. Apply: $\theta \leftarrow \theta + \Delta\theta$

Related: **AdaDelta** goes one step further and removes the global LR ε by also tracking $\mathbb{E}[\Delta\theta^2]$.

Adam: ADAdaptive Moments

Track **first moment** (mean of $\hat{g}$) like momentum, and **second moment** (mean of $\hat{g}^2$) like RMSProp.

Adam

Require: LR ε , decay rates ρ_1, ρ_2 (typically 0.9, 0.999), small δ ; init $\mathbf{s} = \mathbf{0}, \mathbf{r} = \mathbf{0}, t = 0$.

1. Compute gradient $\hat{g}$; increment t .
2. **1st moment**: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \hat{g}$
3. **2nd moment**: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \hat{g} \odot \hat{g}$
4. **Bias correction**: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1^t}, \quad \hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2^t}$
5. Update: $\Delta\theta = -\varepsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}}$
6. Apply: $\theta \leftarrow \theta + \Delta\theta$

Bias correction fixes the cold-start: early $\mathbf{s}, \mathbf{r}$ are biased toward 0 because they start at 0.

Optimizer cheat sheet

Method	Update direction	Per-param LR?	When to use
SGD	$-\varepsilon \hat{g}$	No	Simple losses; baseline.
+ Momentum	$\mathbf{v} = \alpha \mathbf{v} - \varepsilon \hat{g}$	No	Noisy gradients, ravines, local minima.
RMSProp	$-\frac{\varepsilon}{\sqrt{\mathbf{r}} + \delta} \hat{g}$	Yes	RNNs, non-stationary or sparse gradients.
Adam	$-\varepsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}}$	Yes	Default choice in deep learning today.

Tune the learning rate first; pick the optimizer second.

Revisiting learning goals

- **Stochastic gradient descent:** noisy steps, learning-rate schedules, convergence.
- **Momentum** and its **Nesterov** look-ahead variant.
- **Adaptive learning rates:** AdaGrad and RMSProp.
- **Adam:** adaptive moments combining the two ideas.

Search ›

Uncertainty ›

Decisions ›

Learning ›

Recap

Next: L22 — course recap & exam prep

- A tour of everything we learned: search, CSPs, probabilistic reasoning, decision theory, MDPs, RL, ML.
- How the pieces fit into **modern AI systems**.
- Pointers to follow-up courses (CS480/680, CS485) and open research.
- **Final exam logistics:** what to bring, format, and a notation cheat sheet for your study card.

Thanks for a great term — see you on Tuesday!