

# Markov Decision Processes

Christos Dimitrakakis

June 5, 2024

# Outline

## Markov decision processes

The Markov process

## Backwards induction

Utility and value functions

## Examples

Toy examples

## Markov decision processes

### The Markov process

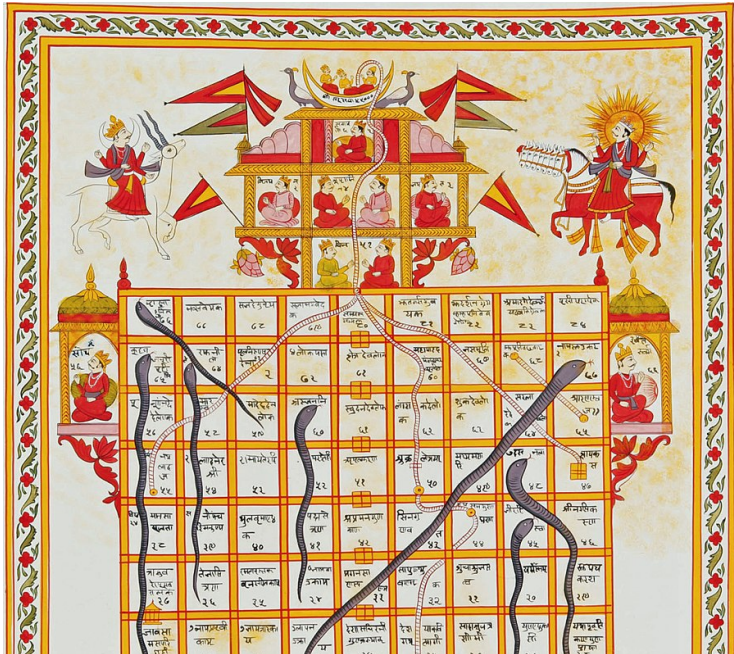
## Backwards induction

### Utility and value functions

## Examples

### Toy examples

# Markov processes



# Markov Process

A sequence of random variables  $s_1, s_2, \dots, s_t$  is a **Markov process** if the next variable  $s_{t+1}$  only depends on the current value  $s_t$

$$P(s_{t+1} \mid s_t, \dots, s_1) = P(s_{t+1} \mid s_t) \quad \forall t$$

# Markov Process

A sequence of random variables  $s_1, s_2, \dots, s_t$  is a **Markov process** if the next variable  $s_{t+1}$  only depends on the current value  $s_t$

$$P(s_{t+1} \mid s_t, \dots, s_1) = P(s_{t+1} \mid s_t) \quad \forall t$$

- ▶ The variable  $s_t \in S$  is the current **state** of the process.
- ▶ For finite  $S$ , the matrix  $p_{i,j} \triangleq P(s_t = j \mid s_{t-1} = i)$  is called the **transition matrix**.

# Markov Process

A sequence of random variables  $s_1, s_2, \dots, s_t$  is a **Markov process** if the next variable  $s_{t+1}$  only depends on the current value  $s_t$

$$P(s_{t+1} \mid s_t, \dots, s_1) = P(s_{t+1} \mid s_t) \quad \forall t$$

- ▶ The variable  $s_t \in S$  is the current **state** of the process.
- ▶ For finite  $S$ , the matrix  $p_{i,j} \triangleq P(s_t = j \mid s_{t-1} = i)$  is called the **transition matrix**.

## Bayesian network

It is sometimes called **Markov chain** because of the structure formed by



# Markov Process

A sequence of random variables  $s_1, s_2, \dots, s_t$  is a **Markov process** if the next variable  $s_{t+1}$  only depends on the current value  $s_t$

$$P(s_{t+1} \mid s_t, \dots, s_1) = P(s_{t+1} \mid s_t) \quad \forall t$$

- ▶ The variable  $s_t \in S$  is the current **state** of the process.
- ▶ For finite  $S$ , the matrix  $p_{i,j} \triangleq P(s_t = j \mid s_{t-1} = i)$  is called the **transition matrix**.

## Bayesian network

It is sometimes called **Markov chain** because of the structure formed by



## Snakes and ladders

- ▶ What is the state?
- ▶ What is the transition matrix?

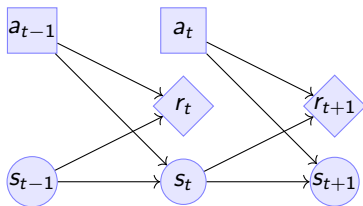


# Markov Decision Process

- ▶ The **state**  $s_t \in S$ .
- ▶ The **action**  $a_t \in A$ .
- ▶ The **reward**  $r_t \in \mathbb{R}$ .

# Markov Decision Process

- ▶ The **state**  $s_t \in S$ .
- ▶ The **action**  $a_t \in A$ .
- ▶ The **reward**  $r_t \in \mathbb{R}$ .



## Definition (Markov Decision Process)

A Markov decision process  $\mu$  on  $(S, A)$  has the property that for any sequence of actions  $a_1, \dots$

$$P_\mu(s_{t+1} \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots) = P_\mu(s_{t+1} \mid s_t, a_t)$$

$$P_\mu(r_{t+1} \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots) = P_\mu(r_{t+1} \mid s_t, a_t)$$

The goal in a **finite-horizon** MDP is to maximise the  $T$ -horizon utility:

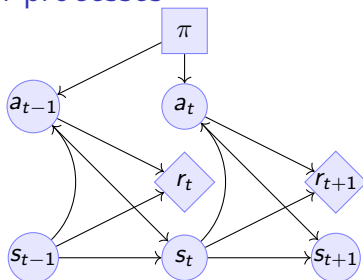
$$U = \sum_{t=1}^T r_t$$

# Policies in Markov decision processes

- ▶ The **policy**  $\pi$
- ▶ The **state**  $s_t \in S$ .
- ▶ The **action**  $a_t \in A$ .
- ▶ The **reward**  $r_t \in \mathbb{R}$ .

# Policies in Markov decision processes

- ▶ The **policy**  $\pi$
- ▶ The **state**  $s_t \in S$ .
- ▶ The **action**  $a_t \in A$ .
- ▶ The **reward**  $r_t \in \mathbb{R}$ .



## Definition (Markov Policy)

A Markov policy takes an action  $a$  at time  $t$  with probability

$$\pi(a_t = a \mid s_t = s)$$

## The expected utility of a policy

$$\mathbb{E}_\pi[U] = \sum_{t=1}^T \mathbb{E}_\pi[r_t]$$

## Markov decision processes

The Markov process

## Backwards induction

Utility and value functions

## Examples

Toy examples

# Value function

- The utility from step  $t$  is  $U_t \triangleq \sum_{k=t}^T r_k$

## The state value function

This is the expected utility obtained by following a **policy**  $\pi$  starting from some **state**  $s$ .

$$V_t^\pi(s) \triangleq \mathbb{E}_\pi(U_t \mid s_t = s)$$

## The state-action value function

This is the expected utility obtained by following a policy  $\pi$  starting from some state  $s$  and **playing action**  $a$

$$Q_t^\pi(s, a) \triangleq \mathbb{E}_\pi(U_t \mid s_t = s, a_t = a)$$

## The optimal value function

There is some policy  $\pi^*$  satisfying

$$V^*(s) \triangleq V^{\pi^*}(s) \geq V^\pi(s) \quad \forall \pi, s$$

$$Q^*(s, a) \triangleq Q^{\pi^*}(s, a) \geq Q^\pi(s, a) \quad \forall \pi, s, a$$

# The expected utility recursion

Value functions satisfy the following recursion

$$\begin{aligned} V_t^\pi(s_t) &= \mathbb{E}_\pi(U_t \mid s_t) \\ &= \mathbb{E}_\pi \left[ \sum_{t=1}^T r_t \mid s_t \right] \\ &= \mathbb{E}_\pi[r_t \mid s_t] + \mathbb{E}_\pi \left[ \sum_{k=t+2}^T r_k \mid s_t \right] \\ &= \mathbb{E}_\pi[r_t \mid s_t] + \mathbb{E}_\pi[U_{t+1} \mid s_t] \\ &= \mathbb{E}_\pi[r_t \mid s_t] + \sum_{s_{t+1} \in S} \mathbb{P}_\pi(s_{t+1} \mid s_t) \mathbb{E}_\pi[U_{t+1} \mid s_{t+1}] \\ &= \mathbb{E}_\pi[r_t \mid s_t] + \sum_a \pi(a \mid s_t) \sum_{s_{t+1} \in S} P_\mu(s_{t+1} \mid s_t, a) V_{t+1}^\pi(s_{t+1}). \end{aligned}$$

## Exercise

Prove that

$$Q_t^\pi(s, a) = r(s, a) + \sum_{s' \in S} P_\mu(s' \mid s, a) \sum_{a' \in A} Q_{t+1}^\pi(s', a') \pi(a_{t+1} = a' \mid s_{t+1} = s')$$

# Backwards induction

## On the state value function

To find the value function of the optimal policy, we can perform the following recursion, after setting  $V_T^*(s) = \max_a r(s, a)$  for all  $s$ .

$$V_t^*(s) = \max_a r(s, a) + \sum_{s' \in S} P_\mu(s' | s, a) V_{t+1}^*(s'),$$

where the optimal action at  $s, t$  is

$$\arg \max_a r(s, a) + \sum_{s' \in S} P_\mu(s' | s, a) V_{t+1}^*(s').$$

## On the state-action value function

Alternatively, we can write this in terms of the Q-value function, where we set  $Q_T^*(s, a) = r(s, a)$  and then recurse:

$$Q_t^*(s, a) = r(s, a) + \sum_{s' \in S} P_\mu(s' | s, a) \max_{a'} Q_{t+1}^*(s', a').$$

Here the optimal action at step  $t$  is just  $\arg \max_a Q_t^*(s, a)$ .



