DQN 和 Q 学习

朱大福

2024/03/31

Outline

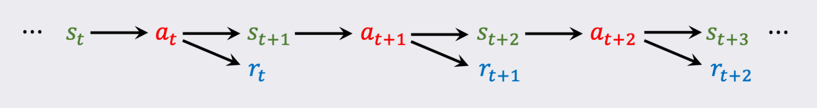
Notation

大写为随机变量，小写为具体观测值

- S_t: 时刻 t 的状态
- A_t: 基于 S_t，在时刻 t 做出的行动
- R_t: 在 S_t 状态下做出 A_t 动作收获的奖励
- U_t: 从时刻 t 开始到回合结束的累积奖励，称为回报
- γ: 折扣因子
- π: 策略函数
- $p_t(s'|s, a)$: 状态转移函数

Outline

基本概念



图：智能体轨迹

随机性有两个来源：动作和状态。动作的随机性来自策略，状态的随机性来自状态转移。

定义 (回报)

假设本回合在时刻 n 结束，则对 $\forall t = 1, 2, \cdots, n$

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} \cdots + \gamma^{t-n} R_n$$

定义 (动作价值函数)

依赖三个因素：s_t, a_t, π

$$Q_\pi(s_t, a_t) = \mathbb{E}_{S_{t+1}, A_{t+1}, \cdots, S_n, A_n}[U_t | S_t = s_t, A_t = a_t]$$

定义 (最优动作价值函数)

$$Q_*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t) \quad \forall s, a$$

有多种策略函数 $\pi \in \Pi$ 可以选择，我们选择最好的策略，记为 π^*。这个策略使得无论在什么状态 s 和动作 a 下都能占优，即

$$Q_*(s_t, a_t) \geqslant Q_\pi(s_t, a_t) \quad \forall \pi \neq \pi^*$$

这里

$$\pi^* = \arg\max \pi \, Q_\pi(s_t, a_t) \quad \forall s, a$$

举个例子，假如有行动空间 $\mathcal{A} = \{a_1, a_2, a_3\}$，且

$$Q_*(s_t, a_1) = 370, Q_*(s_t, a_2) = -21, Q_*(s_t, a_3) = 610$$

这就像放缩...

定义 (状态价值函数)

本质上是动作价值函数的期望

$$V_\pi(s_t) = \mathbb{E}_{A_t \sim \pi(\cdot|s_t)}[Q_\pi(s_t, A_t)]$$

$$V_\pi(s_t) = \mathbb{E}_{A_t, S_{t+1}, A_{t+1}, \cdots, S_n, A_n}[U_t | S_t = s_t]$$

Outline

DQN

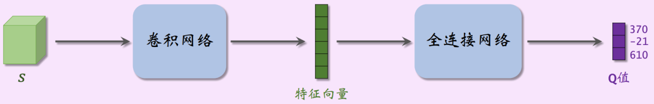


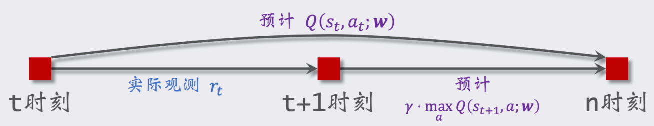
图: DQN 结构

可见 DQN 和普通的神经网络没什么不同，只不过拟合对象变成最优动作价值函数。

$$Q(s, a; w) \to Q_*(s, a)$$

定义 (TD 算法)

训练 DQN 最常见的算法是 TD（temporal difference，时间差分）。TD 算法有很多种，常用的是 Q 学习、SARSA



t 时刻前向传播得到对 n 时刻的预测 $Q(s_t, a_t; w)$，没有用到任何经验数据。t+1 时刻收集到实际观测 r_t，使得对 n 时刻的预测更精准，反复迭代。

贝尔曼公式

定理 (最优贝尔曼方程)

由 U_t 定义和最优动作价值函数

$$U_t = R_t + \gamma \cdot \sum_{k=t+1}^{n}$$

$$Q_* = \max_{\pi} \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$$

经【推导】得

$$Q_*(s_t, a_t) = \mathbb{E}_{S_{t+1} \sim p(\cdot | s_t, a_t)}[R_t + \gamma \cdot \max_{A \in \mathcal{A}} Q_*(S_{t+1}, A) | S_t = s_t, A_t = a_t]$$

