PADERBORN UNIVERSITY

Data Science Engineering

Markov Decision Processes (MDPs)

Sanjay Gupta

Supervisor: Hans Harder

Why MDPs for RL Problems?[1]

- Provides a mathematical *framework* for modelling decision making in situations, where results are partly random or control.
- Captures the Markov *property*.

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$$

- Agents can learn to *act* in complex and dynamic environments. Example Autonomous Driving, Video games.

1. Sutton & Barto, MIT Press, 2018.

Introduction to MDPs[1]

- **RL-Problems:** Learning *how* agent make decision in complex and uncertain environments?

- **Definition:** A MDP is a *tuple* $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}>$,
  - $\mathcal{S}$ is a set of *states*.
  - $\mathcal{A}$ is a set of *actions*.
  - $\mathcal{P}$ is a *state transition probalities*,
    $$\mathcal{P}(s'|s,a) = \mathbb{P}\{S_t = s' | S_{t-1} = s, A_{t-1} = a\}$$
  - $\mathcal{R}$ is a *reward function*,
    $$\mathcal{R}(s,a) = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a]$$
  - Discount factor,
    $$\gamma \in [0,1]$$

- **Goal:** *Find* a policy that maximizes the expected cumulative *reward* over *time*.

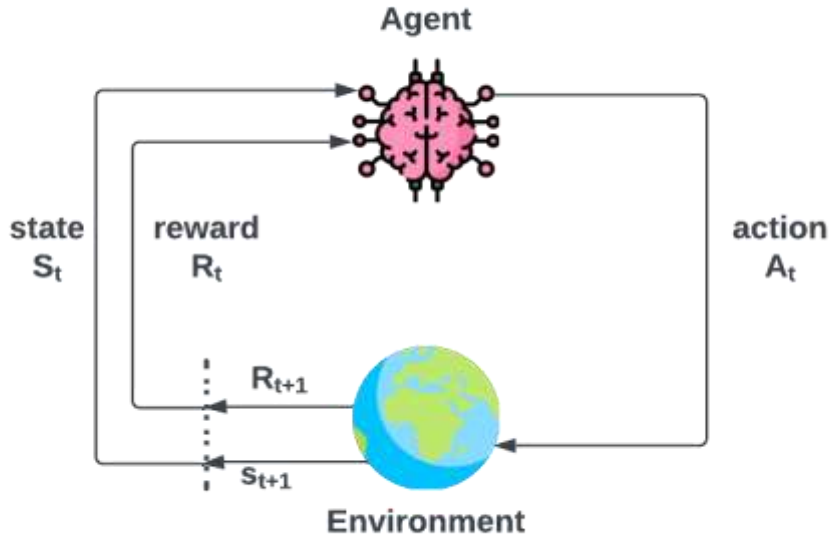

Figure 1: The agent - environment interaction in an MDP.

1. Sutton & Barto, MIT Press, 2018.

Example

- **Goal:** Explore strategies to optimise study decision-making process.
- **Benefits:** Learn an optimal policy that balances academic responsibilities, social interactions, and personal well-being.
- Sample episodes for Student MDPs:
  - **C1, C2, Pass, Sleep.**
  - **C1, Instagram, C1, C2, Pass, Sleep.**
  - **C1, Instagram, C1, C2, Café, C2, Pass, Sleep.**

