优化与正则化技巧实战

2025 年 10 月 22 日

目录

1 自适应优化器：Adam、RMSprop 及其家族

自适应优化器通过为每个参数维护独立的统计量来自适应地调整学习率，使得模型能够在病态曲面或噪声目标函数上更快收敛。常见方法都会累计梯度的一阶与二阶矩信息。

1.1 RMSprop

RMSprop 维护梯度平方的指数滑动平均：

$$\mathbf{v}_t = \rho \mathbf{v}_{t-1} + (1-\rho)\, \nabla_{\boldsymbol{\theta}} \mathcal{L}_t \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}_t, \tag{1}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\nabla_{\boldsymbol{\theta}} \mathcal{L}_t}{\sqrt{\mathbf{v}_t + \epsilon}}, \tag{2}$$

其中 $\rho \approx 0.9$，$\epsilon \approx 10^{-8}$ 防止除零。平方和的累积抑制了在陡峭方向上的步长，有效缓解了学习率选择问题。

1.2 Adam 与 AdamW

Adam 在 RMSprop 的基础上引入动量，计算梯度的一阶与二阶矩：

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\, \nabla_{\boldsymbol{\theta}} \mathcal{L}_t, \tag{3}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\, \nabla_{\boldsymbol{\theta}} \mathcal{L}_t \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}_t. \tag{4}$$

偏置校正项消除了初始为零的影响：

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \qquad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}. \tag{5}$$

参数更新因此写为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}. \tag{6}$$

AdamW 将权重衰减与自适应梯度解耦：

$$\boldsymbol{\theta}_{t+1} = (1 - \eta\lambda)\boldsymbol{\theta}_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \tag{7}$$

其中 λ 为权重衰减系数，能够更稳定地控制参数范数并提升泛化。

1.3 更多改进优化器

- **AdaBelief** 使用梯度与一阶矩之间的偏差来估计方差，在平稳区域具有更小的噪声。
- **AdaFactor** 将二阶矩分解为行、列向量，大幅降低大型模型（如 Transformer）的显存消耗。
- **Yogi** 通过受控的加减更新避免二阶矩无限增大，在稀疏梯度场中更稳定。

