

COMP3702 Artificial Intelligence

Semester 2, 2023

Tutorial 6

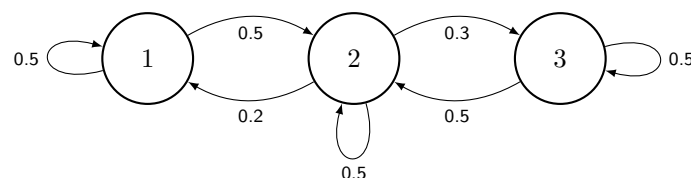
Before you begin, please note:

- Tutorial exercises are provided to help you understand the materials discussed in class, and to improve your skills in solving AI problems.
- Tutorial exercises will not be graded. However, you are highly encouraged to do them for your own learning. Moreover, we hope you get the satisfaction from solving these problems.
- The skills you acquire in completing the tutorial exercises will help you complete the assignments.
- You'll get the best learning outcome when you try to solve these exercises on your own first (before your tutorial session), and use your tutorial session to ask about the difficulties you face when trying to solve this set of exercises.

Background

As a building block for Markov decision processes, we begin by introducing a very useful probability model called a **Markov chain**. A Markov chain is a stochastic discrete-time, discrete-state transition system. Markov chains are used extensively to model systems with predictable, stochastic, state transitions, including activity models, target tracking, regression models with mode- or regime-switching, for simulating wind and weather patterns, and in finance. They are also used extensively as a computational routine for implementing Bayesian models, which require efficient resampling methods, an approach called the *Monte Carlo Markov chain* method.

A three-state Markov chain system is illustrated below:



Here, the three states are illustrated as circles, with chance transitions between them, as indicated by edges. Each edge has associated with it a probability, such that at each time step, the probability of moving from the current state s to the next state s' is given by $P(s'|s)$, and edges indicate transitions that have strictly positive probabilities. Importantly, in the transition probability expression above, state transitions depend on the current state only, and not on the past history. This dependence of the probability of reaching a future state on current state alone is called the **Markov property** — *remember this!*

Note that, in the example above, not all states are directly linked with all others, but:

- each state can be reached from all others by some path, and
- no two states have a periodic cycle between them.

When these two conditions hold, we say that the Markov chain is *ergodic*. We are going to use some probability theory and matrices to analyse the steady state behaviour of an ergodic Markov chain. The techniques described here will be useful later in the course.

Let the one-step state-transition probabilities for the diagram above be given by a matrix:

$$P = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.2 & 0.5 & 0.3 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

The entries in P are read as the probability that, from the current state corresponding to any given row, the system moves to the next state indicated by the column. Importantly, note that all rows sum to 1; this is the definition of a *stochastic matrix*. Indeed, you can think of each row of a stochastic matrix as a valid probability distribution (specifically, this is called a *categorical* distribution), because they sum to 1.

An initial state x_0 can be encoded as row vector, with a value of 1 indicating the initial state and zeros everywhere else. For example, starting in state 3 is represented by:

$$x_0 = [0 \quad 0 \quad 1]$$

Using the two matrices P and x_0 , we can compute the distribution of states of the Markov chain in the next time step, $t = 1$, starting from state 3, by pre-multiplying P by x_0 :

$$x_1 = x_0 P$$

This returns a distribution over the state of the Markov chain at $t = 1$. By repeated application, for $t = 2$ we have:

$$x_2 = x_1 P = x_0 P^2.$$

The same process can be repeated to compute the distribution over the state of the Markov chain k time steps into the future:

$$x_k = x_0 P^k$$

A *stationary distribution* of a Markov chain is a probability distribution that remains unchanged in the Markov chain as time progresses. Typically, it is represented as a row vector whose entries are probabilities summing to 1, and given transition matrix P , it satisfies:

$$xP = x$$

In other words, x is *invariant* by the matrix P .

An iterative approach to computing the stationary distribution of a (irreducible¹ and aperiodic²) Markov chain is called the *power method*, in which $x_k = x_0 P^k$ is computed for a k large enough for $|x_k - x_{k-1}|$ to become very small. You will see a similar “trick” used later in Markov decision processes.

Exercises

Exercise 6.1. Restless robot on a Markov chain. Consider the following gridworld:



Imagine you have a restless robot with no AI implemented on it. When you put your robot on the gridworld above, it moves in a random direction, left or right, from one square to an adjacent square with equal probability at each time step. The only time your restless robot stays still is if it tries to move into one of the ends of the gridworld, i.e. left in state 0 or right in state 5.

The motion of your restless robot can be modeled as a Markov chain. This particular Markov chain is called a *simple random walk* on a finite set of integers. For example, consider starting in state 2, as indicated by the star: the probability of moving left, to state 1, is 0.5, and the probability of moving right, to state 3, is also 0.5.

¹If the state space is finite and all states communicate, i.e. can be reached, eventually, from each other, then the Markov chain is called *irreducible*. We will deal only with irreducible Markov systems in this course.

²Periodic Markov chains have a special structure that forces every state to be reachable only every $m > 1$ time-steps; for an aperiodic Markov chain, $m = 1$. We will deal only with aperiodic Markov system in this course.

- What are the dimensions of this Markov chain's transition matrix?
- Construct the transition matrix, P . (Do this in code). Take care with states 0 and 5.
- Give the initial state probability vector, x_0 for starting in state 2, and compute the distribution of states of the Markov chain at $t = 1$.
- Using matrix multiplication methods, compute the distribution of states of the Markov chain 2 time steps after starting in state 2, x_2 , by multiplying again by P . Repeat the multiplication for 4, 10 and 20 time steps. What is happening to the resulting distribution over states?

Exercise 6.2. Consider the gridworld below:



s_0	s_1	s_2	s_3	s_4	s_5
5		★			10
			0	0	

An agent is currently on grid cell s_2 , as indicated by the star, and would like to collect the rewards that lie on both sides of it. If the agent is on a numbered square (0, 5 or 10), the instance terminates and the agent receives a reward equal to the number on the square. On any other (non-numbered) square, its available actions are to move Left or Right. Note that Up and Down are never available actions. If the agent is in a square with an adjacent square below it, it does not always move successfully: when the agent is in one of these squares and takes a move action, it will only succeed with probability p . With probability $1 - p$, the move action will fail and the agent will instead fall downwards into a trap. If the agent is not in a square with an adjacent space below, it will always move successfully.

- Consider the policy π_R , which is to always move right when possible. For each state $s \in \{s_1, s_2, s_3, s_4\}$ in the diagram above, give the value function V^{π_R} in terms of $\gamma \in [0, 1]$ and $p \in [0, 1]$.
- Consider the policy π_L , which is to always move left when possible. For each state $s \in \{s_1, s_2, s_3, s_4\}$ in the diagram above, give the value function V^{π_L} in terms of γ and p .

Example Gridworld domain

Consider the gridworld below:

	0	1	2	3
0				1
1				-100
2				

States in this environment are represented by the position of the agent on the tiles (represented as [row,col] using zero indexing, starting from the top left corner) as well as the collection status of a key (True/False), located at [2,2] which must be collected before the agent can exit the environment through the goal tile. The world is bounded by a boundary wall, and there is one obstacle, at [1,1]. In addition, there are two terminal

states, indicated by the coloured squares. The red square indicates a hazard, and the blue square indicates the goal.

Terminal states: In practice, we would say that the two coloured tiles are *terminal states* of the MDP. For mathematical convenience, we often prefer to deal with an infinite horizon MDP (see more below). To convert this model to an infinite horizon MDP, we will add an extra pseudo-state to the list of MDP states called *exited*, which is absorbing (the agent cannot leave it irrespective of its action, i.e. $T(\text{exited}, a, \text{exited}) = 1$ for all a).

Actions and Transitions: In this world, an agent can generally choose to move in four directions — *up*, *down*, *left* and *right*. However, the agent moves successfully with only $p = 0.8$, and moves perpendicular to its chosen direction with $p = 0.1$ in each perpendicular direction. If it hits a wall or obstacle, the agent stays where it is. In addition, once the agent arrives on a coloured square with a value, it has only one special action available to it; that is, to *exit* the environment.

Rewards: The values stated on the coloured squares are the reward for *exiting* the square and the environment, so the reward is not repeatedly earned; that is, $R([0, 3, \text{True}], \text{exit}) = 1$, and $R([1, 3], \text{exit}) = -100$. All other states have 0 reward.

Discount factor: $\gamma = 0.9$.

Exercises on Gridworld

You can utilise the sample starter code `grid_world_starter.py` to help solve the following exercises. Your code should be inserted in the indicated methods.

Exercise 6.3. Define and write code to perform an action in the gridworld environment; call it `perform_action()`.

`perform_action()` should do two things: (i) modify the map according to the state transition function given, and (ii) return a reward.

- What is the one-step probability of arriving in state $[1, 0, \text{False}]$ from each state s after taking action a , i.e. what is $P([1, 0, \text{False}] | a, s) \forall (a, s)$?
- What is the one-step probability of arriving in each state s' when starting from $[0, 0, \text{False}]$ for each a , i.e. what is $P(s' | a, [0, 0, \text{False}]) \forall a, s'$?
- Write a function that can compute the probabilities of arriving in any state s' from an initial state s after taking action a , i.e. $P(s' | a, s)$. Rather than hard-coding the probabilities, check if each action moves to a neighbour state or if it results in any collisions with the boundary or the obstacle. Then use this information to compute the transition probabilities. *Hint:* You may find some of the tricks for 8-puzzle from Tutorial 2 useful, e.g. using linear indexing and logical tests for collisions. You may also wish to parameterise this function so that p can be varied too.
- Write a new function to process the gridworld and store all of the transition probabilities, using the transition-probability computing function you have developed above. The aim is to have functions that can be used on an arbitrary gridworld (i.e. do not hard-code your function just for this problem instance!).

Exercise 6.4. Implement VI for this problem, using: $V[s] \leftarrow \max_a \sum_{s'} P(s' | s, a) (R(s, a, s') + \gamma V[s'])$.

Note that $R(s, a, s') = R(s)$ is the reward for landing on a square, which is non-zero for only the red and blue squares at $[1, 3]$ and $[0, 3]$, respectively.

- What is the value function estimate after 4 iterations? What is the policy according to the value function estimate after 4 iterations?
- What is the value function estimate after 10 iterations? What is the policy according to the value function estimate after 10 iterations?

- c) What is the value function estimate after 1000 iterations? What is the policy according to the value function estimate after 1000 iterations?
- d) What is the largest difference in the value function estimates for any state at iteration 999 and 1000? Based on this, can you say that the algorithm has converged?
- e) How long does 1000 iterations of VI take?