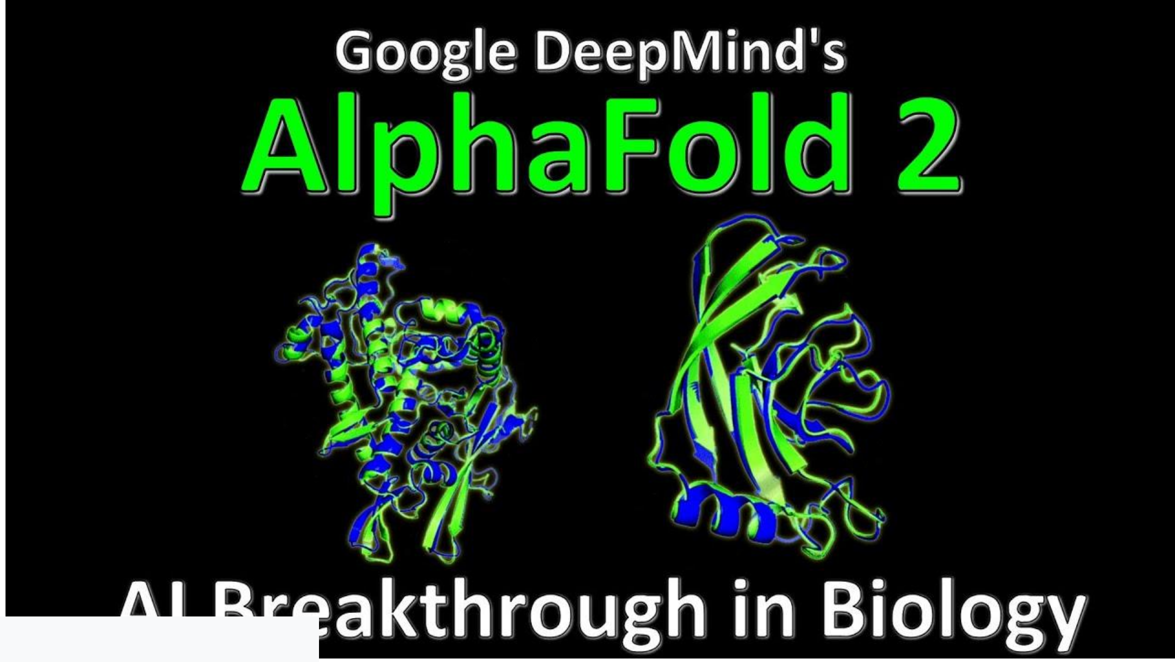
Deep Reinforcement Learning

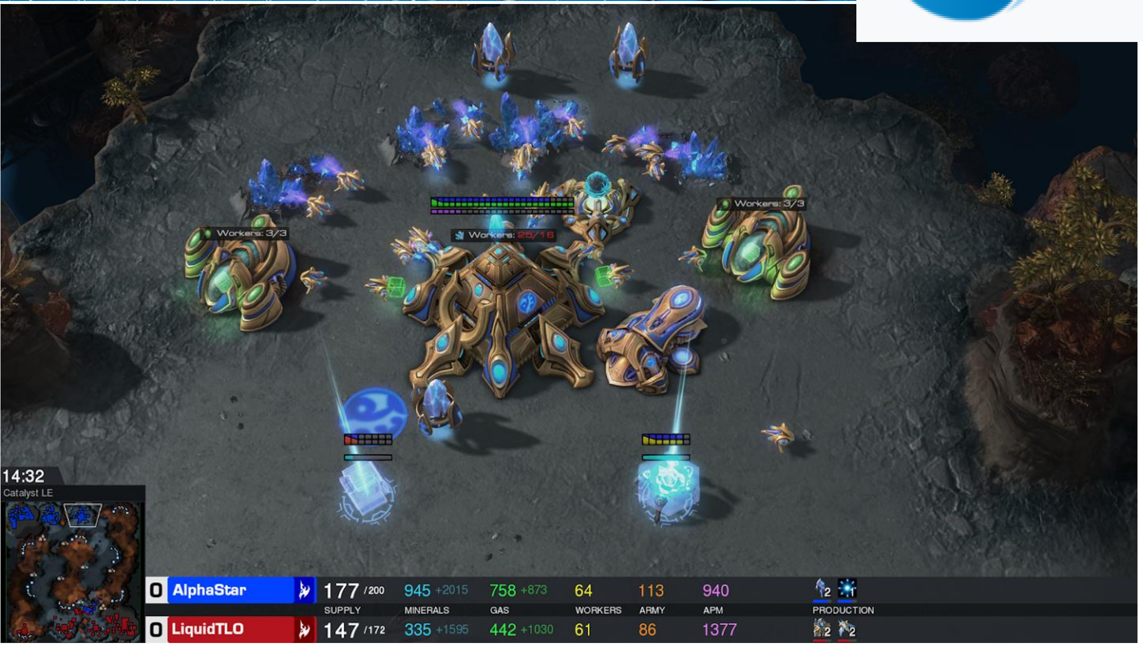
March 20, 2022

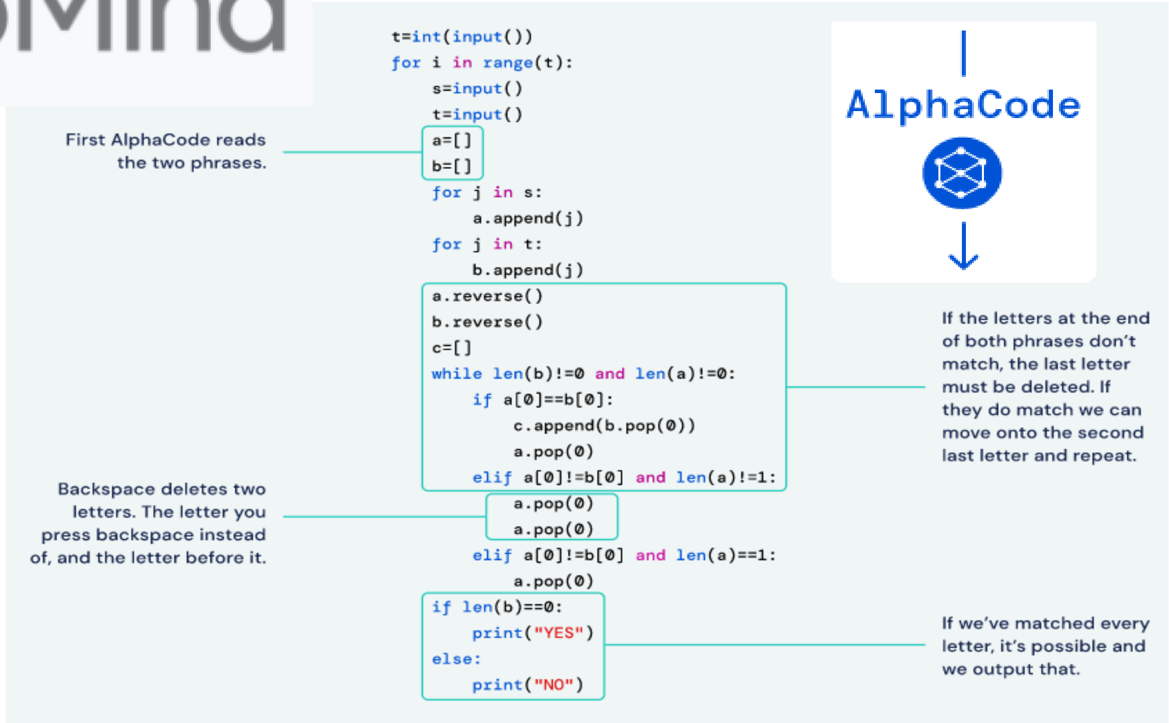
Agenda

- Motivation
- Markov Decision Process
- Reinforcement Learning
- Why Deep?



DeepMind





```
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
    c=[]
    while len(b)!=0 and len(a)!=0:
        if a[0]==b[0]:
            c.append(b.pop(0))
            a.pop(0)
        elif a[0]!=b[0] and len(a)!=1:
            a.pop(0)
            a.pop(0)
        elif a[0]!=b[0] and len(a)==1:
            a.pop(0)
    if len(b)==0:
        print("YES")
    else:
        print("NO")
```

First AlphaCode reads the two phrases.

Backspace deletes two letters. The letter you press backspace instead of, and the letter before it.

If the letters at the end of both phrases don't match, the last letter must be deleted. If they do match we can move onto the second last letter and repeat.

If we've matched every letter, it's possible and we output that.

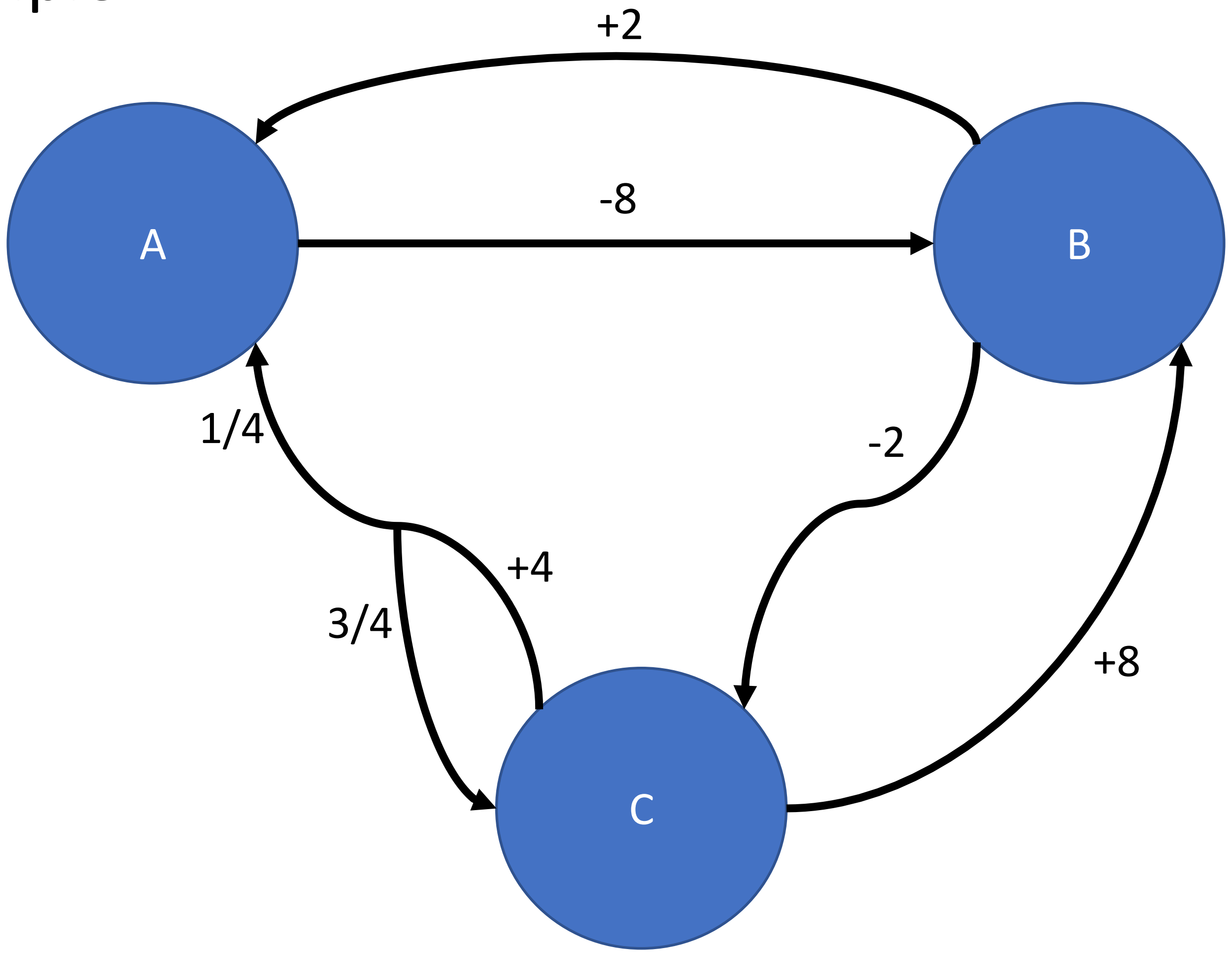
Markov Decision Process

A five-element tuple (T, S, A, P, R)