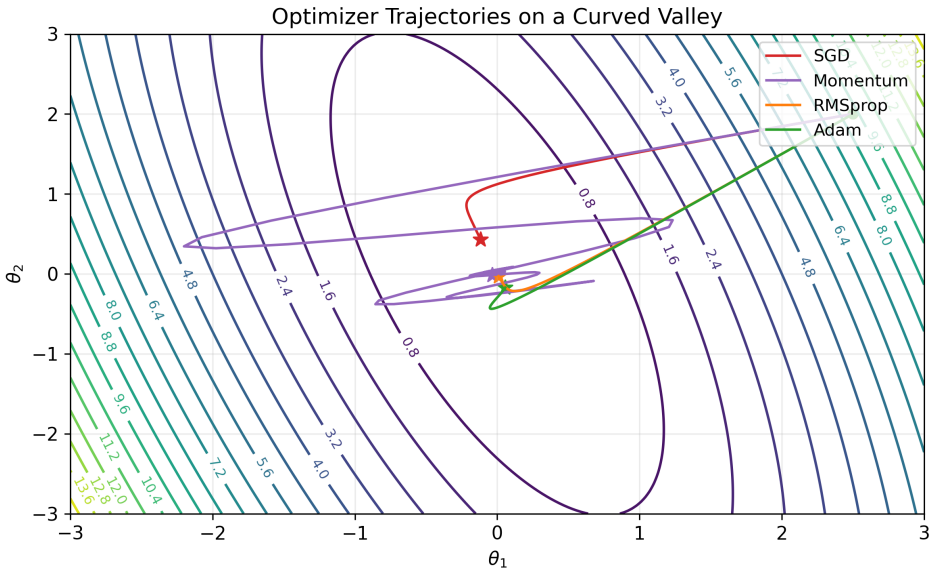

图 1: SGD、动量、RMSprop 与 Adam 在狭长谷底上的优化轨迹对比。自适应方法能够自适应缩放方向梯度。

1.4 实现要点

大规模训练常组合梯度裁剪、混合精度以及解耦权重衰减。如下 PyTorch 代码展示了带有梯度裁剪和余弦退火的 AdamW：

Listing 1: 结合梯度裁剪与余弦调度的 AdamW 训练循环。

```python
import torch
from torch.nn.utils import clip_grad_norm_
from torch.optim.lr_scheduler import CosineAnnealingLR

optimizer = torch.optim.AdamW(model.parameters(), lr=3e-4,
                              betas=(0.9, 0.999), eps=1e-8,
                                 weight_decay=0.01)
scheduler = CosineAnnealingLR(optimizer, T_max=1000, eta_min=1e-5)

for step, batch in enumerate(dataloader, start=1):
    loss = compute_loss(model, batch)
    loss.backward()
    clip_grad_norm_(model.parameters(), max_norm=1.0)
    optimizer.step()
    scheduler.step()
    optimizer.zero_grad()
```

2 Batch Normalization 与 Layer Normalization

归一化层通过稳定激活分布的均值与方差减少层间协变量偏移，显著提升收敛速度并带来一定的正则化效果。

2.1 Batch Normalization

对于一个 mini-batch $\mathcal{B} = \{\mathbf{h}_i\}_{i=1}^{m}$，BatchNorm 在特征维度上标准化：

$$\boldsymbol{\mu}_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{h}_i, \qquad \boldsymbol{\sigma}_{\mathcal{B}}^{2} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{h}_i - \boldsymbol{\mu}_{\mathcal{B}})^{\odot 2}, \tag{8}$$

$$\hat{\mathbf{h}}_i = \frac{\mathbf{h}_i - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\boldsymbol{\sigma}_{\mathcal{B}}^{2} + \epsilon}}, \qquad \mathbf{y}_i = \boldsymbol{\gamma} \odot \hat{\mathbf{h}}_i + \boldsymbol{\beta}. \tag{9}$$

训练阶段使用批统计量，推理阶段则使用滑动平均。由于 BN 会引入批噪声，它在卷积网络中具有良好的正则化效果，但在小批量或序列任务中需要特别处理。

2.2 Layer Normalization

LayerNorm 在单个样本的特征维度上归一化：

$$\mu = \frac{1}{d}\sum_{j=1}^{d} h_j, \quad \sigma^{2} = \frac{1}{d}\sum_{j=1}^{d}(h_j - \mu)^{2}, \quad \hat{h}_j = \frac{h_j - \mu}{\sqrt{\sigma^{2} + \epsilon}}. \tag{10}$$

随后通过可学习参数 (γ, β) 恢复尺度与偏移。LN 不依赖 batch 规模，在 Transformer、语言模型等场景中表现稳定。

2.3 差异与组合实践

- **对批大小敏感度：** BN 对 batch 规模敏感；LN 对任意 batch 均适用。
- **正则化强度：** BN 的噪声具备隐式正则化；LN 需搭配显式正则化（如 dropout）。
- **跨设备同步：**分布式训练中 BN 需同步统计量，LN 则无需通信开销。

