15-780 – Graduate Artificial Intelligence: Markov Decision Processes

Aditi Raghunathan
Carnegie Mellon University

Search (recap)

- Want to reach goal from start state

- Each action takes us to a **deterministic successor** state

- Various algorithms
    - Uninformed search: DFS, BFS, iterative deepening, uniform cost search
    - Informed search: A*, heuristics, relaxations

Decision making under uncertainty

Real-world is not deterministic, there is **randomness** or **uncertainty**

Markov Decision Processes (MDPs) and their extensions provide a general way to think about how we can act optimally under uncertainty

MDPs introduced in 1950s – 60s

Applications: Robotics, self-driving cars, video games, robot soccer, scheduling and many more..

Example MDP

Example: dice game

For each round $r = 1, 2, \ldots$

- You choose stay or quit.
- If quit, you get $\$10$ and we end the game.
- If stay, you get $\$4$ and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.

What policy should you follow?

MDP intuition

States encode all the information of a system needed to determine how it will evolve when taking actions

The system is governed via *transition probabilities*: they only depend on the current state and action; not on the history

Agent starts at a start state, and the goal is to take actions that maximize *expected reward*

MDPs formal definition

States: $s \in S$ assumed to be discrete

Actions: a $\in \mathcal{A}$ (assumed to be discrete)

Transition probabilities: distribution over next states given current state and current action

$T(s, a, s')$ is probability of next state being s' when taking action a at state s

Rewards: mapping states or transitions to reals $R(s, a, s')$

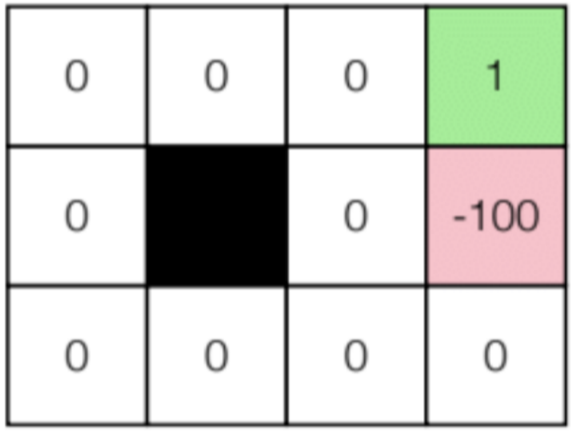
Start state, end state, discount factor γ(default 1)

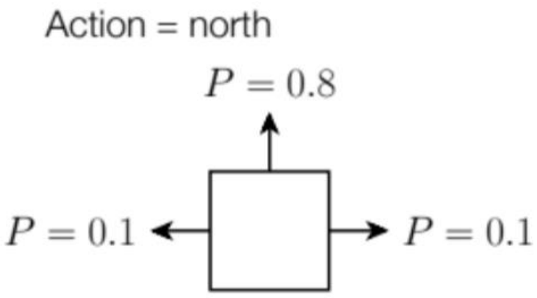
Gridworld domain

Simple grid world with a goal state with reward and a “bad state”with reward -100

Actions move in desired direction with probability 0.8 and one of two perpendicular directions with probability 0.1 each

Taking an action that bumps into a wall leaves agent where it is





Policy and policy evaluation

A **policy** π is a mapping from each state s $\in S$ to an action a $\in \mathcal{A}$ (or distribution of actions)

Suppose you followed a path $s_0, a_1, r_1, s_1, a_2, r_2, s_2, a_2, r_2, s_3$ ...; the expected sum of discounted rewards is $r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$

Value function $V_\pi : S \rightarrow R$ such that $V_\pi(s)$ gives the expected sum of discounted rewards when following policy π from state s

Q-value of a policy $Q_\pi : S \times A \rightarrow R$ such that $Q_\pi(s, a)$ is the expected sum of discounted rewards when taking action a from state s and then following π

Recursive definitions

$V_\pi(s) = Q_\pi(s, \pi(s))$ otherwise *(= 0 if s is the end-state)*

$Q_\pi(s,a) = \sum_{s'} T(s,a,s')\ [\, R(s,a,s') + \gamma V_\pi(s')\,]$

Algorithm for policy evaluation: start with arbitrary initialization $V_\pi(s) = 0\ \forall s$ and then apply the recursion

Example MDP

Example: dice game

For each round $r = 1, 2, \ldots$

- You choose stay or quit.
- If quit, you get $\$10$ and we end the game.
- If stay, you get $\$4$ and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.

Evaluate the “always stay” policy

What is $V_\pi(stay)$*?*

Dice game

For *always stay* policy π

$$V_\pi(\text{end}) = 0$$

$$V_\pi(\text{stay}) = \frac{1}{3}\big(4 + V_\pi(\text{end})\big) + \frac{2}{3}(4 + \ V_\pi(\text{stay}))$$

Solve the recurrence

$$V_\pi(\text{stay}) = 12$$