## Markov decision processes

The Markov process

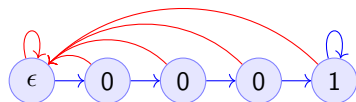
## Backwards induction

Utility and value functions

## Examples

Toy examples

# Chain



In this MDP, there are 5 states, and the transition probabilities are:

$$P(s_{t+1} = \min\{5, i+1\} \mid s_t = i, a = 1) = 1 - \delta, \quad P(s_{t+1} = 1 \mid s_t = i, a = 1) = \delta$$

For the alternative action  $a = 0$ , the probabilities are reversed

$$P(s_{t+1} = \min\{5, i+1\} \mid s_t = i, a = 1) = \delta, \quad P(s_{t+1} = 1 \mid s_t = i, a = 1) = 1 - \delta$$

Further, the reward at state  $s = 1$  is  $\epsilon < 1$  and the reward at state  $s = 5$  is 1.

# Wumpus world

- ▶ State:  $s_t = (x_t, y_t, d_t, w_t)$ , the x-y location of the agent, the direction, and the amount of arrows left.
- ▶ Actions:  $a \in \{L, R, M, S\}$  for left, right, move and shoot.
- ▶ Rewards are given for killing the Wumpus, dying, or finding the treasure.

## Deterministic/Stochastic Wumpus

- ▶ An action/observation is always the same/is random

## Observable/Unobservable Wumpus

- ▶ We know where the holes, the treasure and the Wumpus is/they are unknown

## Static/Dynamic/Strategic/ Wumpus

- ▶ The Wumpus is stationary/moves according to a fixed policy/has goals to achieve

# Deterministic, Observable Wumpus

This is the simplest setting. It is a deterministic planning problem. For this, you can

1. Define a way to describe the Wumpus world
2. Find a policy for solving the Wumpus world as given. This policy is going to be deterministic and Markov.

Of course, the optimal policy for each **instance** of the Wumpus problem is going to be different.

I recommend summarising the Wumpus problem in two parts: (a) A matrix  $G$  where  $G[x, y]$  is a number indicating what is contained in this location, (b)  $x_t, y_t, d_t, w_t$  being the agent-relevant variables.

You can either use any logical planning algorithm, or an MDP algorithm with deterministic transitions for this problem.

# Stochastic, Observable Wumpus

To make the environment stochastic, we can add the following extensions  
(a) The Wumpus moves according to some stochastic policy. For example, the Wumpus could randomly move in a direction, so that on average it moves away from us. (b) Our actions do not always work (e.g. we may turn in the wrong direction) (c) We do not always die when we encounter a hole or the Wumpus.

For this, you can

1. Define a way to describe the Wumpus world
2. Find a policy for solving the Wumpus world as given. This policy is going to be deterministic and Markov.

Of course, the optimal policy for each **instance** of the Wumpus problem is going to be different.

I recommend summarising the Wumpus problem in two parts: (a) A matrix  $G$  where  $G[x, y]$  is a number indicating what is contained in this location, (b)  $x_t, y_t, d_t, w_t$  being the agent-relevant variables.

You can either use any logical planning algorithm, or an MDP algorithm with deterministic transitions for this problem.

# Deterministic, Unobservable Wumpus

This setting is significantly harder to work with. Now we have observations whenever we are near a hole or the Wumpus.

You can either: (a) Use a logical description of the world, and a SAT algorithm. (b) Use a probabilistic description with all probabilities being 0 or 1, and an MDP algorithm.

In either case, a simple idea is to summarise the knowledge of the Wumpus problem as a matrix  $G$  where  $G[x, y]$  indicates one of:

- ▶ Empty.
- ▶ Hole.
- ▶ Wumpus.
- ▶ Treasure.
- ▶ Breeze Observed.
- ▶ Stink Smelled.
- ▶ Unknown.

For simplicity, you can always start with the setting where you know you are dealing with one of a **small number** of possible worlds.

# Static, Stochastic-Observation, Unobservable Wumpus

Here we assume the Wumpus does not move, and observations are stochastic: sometimes we feel a breeze, sometimes not. We assume we know the probability of a breeze.

The first problem is to summarise what we know about the Wumpus problem. Now we can have an entry  $G[x, y]$  in the matrix which is a **vector of probabilities** for the possible contents of the co-ordinate:  
(Empty, Hole, Wumpus, Treasure)

For simplicity, you can always start with the setting where you know you are dealing with one of a **small number** of possible worlds. Then you only need to deal with the probability of each world being the right one.