# Lecture 1.4: Optimization for Deep Networks

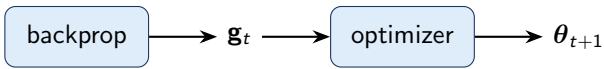
## Module 1: Foundations of Neural Networks
## Deep and Reinforcement Learning

Srikanth Pai

Madras School of Economics

# Where We Are Going

Backprop gives the **gradient**. Optimization decides **how to step**.




1-D picture → what breaks → SGD → momentum → RMSProp → Adam.

# We Optimise a Proxy

**True risk** (want)

$$R(\boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}}[\mathcal{L}]$$

never observed

**Empirical risk** (have)

$$\hat{R}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}^{(i)}$$

training set only

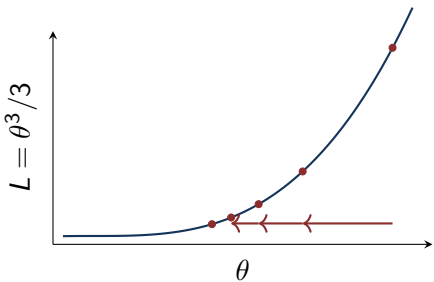
# Gradient Descent in One Variable

Toy objective $L(\theta) = \frac{1}{3}\theta^3$, $L'(\theta) = \theta^2$:

$$\theta_{t+1} = \theta_t - \eta\,\theta_t^2.$$

From $\theta_0 = 3$, $\eta = 0.1$:

| $t$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $\theta_t$ | 3.0 | 2.1 | 1.66 | 1.38 | 1.19 |
| step | .90 | .44 | .28 | .19 | .14 |




**Steps shrink** as the slope flattens; the iterate crawls toward the flat point. Too large an $\eta$ overshoots it.

# Challenge 1: The Learning Rate Is Bounded

Locally the loss is quadratic with curvature $\lambda$; the error obeys

$$e_{t+1} = (1 - \eta\lambda)\, e_t \quad \Longrightarrow \quad \text{stable} \quad \Longleftrightarrow \quad 0 < \eta < \tfrac{2}{\lambda}.$$

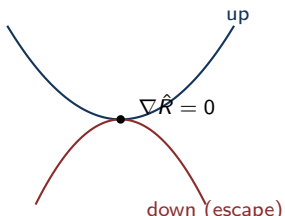
| learning rate | behaviour |
|---|---|
| $\eta > 2/\lambda$ | **diverges** (oscillates, grows) |
| $\eta$ small | stable but **slow** |
| $\eta \approx 1/\lambda$ | **safe and fast** |

Different directions have different $\lambda$, so **one global $\eta$ is a blunt instrument**. Momentum and adaptive methods fix this.

# Challenge 2: Saddle Points Are the Real Bottleneck

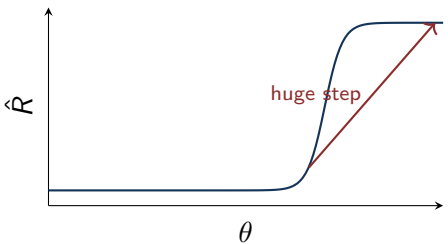
A minimum needs *all* $d$ directions up; one down $\Rightarrow$ **saddle**. In high $d$, saddles far outnumber minima.

At a saddle $\nabla \hat{R} \approx 0$, so the step is tiny and **GD stalls** for many steps before escaping.




**Evidence (Dauphin et al., 2014):** in trained nets, the fraction of negative-curvature directions grows with the loss, so high-loss critical points are saddles. SGD noise helps escape them.

# Challenge 3: Cliffs




A sudden large curvature: a normal step sails over the wall.

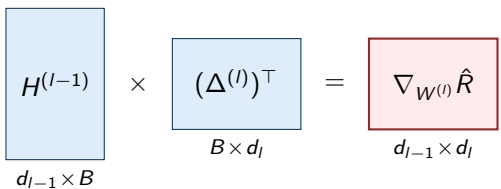
**Fix: gradient clipping**

$$\mathbf{g} \leftarrow \frac{c}{\|\mathbf{g}\|}\mathbf{g} \quad \text{if } \|\mathbf{g}\| > c.$$

Keep the direction, cap the length.

