Week 8 T7

Thursday, 8 September 2022 2:12 PM

Discount - Reward Markov Decision Process

Markov Decision Processes

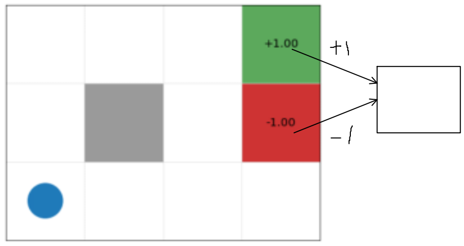
Definition – Markov Decision Process

A **Markov Decision Processes** (MDP) is a *fully observable, probabilistic* state model. The most common formulation of MDPs is a **Discounted-Reward Markov Decision Process**. A discount-reward MDP is a tuple $(S, s_0, A, P, r, \gamma)$ containing:

- a state space S
- initial state $s_0 \in S$
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- **transition probabilities** $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$
- **rewards** $r(s, a, s')$ positive or negative of transitioning from state s to state s' using action a
- a **discount factor** $0 \leq \gamma < 1$

What is different between an MDP and the models from classical planning? There are four main differences:

- The transition function is not deterministic. Each action has a probability of $P_a(s'|s)$ of ending in state s' if a is executed in the state s, whereas in classical planning, the outcome of each action is known in advance.
- There are no goal states. Each action receives a reward when applied. The value of the reward is dependent on the state in which it is applied.
- There are no action costs. Actions costs are modelled as negative rewards.
- We have a *discount factor*.



But! Things can go wrong — sometimes the effects of the actions are not what we want:

- If the agent tries to move north, 80% of the time, this works as planned (provided the wall is not in the way)
- 10% of the time, trying to move north takes the agent west (provided the wall is not in the way);
- 10% of the time, trying to move north takes the agent east (provided the wall is not in the way)
- If the wall is in the way of the cell that would have been taken, the agent stays in the current cell.

discount factor $\gamma \in (0,1)$

- Reduces value of rewards based on how far away they are.

MDPs take place in discrete time steps

r_t : reward at time t

Discounted reward value $= \gamma^t \times r^t$

V_t : value of current state = reward for current state + sum of all future discounted reward values

$V_t = r_t + \gamma \times r_{t+1} + \gamma^2 \times r_{t+2} + \gamma^3 \times r_{t+3} + \ldots$

$\equiv V_t = r_t + \gamma \times V_{t+1}$

e.g. receive \$100 today and every year for the next 4 years (5 total payments) with discount factor $\gamma = 0.9$

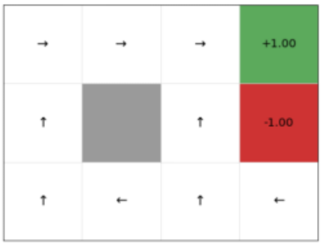
r_t	100	100	100	100	100
Timeline	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$

$$V_0 = 100 + 0.9 \times 100 + 0.9^2 \times 100 + 0.9^3 \times 100 + 0.9^4 \times 100$$
$$= 100 \times [1 + 0.9 + 0.9^2 + 0.9^3 + 0.9^4]$$
$$= 100 \times 3.439 = \$343.9$$

Solution to MDP : A policy π

Tells you which action to take in the current state

AKA Deterministic policy (vs stochastic policy)



How to come up with optimal policy?

Choose policy which maximises expected discount value of current state V(s)

Definition – Expected discounted reward

The **expected discounted reward** from s for a policy π is:

$$V^{\pi}(s) = E_{\pi}[\underbrace{\sum_i \gamma^i r(s_i, a_i, s_{i+1})}_{\text{expected sum of discounted rewards}} \mid s_0 = s, a_i = \pi(s_i)]$$

So, $V^{\pi}(s)$ defines the expected value of following the policy π from state s.

Instead of looking at expected value,
lets consider maximum value for each state

Definition – Bellman equation

The **Bellman equation**, identified by Richard Bellman, describes the condition that must hold for a policy to be optimal. The Bellman equation is defined recursively as:

$$V(s) = \max_{a \in A(s)} \sum_{s' \in S} P_a(s'|s)\,[r(s, a, s') + \gamma\, V(s')]$$

$$V(s) = \overbrace{\max_{a \in A(s)}}^{\text{best action from } s} \underbrace{\sum_{s' \in S}}_{\text{for every state}} P_a(s' \mid s)[\overbrace{\underbrace{r(s, a, s')}_{\text{immediate reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{V(s')}_{\text{value of } s'}}^{\text{expected reward of executing action } a \text{ in state } s}]$$

Idea: Value of state = max value of state since we always want to take action maximising state value

Alternatively, consider the value of each action in a state and choose the best one.

- known as Q-value

Definition – Q-value

The **Q-value** for action a in state s is defined as:

$$Q(s, a) = \sum_{s' \in S} P_a(s' \mid s)\,[r(s, a, s') + \gamma\, V(s')]$$

This represents the value of choosing action a in state s and then following this same policy until termination.

Then, value of state = max Q-value

$$V(s) = \max_{a \in A(s)} Q(s, a)$$

Policy Extraction

pick action maximising state value / has largest Q-value

$$\pi(s) = \text{argmax}_{a \in A(s)} \underline{\sum_{s' \in S} P_a(s' \mid s)\, [r(s,a,s') + \gamma\, V(s')]}$$

$$\pi(s) = \text{argmax}_{a \in A(s)} \underline{Q(s,a)}$$

Value for current state depends on value of future states, so we need to find those values first?

Instead, we can set an initial value for all states, then iteratively update values until convergence

Algorithm – Value Iteration

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s,a,s') \rangle$
Output: Value function V

Set V to arbitrary value function; e.g., $V(s) = 0$ for all s

Repeat
  $\Delta \leftarrow 0$ Track change in value
  For each $s \in S$
    $\underbrace{V'(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} P_a(s' \mid s)\, [r(s,a,s') + \gamma\, V(s')]}_{\text{Bellman equation}}$
    $\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$
  $V \leftarrow V'$
Until $\Delta \leq \theta$

If value does not change or change is under threshold, stop

Alternative using Q-values

$\Delta \leftarrow 0$
For each $s \in S$
  For each $a \in A(s)$
    $Q(s,a) \leftarrow \sum_{s' \in S} P_a(s' \mid s)\, [r(s,a,s') + \gamma\, V(s')]$
  $\Delta \leftarrow \max(\Delta, |\max_{a \in A(s)} Q(s,a) - V(s)|)$
  $V(s) \leftarrow \max_{a \in A(s)} Q(s,a)$