什么意思？在给定 s_t, a_t 的情况下，可根据状态转移函数算出 s_{t+1}，从而奖励 R_t 也被观测到，形成四元组

$$(s_t, a_t, r_t, s_{t+1})$$

可计算出

$$r_t + \gamma \cdot \max_{a \in \mathcal{A}} Q_*(s_{t+1}, a)$$

蒙特卡洛近似期望

$$\mathbb{E}[f(X)] = \int_{\Omega} p(x) \cdot f(x)\, dx$$

1. 随机抽取 N 个样本 $(s_1, a_1), \cdots, (s_N, a_N) \sim p(\cdot)$
2. 根据 $p(S_{t+1}|S_t, A_t)$ 和 $R_t(S_t, A_t, S_{t+1})$，得到 N 个四元组 $(s_1, a_1, s_2, r_1), \cdots, (s_N, a_N, s_{N+1}, r_N) \sim p(\cdot)$
3. 对函数值 $f(s_1, a_1, s_2, r_1), \cdots, f(s_N, a_N, s_{N+1}, r_N)$ 取平均

$$q_N = \frac{1}{N} \sum_{t=1}^{N} f(s_t, a_t, s_{t+1}, r_t)$$

4. 返回 q_N 作为 $\mathbb{E}[f(s_t, a_t, s_{t+1}, r_t)]$ 的估计

其中 $f(s_t, a_t, s_{t+1}, r_t) = r_t + \gamma \cdot \max_{a \in \mathcal{A}} Q_*(s_{t+1}, a)$
由最优贝尔曼方程和蒙特卡洛近似可得

$$Q_*(s_t, a_t) \approx r_t + \gamma \cdot \max_{a \in \mathcal{A}} Q_*(s_{t+1}, a)$$

收集训练数据

至此，我们知道训练 DQN 所需的两个步骤:

1. 收集训练数据，得到四元组 (s_t, a_t, s_{t+1}, r_t)，存入缓存
2. 从缓存中随机取出四元组，更新参数

收集训练数据有多种方法，这里介绍两种。

1. ϵ-greedy 策略

$$\begin{cases} \arg\max_a Q(s_t, a; w) & p = 1 - \epsilon \\ \text{均匀抽取}\mathcal{A}\text{中的一个动作} & p = \epsilon \end{cases}$$

有一变体为 ϵ 随时间递减，即 $\epsilon = 1/t$

2. 上置信界算法

基于霍夫丁不等式（Hoeffding's inequality）。令 $X_1, \cdots, X_n$ 为独立同分布，取值范围为 $[0, 1]$，样本均值为 $\bar{x}_n$

$$P(\mathbb{E}[X] \geqslant \bar{x}_n + u) \leqslant e^{-2nu^2}$$

其中 u 表示不确定性度量。在强化学习中 $u = \hat{U}(s_t, a_t)$，$\bar{x}_n = \hat{U}(s_t, a_t)$。

$$P(Q(s_t, a_t) \geqslant \hat{Q}(s_t, a_t) + \hat{U}(s_t, a_t)) \leqslant e^{-2N(s_t,a_t)\,U^2(s_t,a_t)} = p$$

p 为超参数，根据上述不等式 $Q(s_t, a_t) < \hat{Q}(s_t, a_t) + \hat{U}(s_t, a_t)$ 至少以概率 $1 - p$ 成立。当 p 很小时，前者以很大概率成立。再利用

$$\hat{U}(s_t, a_t) = \sqrt{\frac{-\log p}{2N(s_t, a_t)}}$$

用 Q 学习训练 DQN

把 $Q_*(s,a)$ 替换成 $Q(s,a;w)$

$$Q(s,a;w) \approx r_t + \gamma \cdot \max_{a \in \mathcal{A}} Q(s_{t+1},a;w)$$

左边为预测 $\hat{q}_t$，右边为 TD 目标 $\hat{y}_t$。TD 目标更可靠，因为它借用了实际数据。
根据四元组 (s_t,a_t,r_t,s_{t+1}) 算出 $\hat{q}_t,\hat{y}_t$ 后
定义损失函数

$$L(w) = \frac{1}{2}[Q(s_t,a_t;w) - \hat{y}_t]^2$$

梯度

$$\nabla_w L(w) = (\hat{q}_t - \hat{y}_t) \cdot \nabla_w Q(s_t,a_t;w)$$

梯度下降

$$w \leftarrow w - \alpha \cdot \delta_t \cdot \nabla_w Q(s_t,a_t;w)$$

Outline

经验回放

定义 (经验回放)

把 Agent 与环境交互的记录（经验）存储在一个缓存里，事后反复使用这些经验训练 Agent。具体来说，缓存中有 b 个四元组 (s_t, a_t, r_t, s_{t+1}) 算出 $\hat{q}_t, \hat{y}_t$，这里 b 是超参数，缓存满时会挤掉最旧的数据、放入最新的，b 一般设置为 $10^5 \sim 10^6$

定义 (优先经验回放)

优先经验回放的本质是不均匀抽样。每条经验并不享有同等地位！比如无人驾驶，出事故的数据比正常行驶的重要的多。可通过 $|Q(s_j, a_j; w) - Q_*(s_j, a_j)|$ 判断重要性。但由于 $Q_*(s_j, a_j)$ 未知，可用 TD 误差（即 $\hat{q}_t - \hat{y}_t$）替代。
两个地方做了改进：
1. 抽样方法

$$p_j \propto |\delta_j| + \epsilon \quad or \quad p_j \propto \frac{1}{\text{rank}(j)}$$