# Recall: Backprop Gives the Gradient, in Batches

Per layer, backprop gives $\nabla_{W^{(l)}}\mathcal{L} = \mathbf{h}^{(l-1)}(\boldsymbol{\delta}^{(l)})^\top$. Stack $B$ examples: one matrix multiply per layer.



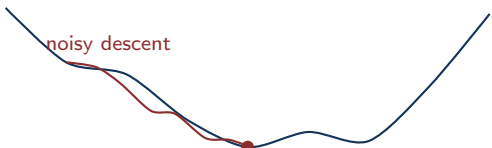

$$\nabla_{W^{(l)}} \hat{R} = \tfrac{1}{B} H^{(l-1)} (\Delta^{(l)})^\top. \qquad \textbf{The batch is just another matrix dimension.}$$

Call the result $\mathbf{g}_t$; every optimiser acts on it.

# SGD: A Mini-Batch Makes It Cheap

$$\mathbf{g}_t = \frac{1}{B}\sum_{i\in\mathcal{B}_t}\nabla\mathcal{L}^{(i)}, \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\,\mathbf{g}_t.$$

$O(B)$ per step, $\mathbb{E}[\mathbf{g}_t] = \nabla\hat{R}$, var $\propto 1/B$.




**The noise helps:** escapes saddles, avoids sharp minima.

**Convergence:** root-finding for $\nabla\hat{R} = 0$ from noisy readings, i.e. Robbins–Monro (1951). Needs $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$: decay the rate, not too fast.

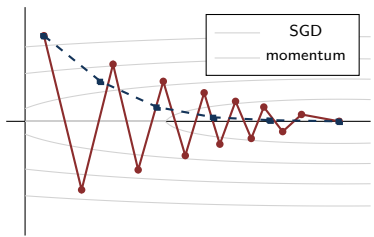
# Momentum: A Heavy Ball

$$\mathbf{v} \leftarrow \beta \mathbf{v} + \mathbf{g}_t, \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \mathbf{v}.$$

Unrolled: $\mathbf{v}_{t+1} = \sum_k \beta^{t-k} \mathbf{g}_k$.

| gradient pattern | velocity |
|---|---|
| $a, a, a, \ldots$ (along) | $\frac{a}{1-\beta}$ |
| $a, -a, a, \ldots$ (across) | $\frac{a}{1+\beta}$ |

Ratio $\frac{1-\beta}{1+\beta} \approx \frac{1}{19}$ at $\beta = 0.9$.



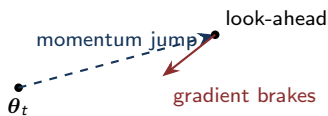
# Nesterov: Look Before You Leap

Evaluate the gradient *after* the inertial jump.
On the quadratic $\hat{R} = \frac{1}{2}\lambda x^2$:

$$\beta \longrightarrow \beta(1 - \eta\lambda).$$

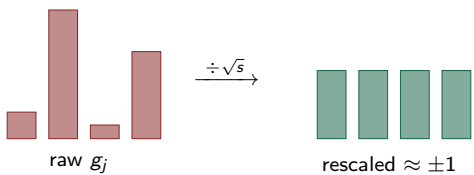
**Curvature-dependent damping:**

- steep ($\lambda$ large): more braking
- shallow ($\lambda$ small): keep momentum



# Coordinates Have Different Scales $\Rightarrow$ RMSProp

One global $\eta$ cannot fit $\mathbf{g}_t = (100,\ 0.1)^\top$: stable for one coordinate, useless for the other.




$$s \leftarrow \rho s + (1-\rho) g^2, \qquad \theta \leftarrow \theta - \frac{\eta}{\sqrt{s} + \varepsilon}\, g.$$

AdaGrad sums $g^2$ forever ($\eta/\sqrt{s} \to 0$, stalls); RMSProp uses a moving average (recent scale).

# Adam: Direction $\times$ Per-Coordinate Scale

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \, \frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{s}}} + \varepsilon}$$

$\hat{\mathbf{m}}$: smoothed gradient
**which way** (momentum)

$\sqrt{\hat{\mathbf{s}}}$: recent gradient size
**how big** (RMSProp)

Bias-correction $/(1-\beta^t)$ undoes the zero start.
**The everyday default** ($\beta_1 = 0.9$, $\beta_2 = 0.999$).

# Which Optimiser to Use?

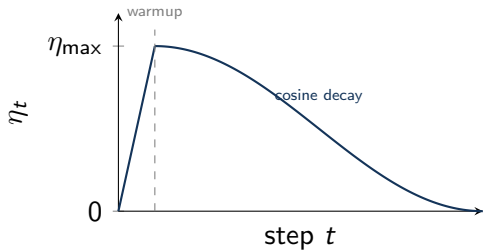
| **Optimiser** | **Adaptive** | **Memory** | **Default use** |
|---|---|---|---|
| SGD + momentum | No | $O(d)$ | vision, ResNets |
| RMSProp | Yes | $O(d)$ | RNNs |
| Adam | Yes | $O(d)$ | most tasks, transformers |

- Default: **Adam + warmup + cosine decay**.
- SGD + momentum often beats Adam on image classification.
- Tune $\eta$ and $B$ first; the rest matters less.

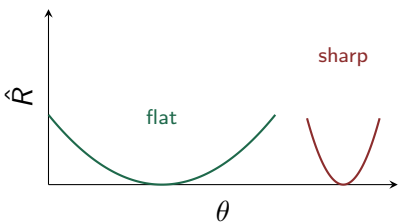
# Learning Rate Schedules

**Warmup** then **cosine decay**.

Warmup: early gradients are noisy, so start small (standard for transformers). Then decay smoothly to settle into a minimum.



# Flat Minima and the Edge of Stability




Flatter minima *tend* to generalise better (Keskar et al., 2017), but flatness is not reparameterisation-invariant: ReLU rescaling makes the same function look sharper or flatter (Dinh et al., 2017). A **rule of thumb**, not a law.

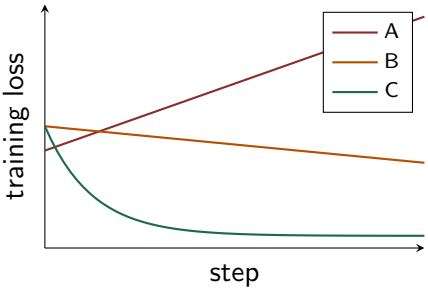
**Edge of stability:** training often sits near sharpness $= 2/\eta$ (Cohen et al., 2021).

# Discussion

## Talk it through

1. Why does the *same* learning rate diverge in one direction and crawl in another? What single number captures this?
2. Momentum suppresses the across-valley direction about $19\times$ at $\beta = 0.9$. Does it *cancel* it? What actually happens?
3. RMSProp divides by $\sqrt{s}$. Is $\sqrt{s}$ a curvature? What is it?
4. You switch from SGD to Adam and training gets faster but the test accuracy drops slightly. Plausible? Why?

# Practice: Read the Training Curves




**Try it (5 min)**

1. Which curve has $\eta$ **too large**? too **small**? well-tuned?
2. For B, what one change would you try first?
3. C plateaus. Name one thing that could lower it further.

# Practice: One Step of Adam

## Try it (5 min)

Cold start: $\mathbf{m}_0 = \mathbf{s}_0 = \mathbf{0}$, $\boldsymbol{\theta}_0 = (1, -1)^\top$, $\mathbf{g}_1 = (0.5, -0.3)^\top$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\eta = 10^{-3}$.

1. Compute $\mathbf{m}_1, \mathbf{s}_1$, then $\hat{\mathbf{m}}_1, \hat{\mathbf{s}}_1$; note how much the correction matters at $t = 1$.
2. Compute $\boldsymbol{\theta}_1$.
3. What is the effective rate per coordinate? Is it $\eta$? Why does Adam still move sensibly from zero moments?

# Key Takeaways

Backprop gives $\mathbf{g}_t$; the optimiser decides the step.

- **GD** has a hard learning-rate bound $\eta < 2/\lambda$; **saddles**, not local minima, are the obstacle.
- **SGD**: a mini-batch is cheap and its noise helps.
- **Momentum**: reinforces persistent directions, suppresses reversals.
- **RMSProp**: per-coordinate scaling by recent gradient size.
- **Adam**: momentum + RMSProp, the everyday default.