Johns Hopkins
Whiting School of Engineering

Algorithms for Data Science

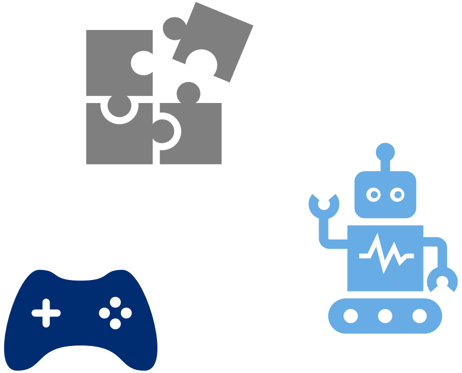
Optimization: Markov Decision Processes

Markov Decision Processes

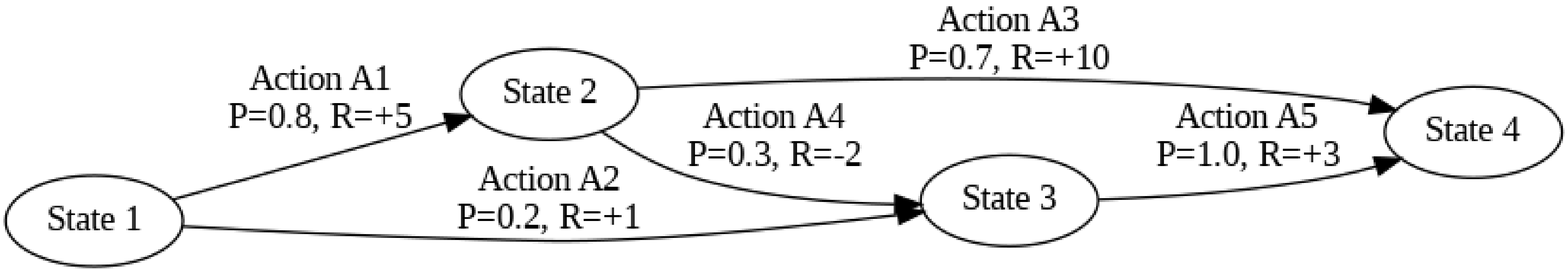
A Markov Decision Process (MDP) is a mathematical framework for modeling sequential decision-making under uncertainty.

Components

- States (S): All configurations of the system.
- Actions (A): Available choices in each state.
- Transition Probabilities $P\left(s' \mid s, a\right)$
  Probability of transitioning to a new state *s'* given current state *s* and action *a*.
- Rewards $R(s, a)$ Immediate reward after taking action *a* in state *s*.
- Policy $\pi(a \mid s)$ Strategy for choosing actions.



Flow of States & Actions in MDP



Value Functions

- State Value Function: ($V(s)$): Expected cumulative reward starting from state *s* and following policy π

$$V(s)=\mathrm{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R\left(s_t,a_t\right)\mid s_0=s\right]$$

- Action Value Function ($Q(s,a)$): Expected cumulative reward starting from State *s*, taking action *a*, and following policy π

$$Q(s,a)=R(s,a)+\gamma\sum_{s'}P\left(s'\mid s,a\right)V\left(s'\right)$$

Bellman Equations (Recursive definitions of value functions)

- State Value Function:

$$V(s)=\max_a\left[R(s,a)+\gamma\sum_{s'}P\left(s'\mid s,a\right)V\left(s'\right)\right]$$

- Action Value Function:

$$Q(s,a)=R(s,a)+\gamma\sum_{s'}P\left(s'\mid s,a\right)\max_{a'}Q\left(s',a'\right)$$

- γ: Discount factor, balancing the trade-off between immediate and future rewards $(0\leq\gamma\leq 1)$

Solving MDPs with DP

1. **Policy Evaluation:** Compute the value function $V^{\pi}(s)$ for a given policy π with the recursive update rule:

$$V^{\pi}(s) = \sum_{s} \pi(a \mid s) \left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) V^{\pi}\left(s'\right) \right]$$

2. **Policy Improvement:** Improve the policy by selecting actions that maximize the value:

$$\pi'(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) V^{\pi}\left(s'\right)\right] \\ 0 & \text{otherwise} \end{cases}$$

3. **Policy Iteration:** Alternate between policy evaluation and policy improvement until the policy converges.
4. **Value Iteration:** Iteratively compute the optimal value function $V^*(s)$ using:

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) V^*\left(s'\right) \right]$$

   - Derive the optimal policy from

$$\pi^*(a \mid s) = \begin{cases} 1, & \text{if } a = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) V^*\left(s'\right)\right] \\ 0, & \text{otherwise} \end{cases}$$

