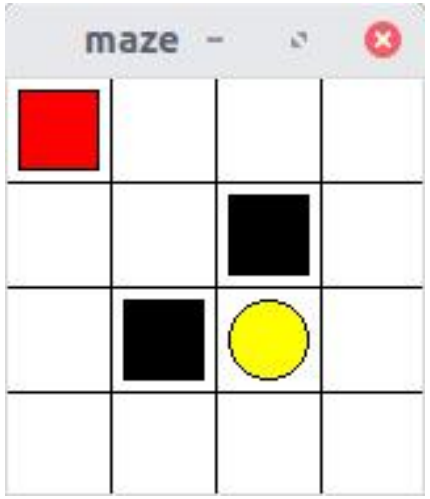
强化学习-时序差分学习

主讲TA：张祖胜

2020/12/11

往期回顾

- 两个交互对象：智能体 & 环境
- 五要素
  - 状态s：对环境的描述
  - 动作a：对智能体行为的描述
  - 策略π(a|s)：根据状态s决定下一步动作a的函数
  - 状态转移概率p(s'|s,a)：环境从状态s转移至状态s'的概率
  - 即时奖励r(s,a,s')：环境反馈给智能体的奖励



往期回顾

- 学习目标：找到一个最优策略$\pi_\theta(a|s)$来最大化期望回报（τ表示一次交互过程的轨迹）

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[G(\tau)] = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_{t=0}^{T-1} \gamma^t r_{t+1}]$$

- 值函数（说白了就是期望回报）
  - 状态值函数$V^\pi(s)$：

$$\begin{aligned}\mathbb{E}_{\tau \sim p(\tau)}[G(\tau)] &= \mathbb{E}_{s \sim p(s_0)}\left[\mathbb{E}_{\tau \sim p(\tau)}[\sum_{t=0}^{T-1} \gamma^t r_{t+1} | \tau_{s_0} = s]\right] \\ &= \mathbb{E}_{s \sim p(s_0)}\left[V^\pi(s)\right],\end{aligned}$$

  - 状态-动作值函数$Q^\pi(s,a)$，也称为Q函数：

$$Q^\pi(s,a) = \mathbb{E}_{s' \sim p(s'|s,a)}\left[r(s,a,s') + \gamma V^\pi(s')\right]$$

往期回顾

- 贝尔曼方程（本质上是递推）
  - 关于状态值函数$V^{\pi}(s)$：

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\tau_{0:T}\sim p(\tau)}\left[r_1 + \gamma \sum_{t=1}^{T-1}\gamma^{t-1} r_{t+1} | \tau_{s_0} = s\right] \\
&= \mathbb{E}_{a\sim\pi(a|s)}\mathbb{E}_{s'\sim p(s'|s,a)}\mathbb{E}_{\tau_{1:T}\sim p(\tau)}\left[r(s,a,s') + \gamma \sum_{t=1}^{T-1}\gamma^{t-1} r_{t+1} | \tau_{s_1} = s'\right] \\
&= \mathbb{E}_{a\sim\pi(a|s)}\mathbb{E}_{s'\sim p(s'|s,a)}\left[r(s,a,s') + \gamma \mathbb{E}_{\tau_{1:T}\sim p(\tau)}\left[\sum_{t=1}^{T-1}\gamma^{t-1} r_{t+1} | \tau_{s_1} = s'\right]\right] \\
&= \mathbb{E}_{a\sim\pi(a|s)}\mathbb{E}_{s'\sim p(s'|s,a)}\left[r(s,a,s') + \gamma V^{\pi}(s')\right].
\end{aligned}
$$

  - 关于状态-动作值函数（Q函数）：

$$
Q^{\pi}(s,a) = \mathbb{E}_{s'\sim p(s'|s,a)}\left[r(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(a'|s')}[Q^{\pi}(s',a')]\right]
$$

往期回顾

- 贝尔曼最优方程（本质上是递推）
  - 关于最优状态值函数V*(s)：

$$V^*(s) = \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s, a, s') + \gamma V^*(s') \right]$$

  - 关于最优状态-动作值函数Q*(s,a)：

$$Q^*(s, a) = \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right]$$

强化学习方法分类

- 基于值函数
  - ➢动态规划算法：p(s'|s,a)和r(s,a,s')已知，通过优化值函数来找最优策略
    - ✓策略迭代：先根据贝尔曼方程更新值函数，再改进策略
    - ✓值迭代：直接根据贝尔曼**最优**方程更新值函数
  - ➢蒙特卡罗方法：p(s'|s,a)和r(s,a,s')未知，需采样多条轨迹来估计Q函数
  - ➢时序差分学习方法
    - ✓SARSA
    - ✓Q学习（延伸：深度Q网络）
- 基于策略函数

