最优学习率搜索（二）：基于贝叶斯优化的最优学习率搜索

朱梦

初稿于 2025-08-15，修改于 2025-08-15

1. 问题定义及其分析

在博客《最优学习率搜索（一）：基于学习率扫描的最优学习率搜索》中，已阐述基于学习率扫描的最优学习率搜索伪代码。书接上文，这篇博客阐述基于贝叶斯优化的最优学习率搜索。

2. 问题解决

```python
class BayesianOptSearchingLR:
    def __init__(self,
        model: nn.Module,
        OptType: Type[torch.optim.Optimizer],
        train_dataloader: DataLoader,
        val_dataloader: DataLoader,
        Criterion: Type[_Loss] = nn.CrossEntropyLoss,
        start_lr: float = 1e-8,
        end_lr: float = 0.1,
        warmup_epochs: int = 2,
        training_epochs: int = 5,
    ) -> None:
        self.model = model
        self.OptType = OptType
        self.train_dataloader = train_dataloader
        self.val_dataloader = val_dataloader
        self.criterion = Criterion()
        self.start_lr = start_lr
        self.end_lr = end_lr
        self.warmup_epochs = warmup_epochs
        self.training_epochs = training_epochs

        self.init_wts = deepcopy(model.state_dict())
        self.device = next(model.parameters()).device

        self.rsts = {}

    def objective(self, trial) -> float:
```

```
        self.model.load_state_dict(self.init_wts)  # 初始化模型，确保权重重置
        lr = trial.suggest_float('lr', self.start_lr, self.end_lr, log=True)

        warmup_batches = self.warmup_epochs * len(self.train_dataloader)
        batch_cnt = 1
        if warmup_batches != 0:
            optimizer = self.OptType(self.model.parameters(), lr=lr * batch_cnt /
                warmup_batches)   # 学习率预热
        else:
            optimizer = self.OptType(self.model.parameters(), lr=lr)


        self.model.train()
        for epoch in range(self.training_epochs):
            for inputs, targets in self.train_dataloader:
                inputs, targets = inputs.to(self.device), targets.to(self.device)
                optimizer.zero_grad()
                outputs = self.model(inputs)
                loss = self.criterion(outputs, targets)
                loss.backward()
                optimizer.step()

                batch_cnt += 1
                if batch_cnt < warmup_batches:
                    for param_group in optimizer.param_groups:
                        param_group['lr'] = lr * batch_cnt / warmup_batches

            # 报告当前epoch的指标，然后检查是否应早停
            val_acc = self.evaluate()
            trial.report(val_acc, epoch)
            if trial.should_prune():
                raise optuna.TrialPruned()  # 终止本次 Trial

        # 记录结果
        val_acc = self.evaluate()
        self.rsts[lr] = val_acc

        return val_acc


    def evaluate(self) -> float:
        self.model.eval()

        correct, total = 0, 0
        with torch.no_grad():
            for inputs, targets in self.val_dataloader:
                inputs, targets = inputs.to(self.device), targets.to(self.device)
                outputs = self.model(inputs)
                _, preds = torch.max(outputs, 1)
                correct += (preds == targets).sum().item()
```

```
                total += targets.size(0)

        acc = correct / total
        return acc
```

```
if __name__ == "__main__":
    '''部分调用代码
    '''
    start_lr, end_lr = 1e-8, 0.1
    opt = BayesianOptSearchingLR(model, AdamX, train_dataloader, val_dataloader,
        start_lr=start_lr, end_lr=end_lr, training_epochs=5)
    study = optuna.create_study(direction="maximize")
    study.optimize(opt.objective, n_trials=10)  # 10次试验

    results = opt.rsts
    best_trial = study.best_trial
```

3. 结论及其反思