深度强化学习：值函数、策略梯度与 AlphaGo

2025 年 10 月 22 日

目录

1 DQN、策略梯度、Actor-Critic

深度强化学习将神经网络与强化学习目标结合，实现对复杂序列决策问题的端到端优化。图 **??** 对比了值函数方法、策略梯度方法与 Actor-Critic 框架的核心流程。

1.1 深度 Q 网络（DQN）

DQN 近似离散动作空间下的最优动作价值函数 $Q^\star(s,a)$。贝尔曼最优方程为

$$Q^\star(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a) + \gamma \max_{a'} Q^\star(s',a')\right]. \tag{1}$$

DQN 使用神经网络 $Q_\theta(s,a)$ 拟合该函数，最小化时序差分（TD）损失：

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[(y - Q_\theta(s,a))^2\right], \quad y = r + \gamma \max_{a'} Q_{\theta^-}(s',a'), \tag{2}$$

其中 θ^- 为周期性复制的目标网络参数，经验回放缓冲 $\mathcal{D}$ 用于打破样本相关性并提升数据效率。

稳定训练技巧

- **Double DQN**：使用在线网络的 $\arg\max$ 选择动作，降低目标值的高估偏差。
- **Dueling 网络**：将 $Q(s,a)$ 分解为状态价值 $V(s)$ 与优势函数 $A(s,a)$，提高不同动作值差异的表达力。
- **优先经验回放**：根据 TD 误差大小调整采样概率，聚焦于训练困难样本。

Listing 1: 带目标网络与经验回放的 DQN 训练循环示例。

伪代码

```
replay = ReplayBuffer(capacity=100_000)
q_net = QNetwork().to(device)
target_net = copy.deepcopy(q_net)
optimizer = torch.optim.Adam(q_net.parameters(), lr=1e-3)

for step in range(total_steps):
    action = epsilon_greedy(q_net, obs, epsilon_schedule(step))
    next_obs, reward, done, info = env.step(action)
    replay.add(obs, action, reward, next_obs, done)
    obs = next_obs if not done else env.reset()

    if step > warmup and step % train_freq == 0:
        batch = replay.sample(batch_size=64)
        target = batch.reward + gamma * target_net(batch.next_obs).max(
            dim=1).values * (1 - batch.done)
        q_values = q_net(batch.obs).gather(1, batch.action.unsqueeze(1)
            ).squeeze(1)
        loss = F.mse_loss(q_values, target.detach())
        optimizer.zero_grad()
        loss.backward()
        clip_grad_norm_(q_net.parameters(), max_norm=10.0)
        optimizer.step()

    if step % target_update == 0:
        target_net.load_state_dict(q_net.state_dict())
```

1.2 策略梯度方法

策略梯度直接最大化期望回报 $J(\theta)=\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$。REINFORCE 梯度估计为

$$\nabla_\theta J(\theta)=\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\,G_t\right],\quad G_t=\sum_{k=t}^{T}\gamma^{k-t}r_k. \tag{3}$$

为了降低方差，引入基线 $b(s_t)$ 后：

$$\nabla_\theta J(\theta)=\mathbb{E}_{\pi_\theta}\left[\sum_{t}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\,\big(G_t-b(s_t)\big)\right]. \tag{4}$$

常见的基线包括学习得到的状态价值函数 $V_\phi(s)$，以及优势函数 $A_t=G_t-V_\phi(s_t)$。

信赖域与 PPO TRPO 通过约束新旧策略间的 KL 散度控制更新幅度。PPO 用易实现的截断代理目标替代：

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right], \tag{5}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}. \tag{6}$$

广义优势估计（GAE）通过 λ-return 平滑优势，兼顾偏差与方差：

$$A_t^{\mathrm{GAE}} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{7}$$

1.3 Actor-Critic 框架

Actor-Critic 同时学习策略（Actor）与价值（Critic）网络。Critic 估计 $V_\phi(s)$ 或 $Q_\phi(s,a)$ 用于降低方差。A2C/A3C 在同步/异步多线程环境下更新；Soft Actor-Critic (SAC) 在连续控制中引入熵正则：

$$J_\pi = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathbb{E}_{a_t\sim\pi}\left[\alpha\log\pi(a_t \mid s_t) - Q_\phi(s_t,a_t)\right]\right], \tag{8}$$

α 调节探索程度。DDPG、TD3 等确定性策略梯度方法针对连续动作设计。

1.4 对比总结

图 ?? 总结了三类方法特点：

- DQN：离散动作、离策略、依赖经验回放与目标网络。
- 策略梯度：在策略、适用于连续动作，但梯度方差较大。
- Actor-Critic：融合优点，实现稳定更新与连续控制。