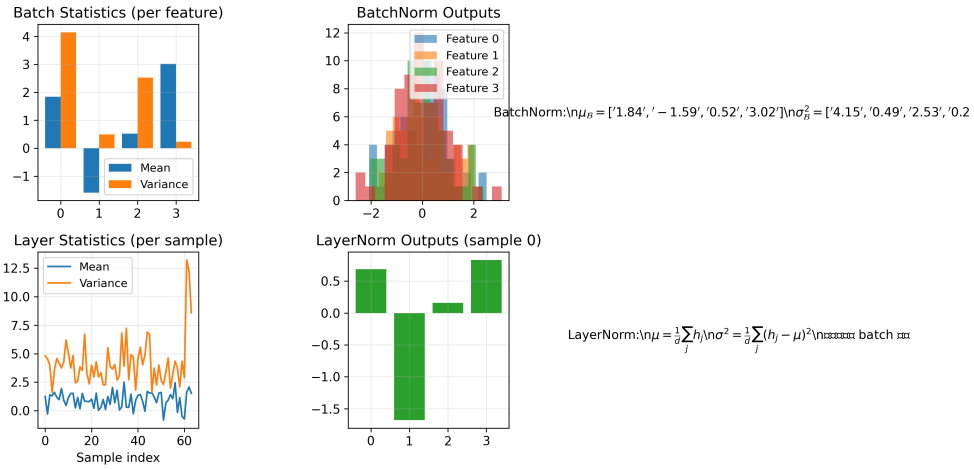

图 2: BatchNorm 与 LayerNorm 归一化前后特征分布的对比。LN 在每个样本内部进行标准化。

3 学习率调度与 Warm-up

合理的学习率策略能够在训练早期快速下降、后期稳定收敛。Warm-up 则避免自适应优化器在统计量尚未稳定时产生过大的步长。

3.1 阶梯、指数与多项式衰减

阶梯衰减每隔 k 个 epoch 将学习率乘以 $\gamma < 1$：

$$\eta_t = \eta_0 \gamma^{\lfloor \frac{t}{k} \rfloor}. \tag{11}$$

指数衰减连续地减小学习率：

$$\eta_t = \eta_0 \exp(-\lambda t). \tag{12}$$

多项式衰减在 T 步内平滑地从 η_0 过渡到 η_{end}：

$$\eta_t = \eta_{\text{end}} + (\eta_0 - \eta_{\text{end}}) \left(1 - \frac{t}{T}\right)^p. \tag{13}$$

3.2 余弦退火与循环策略

余弦退火将学习率平滑衰减到 $\eta_{\min}$：

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_0 - \eta_{\min})\left(1 + \cos\frac{\pi t}{T}\right). \tag{14}$$

循环学习率（CLR）在一个周期内在上下界之间往返，有助于跳出局部极小值。例如三角策略通过线性上升至 $\eta_{\max}$ 再下降到 $\eta_{\min}$。

3.3 Warm-up

Warm-up 将学习率在起始 T_w 步内线性升高：

$$\eta_t = \begin{cases} \eta_{\text{target}}\frac{t}{T_w}, & 0 \le t \le T_w, \\ \text{Schedule}(t - T_w), & t > T_w. \end{cases} \tag{15}$$

Transformer 常采用线性 Warm-up 后接 $\eta_t \propto t^{-1/2}$ 的逆平方衰减。该策略能避免一阶与二阶动量尚未稳定时出现梯度爆炸。

