

深度学习 — 第三、四讲

深度模型的训练

江伟森

- 一 GD 的稳定性、SGD、Momentum、Adam
- 二 AdamW、离散变量的梯度、参数初始化
- 三 学习率调度、Warmup、SAM

本讲主线：一个更新公式

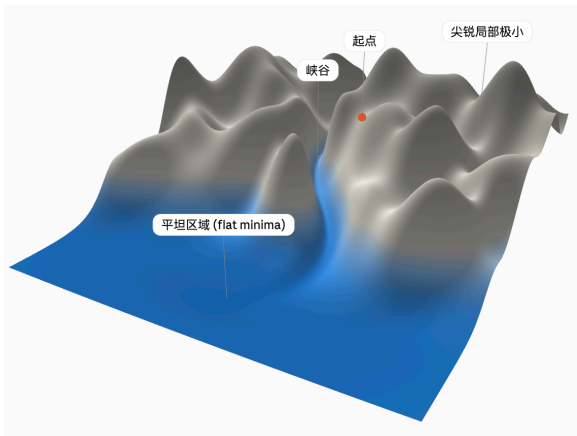
几乎所有深度学习优化器都可以写成

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{U}_t, \quad \mathbf{w}_0 \sim \text{初始化分布}$$

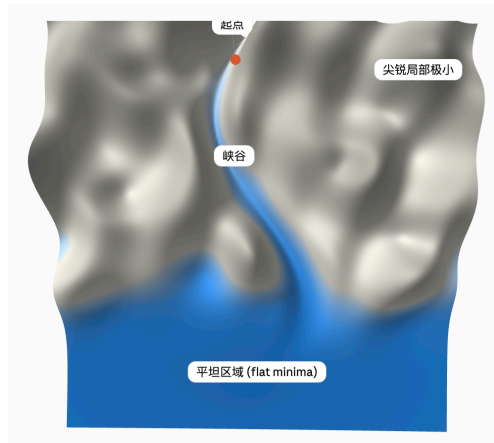
本讲回答三个问题：

对象	问题	本讲内容
$\mathbf{U}_t$	往哪个方向走？	SGD / Momentum / Adam
η_t	每一步走多远？	稳定性条件、学习率调度、Warmup
$\nabla \mathcal{L}$	离散操作如何求梯度？	STE / Gumbel-Softmax / REINFORCE
$\mathbf{w}_0$	从哪里出发？	He 初始化

深度网络的损失景观



立体视角



俯视视角

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化
- 5 学习率调度
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

L-光滑与下降引理

L-光滑（如何证明？）

$$\|\nabla \mathcal{L}(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\| \implies \mathcal{L}(\mathbf{w}') \leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{w}' - \mathbf{w} \rangle + \frac{L}{2}\|\mathbf{w}' - \mathbf{w}\|^2$$

代入 GD 步 $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \mathcal{L}(\mathbf{w}_t)$:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) - \eta \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{L\eta^2}{2} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 \\ &= \mathcal{L}(\mathbf{w}_t) - \underbrace{\eta \left(1 - \frac{L\eta}{2}\right)}_{\text{保证下降量系数}} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\end{aligned}$$

- 系数 $> 0 \iff \boxed{\eta < 2/L}$: 每一步都能保证下降
- 系数在 $\eta = 1/L$ 处最大（为什么），得到第二讲的形式 $\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \frac{1}{2L} \|\nabla \mathcal{L}\|^2$
- $\eta \geq 2/L$: 下降引理不再给出保证，这时会发生什么？

二次模型：把 GD 看清楚

考虑二次损失（任何光滑函数在极小点附近的二阶近似）：

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \mathbf{w}^\top \mathbf{H} \mathbf{w}, \quad \mathbf{H} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^\top, \quad \lambda_1 \geq \dots \geq \lambda_d > 0$$

GD 更新： $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \mathcal{L}(\mathbf{w}_t) = \mathbf{w}_t - \eta \mathbf{H} \mathbf{w}_t = (\mathbf{I} - \eta \mathbf{H}) \mathbf{w}_t$

在特征向量坐标 $\mathbf{c} = \mathbf{Q}^\top \mathbf{w}$ 下（利用 $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$ ）：

$$\mathbf{c}_{t+1} = \mathbf{Q}^\top \mathbf{w}_{t+1} = \mathbf{Q}^\top (\mathbf{I} - \eta \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^\top) \mathbf{w}_t = (\mathbf{I} - \eta \mathbf{\Lambda}) \mathbf{Q}^\top \mathbf{w}_t = (\mathbf{I} - \eta \mathbf{\Lambda}) \mathbf{c}_t$$

$\mathbf{I} - \eta \mathbf{\Lambda}$ 是对角矩阵，所以各方向完全解耦：

$$\boxed{c_{t+1}^{(i)} = (1 - \eta \lambda_i) c_t^{(i)} \implies c_t^{(i)} = (1 - \eta \lambda_i)^t c_0^{(i)}}$$

- 每个方向是一个独立的一维几何序列，公比 $r_i = 1 - \eta \lambda_i$
- 收敛 $\iff$ 对所有 i , $|1 - \eta \lambda_i| < 1$
- 对二次函数， $L = \lambda_1 = \lambda_{\max}(\mathbf{H})$

一维情形：四种行为

$\mathcal{L}(w) = \frac{\lambda}{2} w^2$, $w_{t+1} = (1 - \eta\lambda) w_t$. 取 $\lambda = 1$, $w_0 = 1$:

步长范围	公比 $1 - \eta\lambda$	行为	例: w_t
$0 < \eta < 1/\lambda$	$(0, 1)$	单调收敛	$\eta = 0.5$: $1, 0.5, 0.25, \dots$
$1/\lambda < \eta < 2/\lambda$	$(-1, 0)$	振荡收敛	$\eta = 1.5$: $1, -0.5, 0.25, \dots$
$\eta = 2/\lambda$	-1	等幅振荡	$\eta = 2$: $1, -1, 1, \dots$
$\eta > 2/\lambda$	< -1	振荡发散	$\eta = 2.5$: $1, -1.5, 2.25, \dots$

关键观察

- 发散的方式是越来越剧烈的来回振荡，而不是单调远离
- 阈值 $2/\lambda$ 与下降引理的 $2/L$ 一致：曲率越大，能容忍的步长越小

多维情形：稳定性

稳定性条件

所有方向都收敛 $\iff \boxed{\eta < \frac{2}{\lambda_{\max}}}$ ——由最陡的方向决定

高维空间：用 Hessian，看一步 GD 让 loss 变了多少

当前点 $\mathbf{w}$ ，梯度 $\mathbf{g} = \nabla \mathcal{L}(\mathbf{w})$ ，Hessian $\mathbf{H} = \nabla^2 \mathcal{L}(\mathbf{w})$ 。把 GD 的一步 $\Delta \mathbf{w} = -\eta \mathbf{g}$ 代入二阶泰勒展开：

$$\mathcal{L}(\mathbf{w} - \eta \mathbf{g}) \approx \mathcal{L}(\mathbf{w}) \underbrace{- \eta \mathbf{g}^\top \mathbf{g}}_{\text{一阶：下降}} + \underbrace{\frac{1}{2} \eta^2 \mathbf{g}^\top \mathbf{H} \mathbf{g}}_{\text{二阶：曲率带来的上升}}$$

- 一阶项：沿负梯度走，loss 按直线下降，步长越大降得越多
- 二阶项：曲面是弯的，走远了会“弯回来”，loss 反而上升； $\mathbf{g}^\top \mathbf{H} \mathbf{g}$ 就是沿梯度方向的曲率
- 一阶项 $\propto \eta$ ，二阶项 $\propto \eta^2$ ：步长小时下降占优，步长大时上升占优

loss 下降的条件

$$\eta \|\mathbf{g}\|^2 > \frac{1}{2} \eta^2 \mathbf{g}^\top \mathbf{H} \mathbf{g} \quad \Longleftrightarrow \quad \eta < \frac{2 \|\mathbf{g}\|^2}{\mathbf{g}^\top \mathbf{H} \mathbf{g}}$$

最坏方向：Hessian 的最大特征值

分母 $\mathbf{g}^\top \mathbf{H} \mathbf{g}$ 取决于梯度指向哪个方向：

- 梯度指向平缓方向：曲率小，可以用大步长
- 梯度指向最陡方向（ $\mathbf{H}$ 的最大特征向量）： $\mathbf{g}^\top \mathbf{H} \mathbf{g} = \lambda_{\max} \|\mathbf{g}\|^2$

由于 $\mathbf{g}^\top \mathbf{H} \mathbf{g} \leq \lambda_{\max}(\mathbf{H}) \|\mathbf{g}\|^2$ ，无论梯度指向哪里 loss 都下降的条件是

$$\eta < \frac{2}{\lambda_{\max}(\mathbf{H})}$$

注意： L 是整个参数空间里最坏的曲率； $\lambda_{\max}(\mathbf{H})$ 是当前位置的曲率。

深度网络中的 sharpness

把局部 Hessian 的最大特征值定义为 sharpness（局部最大曲率）：

$$S(\mathbf{w}) = \lambda_{\max}(\nabla^2 \mathcal{L}(\mathbf{w})) \quad (\text{sharpness})$$

与 L -光滑的关系： $L = \sup_{\mathbf{w}} \|\nabla^2 \mathcal{L}(\mathbf{w})\|$ 是全局最大曲率， $S(\mathbf{w}_t)$ 是当前位置的曲率， $S(\mathbf{w}_t) \leq L$ 。

二次模型给出的“经典预言”：

经典稳定性观点

$$\text{GD 在 } \mathbf{w}_t \text{ 附近局部稳定} \iff S(\mathbf{w}_t) < 2/\eta$$

在深度网络中， $S(\mathbf{w}_t)$ 随训练不断变化：同一个 η ，在尖锐的地方可能发散，在平坦的地方则很稳定。

如何选择合适的步长？ → **LR schedule, Warmup, SAM**

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化
- 5 学习率调度
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

SGD: 随机梯度下降

训练目标 $\mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{w}; \mathbf{x}_i, y_i)$, 全量梯度每步要遍历全部 n 个样本, 太贵。

SGD 公式

每步随机抽取 B 个样本 $\mathcal{B}_t$:

$$\mathbf{g}_t = \frac{1}{B} \sum_{i \in \mathcal{B}_t} \nabla \ell(\mathbf{w}_t; \mathbf{x}_i, y_i), \quad \boxed{\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t}$$

- $\mathbf{g}_t$ 是全梯度的无偏估计: $\mathbb{E}[\mathbf{g}_t] = \nabla \mathcal{L}(\mathbf{w}_t)$, 噪声方差约 $\propto 1/B$
- 术语: 一次更新 = 1 个 **step**; 遍历一遍数据 = 1 个 **epoch**

SGD 的问题：振荡与缓慢

回到二次模型：各方向的曲率 λ_i 不同，假如条件数 $\kappa = \lambda_{\max}/\lambda_{\min}$ 很大

- 陡峭方向： η 必须小于 $2/\lambda_{\max}$ ，公比 $1 - \eta\lambda_{\max}$ 接近 $-1 \Rightarrow$ 来回振荡
- 平缓方向：公比 $1 - \eta\lambda_{\min}$ 接近 $1 \Rightarrow$ 几乎不动

例： $\kappa = 100$

GD 最好情况下每步误差缩小为 0.98，误差降到 1% 需要约 **230** 步。

观察相邻两步的梯度：

- 陡峭方向：梯度符号来回翻转（在两侧山坡间跳）
- 平缓方向：梯度符号始终一致（一直朝同一个方向）

改进思路 $\Rightarrow$ Momentum

把历史梯度累加起来：翻转的分量相互抵消，一致的分量不断积累。

Momentum: 累加历史梯度

Momentum 公式 (PyTorch 写法)

$$\mathbf{b}_t = \beta \mathbf{b}_{t-1} + \mathbf{g}_t, \quad \boxed{\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{b}_t}, \quad \beta \in [0, 1) \text{ (常用0.9)}$$

展开: $\mathbf{b}_t = \mathbf{g}_t + \beta \mathbf{g}_{t-1} + \beta^2 \mathbf{g}_{t-2} + \dots$ —— 历史梯度的指数加权和

- 一致方向: 梯度恒为 $\mathbf{g}$ 时 $\mathbf{b}_t \rightarrow \mathbf{g}/(1 - \beta)$, 步长放大 $\frac{1}{1-\beta} = 10$ 倍
- 翻转方向: $\mathbf{g}, -\mathbf{g}, \mathbf{g}, \dots$ 相互抵消, 振荡被抑制

$\kappa = 100$	每步误差缩小为	误差降到 1% 的步数
SGD / GD	0.980	≈ 230
Momentum	0.818	≈ 23

Momentum 的问题：所有方向仍共用一个学习率

Momentum 的稳定条件（二次模型，推导略）：对所有方向 j ,

$$\eta \lambda_j < 2(1 + \beta) \implies \eta < \frac{2(1 + \beta)}{\lambda_{\max}} \propto \frac{1}{\lambda_{\max}}$$

- η 仍由最陡的方向决定，平缓方向仍然走得慢，只是改善了常数和速率
- 深度网络中，不同参数（不同层、embedding 等）的曲率可以相差好几个数量级

误差降到 1% 的步数	$\kappa = 10^2$	$\kappa = 10^4$	$\kappa = 10^6$
SGD / GD ($\approx 2.3 \kappa$)	230	2.3×10^4	2.3×10^6
Momentum ($\approx 2.3\sqrt{\kappa}$)	23	230	2300

根本原因：Momentum 对所有方向一视同仁，同一个 η 、同一个 β 。

改进思路 $\Rightarrow$ Adam

每个方向用自己的学习率：不稳定（陡峭）的方向 lr 小，稳定（平缓）的方向 lr 大。

Adam 的想法：除以 $\sqrt{\mathbb{E}[g_j^2]}$

Idea：用每个参数梯度的典型大小去除梯度

$$w_j \leftarrow w_j - \eta \frac{g_j}{\sqrt{\mathbb{E}[g_j^2]}} \quad \Longleftrightarrow \quad \text{每个方向的有效学习率 } \eta_j = \frac{\eta}{\sqrt{\mathbb{E}[g_j^2]}}$$

为什么这样选？理想做法是 $\eta_j \propto 1/\lambda_j$ ，但 λ_j 未知（Hessian 太贵）。梯度大小可以作为代理：二次模型中 $g_j = \lambda_j \delta_j$ ，陡峭方向梯度更大，振荡时还会正负翻转。

方向	梯度的特征	$\sqrt{\mathbb{E}[g_j^2]}$	有效学习率 η_j
陡峭、振荡（不稳定）	大，且正负翻转	大	小
平缓（稳定）	小，方向一致	小	大

为什么要平方：振荡方向上 $\mathbb{E}[g_j] \approx 0$ （正负抵消），只看均值会误以为“梯度很小”； $\mathbb{E}[g_j^2]$ 不受符号影响，能察觉到这种不稳定。

Adam: Momentum + 按参数归一化

Adam 公式 (Kingma & Ba, 2015; 所有运算按坐标进行)

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$$

一阶矩：梯度的平均（即 Momentum）

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$$

二阶矩：梯度平方的平均（梯度尺度）

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}$$

偏差修正 ($\mathbf{m}_0 = \mathbf{v}_0 = \mathbf{0}$, 初期偏小)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}$$

对应关系：分子 $\hat{\mathbf{m}}_t$ 就是 Momentum（梯度的平均）；分母 $\sqrt{\hat{\mathbf{v}}_t}$ 就是上一页的 $\sqrt{\mathbb{E}[\mathbf{g}_j^2]}$ ，用滑动平均来估计。

默认值 $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ (LLM 常用 $\beta_2 = 0.95$)；代价：每个参数多存 $\mathbf{m}, \mathbf{v}$ (内存 $2\times$)。

三种优化器对比

	SGD	Momentum	Adam
更新量 U_t	$\mathbf{g}_t$	$\mathbf{b}_t = \beta \mathbf{b}_{t-1} + \mathbf{g}_t$	$\hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon)$
引入的改进	——	抑制振荡、加速	每个参数自己的 lr
解决的问题	——	SGD 振荡、慢	各方向共用一个 lr
额外内存	无	1×	2×
超参数	η	η, β	η, β_1, β_2
典型用途	简单问题	CNN	Transformer、LLM

SGD $\xrightarrow{+ \text{历史梯度}}$ Momentum $\xrightarrow{\div \text{梯度尺度}}$ Adam

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度**
- 4 参数初始化
- 5 学习率调度
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

问题：离散操作“挡住”了梯度

例：二值化网络（权重只取 ± 1 ：内存省 32 倍，乘法变成加减）

目标：学习一个网络参数 θ ，使得二值化后的 θ_b 效果好

$$\min_{\theta} \mathcal{L}(\theta_b), \quad \theta_b = \text{sign}(\theta)$$

前向用 θ_b 计算预测和损失；反向时由链式法则

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \theta_b} \cdot \underbrace{\frac{\partial \text{sign}(\theta)}{\partial \theta}}_{=0} = 0$$

- sign 是阶梯函数：只要 θ_j 不跨过 0， $\theta_{b,j}$ 就不变，导数几乎处处为 0
- $\partial \mathcal{L} / \partial \theta_b$ 明明告诉我们哪些权重该从 -1 翻到 $+1$ ，却传不到 $\theta \Rightarrow \theta$ 永远不会更新
- 为什么不直接学 θ_b ? $\{-1, +1\}^d$ 是离散的，无法做梯度下降

离散操作无处不在

离散操作	出现的场景
四舍五入 $\text{round}(x)$ 、符号 $\text{sign}(x)$	模型量化、低比特网络
从码本中选最近的向量	VQ-VAE、图像 tokenizer
从类别分布中采样 $z \sim \text{Cat}(\pi_\theta)$	离散隐变量、MoE 路由
LLM 生成一段文本，再打分	RLHF、推理模型训练

两类问题

- ① 网络中有确定性的离散函数： $y = \text{round}(x)$ $\rightarrow$ **STE**
- ② 对离散随机变量求期望： $\min_\theta \mathbb{E}_{z \sim p_\theta}[f(z)]$ $\rightarrow$ **Gumbel-Softmax / REINFORCE**

方法一：Straight-Through Estimator (STE)

STE (Bengio et al., 2013)

前向照常使用离散函数，反向时假装它是恒等映射：

$$\text{前向: } y = \text{round}(x), \quad \text{反向: } \frac{\partial \mathcal{L}}{\partial x} := \frac{\partial \mathcal{L}}{\partial y} \quad \left(\text{即令 } \frac{\partial y}{\partial x} = 1 \right)$$

PyTorch 中的一行实现 (`detach()` 部分不参与求导)：

```
y = x + (torch.round(x) - x).detach()  
# forward: y == round(x); backward: dy/dx == 1
```

- 优点：极其简单，适用于任何确定性的离散函数
- 缺点：梯度是有偏的（“假装”的导数），没有理论保证，但实践中很好用
- 回到二值化的例子：STE 令 $\partial \text{sign}(\theta) / \partial \theta = 1$ （通常只在 $|\theta| \leq 1$ 时传梯度），即 $\partial \mathcal{L} / \partial \theta \approx \partial \mathcal{L} / \partial \theta_b$
⇒ 用二值权重处的梯度更新实值 θ ， θ 跨过 0 时 θ_b 翻转 (BinaryConnect, 2015)

连续变量的重参数化技巧

若 z 是连续随机变量，例如 $z \sim \mathcal{N}(\mu, \sigma^2)$ ， $\theta = (\mu, \sigma)$ ，可以把随机性“移出去”：

$$z = \mu + \sigma \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

于是期望对 θ 求导可以交换到期望里面：

$$\nabla_{\mu} \mathbb{E}_{z \sim \mathcal{N}(\mu, \sigma^2)} [f(z)] = \nabla_{\mu} \mathbb{E}_{\epsilon} [f(\mu + \sigma \epsilon)] = \mathbb{E}_{\epsilon} [\nabla_{\mu} f(\mu + \sigma \epsilon)]$$

用几个样本 ϵ 估计即可，这就是 VAE 训练的关键 (Kingma & Welling, 2014)。

离散变量为什么不行？

离散变量也可以写成 $z = g(\theta, \epsilon)$ ，但 g 对 θ 是分段常数： θ 稍微变化， z 要么不变，要么突然跳到另一个类别。 $\Rightarrow \nabla_{\theta} g = 0$ ，重参数化给不出有用的梯度。

方法二：ST Gumbel-Softmax（第一步：前向用 argmax 采样）

从 θ 到 **loss** 的计算过程（ z 是 one-hot 向量）：

$$\theta \xrightarrow{\text{网络}} \mathbf{s} = h_{\theta}(\mathbf{x}) \xrightarrow{\text{softmax}} \boldsymbol{\pi} \xrightarrow{\text{采样}} z \sim \text{Cat}(\boldsymbol{\pi}) \xrightarrow{\text{后续网络}} \mathcal{L} = f(z)$$

目标 $\min_{\theta} \mathbb{E}_{z \sim \text{Cat}(\boldsymbol{\pi}_{\theta})}[f(z)]$ ，但“采样”这一步对 $\boldsymbol{\pi}$ 没有导数。

Gumbel-Max 技巧（证明略）：把采样写成 argmax

$$z = \text{onehot}\left(\arg\max_i (\log \pi_i + G_i)\right) \sim \text{Cat}(\boldsymbol{\pi}), \quad G_i = -\log(-\log U_i), \quad U_i \sim U(0, 1)$$

argmax 返回最大分量的下标，例： $\arg\max(1.2, 3.0, 0.5) = 2 \rightarrow (0, 1, 0)$

随机性全在 G_i 里，与 $\boldsymbol{\pi}$ 无关（类似重参数化）。链式法则：
$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial f}{\partial z} \cdot \underbrace{\frac{\partial z}{\partial \boldsymbol{\pi}}}_{=0} \cdot \frac{\partial \boldsymbol{\pi}}{\partial \theta} = 0$$

argmax 仍然不可导： $\boldsymbol{\pi}$ 稍微变化，最大值的下标通常不变， z 不变（和 sign 一样是阶梯函数）。

方法二：ST Gumbel-Softmax（第二步：反向用 softmax 的梯度）

带温度的 **softmax** 是 **argmax** 的光滑近似： $\text{softmax}(\mathbf{a}/\tau)_i = \exp(a_i/\tau) / \sum_j \exp(a_j/\tau)$