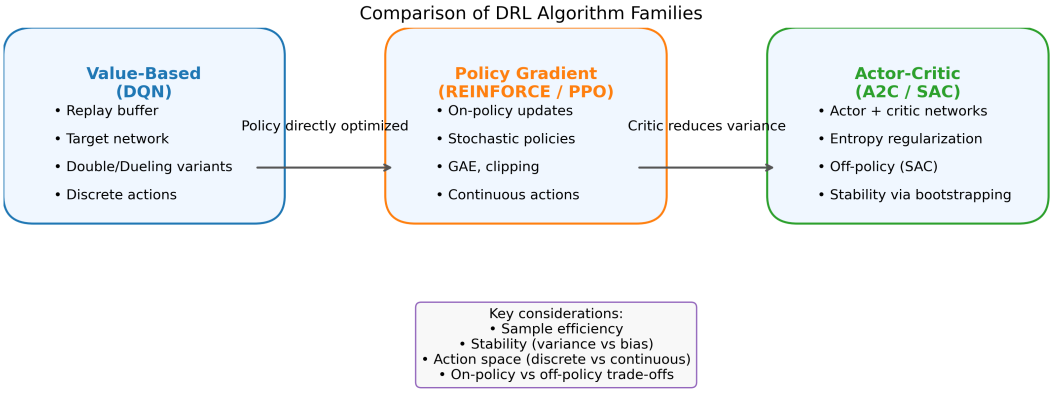

图 1: DQN、策略梯度与 Actor-Critic 流水线对比。值函数方法依靠经验回放；策略梯度直接更新策略；Actor-Critic 融合两者优势。

2 AlphaGo 案例

AlphaGo 首次将深度神经网络与蒙特卡罗树搜索（MCTS）结合，实现围棋超越人类顶尖水平。图 ?? 展示了监督学习、强化学习与树搜索的组合流程。

2.1 策略网络训练

监督策略网络 $p_\theta(a \mid s)$ 在职业棋谱上通过交叉熵训练：

$$\mathcal{L}_{\mathrm{SL}}(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}_{\mathrm{human}}}[\log p_\theta(a \mid s)]. \tag{9}$$

随后基于自对弈进行策略梯度优化，最大化胜率 $\rho(\theta)$：

$$\nabla_\theta \rho(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[(z-b)\sum_{t}\nabla_\theta \log p_\theta(a_t \mid s_t)\right], \tag{10}$$

其中 $z \in \{-1,+1\}$ 为对局结果，b 为基线减少方差。

2.2 价值网络与快速模拟

价值网络 $v_\phi(s)$ 预测当前局面的胜率，使用自对弈样本与蒙特卡罗结果训练：

$$\mathcal{L}_{\mathrm{value}}(\phi) = \mathbb{E}_{s\sim\mathcal{D}_{\text{self-play}}}\left[\left(v_\phi(s)-z\right)^2\right]. \tag{11}$$

搜索过程中还使用轻量快速走子策略进行随机模拟，以提升评估多样性。

2.3 MCTS 融合

AlphaGo 在树搜索中采用改进的上置信界（PUCT）选择：

$$a_t = \arg\max_a \left(Q(s_t,a) + c_{\mathrm{puct}}\, P(s_t,a)\frac{\sqrt{\sum_b N(s_t,b)}}{1+N(s_t,a)} \right), \tag{12}$$

其中 Q 为平均回报，P 为策略网络先验，N 为访问次数。策略网络用于引导扩展节点，价值网络提供评估，大幅压缩搜索空间。

2.4 AlphaGo Zero 与 AlphaZero

AlphaGo Zero 取消监督阶段，通过纯自对弈学习，将策略和值合并为一个残差网络输出 (p,v)。策略目标为 MCTS 访问频率 π，损失函数为

$$\mathcal{L}(\theta) = (z - v_\theta(s))^2 - \pi^\top \log p_\theta(s) + \lambda\|\theta\|^2. \tag{13}$$

AlphaZero 将该框架推广至国际象棋、将棋，证明树搜索 + 自对弈的通用性。

2.5 工程启示

- **硬件集群：** AlphaGo 结合 GPU 评估网络与分布式 CPU 树搜索，高效调度计算资源。
- **训练效率：**自对弈数据进入回放缓冲，确保训练样本多样且可重复使用；批量化树搜索调用提升硬件利用率。
- **评估体系：**使用 Elo 评分对比历史版本，进行模块消融试验，并与顶级棋手对战检验实力。

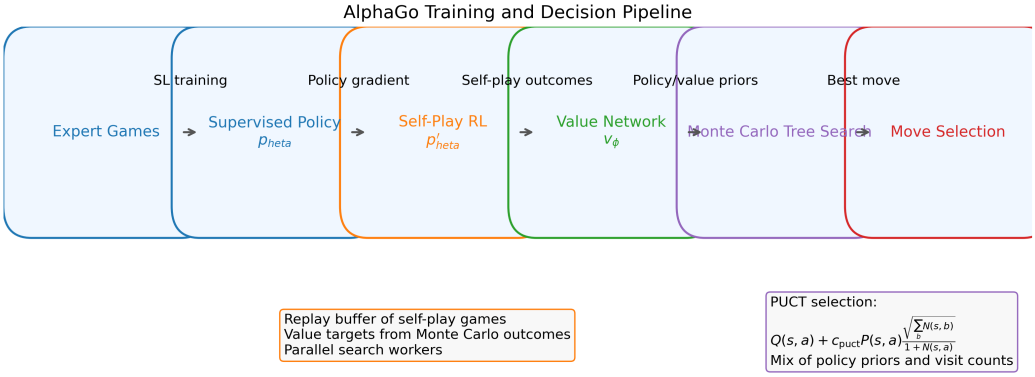

图 2: AlphaGo 训练与推理流程：监督策略初始化、自对弈强化学习、价值网络训练与 MCTS 决策。

延伸阅读

- Volodymyr Mnih 等：《Playing Atari with Deep Reinforcement Learning》，2013。
- John Schulman 等：《Proximal Policy Optimization Algorithms》，2017。
- Tuomas Haarnoja 等：《Soft Actor-Critic Algorithms and Applications》，2018。
- Silver 等：《Mastering the game of Go with deep neural networks and tree search》，Nature 2016。
- Silver 等：《Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm》，2017。