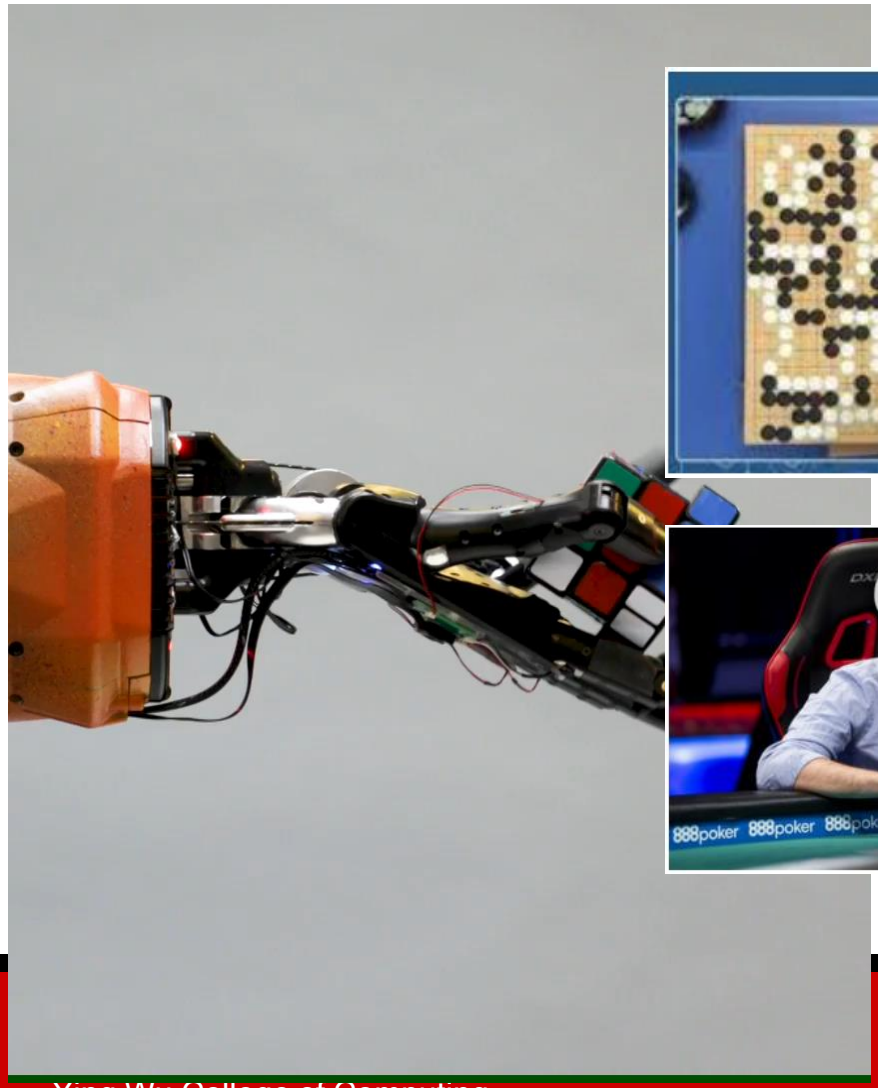
Reinforcement Learning

Presenter: Liang Zhang

Introduction

- Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to learn based on data, such as from sensor data or databases.