$\mathbf{a} = (1.2, 3.0, 0.5)$	$\text{softmax}(\mathbf{a}/\tau)$	
$\tau = 10$	$(0.32, 0.38, 0.30)$	接近均匀：梯度平滑，但偏差大
$\tau = 1$	$(0.13, 0.80, 0.07)$	
$\tau = 0.1$	$(0.00, 1.00, 0.00)$	接近 one-hot：偏差小，但梯度方差大

Straight-Through Gumbel-Softmax (Jang et al., 2017)

前向：用 **argmax** 得到真正的 one-hot 样本 $\mathbf{z} = \text{onehot}(\arg \max_i (\log \pi_i + G_i))$

反向：用软向量 $\mathbf{y} = \text{softmax}((\log \boldsymbol{\pi} + \mathbf{G})/\tau)$ 的梯度代替，即令 $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\pi}} := \frac{\partial \mathbf{y}}{\partial \boldsymbol{\pi}}$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \approx \frac{\partial f}{\partial \mathbf{z}} \cdot \underbrace{\frac{\partial \mathbf{y}}{\partial \boldsymbol{\pi}}}_{\text{代替0}} \cdot \frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\theta}} \quad (\text{上一页中这一项为 0})$$

ST Gumbel-Softmax 的实现与应用

实现：与 STE 的写法完全相同，前向值是 y_{hard} ，梯度来自 y_{soft}

```
G = -torch.log(-torch.log(torch.rand_like(logits))) # Gumbel noise
y_soft = torch.softmax((logits + G) / tau, dim=-1)
y_hard = F.one_hot(y_soft.argmax(-1), K).float()
z = y_hard - y_soft.detach() + y_soft
# same as: F.gumbel_softmax(logits, tau=tau, hard=True)
```

与 **STE** 的关系：同一个思路——前向用离散值，反向用可导的替代。STE 假装导数是 1；ST Gumbel-Softmax 借用 softmax 的导数。

应用（网络内部需要离散选择，又想端到端训练）：

- 语音：wav2vec 2.0（2020）从码本中选码字，把语音特征离散化
- 图像 tokenizer：DALL·E（2021）的离散 VAE，每个位置从 8192 个码字中选一个
- 神经架构搜索：SNAS、GDAS、FBNet（2019）为每层“选一个”候选操作
- 动态网络：DynamicViT（2021）决定每个 token 保留还是丢弃

变体：若后续网络可以接受软向量，也可以前向直接用 y （原始 Gumbel-Softmax）。

方法三：REINFORCE / Score Function 估计

目标： $J(\theta) = \mathbb{E}_{z \sim p_\theta} [f(z)] = \sum_z p_\theta(z) f(z)$ 。直接求导：

$$\begin{aligned}\nabla_\theta J &= \sum_z f(z) \nabla_\theta p_\theta(z) = \sum_z f(z) p_\theta(z) \nabla_\theta \log p_\theta(z) & (\nabla p = p \nabla \log p) \\ &= \boxed{\mathbb{E}_{z \sim p_\theta} [f(z) \nabla_\theta \log p_\theta(z)]}\end{aligned}$$

估计：采样 $z_1, \dots, z_N \sim p_\theta$, $\nabla_\theta J \approx \frac{1}{N} \sum_{k=1}^N f(z_k) \nabla_\theta \log p_\theta(z_k)$

特点 (Williams, 1992)

- 只需要 $f(z)$ 的值： f 可以不可导，甚至是黑盒（人类打分、程序是否通过测试）
- 无偏估计
- 直观理解： $f(z)$ 大的样本，提高它的概率； $f(z)$ 小的样本，降低它的概率
- 缺点：方差很大，需要很多样本

降低方差：Baseline

关键事实：score function 的期望为 0

$$\mathbb{E}_{z \sim p_\theta} [\nabla_\theta \log p_\theta(z)] = \sum_z \nabla_\theta p_\theta(z) = \nabla_\theta \sum_z p_\theta(z) = \nabla_\theta 1 = 0$$

所以对任意常数 b ，减去它不改变期望：

$$\nabla_\theta J = \mathbb{E}_{z \sim p_\theta} \left[(f(z) - b) \nabla_\theta \log p_\theta(z) \right]$$

- 常用 $b =$ 样本上 f 的平均值
- $f(z) - b$ 称为**优势** (advantage)：比平均好多少
- 例： f 恒在 $[99, 101]$ 之间。不减 baseline 时，每个样本都在“提高概率”，只是程度略有差别；减去 $b = 100$ 后，好样本提高、差样本降低，信号清晰得多

REINFORCE 的应用：大模型强化学习

把 LLM 生成看作从离散分布中采样：

- z = 模型对问题 x 生成的回答 $y = (y_1, \dots, y_T)$ ，每个 token 都是离散的
- $\log \pi_{\theta}(y | x) = \sum_{t=1}^T \log \pi_{\theta}(y_t | x, y_{<t})$ ：可以用反向传播求导
- f = 奖励 $r(x, y)$ ：答案对不对、人类偏好打分.....不可导

$$\nabla_{\theta} \mathbb{E}_{y \sim \pi_{\theta}} [r(x, y)] = \mathbb{E}_{y \sim \pi_{\theta}} \left[(r(x, y) - b) \nabla_{\theta} \log \pi_{\theta}(y | x) \right]$$

GRPO (DeepSeekMath 2024; DeepSeek-R1 2025)