2. 学习率

$$w_{new} \leftarrow w_{now} - \alpha \cdot \nabla$$

$$\alpha_j = \frac{\alpha}{(b \cdot p_j)^\beta}, \beta \in (0, 1)$$

特殊地，当所有概率相等时，$(b \cdot p_j)^\beta = 1$，学习率也相等

自举和高估问题

回顾 DQN 参数更新的过程

1. 计算 TD 目标

$$\hat{y}_j = r_j + \gamma \cdot \underbrace{\max_{a_{j+1}\in\mathcal{A}} Q(s_{j+1}, a_{j+1}; w_{now})}_{\text{不准!}}$$

2. 定义损失函数

$$L(w) = \frac{1}{2}[Q(s_j, a_j; w) - \hat{y}_j]^2$$

3. 把 $\hat{y}_j$ 看作常数，做梯度下降

$$w_{new} \leftarrow w_{now} - \alpha \cdot \nabla_w L(w_{now})$$

$Q(s_{j+1}, a_{j+1}; w_{now})$ 是对 $Q_*(s_{j+1}, a_{j+1})$ 的近似，假如前者被低估/高估，将通过 TD 目标和反向传播传递下去。这种情况被称为“自举”，即自己把自己举起来。
max 导致高估

$$\mathbb{E}[\max(Z_1, \cdots, Z_d)] \geqslant \max(\mathbb{E}(Z_1), \cdots, \mathbb{E}(Z_d)) = \max(x_1, \cdots, x_d)$$

高估的危害在于高估不均匀

用另一个网络切断自举

自举的问题在于，$Q(s_j, a_j; w)$ 和近似目标 $\hat{y}_t$ 都是自己的，相当于自己在近似自己。如果新增一个网络（目标网络），由它计算 $\hat{y}_t$，可以切断自举。设该网络为

$$Q(s, a; w^-)$$

1. 对 DQN 正向传播

$$\hat{q}_j = Q(s_j, a_j; w_{now})$$

2. 对目标网络正向传播

$$\hat{q}_{j+1}^- = \max_{a \in \mathcal{A}} Q(s_{j+1}, a; w_{now}^-)$$

3. 计算 TD 目标和 TD 误差

$$\hat{y}_j^- = r_j + \gamma \cdot \hat{q}_{j+1}^- \qquad \delta_j = \hat{q}_j - \hat{y}_j^-$$

4. 梯度下降

$$w_{new} \leftarrow w_{now} - \alpha \cdot \delta_j \cdot \nabla_w Q(s_j, a_j; w_{now})$$

5. 更新目标网络的参数

$$w_{new}^- \leftarrow \tau \cdot w_{new} + (1 - \tau) \cdot w_{now}^-$$

DDQN

把最大化拆成两步

1. $a^* = \arg\max_{a\in\mathcal{A}} Q(s_{j+1}, a; w)$
2. $\hat{y}_j = r_j + Q(s_{j+1}, a^*; w)$

第一步让原 DQN 做，第二步让目标网络来做

1. 对 DQN 正向传播

$$\hat{q}_j = Q(s_j, a_j; w_{now})$$

2. 对目标网络正向传播

2.1 选择

$$a^* = \arg\max_{a\in\mathcal{A}} Q(s_{j+1}, a; w_{now})$$

2.2 求值

$$\hat{q}_{j+1}^- = Q(s_{j+1}, a^*; w_{now}^-)$$

3. 计算 TD 目标和 TD 误差

$$\hat{y}_j^- = r_j + \gamma \cdot \hat{q}_{j+1}^- \qquad \delta_j = \hat{q}_j - \hat{y}_j^-$$

4. 梯度下降

$$w_{new} \leftarrow w_{now} - \alpha \cdot \delta_j \cdot \nabla_w Q(s_j, a_j; w_{now})$$

5. 更新目标网络的参数

$$w_{new}^- \leftarrow \tau \cdot w_{new} + (1-\tau) \cdot w_{now}^-$$

OpenAI Gym

Gym
Documentation

Search

INTRODUCTION

Basic Usage

API

Core

Spaces

Wrappers

Vector

Utils

ENVIRONMENTS

Atari

MuJoCo

Toy Text

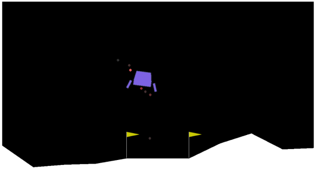
Classic Control

Box2D

Third Party Environments

TUTORIALS

Gym is a standard API for reinforcement learning, and a diverse collection of reference environments



The Gym interface is simple, pythonic, and capable of representing general RL problems:

```
import gym
env = gym.make("LunarLander-v2", render_mode="human")
observation, info = env.reset(seed=42)
for _ in range(1000):
   action = policy(observation)  # User-defined policy function
   observation, reward, terminated, truncated, info = env.step(action)

   if terminated or truncated:
      observation, info = env.reset()
env.close()
```

参考文献 I

1. 王树森：深度强化学习
2. 动手学强化学习