- Time
- State
- Action
- Transition probability
- Reward

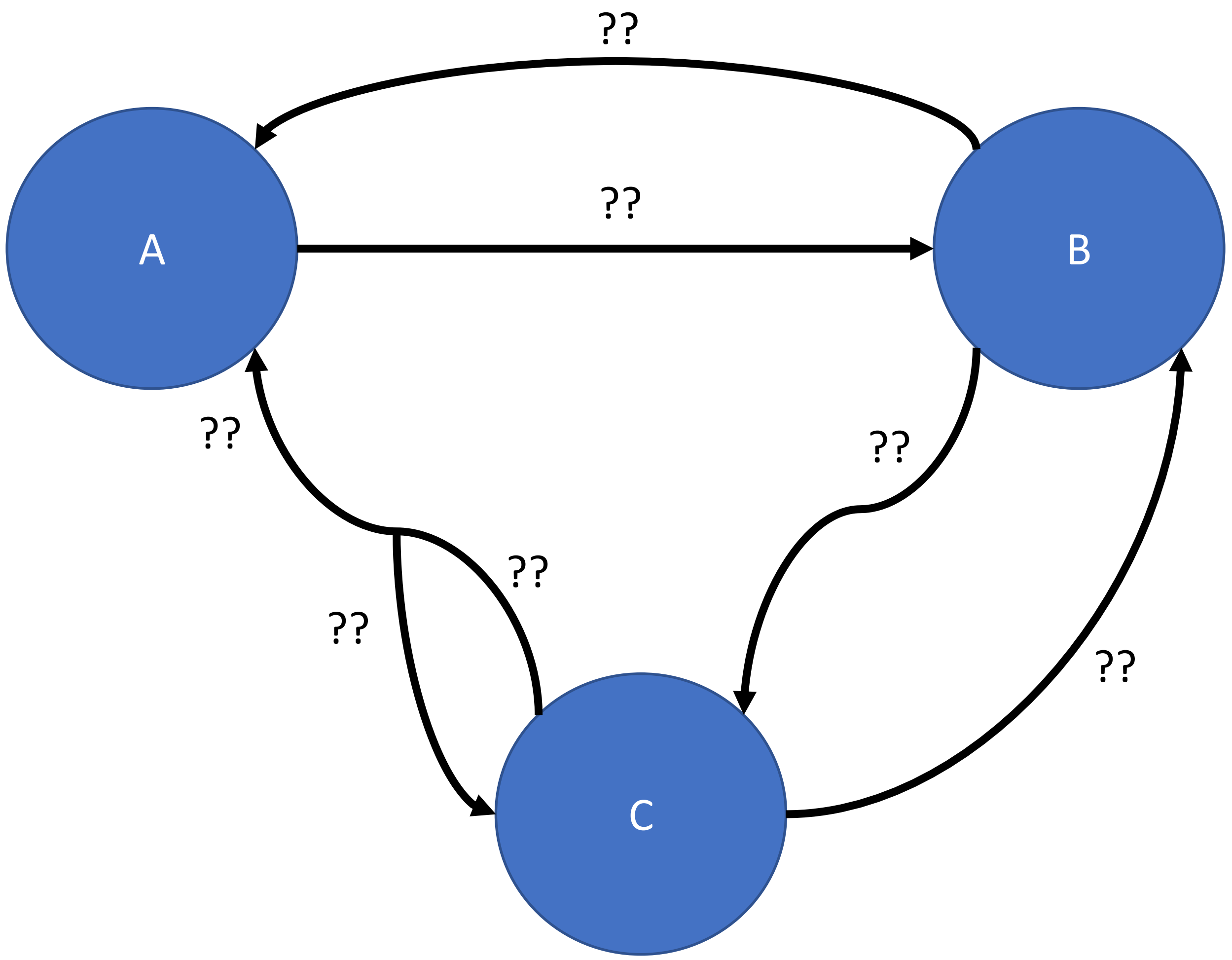
Markov Decision Process

- A simple example



Reinforcement Learning

- What if the reward/transition probability is unknown to us?



Reinforcement Learning

- *Model-based* algorithm
  - The method directly uses environment dynamics
  - Disadvantage: works only when the model is fixed or easy to learn

- *Model-free* algorithm
  - The method explicitly learns the environment dynamics
  - Disadvantage: requires many samples

Reinforcement Learning

- *Temporal-Difference (TD) Learning* algorithm
  - *"one idea central and novel to reinforcement learning"* by Sutton and Barto
- Algorithm:
  - In state s, take action a, observe reward r, and state s' (learning rate is α)

$$v_{k+1}(s) = v_k(s) + \alpha[r + discount\ factor \times v_k(s') - v_k(s)]$$

Reinforcement Learning

- *Q-Learning* algorithm
- Algorithm:
    - In state s, take action a, observe reward r, and state s' (learning rate is α)

$$Q_{k+1}(s,a)$$
$$= Q_k(s,a) + \alpha\left[r + discount\ factor \times \max_{a'} Q_k(s',a') - Q_k(s,a)\right]$$

Why Deep?

- What if the state is continuous or very large?
  - *Discretization* and *Curse of Dimensionality*
- Solution: *Approximate* the value function

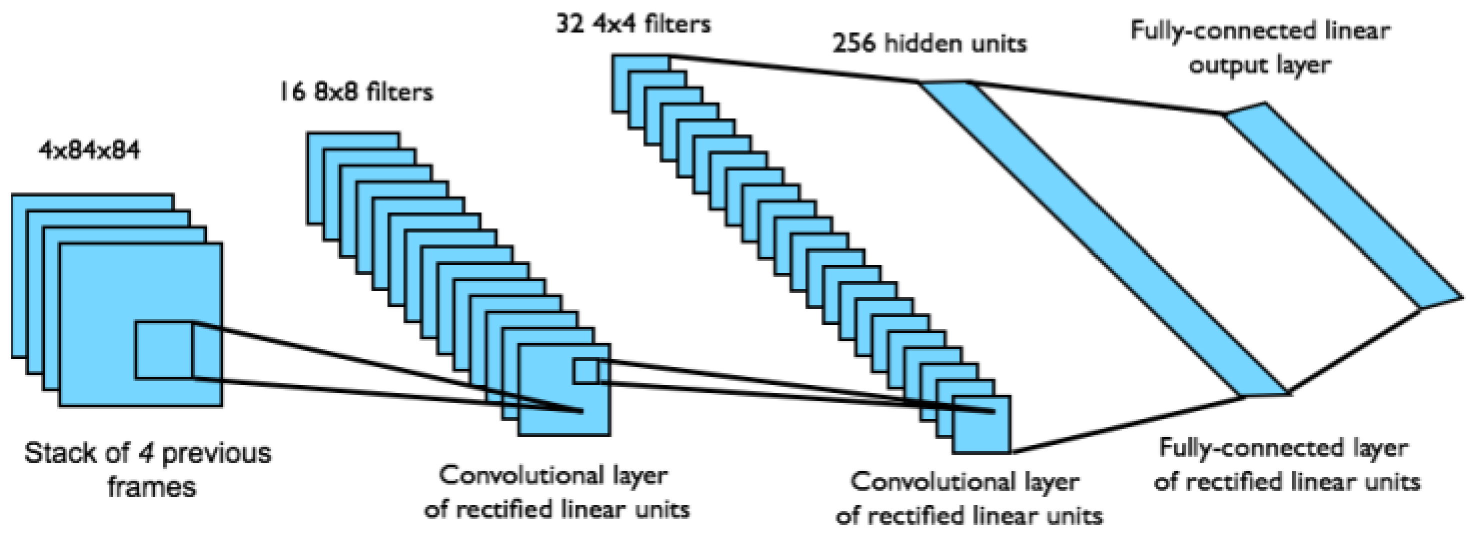
$$Value = f(State, Action)$$

Any curve fitting method ranging from linear/polynomial regression to neural network

- In 2013, DeepMind demonstrated the power of using a deep neural network to approximate the value function, called Deep Q-Network (DQN)

Why Deep?

- DQN in Atari


4x84x84
16 8x8 filters
32 4x4 filters
256 hidden units
Fully-connected linear output layer
Stack of 4 previous frames
Convolutional layer of rectified linear units
Convolutional layer of rectified linear units
Fully-connected layer of rectified linear units

Input
State s

Output
Value $Q(s, a)$

Policy Gradient

- DQN represent value-based RL
- Another stream is policy-based RL
  - Under the policy π_θ which is specified by the policy parameter θ, the agent maximizes the total discounted rewards

$$\eta(\theta) = E\left[\sum_{t=0}^{T} \gamma^t \cdot r(s_t, a_t)\right]$$

  - Policy gradient methods then apply gradient ascent to maximize it

$$\theta_{new} = \theta_{old} + learning\ rate \times \nabla\eta(\theta)$$

Policy Gradient

A simple example

- Consider a robotic vacuum cleaner
  - reward is the amount of dust it picks up in 30 minutes.
  - Policy: move forward with some probability p every second, or randomly rotate left or right with probability 1 – p.
- PG: slightly increase p and evaluate whether this increases the amount of dust picked up by the robot in 30 minutes
  - if it does, then increase p some more.

Policy Gradient

- REINFORCE

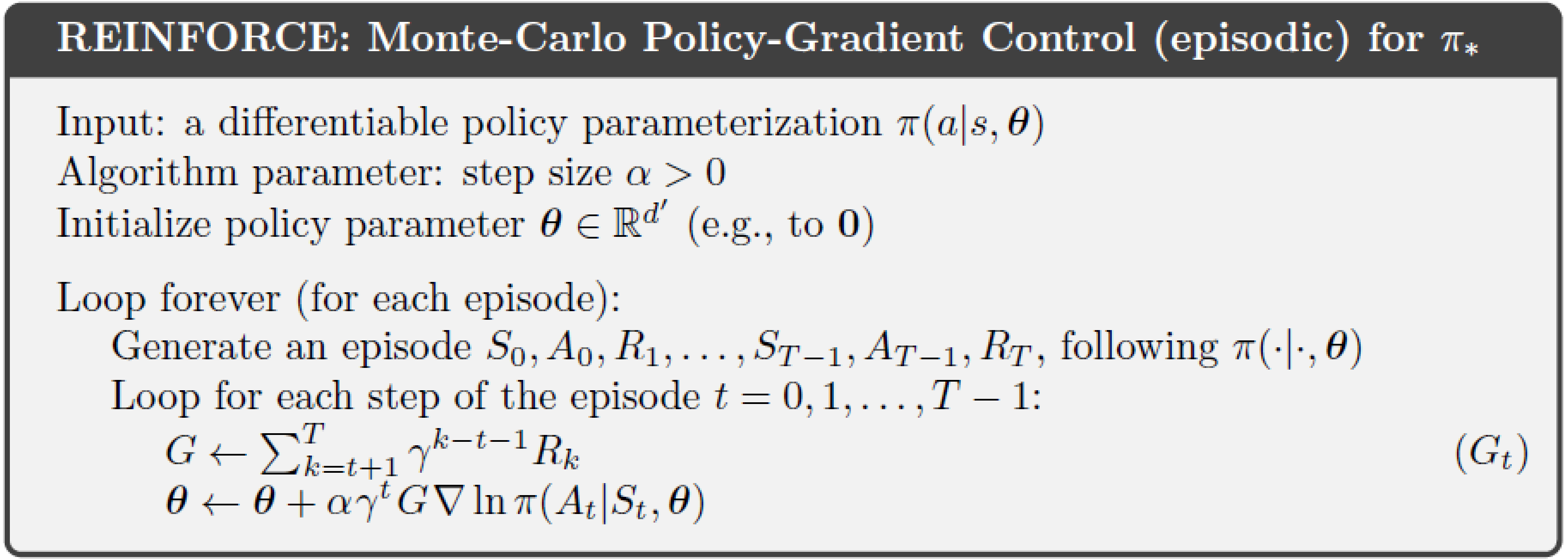
$$\nabla\eta(\theta) \propto \mathbb{E}_\pi\left[\sum_a q_\pi(S_t, a)\nabla\pi(a|S_t, \theta)\right]$$

$$\nabla\eta(\theta) \propto \mathbb{E}_\pi\left[\sum_a \pi(a|S_t, \theta)q_\pi(S_t, a)\frac{\nabla\pi(a|S_t, \theta)}{\pi(a|S_t, \theta)}\right]$$

$$\nabla\eta(\theta) \propto \mathbb{E}_\pi\left[q_\pi(S_t, A_t)\frac{\nabla\pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)}\right]$$

Policy Gradient

- REINFORCE



REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

```
Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                                        $(G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

Importance Sampling

- In Monte Carlo simulation, we often need to compute $E[f(X)]$

$$E[f(X)] = \int f(x)p(x)dx$$

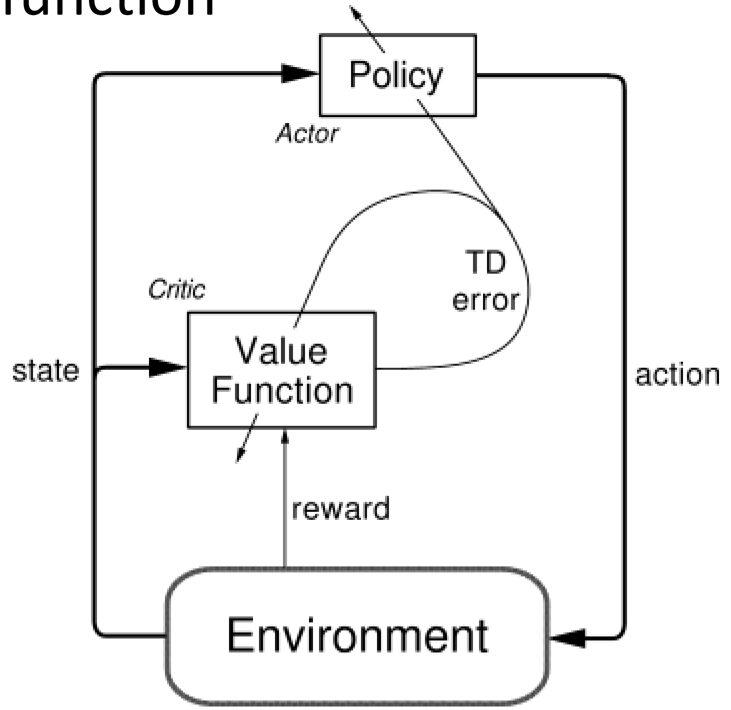
  - Challenge: sampling from p could be hard
- Solution: Importance sampling

$$E_p[f(X)] = \int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)} \cdot q(x)dx = E_q\left[\frac{f(x)p(x)}{q(x)}\right]$$

  - $p(x)$: *nominal* or *target* distribution
  - $q(x)$: *importance* or *proposal* distribution

Actor-Critic

- Like SGD, policy-based RL can have high variance
- The third mainstream method: *Actor-Critic*
  - *Actor* learns the policy
  - *Critic* learns the value function



Actor is used to select the action

Critic is used to criticize the actions made by the actor

Advantage

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right], \text{ where}$$
$$s_0 \sim \rho_0(s_0),\; a_t \sim \pi(a_t|s_t),\; s_{t+1} \sim P(s_{t+1}|s_t, a_t).$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right]$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right],$$
$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right],$$
$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s), \text{ where}$$
$$a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t) \text{ for } t \geq 0.$$

$$\begin{aligned}
\eta(\tilde{\pi}) &= \eta(\pi) + \sum_{t=0}^{\infty} \sum_s P(s_t = s|\tilde{\pi}) \sum_a \tilde{\pi}(a|s)\gamma^t A_\pi(s, a) \\
&= \eta(\pi) + \sum_s \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\tilde{\pi}) \sum_a \tilde{\pi}(a|s) A_\pi(s, a) \\
&= \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \qquad (2)
\end{aligned}$$