对每个问题采样 G 个回答，得到奖励 $r_1, \dots, r_G$ ，用组内平均作 baseline：

$$A_i = \frac{r_i - \text{mean}(r_1, \dots, r_G)}{\text{std}(r_1, \dots, r_G)}$$

核心就是 **REINFORCE + baseline**（再加上裁剪和 KL 约束等稳定性技巧）。

三种方法对比

	STE	ST Gumbel-Softmax	REINFORCE
适用对象	确定性离散函数	类别采样	任意离散采样
要求 f 可导	是	是	否
梯度偏差	有偏	有偏 ($\tau > 0$)	无偏
梯度方差	低	中	高 (需 baseline)
实现难度	一行代码	一行代码	需要采样多个样本
典型应用	量化、VQ-VAE	NAS、wav2vec 2.0	强化学习、LLM 后训练

选择原则：

- f 可导、只想让梯度“通过” $\Rightarrow$ STE 或 Gumbel-Softmax
- f 不可导（奖励、黑盒评价） $\Rightarrow$ REINFORCE

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化**
- 5 学习率调度
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

为什么初始化很重要？

全零（或全相同）初始化：

- 同一层所有隐藏单元计算相同的函数，得到相同的梯度
- 更新后仍然相同 $\Rightarrow$ 对称性永远无法打破，相当于每层只有一个神经元

$\Rightarrow$ 必须随机初始化。

随机初始化的尺度：

- 太小：信号逐层衰减 $\Rightarrow$ 梯度消失
- 太大：信号逐层放大 $\Rightarrow$ 梯度爆炸 / 激活饱和

目标

选择权重方差 σ^2 ，使前向激活和反向梯度的方差在各层之间保持不变。

前向传播中的方差

考虑一层 $z_i = \sum_{j=1}^{n_{\text{in}}} W_{ij} x_j$, 假设:

- W_{ij} i.i.d., $\mathbb{E}[W_{ij}] = 0$, $\text{Var}(W_{ij}) = \sigma^2$
- W 与 x 独立, x_j 同分布

由于 $\mathbb{E}[W_{ij}] = 0$, 各项零均值且不相关:

$$\text{Var}(z_i) = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[W_{ij}^2] \mathbb{E}[x_j^2] = \boxed{n_{\text{in}} \sigma^2 \mathbb{E}[x^2]}$$

若激活函数在 0 附近近似线性且对称, $\mathbb{E}[x^2] \approx \text{Var}(z^{\text{上一层}})$, 于是

$$\text{Var}(z^{(l)}) = n_{\text{in}} \sigma^2 \cdot \text{Var}(z^{(l-1)}) \implies \text{保持方差需要 } \sigma^2 = \frac{1}{n_{\text{in}}}$$

(LeCun 初始化, 1998)

ReLU 网络: He Kaiming 初始化 (2015)

ReLU 把一半的输入置零, 上面的 “ $\mathbb{E}[x^2] \approx \text{Var}(z)$ ” 不再成立。

设 $x = \max(0, z')$, z' 关于 0 对称:

$$\mathbb{E}[x^2] = \mathbb{E}[z'^2 : \{z' > 0\}] = \frac{1}{2} \mathbb{E}[z'^2] = \frac{1}{2} \text{Var}(z')$$

代入前向方差公式:

$$\text{Var}(z^{(l)}) = n_{\text{in}} \sigma^2 \cdot \frac{1}{2} \text{Var}(z^{(l-1)}) \quad \Longrightarrow \quad \boxed{\sigma^2 = \frac{2}{n_{\text{in}}}}$$

He Kaiming 初始化

- 正态: $W \sim \mathcal{N}(0, 2/n_{\text{in}})$; 均匀: $a = \sqrt{6/n_{\text{in}}}$
- 使 30 层的 ReLU 网络可以从头训练达到收敛

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化
- 5 学习率调度**
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

为什么学习率要衰减？

一维二次损失 $\mathcal{L} = \frac{\lambda}{2} w^2$ ，SGD 的梯度带噪声： $g_t = \lambda w_t + \xi_t$ ， $\mathbb{E}[\xi_t] = 0$ ， $\text{Var}(\xi_t) = \sigma^2$

$$w_{t+1} = (1 - \eta\lambda) w_t - \eta \xi_t$$

稳态时 $\mathbb{E}[w_{t+1}^2] = \mathbb{E}[w_t^2] = s$:

$$s = (1 - \eta\lambda)^2 s + \eta^2 \sigma^2 \quad \Rightarrow \quad s = \frac{\eta \sigma^2}{\lambda(2 - \eta\lambda)} \approx \frac{\eta \sigma^2}{2\lambda}$$

- 固定学习率下，SGD 不会收敛到极小点，而是在其周围一个“噪声球”中游走
- 噪声球的大小 $\propto \eta$

两个阶段的矛盾

前期： η 大，快速接近极小点所在区域

后期： η 小，缩小噪声球、降低 loss

经典学习率调度

记峰值学习率 $\eta_{\max}$ ，总步数 T :

名称	公式	典型用途
Step decay	$\eta_t = \eta_{\max} \cdot \gamma^{\lfloor t/T_s \rfloor}$	ResNet (每 30 epoch $\times 0.1$)
指数衰减	$\eta_t = \eta_{\max} \cdot \gamma^t$	
Cosine	$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos \frac{\pi t}{T})$	ViT、LLM
线性衰减	$\eta_t = \eta_{\max}(1 - t/T)$	BERT、LLM
逆平方根	$\eta_t \propto 1/\sqrt{t}$	原始 Transformer

- Cosine (Loshchilov & Hutter, SGDR 2017): 前期下降慢、中间快、末尾又慢，几乎没有需要调的额外超参数
- LLM 常见做法: warmup + cosine 衰减到峰值的 10%
- 以上调度都需要事先知道总步数 T

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化
- 5 学习率调度
- 6 Warmup 为什么有效**
- 7 Sharpness-Aware Minimization (SAM)

Warmup: 希望用大的 η_{trgt} , 但为什么一开始要用小学习率

什么是 **warmup**: 训练最开始的 T_w 步, 学习率从很小逐渐增大到目标值 η_{trgt} , 之后再按正常的调度 (如 cosine) 衰减。最常用的是线性 warmup:

$$\eta_t = \begin{cases} \eta_{\text{trgt}} \cdot \frac{t}{T_w}, & t \leq T_w \text{ (warmup: 线性增大)} \\ \text{正常的调度 (如 cosine 衰减)}, & t > T_w \end{cases}$$

- T_w 通常取总步数的 1%–5% (LLM 常用几百到几千步)
- 起源: 大 batch 训练 ImageNet (Goyal et al., 2017), 之后成为 Transformer / LLM 训练的标配

为什么有效: 由稳定条件 $\eta < 2/S(\mathbf{w})$, 初始化点往往很尖锐 (S 大), 直接用 η_{trgt} 会发散; 先用小学习率, 等参数走到平坦区域 (S 变小) 再逐渐加大。

一句话 (Kalra & Barkeshli, NeurIPS 2024)

Warmup 把参数从尖锐的初始区域, 平稳地带到能容忍大学习率的平坦区域。

Warmup 是在找更平坦的点吗？

是，但要分清直接作用和最终效果：

- ① 直接作用：在训练早期降低 sharpness，让训练不发散
- ② 真正的收益：能用更大的目标学习率 η_{trgt} ，而且对 η_{trgt} 的选择更稳健（不容易调坏）
- ③ 最终停在多平坦的地方，主要由之后的学习率决定：稳定条件要求 $S < 2/\eta$
 $\Rightarrow$ 大 η 只能停在平坦的区域；warmup 让大 η 成为可能，间接得到更平坦的解
warmup $\Rightarrow$ 早期 sharpness $\downarrow \Rightarrow$ 可以用更大的 $\eta \Rightarrow$ 停在更平坦的区域

实践：

- 线性 warmup，长度约为总步数的 1%–5%（LLM 常用几百到几千步）
- 对 Adam 还有一个原因：初期 $\mathbf{v}$ 只由很少几个梯度估计，不准确，warmup 起缓冲作用

目录

- 1 梯度下降与稳定性
- 2 优化器：从 SGD 到 AdamW
- 3 离散变量的梯度
- 4 参数初始化
- 5 学习率调度
- 6 Warmup 为什么有效
- 7 Sharpness-Aware Minimization (SAM)

动机：平坦的极小点泛化更好？

- Hochreiter & Schmidhuber (1997)：平坦的极小点对应更“简单”的模型
- Keskar et al. (2017)：大 batch 训练收敛到更尖锐的极小点，泛化更差
- 稳定性条件：大学习率 $\Rightarrow$ 只能停在 $S < 2/\eta$ 的平坦区域

直觉：训练集与测试集的损失曲面有小的偏移

- 尖锐极小点：偏移一点，loss 就大幅上升
- 平坦极小点：对偏移不敏感

问题

大学习率和 warmup 只是隐式地偏好平坦区域。能否显式地优化平坦度？

SAM 的目标与一阶近似 (Foret et al., ICLR 2021)

SAM 目标：最小化邻域内的最坏 loss

$$\min_{\mathbf{w}} \mathcal{L}^{\text{SAM}}(\mathbf{w}), \quad \mathcal{L}^{\text{SAM}}(\mathbf{w}) = \max_{\|\epsilon\|_2 \leq \rho} \mathcal{L}(\mathbf{w} + \epsilon)$$

内层 max 的一阶近似：

$$\mathcal{L}(\mathbf{w} + \epsilon) \approx \mathcal{L}(\mathbf{w}) + \epsilon^\top \nabla \mathcal{L}(\mathbf{w}) \implies \hat{\epsilon}(\mathbf{w}) = \rho \frac{\nabla \mathcal{L}(\mathbf{w})}{\|\nabla \mathcal{L}(\mathbf{w})\|}$$

(在球内沿梯度方向走到边界)

外层梯度：代入 $\hat{\epsilon}$ ，并忽略 $\hat{\epsilon}$ 对 $\mathbf{w}$ 的导数（避免二阶项）：

$$\nabla \mathcal{L}^{\text{SAM}}(\mathbf{w}) \approx \nabla \mathcal{L}(\mathbf{w}) \big|_{\mathbf{w} + \hat{\epsilon}(\mathbf{w})}$$

即：先往上坡走一步到最坏点，再用那里的梯度更新原来的参数。

每步两次前向-反向

- ① 在 $\mathbf{w}_t$ 处计算梯度 $\mathbf{g}_t = \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w}_t)$
- ② 计算扰动 $\hat{\epsilon}_t = \rho \mathbf{g}_t / \|\mathbf{g}_t\|$
- ③ 在 $\mathbf{w}_t + \hat{\epsilon}_t$ 处计算梯度 $\mathbf{g}_t^{\text{SAM}} = \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w}_t + \hat{\epsilon}_t)$
- ④ 用任意基础优化器更新: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t^{\text{SAM}}$ (SGD / Adam 均可)

- 超参数 ρ : 邻域半径, 常用 0.05 (ResNet) $\sim$ 0.5
- 计算代价: 约为基础优化器的 2 倍
- 在 CIFAR、ImageNet 以及 ViT 从头训练上显著提升泛化
- Chen et al. (ICLR 2022): 用 SAM 训练, ViT 不需要大规模预训练也能超过 ResNet

SAM 在惩罚什么？

一阶：代入 $\hat{\epsilon}$,

$$\mathcal{L}^{\text{SAM}}(\mathbf{w}) \approx \mathcal{L}(\mathbf{w}) + \rho \|\nabla \mathcal{L}(\mathbf{w})\| \quad \Rightarrow \quad \text{惩罚梯度范数}$$