MDPs: Foundation for Reinforcement Learning

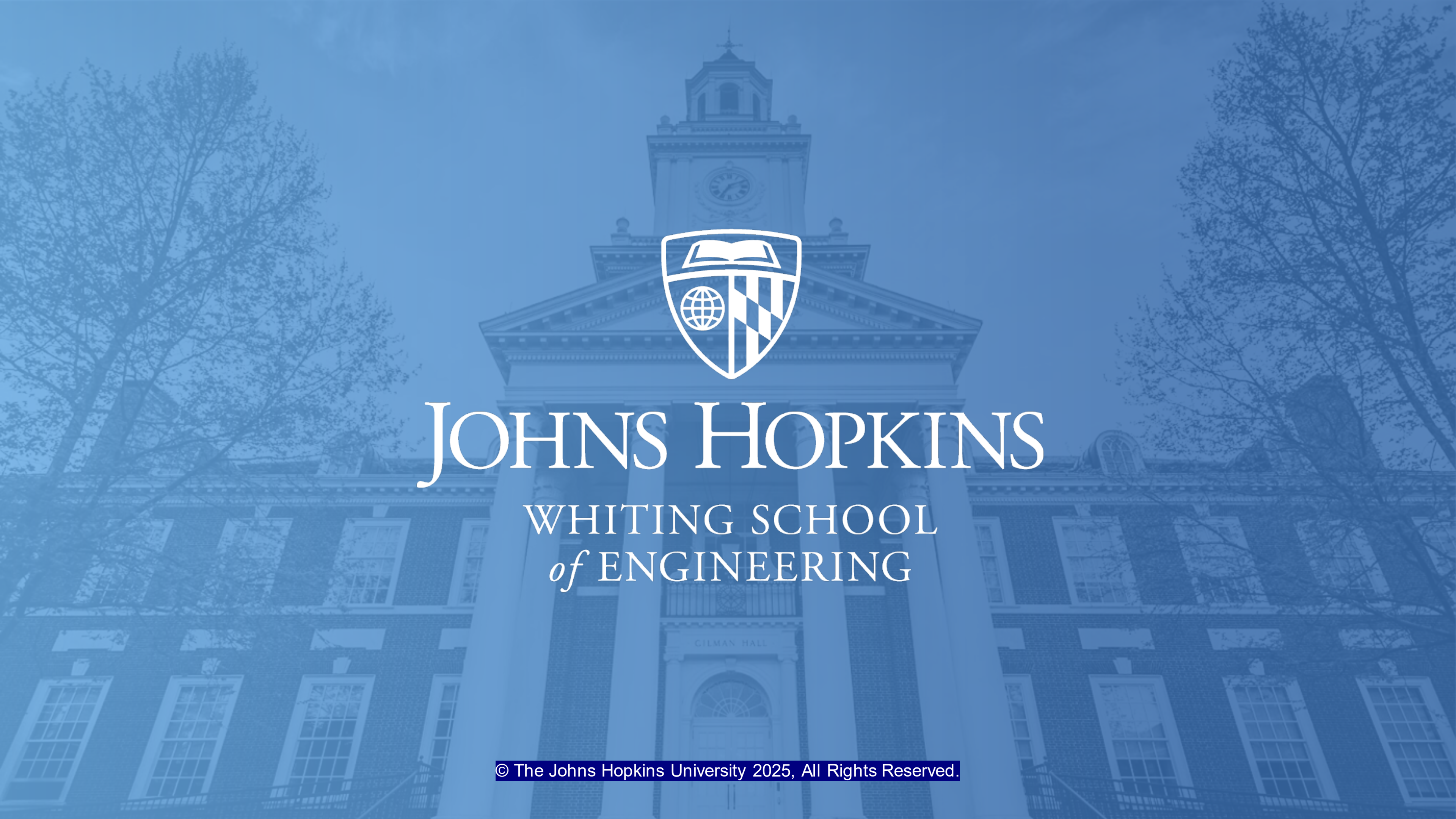
Reinforcement learning (RL) builds on the framework of MDPs for decision-making where MDPs provide the mathematical foundation for defining states, actions, rewards, and policies.

Transition Probabilities

- MDPs assume $P\left(s' \mid s, a\right)$ are fully known.
- RL assumes $P\left(s' \mid s, a\right)$ is unknown and must be learned.

Learning Through Exploration

- RL agents interact with the environment to estimate value functions and policies.
- Balance between exploration and exploitation.


JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING