

入门

1. 强化学习两个随机性来源

1. 动作根据给定的环境，计算**下一步每种动作**的概率，而随机生成
2. 下一时间的**环境状态**根据当前的环境状态和动作而随机生成
2. 三元组 $(s_1, a_1, r_1, \dots, s_t, a_t, r_t)$ (state, action, reward)
3. **折扣汇报**，未来的奖励没现在的奖励值钱

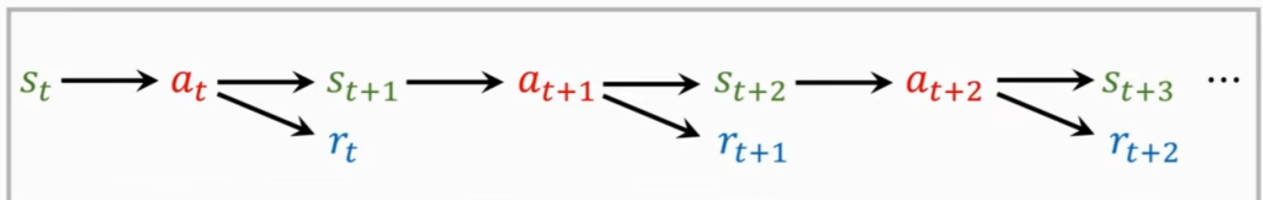
$$G_t = R_t + \gamma R_{t+1} + \dots$$

为什么回报与未来的奖励有关，与过去的奖励无关？

过去的因为是已经改变不了的了，不是个变量了，而强化学习做的是在当前状态下该采取什么动作，做了当前的动作是为了未来更好才去做的。

1. **动作价值函数 (Action-value function)** $Q_{\pi}(s_t, a_t) = E[G_t | S_t = s_t, A_t = a_t]$ ，因为没办法直接求 G_t ，所以只能通过概率去求其期望，转化为状态价值函数（与policy π 有关），这个函数是**用来评价当前的动作好不好**，这里的期望是根据未来的所有可能的状态 A_{t+1}, A_{t+2}, \dots 和动作 $(S_{t+1}, S_{t+2}, \dots)$ 求的
2. **optimal Action-value function**: $Q^*(s_t, a_t) = \underset{\pi}{\max} Q_{\pi}(s_t, a_t)$ ，作用为评价当前的动作，找一个最好的policy π ，使得当前的动作最好
3. **State-value function**: $V_{\pi}(s_t) = E_{\pi}[Q_{\pi}(s_t, A)]$ ，可以判断**当前局势好不好**，因为是对所有动作求期望，所以是对当前局势的评价 $V_{\pi}(s_t) = E_{\pi}[Q_{\pi}(s_t, A)] = \begin{cases} \sum_a Q_{\pi}(s_t, a) \pi(a|s_t) & \text{if discrete} \\ \int_a Q_{\pi}(s_t, a) \pi(a|s_t) da & \text{if continuous} \end{cases}$ $E_s[V_{\pi}(S)]$ 也可以**评价当前 policy π 的好坏**，因为是对所有状态求期望，所以是对整个policy的评价
4. **agent学习方式**：根据当前的观察状态 s_t
 1. 策略学习(policy $\pi(a|s)$)
 2. 价值学习(optimal value function $Q^*(s, a)$)

5. 学习过程

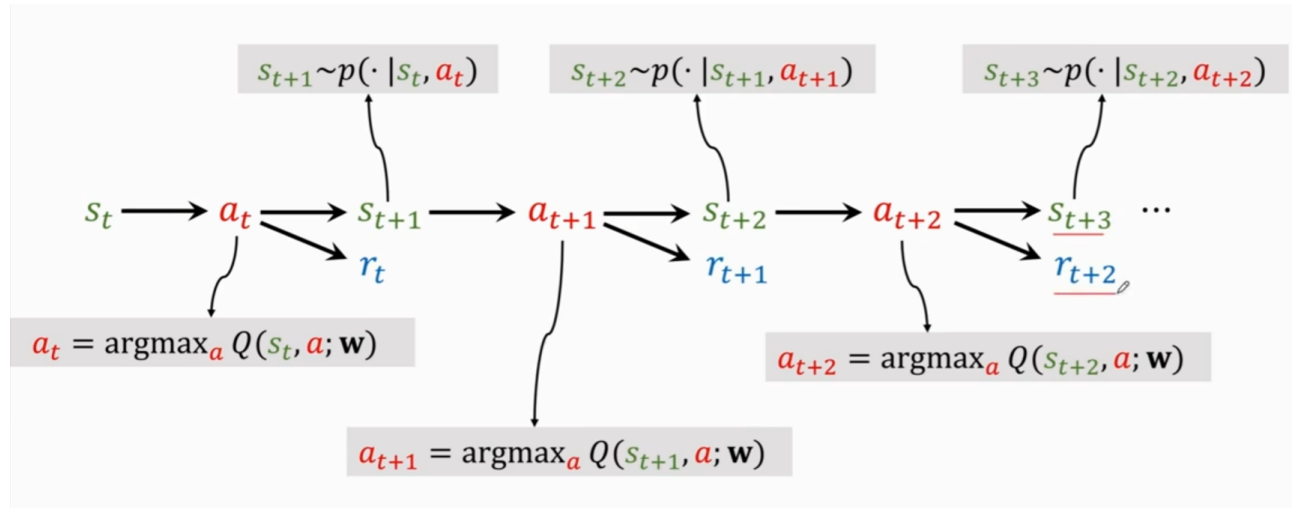


6. 贝尔曼方程及其推导

1. 状态值函数的贝尔曼方程 $V^{\pi}(s) = \mathbb{E}_{\pi} \left[R(s, a) + \gamma \mathbb{E}_{\pi} [V^{\pi}(s') | s, a] \right]$
 1. 状态值函数的贝尔曼方程是一个递归方程，它表示了一个状态的值与其后继状态的值之间的关系。

2. 推导过程：
$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[G_t \mid S_t = s \right] = \mathbb{E}_{\pi} \left[R_{t+1} + \gamma V^{\pi}(S_{t+1}) \mid S_t = s \right] = \mathbb{E}_{\pi} \left[R(s, a) + \gamma V^{\pi}(S') \mid S = s \right] = \mathbb{E}_{\pi} \left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^{\pi}(s') \right]$$
2. 动作值函数的贝尔曼期望方程 $Q^{\pi}(s, a) = R(s, a) + \gamma \mathbb{E}_{\pi} \left[\sum_{s'} P(s'|s, a) Q^{\pi}(s', a) \right]$
1. 动作值函数的贝尔曼方程是一个递归方程，它表示了一个状态动作对的值与其后继状态动作对的值之间的关系。
2. 推导过程：
$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_{\pi} \left[R_{t+1} + \gamma Q^{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] = \mathbb{E}_{\pi} \left[R(s, a) + \gamma Q^{\pi}(S', A') \mid S = s, A = a \right] = R(s, a) + \gamma \mathbb{E}_{\pi} \left[\sum_{s'} P(s'|s, a) \mathbb{E}_{\pi} \left[Q^{\pi}(s', a) \right] \right]$$
3. 状态值函数的贝尔曼最优方程 $V^*(s) = \max_a \left[R(s, a) + \gamma \mathbb{E}_{\pi^*} \left[\sum_{s'} P(s'|s, a) V^*(s') \right] \right]$
1. 状态值函数的贝尔曼最优方程是一个递归方程，它表示了一个状态的值与其后继状态的值之间的关系。
2. 推导过程：
$$V^*(s) = \max_a \left[Q^*(s, a) \right] = \max_a \left[R(s, a) + \gamma \mathbb{E}_{\pi^*} \left[\sum_{s'} P(s'|s, a) V^*(s') \right] \right]$$
4. 动作值函数的贝尔曼最优方程 $Q^*(s, a) = R(s, a) + \gamma \mathbb{E}_{\pi^*} \left[\sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \right]$
1. 动作值函数的贝尔曼最优方程是一个递归方程，它表示了一个状态动作对的值与其后继状态动作对的值之间的关系。
2. 推导过程：
$$Q^*(s, a) = \mathbb{E}_{\pi^*} \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_{\pi^*} \left[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a \right] = \mathbb{E}_{\pi^*} \left[R(s, a) + \gamma \max_{a'} Q^*(S', a') \mid S = s, A = a \right] = R(s, a) + \gamma \mathbb{E}_{\pi^*} \left[\sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \right]$$
7. DQN (Deep Q-Network) (离散型) 用深度学习来近似Q函数 目标：最大化奖励函数
- 问题1：如何通过已知的 $Q^*(s, a)$ 来寻找最好动作 答： $a^* = \underset{a}{\operatorname{argmax}} Q^*(s, a)$
- 挑战： $Q^*(s, a)$ 是未知的，只能通过已知的数据来近似 答：用神经网络 $Q(s, a; w)$ 来近似 $Q^*(s, a)$ (DNQ)

训练过程：



8. Temporal Difference(TD) Learning 最简单的例子

Example

- I want to drive from NYC to Atlanta.
- Model $Q(w)$ estimates the time cost, e.g., 1000 minutes.

Question: How do I update the model?

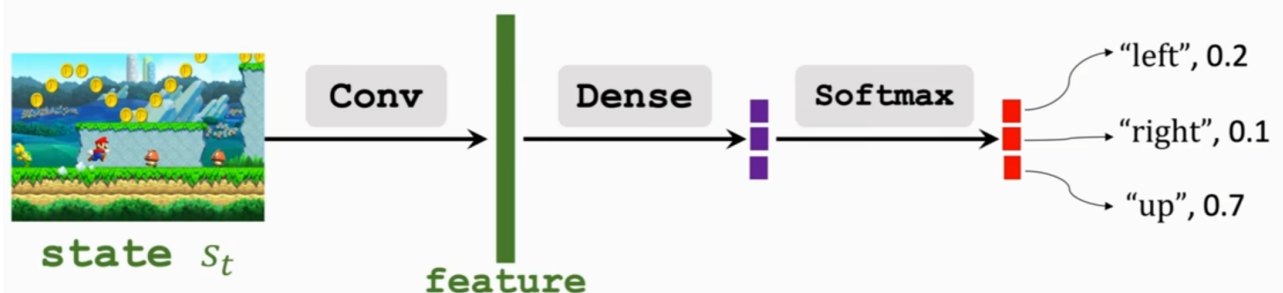
- Make a prediction: $q = Q(w)$, e.g., $q = 1000$.
- Finish the trip and get the target y , e.g., $y = 860$.
- Loss: $L = \frac{1}{2}(q - y)^2$.
- Gradient: $\frac{\partial L}{\partial w} = \frac{\partial q}{\partial w} \cdot \frac{\partial L}{\partial q} = (q - y) \cdot \frac{\partial Q(w)}{\partial w}$.
- Gradient descent: $w_{t+1} = w_t - \alpha \cdot \frac{\partial L}{\partial w} \Big|_{w=w_t}$.



Q: Can I update the model before the end of the episode? A: 前半段实际值，后半段预测值，加起来作为一个值去与模型的整体估计值作比较

9. apply TD to DQN 因为 $G_t = R_t + \gamma G_{t+1}$ 所以 $y_t = Q(s_t, a_t; w) \approx r_t + \gamma Q(s_{t+1}, a_{t+1}; w) = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; w)$ (a_{t+1} 的选择原理) \ Loss: $L_t = \frac{1}{2} (y_t - Q(s_t, a_t; w))^2$ 所以 $EG_t \approx E[r_t + \gamma EG_{t+1}]$ (TD target)

10. Policy Network $\pi(a|s, \theta)$



近似状态价值函数 $V(s_t, \theta) = \sum_a \pi(a|s_t, \theta) Q(s_t, a)$ 问题1：如何学习参数

θ ? 答: 学习 θ , 使得 $J(\theta) = E_s[V(S; \theta)]$ 最大化

策略梯度算法(随机梯度, 随机性来源于 s) 1. 观察当前状态 s 2. 更新参数 $\theta \leftarrow \theta + \beta \cdot \frac{\partial V(s; \theta)}{\partial \theta}$

问题2: 如何求 $\frac{\partial V(s; \theta)}{\partial \theta}$? (此处并不严谨) 答: $\begin{array}{l} \frac{\partial V(s; \theta)}{\partial \theta} = \sum_a \frac{\partial \pi(a|s; \theta)}{\partial \theta} Q_\pi(s, a) \end{array}$

(离散形式) $= \sum_a \pi(a|s; \theta) \frac{\partial \log \pi(a|s; \theta)}{\partial \theta} Q_\pi(s, a)$ (chain rule: $\frac{\partial \log \pi(a|s; \theta)}{\partial \theta} = \frac{1}{\pi(a|s; \theta)} \frac{\partial \pi(a|s; \theta)}{\partial \theta}$)

$= E_a[\frac{\partial \log \pi(a|s; \theta)}{\partial \theta} Q_\pi(s, a)]$ (连续形式) \end{array} 问题

3: 如何在连续的情况下求得 $E_a[\frac{\partial \log \pi(a|s; \theta)}{\partial \theta} Q_\pi(s, a)]$? 答: **蒙特**

卡洛近似, 用采样的方法来估计期望 3. 采样动作 \hat{a} , 根据 $\pi(\hat{a}|s; \theta)$ 4. 计算

$g(\hat{a}, \theta) = \frac{\partial \log \pi(\hat{a}|s; \theta)}{\partial \theta} Q_\pi(s, \hat{a})$ 5. 可以证明

$g(\hat{a}, \theta)$ 为 $\frac{\partial V(s; \theta)}{\partial \theta}$ 的无偏估计

Algorithm

1. Observe the state s_t .
2. Randomly sample action a_t according to $\pi(\cdot | s_t; \theta_t)$.
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate).
4. Differentiate policy network: $\mathbf{d}_{\theta, t} = \frac{\partial \log \pi(a_t | s_t, \theta)}{\partial \theta} \big|_{\theta = \theta_t}$.
5. (Approximate) policy gradient: $\mathbf{g}(a_t, \theta_t) = q_t \cdot \mathbf{d}_{\theta, t}$.
6. Update policy network: $\theta_{t+1} = \theta_t + \beta \cdot \mathbf{g}(a_t, \theta_t)$.

问题4: 如何近似计算 q_t ? 答: 两种方法: 6. REINFORCE

Option 1: REINFORCE.

- Play the game to the end and generate the trajectory:

$$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T.$$

- Compute the discounted return $u_t = \sum_{k=t}^T \gamma^{k-t} r_k$, for all t .
- Since $Q_\pi(s_t, a_t) = \mathbb{E}[U_t]$, we can use u_t to approximate $Q_\pi(s_t, a_t)$.
- \rightarrow Use $q_t = u_t$.

7. Actor-Critic 用神经网络近似价值函数 Q_π