

深度学习中学习率策略的研究（实践篇）

朱梦

初稿于 2025-08-15，修改于 2025-08-15

在博客《深度学习中学习率策略的研究（理论篇）》中，分析了四种不同的学习率策略，本博客通过不同深度学习框架进行实现。

1. 基于 PyTorch 实现 batch 细粒度的学习率策略

PyTorch 官方推荐将优化器和学习率调度器解耦合，有助于代码维护，各自职责更清晰，且推荐先优化器更新（optimizer.step()）后学习率调度器更新（scheduler.step()）。另外，PyTorch 官方提供的学习率调度器是 epoch 细粒度的。综上，基于 PyTorch 实现 batch 细粒度的学习率策略，需注意如下三点：

- （1）初始学习率的设置、最大学习率的设置。
- （2）epoch 细粒度的学习率调度器如何修改为 batch 细粒度的学习率调度器。
- （3）学习率调度器自动维护的计数器是从零计数吗？

```
1 from torch.optim.lr_scheduler import LRScheduler
2
3 class BatchScheduler(LRScheduler):
4     """batch细粒度学习率调度器"""
5
6     def __init__(self, optimizer, lr_lambda, last_epoch=-1):
7         self.lr_lambda = lr_lambda
8         super().__init__(optimizer, last_epoch)
9
10    def get_lr(self):
11        step = self.last_epoch
12        return [self.lr_lambda(step)] * len(self.base_lrs)
13
14    def step(self, epoch=None):
15        # 更新计数器
16        if epoch is None:
17            self.last_epoch += 1
18        else:
19            self.last_epoch = epoch
20
21        # 计算并应用新学习率
22        values = self.get_lr()
23        self._update_lr(values)
```

```
24
25     def _update_lr(self, lrs):
26         for param_group, lr in zip(self.optimizer.param_groups, lrs):
27             param_group['lr'] = lr
28             self._last_lr = lrs
```

学习率策略模板示例代码:

```
1 class MyLRLambda:
2     def __init__(self, lr: float, t1: int, r_min: float = 0.01) -> None:
3         self.lr = lr
4         self.t1 = t1
5         self.r_min = r_min
6
7     def __call__(self, batch_idx: int) -> float:
8         if batch_idx > self.t1:
9             r_t = self.r_min
10        else:
11            r_t = 1 + (self.r_min - 1) / self.t1 * batch_idx
12
13        return self.lr * r_t
14
15 if __name__ == '__main__':
16     lr_lambda = MyLRLambda(0.1, 100)
17     init_lr = lr_lambda(0)
18     print(init_lr)
```

2. 基于 JAX 实现 batch 细粒度的学习率策略