Outline

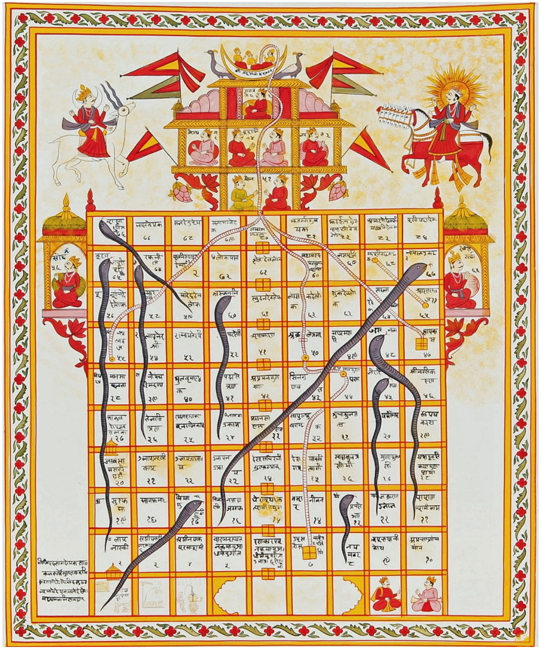
Markov processes
  The Markov process

Markov decision processes

Backwards induction
  Utility and value functions

Examples
  Toy examples

Snakes and Ladders



Markov Process

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

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

Markov Process

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

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

- 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

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

- 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

$$\begin{tikzcd} s_{t-1} \arrow[r] & s_t \arrow[r] & s_{t+1} \end{tikzcd}$$

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

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

- 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

$$\begin{tikzcd} s_{t-1} \arrow[r] & s_t \arrow[r] & s_{t+1} \end{tikzcd}$$

Example (Snakes and ladders)

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

Snakes and ladders workout

State variable

- $s_t = (m_t, x_{0,t}, x_{1,t})$
- m_t: whose turn it is
- $x_{0,t}$: location of player 0
- $x_{1,t}$: location of player 1

Transition matrix (with no snakes or ladders)

- $x_{1-m_t,t+1} = x_{1-m_t,t}$ w.p. 1
- $m_{t+1} = 1 - m_t$ w.p. 1.
- $P(x_{m_t,t+1} = x_{m_t,t} + k) = 1/6$ for all $k < 100 - x_{m_t,t}$.
- $P(x_{m_t,t+1} = 100) = [(100 - x_{m_t,t+1})/6]_+$

Taking snakes/ladders into account

- $f : \{1, 100\} \rightarrow \{1, 100\}$ tells us where snakes/ladders take us.
- $P(x_{m_t,t+1} = {\color{red}f}(x_{m_t,t} + k)) = 1/6$ for all $k < 100 - x_{m_t,t}$.

Simple betting game

- You have s_t CHF.
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

Simple betting game

- You have s_t CHF.
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Simple betting game

- You have s_t CHF.
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Transition matrix

For all $k > 1$.

- $P(s_{t+1} = k + 1 \mid s_t = k) = p$
- $P(s_{t+1} = k - 1 \mid s_t = k) = 1 - p$

If $s_t = 0$ the game ends.

Simple betting game

- You have s_t CHF.
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Transition matrix

For all $k > 1$.

- $P(s_{t+1} = k + 1 \mid s_t = k) = p$
- $P(s_{t+1} = k - 1 \mid s_t = k) = 1 - p$

If $s_t = 0$ the game ends.

Martingale property (*)

If $p = 1/2$ then the process is a martingale, i.e.

$$\mathbb{E}[s_{t+1} \mid s_t] = s_t$$

This is because $\mathbb{E}[s_{t+1} \mid s_t = k] = (k + 1)p + (k - 1)(1 - p) = k$.

Python

Simple betting game

- You have s_t CHF.
- At time t, you decide to stop or play
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

Simple betting game

- You have s_t CHF.
- At time t, you decide to stop or play
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Simple betting game

- You have s_t CHF.
- At time t, you decide to stop or play
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Transition matrix

For all $k > 1$.

- $P(s_{t+1} = k + 1 \mid s_t = k) = p$
- $P(s_{t+1} = k - 1 \mid s_t = k) = 1 - p$

If $s_t = 0$ the game ends. This should only appear in the text.}

Simple betting game

- You have s_t CHF.
- At time t, you decide to stop or play
- With probability p you gain 1 CHF, and otherwise you lose 1 CHF.

State space

- $s_t \in \mathbb{N}$ (i.e. we cannot owe money)

Transition matrix

For all $k > 1$.

- $P(s_{t+1} = k + 1 \mid s_t = k) = p$
- $P(s_{t+1} = k - 1 \mid s_t = k) = 1 - p$

If $s_t = 0$ the game ends. This should only appear in the text.}

Martingale property (*)

If $p = 1/2$ then the process is a martingale, i.e.

$$\mathbb{E}[s_{t+1} \mid s_t] = s_t$$

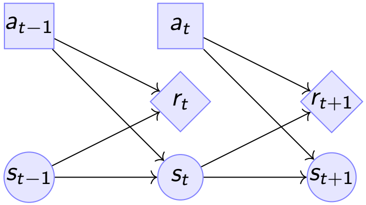
This is because $\mathbb{E}[s_{t+1} \mid s_t = k] = (k + 1)p + (k - 1)(1 - p) = k$.

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 μ on (S, A) has the property that for any sequence of actions $a_1, \ldots$

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

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

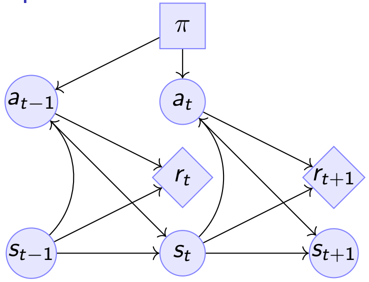
$$U = \sum_{t=1}^{T} r_t$$

Policies in Markov decision processes

- The policy π
- 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 π
- 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

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

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 π 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 π 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 π^* satisfying

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

The expected utility recursion

Value functions satisfy the following recursion

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

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.

$$V^*_t(s) = \max_a r(s, a) + \sum_{s' \in S} P_\mu(s' \mid s, a) V^*_{t+1}(s'),$$

where the optimal action at s, t is
$\arg\max_a r(s, a) + \sum_{s' \in S} P_\mu(s' \mid s, a) V^*_{t+1}(s')$.

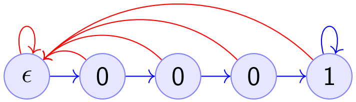
On the state-action value function

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:

$$Q^*_t(s, a) = r(s, a) + \sum_{s' \in S} P_\mu(s' \mid s, a) \max_{a'} Q^*_{t+1} Q(s', a').$$

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

Chain



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

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

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

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

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.