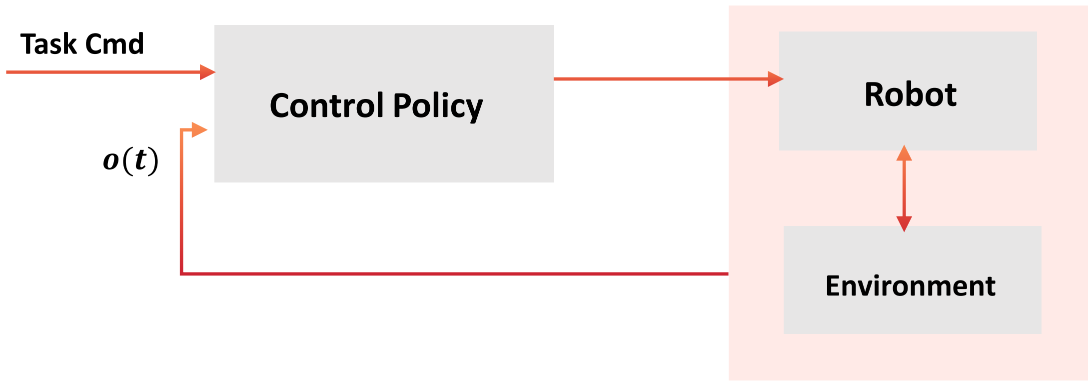
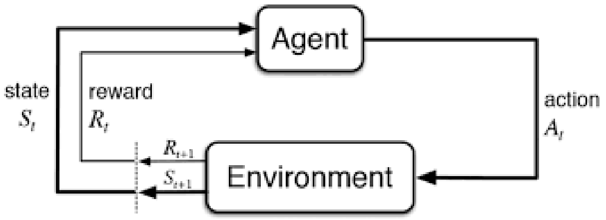
Advanced Control for Robotics (Fall 2024)

Lecture Note 10
Markov Decision Process for Reinforcement Learning

Prof. Wei Zhang
Southern University of Science and Technology

- **From Classical Control to RL**





- **Classical and Modern Control**
  - Require model (often analytical form)
  - Require analysis, only feasible for simple model
  - Hard to deal with uncertainty

- From Classical Control to RL

- From Classical Control to RL

  - Model Predictive Control

- From Classical Control to RL

  - Reinforcement Learning

▪ From Classical Control to RL

Classical control

Modern Control

MPC

Reinforcement Learning

- **Our Plan for Reinforcement Learning**
  - Markov decision problem
  - Value evaluation via sampling
  - Policy gradient theoretical foundation and derivation
  - Advanced PG:
    - Baseline
    - Actor-critique
    - PPO
  - Legged robot examples

- **This lecture**: general decision/control problem formulation of stochastic system
  - Markov chain
  - Markov decision process
  - Bellman equations
  - Simulations

Markov Chain: MC = $(\boldsymbol{S}, \boldsymbol{\Gamma})$

- S – state space (discrete or continuous)

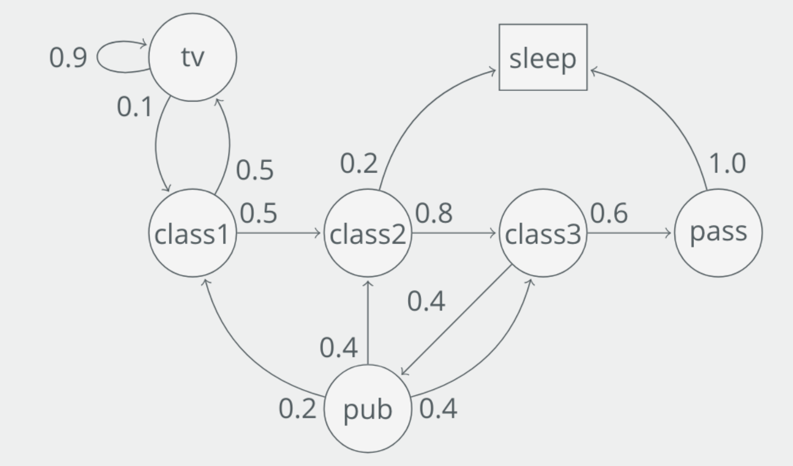
- Initial distribution $p_0(s) = \Pr(S_0 = s)$

- Γ - transition operator, i.e. $\Gamma(x|y) = \Pr(s_{t+1} = x|\, s_t = y)$

- For discrete state space, the transition operator has a matrix representation:

▪ Markov Chain Example

Example: Markov Chain



State Transition Matrix

$$\mathcal{P} = \begin{array}{cc} & \begin{array}{ccccccc} \text{c1} & \text{c2} & \text{c3} & \text{pass} & \text{pub} & \text{tv} & \text{sleep} \end{array} \\ \begin{array}{c} \text{c1} \\ \text{c2} \\ \text{c3} \\ \text{pass} \\ \text{pub} \\ \text{tv} \\ \text{sleep} \end{array} & \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1.0 \end{bmatrix} \end{array}$$

- $\boldsymbol{MC}(\boldsymbol{S},\boldsymbol{\Gamma})$ with P_0 specifies a way to generate sequential random samples $s_0, s_1, \dots$
  - $s_0, s_1, \dots$ is a called a realization/trajectory of the MC
  - Markov property: $\Pr(s_t | s_0, s_1, \dots, s_{t-1}) = \Pr(s_t | s_{t-1})$

- (Autonomous) Stochastic dynamical system:
    - $s_{k+1} = f(s_k, w_k)$ or $s_{k+1} = f(s_k) + w_k$
    - where w_k is a random variable (process noise)
  - The above stochastic dynamical system is a Markov chain

- MC is a more general (less explicit) way to write a stochastic dynamic system

- **Markov Decision Process:**
  - MDP = $(S, \mathcal{A}, \Gamma, \mathrm{r})$
  - S – state space (discrete or continuous)
  - $\mathcal{A}$- action/control space (discrete or continuous)
  - Γ - Transition kernel/operator
    - $\Gamma(s'|s,a) = Pr(S_{t+1} = s'|S_t = s, A_t = a) = p(s'|s,a)$
  - r - reward function: $r(s,a,s')$ or typically $r(s,a)$

- **Policy**
  - Markov decision: agent makes decision based on the current state
  - $\pi(a|s)$: is the pdf/pmf of action a given the current state s, i.e., $\pi(a|s) = \Pr(A = a|S = s)$
  - Deterministic policy $a = \pi(s)$
    - is a deterministic function of state
  - The policy can also be time varying in general, $\pi_t(a|s)$ or $\pi(a|s, t)$
  - $\pi_\theta(a|s)$ - represent a policy within a class of functions with certain parameter θ

- **Trajectories** of MDP
  - Given a policy π, a finite horizon T
  - MDP becomes a MC with "closed-loop" transition operator Γ_{cl}

  - Trajectory $\tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$ is a trajectory of MDP under a policy π

  - **Probability of a trajectory**: $P(\tau|\pi) = P(s_0, a_0, \dots, s_T, a_T|\pi) = \Pr(S_0 = s_0, A_0 = a_0, \dots, S_T = s_T, A_T = a_T|\pi)$
    - We will write as $P(\tau)$ whenever the underlying policy π is clear from the context

- **Some notations and facts:**

- **Some notations and facts:**

- **Return:** Cumulative rewards over a trajectory, which may take several different forms.

  - Finite-horizon (undiscounted) return:

  - Infinite-horizon discounted return:

  - MDP (RL) Problem: $\max_{\pi} E_{\tau\sim\pi}[R(\tau)]$

  - In deep RL, researchers often mix the two types of returns within one problem, despite the two are significantly different. (e.g. one may set up algorithms to optimize the undiscounted return, but use discount factors in estimating **value functions**).

- **Value functions:**
  - On-policy (state)-value function:
  - On-policy action-value function (Q-function):
  - Optimal value function:
  - Optimal action-value function:

- **Bellman Equations:**
  - Let’s focus on the infinite-horizon discounted return case

- **Bellman equations:**

- **Bellman equations**

- **Value function and Bellman equations summary:**

- $V_\pi(s) = E_{a\sim\pi}[Q_\pi(s,a)]$

- $V^*(s) = \max_a Q^*(s,a)$

- $V_\pi(s) = E_{a\sim\pi}\left[r(s,a) + \alpha\ E_{s'\sim p(|s,a)}[V_\pi(s')]\right]$

- $Q_\pi(s,a) = E_{s'\sim p}\left[r(s,a) + \alpha\ E_{a'\sim\pi}[Q_\pi(s',a')]\ \right]$

- $V^*(s) = \max_a E_{s'\sim p}[r(s,a) + \alpha V^*(s')]$

- $Q^*(s,a) = E_{s'\sim p}\left[r(s,a) + \alpha \max_{a'}[Q_\pi(s',a')]\right]$

- **Regarding Optimal policies:**
  - Directly from the Bellman equation, we have

$$\pi^*(s) = \mathrm{argmax_a}\left\{r(s,a) + \gamma \sum_{s'} p(s'|a,s)V^*(s')\right\}$$

- Optimal deterministic policy:
  - In finite, **fully observable MDPs** with typical reward maximization criteria, there is always at least one optimal deterministic policy.