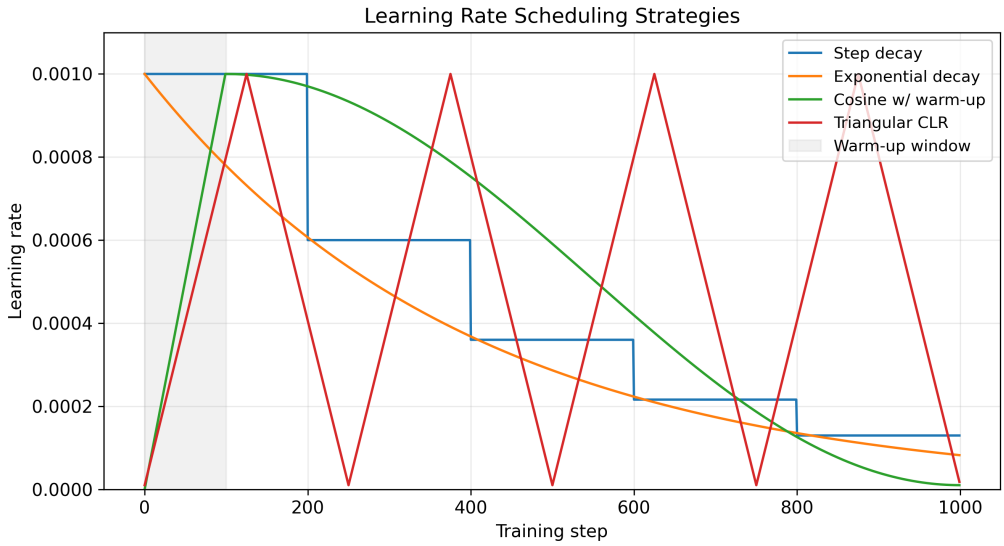

图 3: 阶梯、余弦、循环与 Warm-up 组合学习率策略示意。

4 数据增强与迁移学习

数据增强扩展了有效训练分布，迁移学习则复用预训练特征，在小样本场景中显著提升性能。

4.1 经典与高级增强方法

- **几何增强**：随机裁剪、翻转、旋转、形变等。

- **光照增强**：颜色抖动、伽马校正、Cutout、随机遮挡。
- **混合增强**： Mixup 构造 $\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$、$\tilde{\mathbf{y}} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$。
- **分布扰动**： RandAugment 随机挑选若干操作级联，AugMix 则将多条增强路径加权融合并引入一致性正则。

4.2 迁移学习流程

迁移学习从来源数据集 $\mathcal{D}_{\text{src}}$ 的预训练权重 $\boldsymbol{\theta}_{\text{pre}}$ 出发，在目标数据集 $\mathcal{D}_{\text{tgt}}$ 上微调：

$$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{\text{pre}}, \tag{16}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} \big[\mathcal{L}_{\text{tgt}}(\boldsymbol{\theta}_t) + \lambda \mathcal{R}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{\text{pre}}) \big], \tag{17}$$

其中 $\mathcal{R}$ 可设计为 L2-SP、Fisher 正则等，以限制过度偏离预训练表示。层级自适应学习率（LARS、LAMB）则常用于大模型微调。

4.3 自监督预训练

对比学习与掩码预测任务提供了无监督的可迁移特征。SimCLR 的 NT-Xent 损失定义为

$$\mathcal{L}_{\text{NT-Xent}} = -\sum_{i} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_i'/\tau)}{\sum_j \mathbf{1}_{[j \neq i]} \exp(\mathbf{z}_i \cdot \mathbf{z}_j/\tau)}, \tag{18}$$

通过最大化不同增强视角的相似度来学习表征。后续只需少量标注即可完成下游微调。

4.4 综合流程示意

图 ?? 总结了一条常见的管线：数据增强生成多样样本，特征骨干网络进行预训练，再通过任务头进行微调或蒸馏。

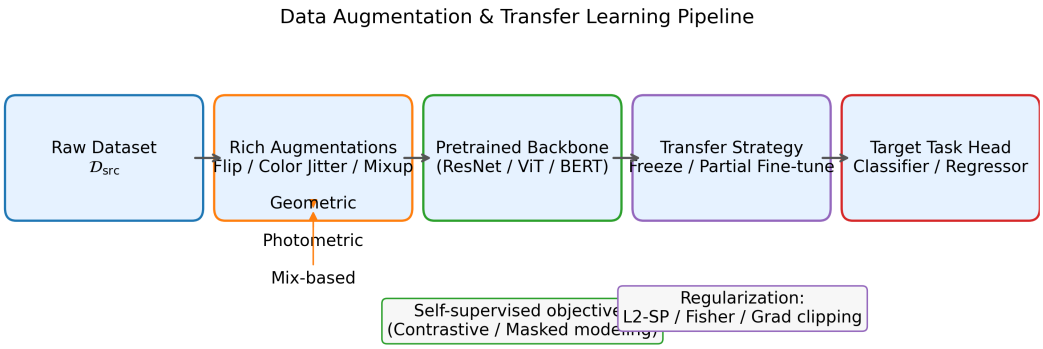

图 4: 数据增强与迁移学习协同的工作流。可选择冻结主干或进行分层微调。

延伸阅读

- Ilya Loshchilov & Frank Hutter：《Decoupled Weight Decay Regularization》，ICLR 2019。
- Sergey Ioffe & Christian Szegedy：《Batch Normalization》，ICML 2015。
- Leslie N. Smith：《Cyclical Learning Rates for Training Neural Networks》，WACV 2017。
- Tong He 等：《Bag of Tricks for Image Classification with Convolutional Neural Networks》，CVPR 2019。