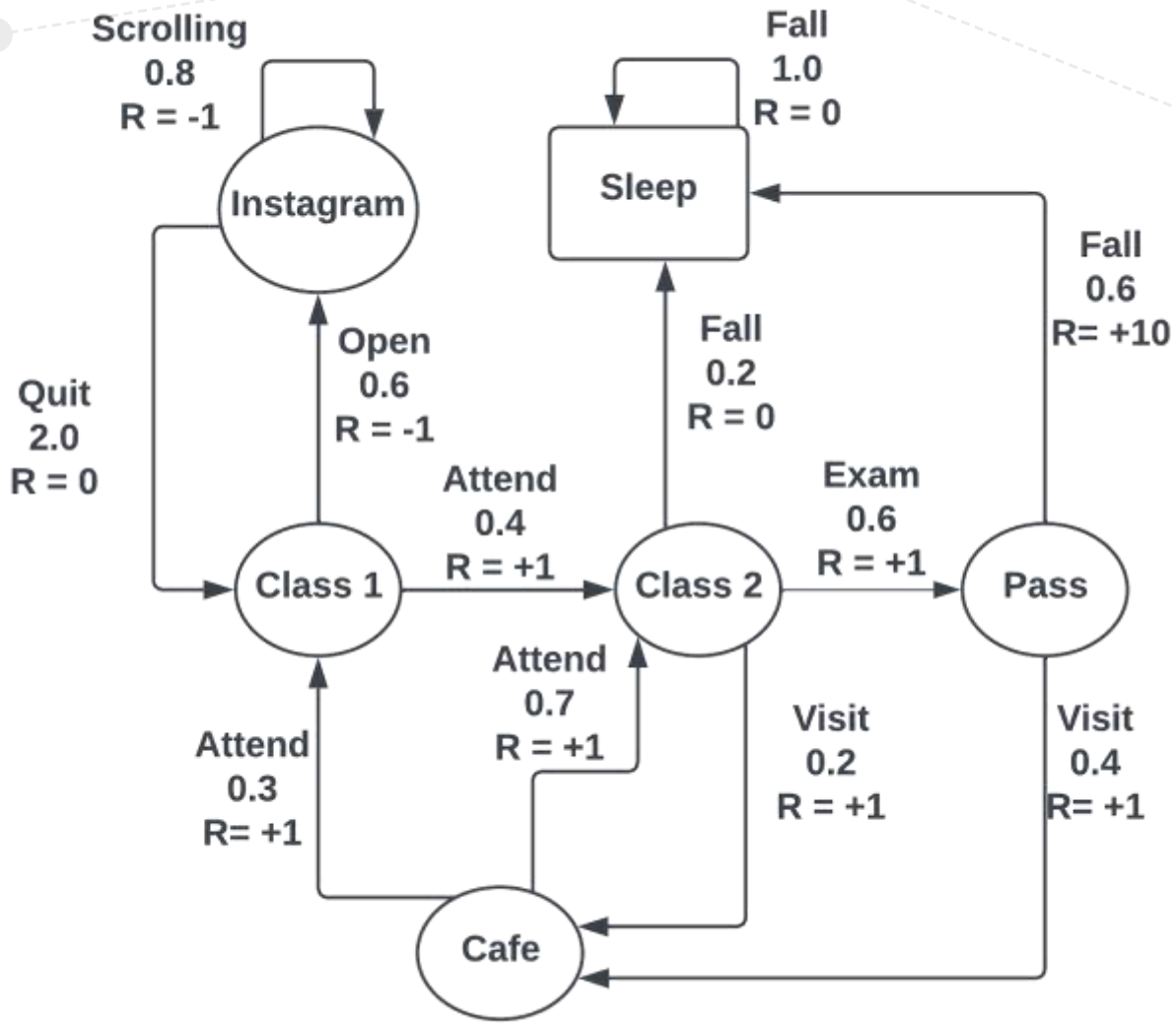

Figure 2: Paderborn University Student MDP.

Finite MDPs[1]

- Both the state and action spaces are finite and discrete. $s \in S$ and $a \in A(s)$
- The dynamics of the environment,

$$p(s', r|s, a) = \mathbb{P}\{S_t = s', R_t = r \,|S_{t-1} = S, A_{t-1} = a\}$$

- **Policy** π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\{A_t = a \,|S_t = s\}$$

- **Value functions** v_π capture the *expected cumulative rewards* an agent can achieve under a given policy π.
- **State-value function** $v_\pi(s)$,

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

- **Action-value function** $q_\pi(s, a)$,

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

- Optimal policy π_*, is a policy that maximizes the expected cumulative reward over the long run in an MDPs.

1. Sutton & Barto, MIT Press, 2018.

Solving Finite MDPs[1]

- **Policy Evaluation:**
  - Process of determining the value function for a given policy.
  - Involves estimating the expected cumulative rewards starting from each state under the given policy.

  - **Bellman Equation:**
    - Provide recursive relationship between the value functions, immediate rewards and future values.
    - Allow agent to learn optimal policies.
    - Equation,

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p\,(s', r|s, a)\,[r + \gamma\, v_\pi(s')]$$

    - State-value function,

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma\, v_\pi(S_{t+1})|S_t = s]$$

    - Action-value function,

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma\, v_\pi(S_{t+1})|S_t = s, A_t = a]$$

1. Sutton & Barto, MIT Press, 2018.

Solving Finite MDPs[1] continue..

- **Iterative Methods:**
  - Commonly use for solving Bellman Equation.
  - Estimate the value functions.
  - **Iterative policy evalution** method, recursively updates the value function estimates untill convergence.
  - The iterative process steps,
    1. Initialize the value function $V(s)$ arbitrarily for all states s.
    2. Iterate untill convergence:

$$V(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

    3. Repeat Step 2 untill the value function estimates no longer change significantly.
  - Converge to the true value function for the policy.

1. Sutton & Barto, MIT Press, 2018.

Solving Finite MDPs[1] continue..

- **Value iteration algorithm:**
  - Combines the ideas of policy evalution and improvement.
  - The value iteration algorithm steps,
    1. Initialize the value function $V(s)$ arbitrarily for all states s.
    2. Repeat untill convergence:

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

    3. Repeat Step 2 untill the value function estimates no longer change significantly.
  - After converges, using resulting value function to derive optimal policy.