- Stochastic policies can be optimal or necessary:
  1. **POMDP**: **Belief States:** The agent maintains a probability distribution over possible states (belief state), stochastic policy may be necessary because the same observation may correspond to different actual states, requiring a randomized strategy to maximize expected rewards

2. Risk Sensitivity and Exploration Objectives: May necessitate stochastic actions as hedge against high variability and uncertain outcomes.

3. Game-Theoretic Settings: Adversarial environments can require mixed (stochastic) strategies.

- In RL Practice:
  - Learning Phase:
    - Stochastic Policies: Useful for exploration and learning the optimal policy.
    - Adaptability: Allows the agent to improve its understanding of the environment.
  - Execution Phase:
    - Deterministic Optimal Policy: Once the environment is well-understood, the agent may employ a deterministic policy for optimal performance.

Sampling

- Random sampling can be used to simulate a MC or MDP

- Random sampling is also used to evaluate high-dim expectations (or integrations) involved in reinforcement learning of MDP

- How to draw a random sample from a given pdf or pmf?
  - Many different ways to draw random samples: such as inverse transform sampling

- Customized sampling can be very slow. Try to use python built-in sampling functions

- Sampling in Python
  - numpy.random
    - rand(d0, d1, ..., dn)   Random uniform values in a given shape.
    - randn(d0, d1, ..., dn) Return a sample (or samples) from the “standard normal” distribution.
    - choice(a[, size, replace, p]) Generates a random sample from a given 1-D array
  - Custom distributions: scipy.stats
    - rv_continuous
    - rv_discrete
    - rv_histogram
  - Many more…

Monte Carlo Method

- a class of simulation-based methods that seek to avoid complicated/ intractable mathematical computations (e.g. integration)
- Consider $X_1, X_2, \dots, X_n$ i.i.d. random vectors
- $E(X_i) = \mu_X \,, Cov(X_i) = Q_X$
- Sample mean: $\bar{X}_{\boldsymbol{n}} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum \boldsymbol{X_i} = \frac{\boldsymbol{X_1 + X_2 \cdots + X_n}}{\boldsymbol{n}}$
- Sample covariance: $\bar{Q}_n = \frac{1}{n-1} \sum_i (X_i - \bar{X}_n)(X_i - \bar{X}_n)$
  - The use of $n-1$ instead of n is called Bessel's correction

- They are unbiased estimate, i.e.

$$E(\bar{X}_n) = \mu_X, \text{ and } E(\bar{Q}_n) = Q_X$$

- Effect of Bessel's correction becomes less significant as $n \to \infty$, we can also use the uncorrected empirical covariance $\widetilde{\mathrm{Q}}_n = \frac{1}{n}\sum_i (X_i - \bar{X}_n)(X_i - \bar{X}_n)$

- Recall (strong) law of large number:

$$\bar{X}_n \to \mu_X = E(X), \text{ a.s.}$$

- **Central limit theorem (CLT):**

$$\sqrt{n}(\bar{X}_n - \mu_X) \to \mathcal{N}(0, Q_X) \text{ in distribution}$$

  - In other words, $\bar{X}_n$ can be well approximated by Gaussian distribution $\mathcal{N}\left(\mu_X, \frac{Q_x}{n}\right)$
  - Covariance: $E[(\bar{X}_n - \mu_X)(\bar{X}_n - \mu_X)^T] \approx \frac{Q_X}{n}$
  - MSE of the estimate $\bar{X}_n$ is $trace\left(\frac{Q_X}{n}\right)$

- **Monte Carlo Integration:**
  - $E\left(\phi(X)\right)$ can be estimated by

$$\frac{1}{n}\sum_{i}\phi(X_i) \approx E\left(\phi(X)\right)$$

where $X_i \sim f_X(x)$ are iid samples

- **Importance sampling**
  - $E_f(X) = \sum_x x f(x)$ → sample mean estimate:
  - Suppose we want to estimate $E_g(X) = \sum_x x g(x)$, but we can only sample from $f(x)$ distribution

- More Discussions