时序差分学习

- 背景：蒙特卡罗方法需要拿到完整的轨迹，才能评估和更新策略，效率较低

- 改进：引入动态规划；模拟一段轨迹，每行动一步，就利用贝尔曼方程评估状态的价值

时序差分学习

- Q函数估计：从蒙特卡罗到时序差分

$$\hat{Q}_N^{\pi}(s,a) = \frac{1}{N}\sum_{n=1}^{N} G(\tau_{s_0=s,a_0=a}^{(n)})$$

$$= \frac{1}{N}\left(G(\tau_{s_0=s,a_0=a}^{(N)}) + \sum_{n=1}^{N-1} G(\tau_{s_0=s,a_0=a}^{(n)})\right)$$

$$= \frac{1}{N}\left(G(\tau_{s_0=s,a_0=a}^{(N)}) + (N-1)\hat{Q}_{N-1}^{\pi}(s,a)\right)$$

$$= \hat{Q}_{N-1}^{\pi}(s,a) + \frac{1}{N}\left(G(\tau_{s_0=s,a_0=a}^{(N)}) - \hat{Q}_{N-1}^{\pi}(s,a)\right)$$

- 不失一般性，将1/N改为一个较小的正数α：

$$\hat{Q}^{\pi}(s,a) \leftarrow \hat{Q}^{\pi}(s,a) + \alpha\left(G(\tau_{s_0=s,a_0=a}) - \hat{Q}^{\pi}(s,a)\right)$$

蒙特卡罗误差：真实回报与期望回报间的差距

时序差分学习

- 不失一般性，将1/N改为一个较小的正数α：

$$\hat{Q}^{\pi}(s,a) \leftarrow \hat{Q}^{\pi}(s,a) + \alpha\left(G(\tau_{s_0=s,a_0=a}) - \hat{Q}^{\pi}(s,a)\right)$$

- 利用贝尔曼方程（动态规划）估计真实回报$G(\tau_{s0=s,a0=a})$：

$$\begin{aligned} G(\tau_{s_0=s,a_0=a,s_1=s',a_1=a'}) &= r(s,a,s') + \gamma G(\tau_{s_0=s',a_0=a'}) \\ &\approxeq r(s,a,s') + \gamma \hat{Q}^{\pi}(s',a'), \end{aligned}$$

- SARSA算法：

$$\hat{Q}^{\pi}(s,a) \leftarrow \hat{Q}^{\pi}(s,a) + \alpha\left(r(s,a,s') + \gamma \hat{Q}^{\pi}(s',a') - \hat{Q}^{\pi}(s,a)\right)$$

时序差分学习（SARSA & Q学习）

- SARSA算法估计Q函数：

$$\hat{Q}^{\pi}(s,a) \leftarrow \hat{Q}^{\pi}(s,a) + \alpha\Big(r(s,a,s') + \gamma\hat{Q}^{\pi}(s',a') - \hat{Q}^{\pi}(s,a)\Big)$$

- Q学习算法估计Q函数：

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big)$$

时序差分学习（SARSA & Q学习）

算法 14.3 SARSA：一种同策略的时序差分学习算法

输入: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

$\forall s, \forall a$，随机初始化 $Q(s,a)$; 初始化策略 $\pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  初始化起始状态 s; 选择动作 $a = \pi^{\epsilon}(s)$;

  repeat

    执行动作 a，得到即时奖励 r 和新状态 s';

    在状态 s'，选择动作 $a' = \pi^{\epsilon}(s')$;

    $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$;

    $\pi(s) = \arg\max_{a \in |\mathcal{A}|} Q(s,a)$;

    $s \leftarrow s', a \leftarrow a'$;

  until s 为终止状态;

until $\forall s, a, Q(s,a)$ 收敛;

输出: 策略 $\pi(s)$

算法 14.4 Q学习：一种异策略的时序差分学习算法

输入: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

$\forall s, \forall a$，随机初始化 $Q(s,a)$; 初始化策略 $\pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  初始化起始状态 s;

  repeat

    在状态 s，选择动作 $a = \pi^{\epsilon}(s)$;

    执行动作 a，得到即时奖励 r 和新状态 s';

    $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$;

    $s \leftarrow s'$;

  until s 为终止状态;

until $\forall s, a, Q(s,a)$ 收敛;

输出: 策略 $\pi(s) = \arg\max_{a \in |\mathcal{A}|} Q(s,a)$

时序差分学习（SARSA & Q学习）

- SARSA：采样和优化的策略都是π^{ε}，因此是一种同策略算法

- Q学习：不通过π^{ε}来选择下一步动作a'，而直接选择最优Q函数，所以更新后的Q函数是关于策略π而非π^{ε}的，因此是一种异策略算法

深度Q网络

- SARSA和Q学习适合在**离散**的状态和动作空间中计算值函数$Q^{\pi}(s,a)$，例如通过查表的方式获取Q值

- 为了适应连续的状态和动作空间，可以用一个函数$Q_{\phi}(\mathbf{s},\mathbf{a})$来表示近似计算：

$$Q_{\phi}(\boldsymbol{s},\boldsymbol{a}) \approx Q^{\pi}(s,a)$$

- 以Q学习为例，学习目标为：

$$\mathcal{L}(s,a,s'|\phi) = \left(r + \gamma \max_{a'} Q_{\phi}(\boldsymbol{s}',\boldsymbol{a}') - Q_{\phi}(\boldsymbol{s},\boldsymbol{a})\right)^2$$

深度Q网络

- 学习目标：

$$\mathcal{L}(s, a, s'|\phi) = \left(r + \gamma \max_{a'} Q_\phi(\boldsymbol{s}', \boldsymbol{a}') - Q_\phi(\boldsymbol{s}, \boldsymbol{a})\right)^2$$

- 存在问题
  - 目标不稳定：参数学习的目标依赖于参数本身
  - 样本间具有强相关性：相邻训练样本具有相似性，模型可能陷入局部最优
- 解决方案：深度Q网络（DQN）
  - 目标网络冻结：在一个时间段内固定目标参数，稳定学习目标
  - 经验回放：构建经验池，池中存放智能体最近的经历，训练时随机从中选取样本，去除数据相关性

深度Q网络

算法 14.4　Q学习：一种异策略的时序差分学习算法

输入： 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

$\forall s, \forall a$，随机初始化 $Q(s,a)$；初始化策略 $\pi(a|s) = \frac{1}{|\mathcal{A}|}$；

repeat

　初始化起始状态 s；

　repeat

　　在状态 s，选择动作 $a = \pi^{\epsilon}(s)$；

　　执行动作 a，得到即时奖励 r 和新状态 s'；

　　$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$；

　　$s \leftarrow s'$；

　until s 为终止状态；

until $\forall s, a, Q(s,a)$ 收敛；

输出： 策略 $\pi(s) = \arg\max_{a \in |\mathcal{A}|} Q(s,a)$

算法 14.5　带经验回放的深度Q网络

输入： 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α，参数更新间隔 C；

初始化经验池 $\mathcal{D}$，容量为 N；

随机初始化 Q 网络的参数 ϕ；

随机初始化目标 Q 网络的参数 $\hat{\phi} = \phi$；

repeat

　初始化起始状态 s；

　repeat

　　在状态 s，选择动作 $a = \pi^{\epsilon}$；

　　执行动作 a，观测环境，得到即时奖励 r 和新的状态 s'；

　　将 s, a, r, s' 放入 $\mathcal{D}$ 中；

　　从 $\mathcal{D}$ 中采样 ss, aa, rr, ss'；

　　$$y = \begin{cases} rr, & ss' \text{为终止状态}, \\ rr + \gamma \max_{a'} Q_{\hat{\phi}}(\mathbf{ss}', \boldsymbol{a}'), & \text{否则} \end{cases};$$

　　以 $\left(y - Q_{\phi}(\boldsymbol{ss}, \boldsymbol{aa})\right)^2$ 为损失函数来训练 Q 网络；

　　$s \leftarrow s'$；

　　每隔 C 步，$\hat{\phi} \leftarrow \phi$；

　until s 为终止状态；

until $\forall s, a, Q_{\phi}(\boldsymbol{s}, \boldsymbol{a})$ 收敛；

输出： Q 网络 $Q_{\phi}(\boldsymbol{s}, \boldsymbol{a})$

参考资料

- 莫烦Python：https://mofanpy.com/tutorials/machine-learning/ML-intro/RL/

- 开源代码：https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow