# CS 486/686
# Neural Networks III

Yuntian Deng

Lecture 21

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

# 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.

GD → SGD → + Momentum → + Nesterov → AdaGrad →

RMSProp → Adam

- **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 $\tau$:

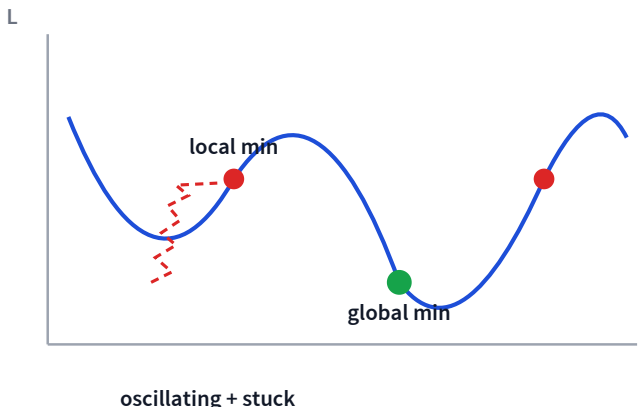
$$\varepsilon_k = (1 - \alpha)\, \varepsilon_0 + \alpha\, \varepsilon_\tau, \quad \alpha = k/\tau$$

**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

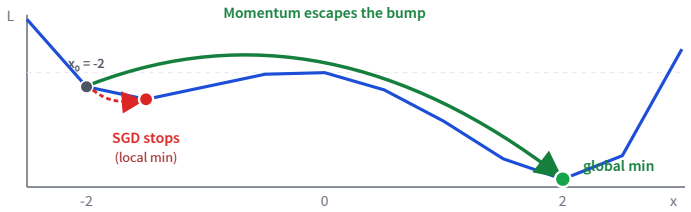
Introduce a **velocity** vector $\boldsymbol{v}$ — an exponentially weighted average of past gradients:

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

- $\alpha \in [0, 1)$ 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 | $g_i$ | $\theta_i$ |
|---|---|---|
| 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 | $g_i$ | $v_i$ | $\theta_i$ |
|---|---|---|---|
| 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 $\varepsilon \left\| g \right\|$. With momentum, recent gradients compound:

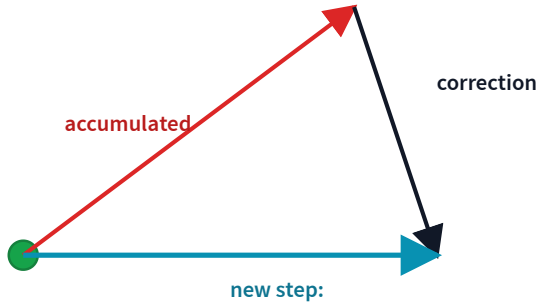
$$\text{step} \approx \varepsilon \|g_1\| + \alpha\, \varepsilon \|g_2\| + \alpha^2\, \varepsilon \|g_3\| + \cdots \approx \frac{\varepsilon \left\| \hat{g} \right\|}{1 - \alpha}$$

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


In oscillating directions, opposite gradients cancel in $\boldsymbol{v}$ — 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:**

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

**Nesterov momentum:**

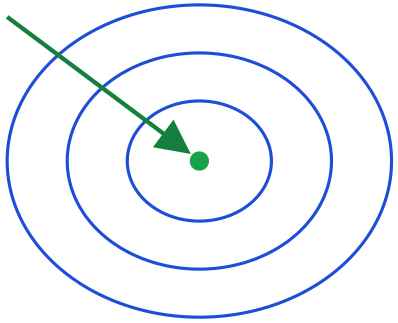
$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \varepsilon \nabla_\theta L(\theta + \alpha \boldsymbol{v})$$

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

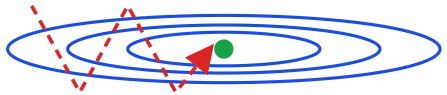
Gradient evaluated at the look-ahead point gives a better local correction.

# Why one learning rate isn't enough

Features differ in scale and frequency. A single $\varepsilon$ 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 $\varepsilon$, small $\delta$; init $\boldsymbol{r} = 0$.*

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

**Catch: $\boldsymbol{r}$** 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 $\varepsilon$, decay $\rho$ (typically 0.9), small $\delta$; init $\boldsymbol{r} = 0$.*

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

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

# Adam: ADAptive Moments

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

**Adam**

*Require: LR $\varepsilon$, decay rates $\rho_1, \rho_2$ (typically 0.9, 0.999), small $\delta$; init $\boldsymbol{s} = 0, \boldsymbol{r} = 0, t = 0$.*

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

Bias correction fixes the cold-start: early $\boldsymbol{s}, \boldsymbol{r}$ 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** | $\boldsymbol{v} = \alpha\boldsymbol{v} - \varepsilon\hat{g}$ | No | Noisy gradients, ravines, local minima. |
| **RMSProp** | $-\dfrac{\varepsilon}{\sqrt{\boldsymbol{r}} + \delta}\hat{g}$ | **Yes** | RNNs, non-stationary or sparse gradients. |
| **Adam** | $-\varepsilon\,\dfrac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ | **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.

# 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!