1. Sutton & Barto, MIT Press, 2018.

Solving Finite MDPs[1] continue..

- **Policy Improvement:**
  - Process of finding optimal policy in MDPs.
  - Involves modifying the current policy to improve its expectecd cumulative reward by making greedy choices.
  - Policy improvement steps,
    1. Given a policy π, evalute ist value function $v_\pi(s)$ for all states using policy evaluation techniques.
    2. For each state s, update the policy,

$$\pi'(s) = \max_a v_\pi(s, a)$$

- It ensures that updated policy π' is at least as good as the current policy π.

1. Sutton & Barto, MIT Press, 2018.

Infinite MDPs[2]

- Both the state and action space are infinite or unbounded.

$$S = \{s_1, s_2, s_3, \dots.\} \quad \text{Where } s \in \mathbb{R}^n$$
$$A = \{a_1, a_2, a_3, \dots.\} \quad \text{Where } a \in \mathbb{R}^m$$

- Probability Density Function (PDFs) described the transition probabilities between states.

- The transition probability $p(s'|s,a)$ as a PDFs,

$$p(s'|s,a) = f(s'|s,a)$$

  where $f(s'|s,a)$ is PDFs.

- **Challenges:**
  1. Exact calculation of value functions becomes impractical.
  2. Might not converge to a unique solution.
  3. Computationally expensive.
  4. Difficult to determine a well-defined optimal policy.

Approximation techniques for infinite MDPs[1]

- Allow us to handle infinite MDPs by reducing the computational complexity and enabling efficient learning.
- Make RL problems feasible in real-world scenarios.
- **Value function approximation:**
  - Estimate the value function of MDPs using a parameterized function.
  - It approximates the values using set of parameters and function approximations.
  - **Linear function approximation**
    - Represents the value function using a linear combination of features.
    - Features derived from the state or state-action pairs.
    - Bellman equation is used to update the weights.
    - Adjusting the weights to minimize the prediction error.
    - May not capture complex non-linear relationships between states and actions effectively.

1. Sutton & Barto, MIT Press, 2018.

Approximation techniques for infinite MDPs[1] continue..

➢ **Neural Networks (NNs):**

- State or state-action pair as input.
- Estimate of the value function as output.
- Use techniques like stochastic gradient descent for process optimization by adjusting the weights and biases.
- Bellman equation guides the training process.
- NNs genralize across the states.
- Netowks aims to minimize the difference between the predicted values and the Bellman equations target values.
- It capture the non-linear realtionships between states and values.

1. Sutton & Barto, MIT Press, 2018.

Approximation techniques for infinite MDPs[1] continue..

- **Policy Approximation:**
  - It use parameterized function to approximate the policy function.
  - Uses a function approximator to estimate the policy based on the observed states and actions.
  - **Policy Gradient Methods:**
    - Use gradient-based optimization methods to directly optimize the Policy function.
    - Search for the optimal policy by iteratively updating the policy parameters.
    - Estimate the gradient of the expected cumulative reward with respect to policy parameters.
    - Then perform gradient ascent to maximize the reward.

1. Sutton & Barto, MIT Press, 2018.

Approximation techniques for infinite MDPs[1] continue..

➢ **Monte Carlo Tree Search (MCTS):**

- Tree-based search algorithms.
- Four main steps: Selection, Expansion, Simulation, and Back propagation.
- Steps are performed in a tree structure that grows as algorithm explores the state-action space.
- Uses Upper Confidence Bound (UCB) strategies in selection step.
- Simulates the transition to the next state after selecting an action.
- The result of simulation are backpropagated up to tree to update the value estimates of the function.
- MCTS builds a search tree that represents the quality of different actions and states.

1. Sutton & Barto, MIT Press, 2018.

Comparative Analysis

Characteristics	Finite MDPs	Infinite MDPs
The complexity of the environment	Small and discrete state and action space	Infinite or Unbounded state or action space
Convergence	Optimal policy	Pose challenges
Resource Requirements	Computationally efficient	Computational complexity
Trade-offs in Modeling	Concise and interpretable	Complex
Applications	Grid-world navigation, Puzzle-solving, Board games	Autonomous driving, Robotic control, Economic decision-making

Table 1: Comparative analysis of finite MDPs and Infinite MDPs

Summary

- MDPs provide a fundamental framework for modelling and solving RL problems.
- Concepts and implications of MDPs in the context of RL, with a focus on finite and infinite MDPs.
- The Bellman equation provides a recursive relationship between the value function and the immediate rewards.
- Study different approximation techniques to approximate the infinite MDPs.
- The comparative analysis between finite and infinite MDPs.

References

1. **Sutton & Barto, MIT Press, 2018.** Reinforcement Learning: An Introduction.

2. **Frédérick Garcia and Emmanuel Rachelson.** Markov Decision Processes, chapter 1, pages 1–38. John Wiley and Sons, Ltd, 2013.