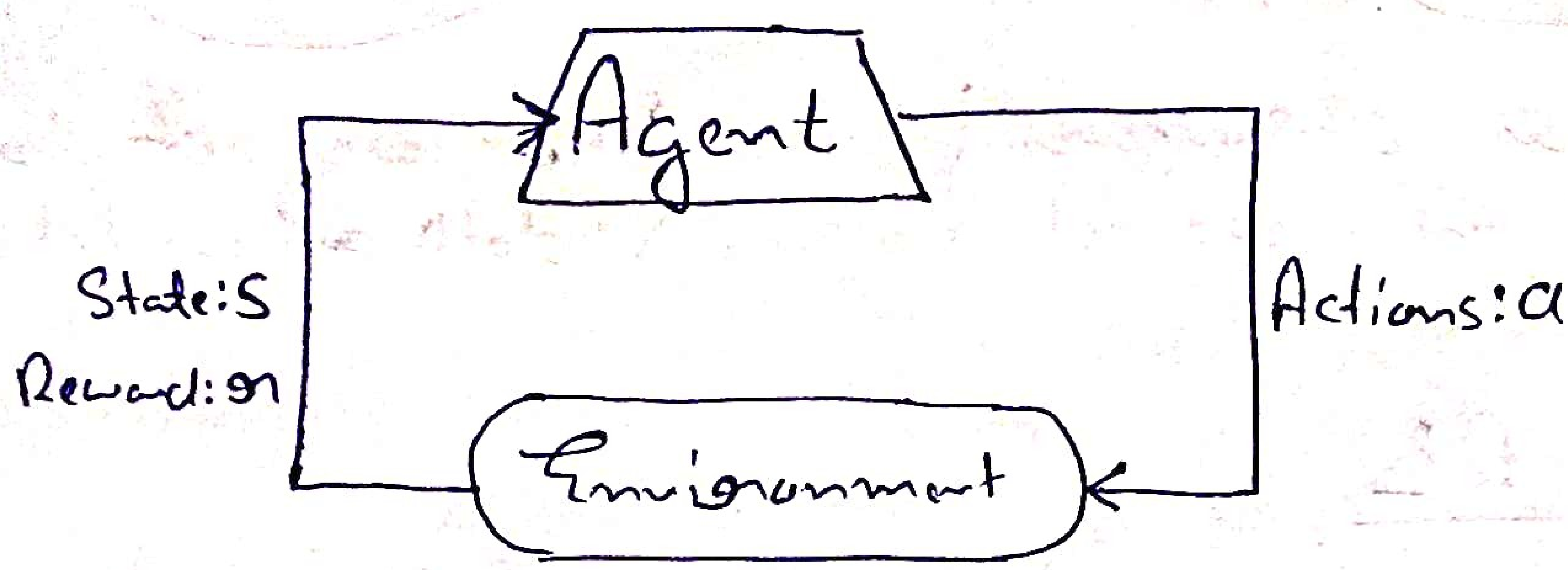
⑨

Reinforcement Learning (Part 1)



→ Receive feedback in form of rewards
→ Agent's utility is defined by the reward function.
→ Must (learn to) act so as to maximize expected rewards.

→ All learning is based on observed samples of outcomes!

⇒ In RL we don't know $T(s,a,s')$ & $R(s,a,s')$ ahead of time.
↳ agent is just allowed to interact with the world. (Observe state transitions & rewards obtained)

★ Model-Based Learning

Step 1: Learn empirical MDP model
→ Count outcomes s' for each s, a
→ Normalize to give an estimate of $\hat{T}(s,a,s')$
→ Discover each $\hat{R}(s,a,s')$ when we experience (s,a,s')

Step 2: Solve the learned MDP
{Value iteration, policy iteration etc...}

* Model-Free learning

Passive RL

{we just observe and based on it we act}

Active RL

{Here we act to collect data and we act based on it}

(#) Passive RL

⇒ Just execute a fixed policy and learn from experience.

⇒ This is NOT offline planning!

↳ You actually take actions in the world.

(a) Direct Evaluation

- Goal: Compute values for each state under π
- Idea: Average together observed sample values
  - ↳ Act according to π
  - ↳ Every time you visit a state, write down what the sum of discounted rewards turned out to be.
  - ↳ Average those samples.
- Problem:
  - ↳ It wastes information about state connections
  - ↳ Each state must be learned separately.
  - ↳ So, it takes long time to learn.

Temporal Difference Learning

Idea: Learn from every experience!

{Update V(s) each time we experience a transition (s, a, s', r)}

→ Initialize $V^{\pi}(s) \; \forall s \in S$ $\{\alpha = 0.1\}$

→ $\text{Sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

→ Update to $V^{\pi}(s) \leftarrow (1-\alpha) V^{\pi}(s) + (\alpha)\, \text{Sample}$

{Running avg, makes recent samples more important}

⇒ If we want to turn values into a (new) policy, we're stuck!

↳ Because we don't know the Q values.

$$\pi(s) = \arg\max_{a} Q(s,a)$$

$$Q(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$

⇒ **Idea:** Learn Q-Values not values.

↳ Make action selection model-free too!

Active RL

→ Learner makes choices!

→ Fundamental tradeoff: Exploration Vs Exploitation

@ Q-Value Iteration

- Start with $Q_0(s,a) = 0$
- Given Q_k, Calculate the depth $k+1$ q-values for all q-state.

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

⇒ Learn $Q(s,a)$ values as you go

- Receive a sample (s,a,s',r)
- Consider your old estimate $Q(s,a)$
- Consider your new sample estimate:

$$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

- Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + (\alpha)[\text{sample}]$$

⇒ Q-learning Converges to optimal policy -- even if you're acting suboptimally.

↳ This is called **off-policy learning**.