Different Types of Machine Learning

- Supervised learning

Reinforcement Learning

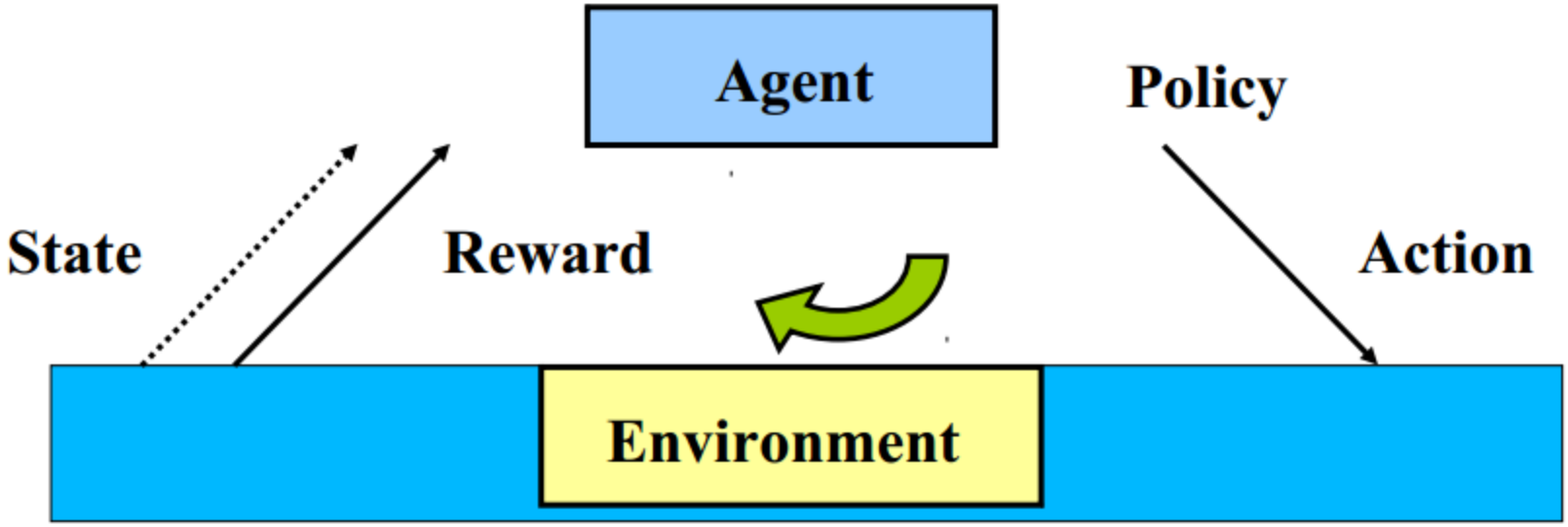
Step: 1

World: You are in state 9. Choose action A or C.

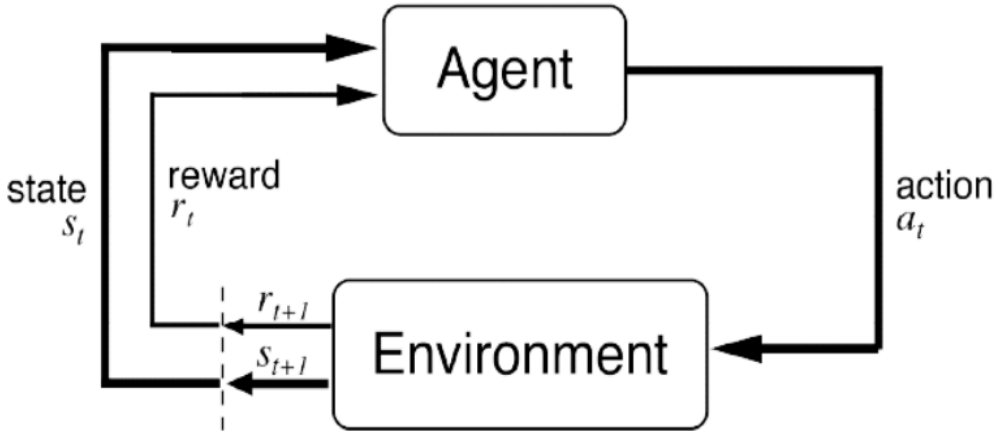
	Supervised	**Unsupervised**	**Reinforcement**
Data	(x, labels y)	x	(State s, action a, reward r)
Goal	Learn a mapping x->y	Learn structure of x	Maximize future reward
Example	Classification	Clustering	Game playing -- win the game

Elements of Reinforcement Learning

- Agent: Intelligent programs
- Environment: External condition
- Policy:
  1) Defines the agent's behavior at a given time
  2) A mapping from states to actions
  3) Lookup tables or simple function
- Reward function



RL Procedures



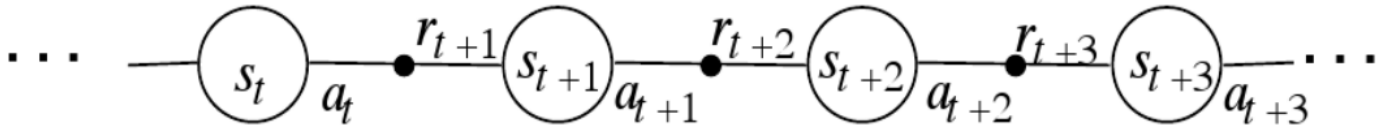
Agent and environment interact at discrete time steps : $t = 0, 1, 2, \ldots$

Agent observes state at step t: $s_t \in S$

produces action at step t: $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \Re$

and resulting next state : s_{t+1}



Markov Decision Process (MDP)

- The *environment* is in a *state*, $s_t \in \mathcal{S}$
- An *agent* takes an *action* $a_t \in \mathcal{A}$ according to a *policy* $a_t = \pi(s_t)$ (or $a_t \sim \pi(\cdot|s_t)$)
- For each (state, action), the environment gives the agent *reward* $r_t \in \mathbb{R}$, $r_{t+1} \sim \mathcal{R}(\cdot|s_t, a_t)$
- Then transitions to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$
- Can be finite time, or indefinite
- The goal is to find a policy to maximize expected discounted future reward:

$$\mathcal{G}_t = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right)$$

$0 \leq \gamma \leq 1$ is a *discount factor*.

A Case Study

IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 68, NO. 8, AUGUST 2019

Intelligent Trajectory Design in UAV-Aided Communications With Reinforcement Learning

Sixing Yin, Shuo Zhao, Yifei Zhao, and F. Richard Yu

[1] S. Yin, S. Zhao, Y. Zhao and F. R. Yu, "Intelligent Trajectory Design in UAV-Aided Communications With Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8227-8231, Aug. 2019.

Problem Statement

- The authors studied an uplink cellular network, where a UAV travelling in the air serves as an aerial base station for *K* ground users through frequency division multiplexing.
- The objective is to maximize the sum rate of all users.

$$\max_{\boldsymbol{q}} \quad J = \sum_{t=1}^{T} \sum_{k=1}^{K} \log\left(1 + \frac{p_k \gamma_0}{\sigma ||\boldsymbol{q}_t - \boldsymbol{w}_k||^{\beta}}\right), \tag{1}$$

$$\text{s.t.} \quad ||\boldsymbol{q}_t - \boldsymbol{q}_{t-1}|| \leq V, \forall t = 1, .., T,$$

- Since the UAV is unable to predict the exact uplink rate for all the ground users, one intuitive solution for its trajectory design is learning in a "try-and-error" way.

- The UAV may first try a few steps of movement, then make evaluation with the feedback to see whether the movement is a wise action. Such a learning process falls into a typical RL framework.

Solution

- An MDP M can be defined by five elements, $M = \{S, A, P, R, y\}$, where S is the state space, A is the action space, P is the state transition probability, R is the reward (also known as cost) at each step and y is the discount factor.
- The system state is the temporal difference in L timeslots, $s_t = \{\Delta e_t, \Delta e_{t-1}, \dots, \Delta e_{t-l+1}\}$. Here, $\Delta e_t = e_t - e_{t-1}$ and $e_t = \{e_{1,t}, e_{2,t}, \dots, e_{K,t}\}$ is the UAV's received signal strength on all the channels.
- Action: the UAV's action can be defined as its movement in each timeslot.
- Reward: the reward at each step can be well defined by the uplink sum rate in each timeslot, $R_t = \sum_{k=1}^{K} log[1 + p_k \gamma_0 / (\sigma ||q_t - w_k||^{\beta})]$. The state-action value function $Q^{\pi}(s_t, A_t)$ can be defined as the expected uplink sum rate with policy π at state s_t when action A_t is taken.
- State Transition Probability: $P(s_{t+1}|s_t, A_t)$ is defined as the probability distribution of the next state given the current state and action taken and characterizes dynamics of the whole system. The optimal policy that maximizes the expected uplink sum rate $E[\sum_{t=1}^{T} R_t]$ is learned by interacting with the environment in a try-and-error fashion without exact state transition probability.

An RL Algorithm

Algorithm 1: DPG for Trajectory Design.

Randomly initialize the parameters of main value and policy networks as well as the feature network, set $\boldsymbol{\theta}_{Q'} = \boldsymbol{\theta}_Q$ and $\boldsymbol{\theta}_{\mu'} = \boldsymbol{\theta}_\mu$;
for $m = 1, \ldots, M$ **do**
  Randomly generate ground user locations;
  Randomly choose an action for $t = \{1, \ldots, L\}$
  for $t = L+1, \ldots, T$ **do**
    Choose and execute action $\boldsymbol{A}_t = \mu(F(\boldsymbol{S}_t)) + N_t$ and store tuple $\{\boldsymbol{S}_t, \boldsymbol{A}_t, R_t, \boldsymbol{S}_{t+1}\}$ in the replay buffer;
    if The replay buffer size is sufficiently large **then**
      Sample a minibatch from the replay buffer;
      Update the parameters of feature network and main value network by minimizing (2);
      Update the parameters of feature network and main policy network based on (3);
      if m is a multiple of C **then**
        Set $\boldsymbol{\theta}_{Q'} = \boldsymbol{\theta}_Q$ and $\boldsymbol{\theta}_{\mu'} = \boldsymbol{\theta}_\mu$.
      end if
    end if
  end for
end for

F denotes the feature network;
Q and Q' denote main and target value networks;
μ and μ' denote main and target value networks;
θ denotes their parameters;
N_t is the additive normal-distributed noise;
N is the batch size.

$$L = \frac{1}{2N}\sum_{i=1}^{N}[y_i - Q(F(\boldsymbol{S}_i), \boldsymbol{A}_i)]^2, \tag{2}$$

$$\nabla_{\boldsymbol{\theta}_\mu} J = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}_\mu}\mu(F(\boldsymbol{S}_i))\nabla_{\boldsymbol{A}_i} Q(F(\boldsymbol{S}_i), \boldsymbol{A}_i), \tag{3}$$