二阶（在极小点附近， $\nabla \mathcal{L} = 0$ ）：

$$\mathcal{L}^{\text{SAM}}(\mathbf{w}) \approx \mathcal{L}(\mathbf{w}) + \max_{\|\epsilon\| \leq \rho} \frac{1}{2} \epsilon^\top \mathbf{H} \epsilon = \mathcal{L}(\mathbf{w}) + \frac{\rho^2}{2} \lambda_{\max}(\mathbf{H}) \quad \Rightarrow \quad \text{惩罚 sharpness}$$

方法	对 sharpness 的作用
大学习率 GD	隐式： $S < 2/\eta$
Warmup	训练前期逐步把 S 降下来
SAM	显式：目标中加入 $\frac{\rho^2}{2} \lambda_{\max}(\mathbf{H})$

SAM 的实现

```
# base_opt: any optimizer, e.g. SGD(momentum=0.9)
loss_fn(model(x), y).backward()           # 1st pass:  $g_t$ 

with torch.no_grad():
    grads = [p.grad for p in model.parameters()]
    norm = torch.norm(torch.stack([g.norm() for g in grads]))
    eps = [rho * g / (norm + 1e-12) for g in grads]
    for p, e in zip(model.parameters(), eps):
        p.add_(e)                          #  $w \leftarrow w + eps$ 
model.zero_grad()

loss_fn(model(x), y).backward()           # 2nd pass:  $g_{SAM}$ 

with torch.no_grad():
    for p, e in zip(model.parameters(), eps):
        p.sub_(e)                          # back to  $w$ 
base_opt.step()                           # update with  $g_{SAM}$ 
base_opt.zero_grad()
```

第三讲知识地图

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{U}_t, \quad \mathbf{w}_0 \sim \text{初始化}$$

方向 $\mathbf{U}_t$

- SGD: $\mathbf{g}_t$
- Momentum: $\sum \beta^{t-s} \mathbf{g}_s$
 $O(\kappa) \rightarrow O(\sqrt{\kappa})$
- Adam: $\hat{\mathbf{m}}/\sqrt{\hat{\mathbf{v}}}$
每个方向自己的学习率
- 离散变量: STE、
ST Gumbel-Softmax、
REINFORCE

步长 η_t

- 稳定性: $\eta < 2/\lambda_{\max}$
- 衰减: 噪声球 $\propto \eta$
- step / cosine
- Warmup: 先小后大,
避开尖锐的初始区域

起点 $\mathbf{w}_0$

- 随机初始化打破对称性
- 保持各层方差不变
- LeCun: $\sigma^2 = \frac{1}{n_{\text{in}}}$
- He (ReLU): $\sigma^2 = \frac{2}{n_{\text{in}}}$

Sharpness 贯穿全讲: 二次模型的稳定性 $\rightarrow$ Warmup $\rightarrow$ SAM

参考文献（按出现顺序） I

优化基础与优化器

- Goodfellow, Bengio, Courville (2016). *Deep Learning*. MIT Press. (§4.3: 泰勒展开与步长)
- Kingma & Ba (2015). Adam: A Method for Stochastic Optimization. *ICLR*.

离散变量的梯度

- Kingma & Welling (2014). Auto-Encoding Variational Bayes. *ICLR*.
- van den Oord, Vinyals, Kavukcuoglu (2017). Neural Discrete Representation Learning. *NeurIPS*.
- Bengio, Léonard, Courville (2013). Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432.
- Courbariaux, Bengio, David (2015). BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations. *NeurIPS*.
- Jang, Gu, Poole (2017). Categorical Reparameterization with Gumbel-Softmax. *ICLR*.
- Baevski et al. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *NeurIPS*.
- Ramesh et al. (2021). Zero-Shot Text-to-Image Generation. (DALL·E) *ICML*.
- Xie et al. (2019). SNAS: Stochastic Neural Architecture Search. *ICLR*.
- Dong & Yang (2019). Searching for a Robust Neural Architecture in Four GPU Hours. (GDAS) *CVPR*.
- Wu et al. (2019). FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. *CVPR*.

参考文献（按出现顺序） II

- Rao et al. (2021). DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *NeurIPS*.
- Williams (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*.
- Shao et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. (GRPO) arXiv:2402.03300.
- DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

参数初始化

- LeCun, Bottou, Orr, Müller (1998). Efficient BackProp. In *Neural Networks: Tricks of the Trade*.
- Glorot & Bengio (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. *AISTATS*.
- He et al. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *ICCV*.

学习率调度与 Warmup

- Loshchilov & Hutter (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *ICLR*.
- Goyal et al. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:1706.02677.
- Kalra & Barkeshli (2024). Why Warmup the Learning Rate? Underlying Mechanisms and Improvements. *NeurIPS*.

SAM 与平坦极小点

- Hochreiter & Schmidhuber (1997). Flat Minima. *Neural Computation*.
- Keskar et al. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *ICLR*.
- Foret et al. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization. *ICLR*.
- Chen, Hsieh, Gong (2022). When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations. *ICLR*.
- Dinh et al. (2017). Sharp Minima Can Generalize For Deep Nets. *ICML*.
- Kwon et al. (2021). ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. *ICML*.
- Andriushchenko et al. (2023). A Modern Look at the Relationship between Sharpness and Generalization. *ICML*.

课后思考题

- ① 经典的 L -光滑条件和“局部 sharpness”有什么区别？为什么深度学习更关心后者？
- ② Adam 比 Momentum 多做了什么？为什么要除以梯度平方平均的平方根 $\sqrt{\mathbb{E}[g_j^2]}$ ？
- ③ 为什么离散操作会让反向传播失效？STE 是怎么“绕过去”的？它的代价是什么？
- ④ 为什么平坦的极小点被认为泛化更好？SAM 是怎么显式追求平坦的？这个观点有什么局限？