StarAi: Deep Reinforcement Learning

Markov Decision Process

Content

- Markov Process
- Markov Reward Process
- Markov Decision Process
  - Bellman Equations
  - Optimality Equations
- Implementation techniques

What is MDP?

MDP - Why?

- Formally describes an environment for RL
- Environment is fully observable
- Current state completely characterises the process

Markov Process

A memoryless random process, a sequence of states with **Markov Property**

$$\langle S, P \rangle$$

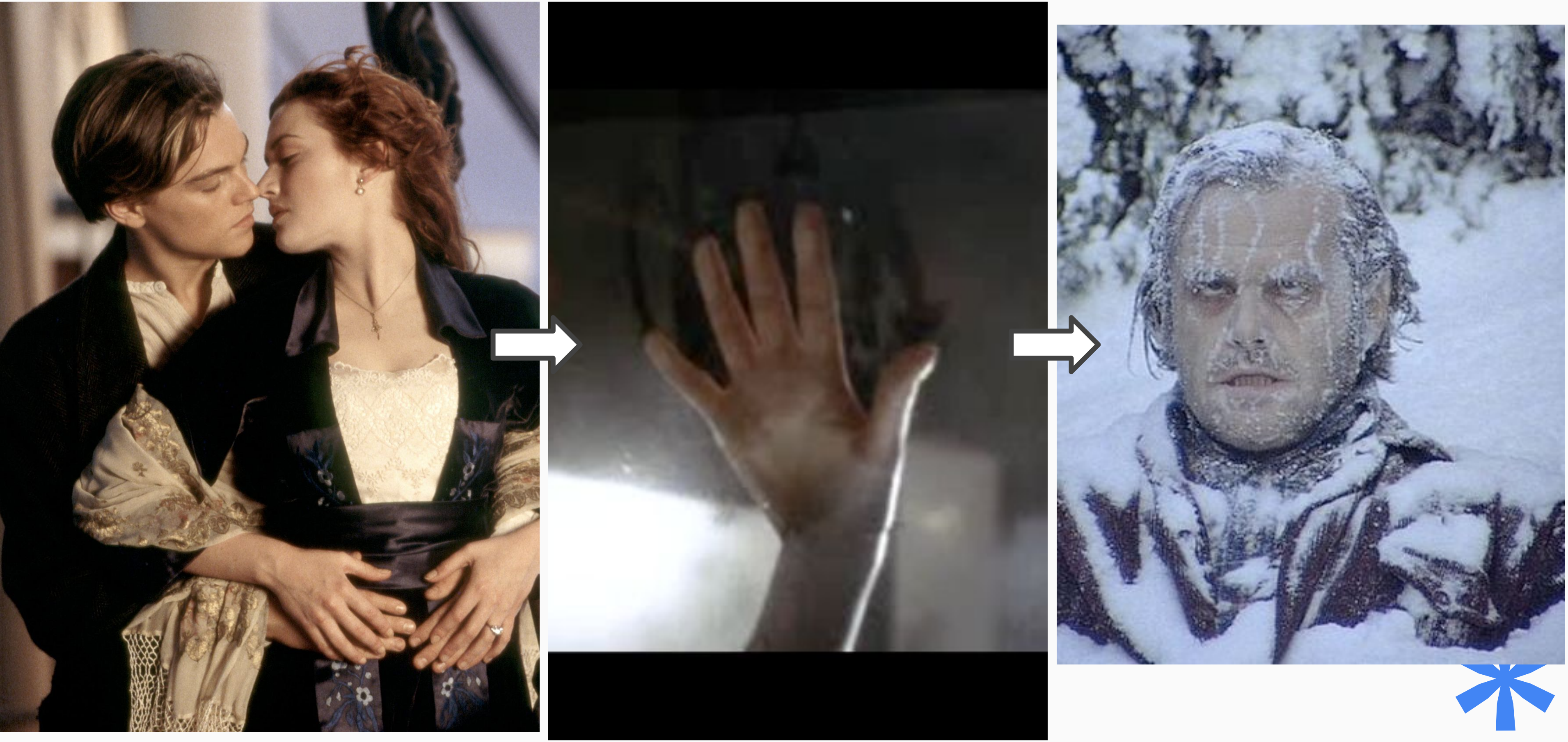
Markov Property

"The future is independent of the past given the present."

- *State* has all necessary information from the previous states
- Therefore no need to keep the history

$$\mathbb{P}[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_1, S_2, \ldots, S_t]$$

Markov Chain - Example



State Transition Probability

Defines transition probabilities from all states to all successor states

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

Example

Airplane at Airport

If Airplane departed now is of certain airline, then there is less probability of having next airplane from same airline. That means if you know the airplane departed then you can predict the probability of airplane from certain airline

Markov Reward Process

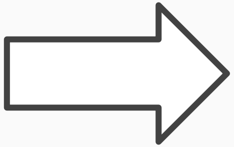
Markov chain with values

$$\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$$

where,

- $\mathcal{R}$ - reward function
- γ - discount factor, $\gamma \in [0, 1]$

MRP - Positive Reward



MRP - Negative Reward

MRP - Example

Airport and Airplanes

Every time the airplane takes off the n sum of money is received by airport

Reward Function

$$\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$$

Return

Total discounted reward from state time step *t*

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Total Reward

Zero not always bad!

State-Value Function

Expected return starting from time step *t*

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

Bellman Equation for MRP

State-value function can be decomposed into

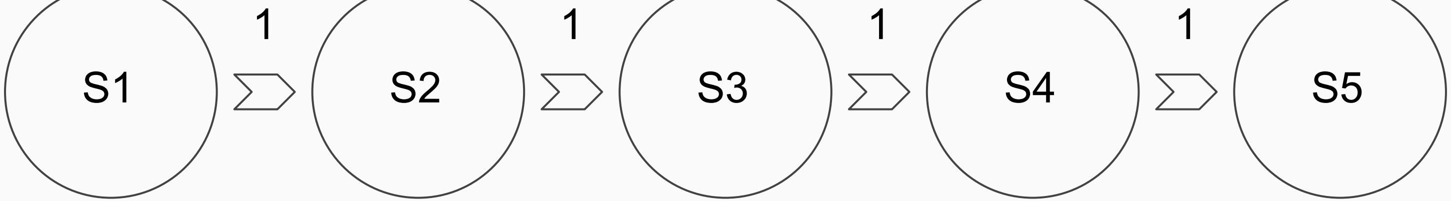
- *immediate reward*
- *discounted value of successor state*

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

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} v(s')$$

State Values - Example

$v(s) =$	4	3	2	1	0



$v(s) =$	1.875	1.75	1.5	1	0

Markov Decision Process

Markov Reward process with decisions

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$$

where,

- $\mathcal{A}$ - set of available actions
- $\mathcal{P}$ - transition probability matrix, **with respect to action**
    - $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $\mathcal{R}$ - reward function, **with respect to action**
    - $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$

Decision, decisions, decisions

MDP - Example

Policy

Distribution over actions, given state *s*

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

- fully defines the behavior of the agent
- depends on current state, not history
- stationary, i.e. *time-independent*

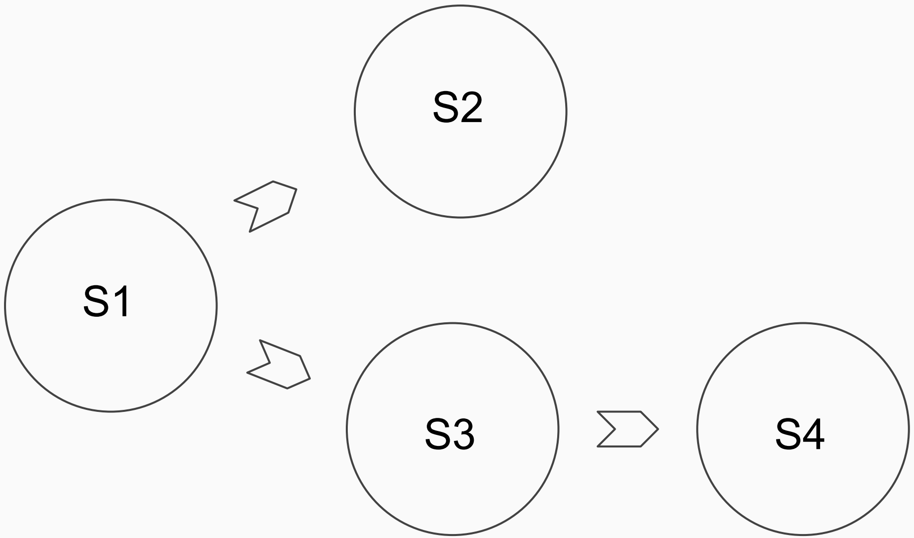
Value Functions

State-Value Function

Expected return, starting from state *s* and following policy ππ

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

State Value Function - Example



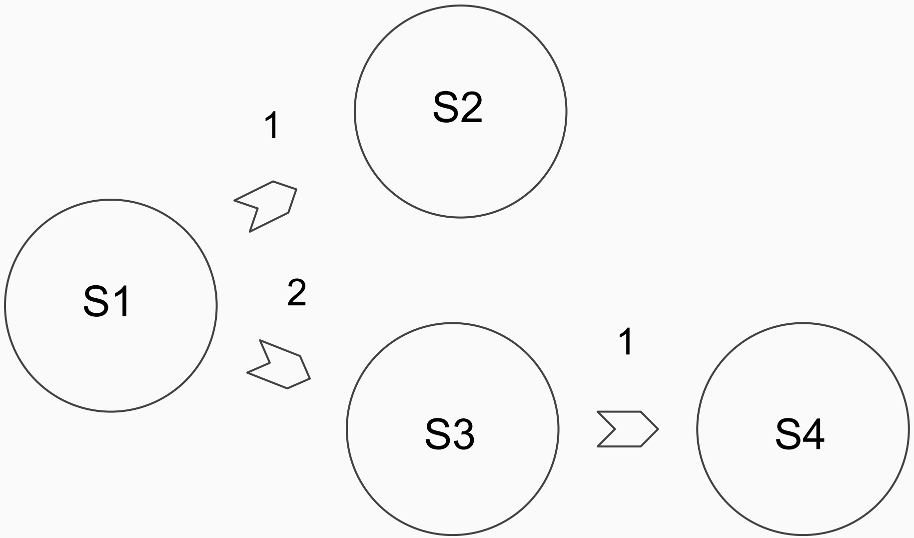
Value Functions

Action-Value Function

Expected return, starting from state s, **taking action a**, and then following policy ππ

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

Action Value Function - Example



Bellman Equations

For state value function

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

Bellman Equations

For action value function

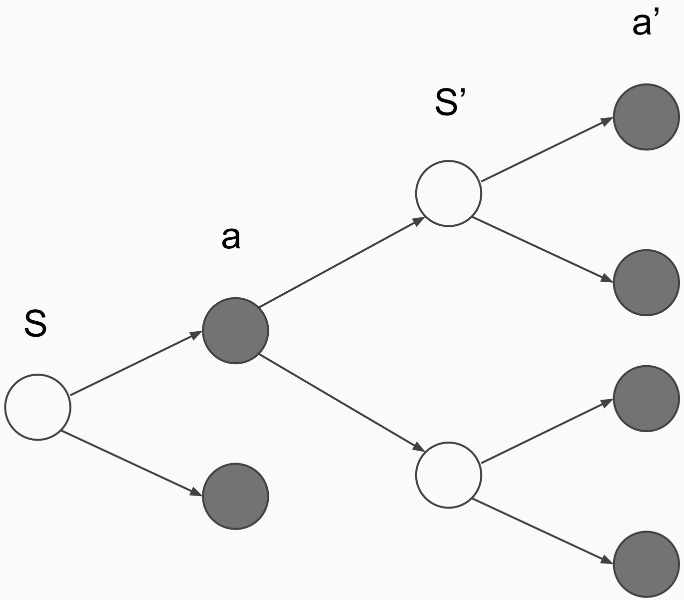
$$Q(s, a) = \mathbb{E}[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1} | S_t = s, A_t = a)]$$

Belman Expectation Equations

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Example



Optimality Value Functions

Specifies the best possible performance in MDP

MDP is solved when optimal value function is solved.

Optimal *state-value* function is the max value function over all policies

Optimal *action-value* function is the max action-value function over all policies.

Bellman Optimality Equations

$$V_*(s) = maxQ_*(s,a) = max(R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_*(s'))$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s \in S} P^a_{ss'} maxQ_*(s',a')$$

Optimal Policy

All optimal policies achieve optimal value functions.

To be continued...

Dynamic Programming

Bellman Optimality Equations

$$V_*(s) = maxQ_*(s,a) = max(R(s,a) + \gamma \sum_{s' \in S} P^a_{ss'} V_*(s'))$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s \in S} P^a_{ss'} maxQ_*(s',a')$$

Dynamic Programming

"dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions."wiki

Necessary properties:

- optimal substructure
- overlapping subproblems

MDP satisfies both.

DP Applications

- Predictions:
  - Input: MDP and policy, or MRP
  - Output: $V\pi$
- Control:
  - Input: MDP
  - Output: $v*$ and $\pi*$

Example

Gridworld - Formula reminder

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

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} v(s')$$

Gridworld Example - State Value Function

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-1
-2	-2	-1	0

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

Gridworld Example - Policy Evaluation

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

0	-2.44	-2.85	-3
-2.44	-2.85	-3	-2.85
-2.85	-3	-2.85	-2.44
-3	-2.85	-2.44	0

Visual

https://cs.stanford.edu/people/karpathy/reinforcejs/index.html


LET'S
ROCK THIS JOINT!
quickmeme.com