COMP3702: Tutorial 7

Aryaman Sharma
aryaman.sharma@uq.edu.au

Rewards and Values

Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \ldots$ in time. What utility should be assigned? "Return" or "value"

- total reward $V = \sum_{i=1}^{\infty} r_i$
- average reward $V = \lim_{n \to \infty} (r_1 + \cdots + r_n)/n$

Rewards and Values

Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \ldots$ in time.

- discounted return $V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$
  γ is the discount factor $0 \leq \gamma \leq 1$.

Terminology reminder

An expected value of a discrete random variable, X is:

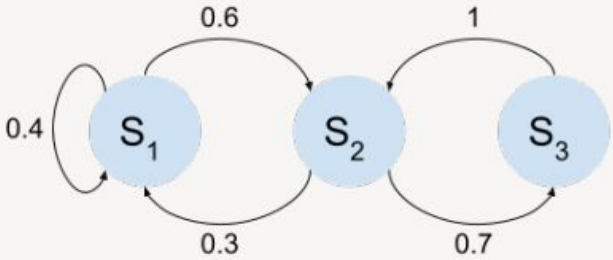
$$E(X) = \sum_x xP(x)$$

Use measures of value for outcomes such that the value of a lottery can be taken as the expected value of the outcomes of the lottery:

$$\begin{aligned} & value([p : o_1, 1 - p : o_2]) \\ = \quad & p \times value(o_1) + (1 - p) \times value(o_2) \end{aligned}$$

Markov Chains

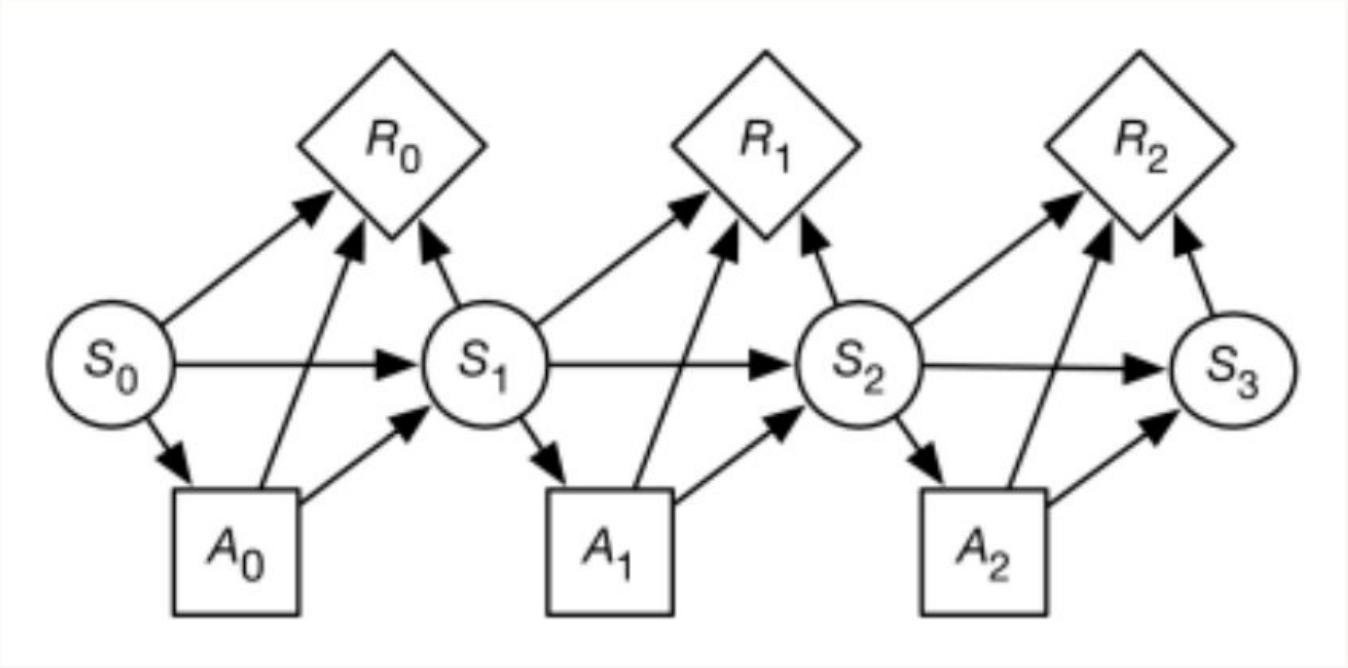
- Markov chains have a set of states, and at each time step they transition to a state (either a new state or the same state) according to some transition probabilities.
- Markov property: Probability distribution for next state depends only on current state.
- Probabilities of outgoing arrows from each state must sum to 1.



Can write the transition probabilities as a stochastic matrix (i.e. rows sum to 1).

$$P = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.3 & 0 & 0.7 \\ 0 & 1 & 0 \end{bmatrix}$$

- A Markov decision process augments a Markov chain with actions and values:



World State

- The world state is the information such that if the agent knew the world state, no information about the past is relevant to the future. Markovian assumption.
- S_t is state at time t, and A_t is the action at time t:

$$P(S_{t+1} \mid S_0, A_0, \ldots, S_t, A_t) = P(S_{t+1} \mid S_t, A_t)$$

  $P(s' \mid s, a)$ is the probability that the agent will be in state s' immediately after doing action a in state s.
  - The state captures all relevant information from the history
  - The state is a sufficient statistic of the future
- The dynamics are stationary if the distribution is the same for each time point.

Markov Decision Processes

An MDP consists of:

- set S of **states**.
- set A of **actions**.
- $P(S_{t+1} \mid S_t, A_t)$ specifies the dynamics or **transition function**. Can denote as $P(s'|s,a)$, the probability of ending up in state s' given that we were in state s and did action a
- $R(S_t, A_t, S_{t+1})$ specifies the **reward** at time t.
  Can denote as $R(s,a,s')$, which is the reward for being in state s, performing action a, and ending up in state s'.
  Sometimes we use the expected reward for being in s and doing a, which is $R(s,a) = \sum_{s'} P(s' \mid s,a) R(s,a,s')$.
- γ is **discount factor**.
- An MDP's **objective** is: $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]$.

Policy

What is the solution of an MDP problem?

- A policy is a sequence of actions, taken to move from each state to the next state over the whole time horizon.
- A stationary policy is a function or a map:

$$\pi : S \to A$$

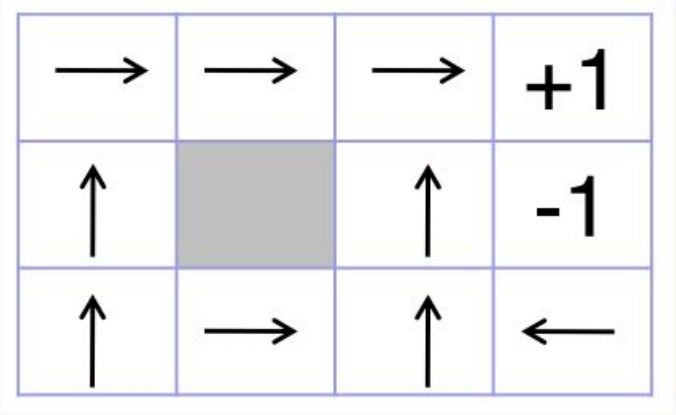
  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.
- An optimal policy, usually denoted as π^*, is one with maximum expected discounted reward.

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0} \gamma^t R(s_t, \pi(s_t))\right]$$

  where $\pi(t)$ is the action taken at time t.
- For a fully-observable MDP with stationary dynamics and rewards with infinite or indefinite horizon, there is always an optimal stationary policy.

What is the solution of an MDP problem?

- For the simple navigation "grid world":
- A **policy**, or mapping from states to actions $\pi : S \rightarrow A$ could be given as below



  - A policy, $\pi(s)$, tells us what action the agent should perform for each state
  - The optimal policy, $\pi^*(s)$, tells us what the **best** action the agent should perform for each state

Value of a Policy

We first approach MDPs using a recursive reformulation of the objective called a value function

- The value function of an MDP, $V^{\pi}(s)$, is the expected future reward of following an (arbitrary) policy, π, starting from state, s, given by:

$$V^{\pi}(s) = \sum_{s' \in \mathcal{S}} P(s' \mid \pi(s), s) \left[R(s, \pi(s), s') + \gamma V^{\pi}(s') \right].$$

  where the policy $\pi(s)$ determines that action taken in state s.
- Here we have dropped the time index, as it is redundant, but note that $a_t = \pi(s_t)$.

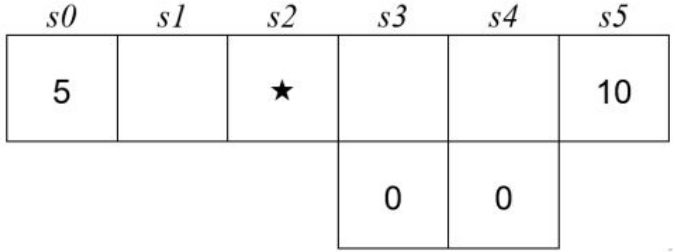
Value of a Policy

Given a policy π:

- The Q-function represents the value of choosing an action and then following policy π in every subsequent state.
- $Q^\pi(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- Q^π and V^π can be defined mutually recursively:

$$\begin{aligned} Q^\pi(s,a) &= \sum_{s'} P(s' \mid a, s)\left(R(s,a,s') + \gamma V^\pi(s')\right) \\ V^\pi(s) &= Q(s, \pi(s)) \end{aligned}$$

Exercise 7.1. Consider the gridworld below:



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 and 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.

a) 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 $p \in [0, 1]$.

b) 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.

s0	*s1*	*s2*	*s3*	*s4*	*s5*
5	$10\gamma^4 p^2$	$10\gamma^3 p^2$	$10\gamma^2 p^2$	$10\gamma p$	10
			0	0	

$$V^{\pi_R}(s_1) = 10\gamma^4 p^2$$
$$V^{\pi_R}(s_2) = 10\gamma^3 p^2$$
$$V^{\pi_R}(s_3) = 10\gamma^2 p^2$$
$$V^{\pi_R}(s_4) = 10\gamma p$$

Value Iteration

Value Iteration

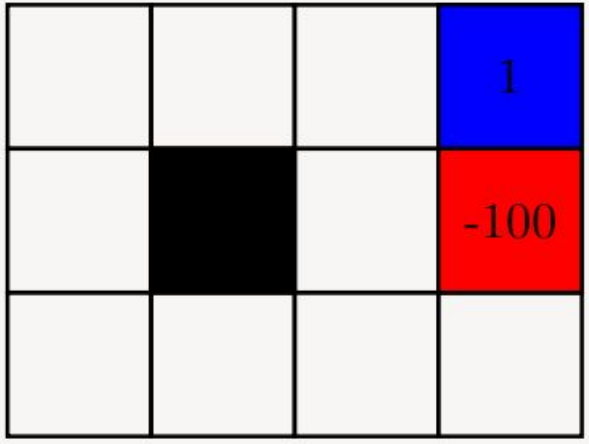
- Let V_k be k-step lookahead value function.
- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.

1. Set V_0 arbitrarily, e.g.:
$$V_0(s) \leftarrow 0$$
2. Until convergence: Compute V_{i+1} from V_i.
Loop for all s {
$$V_{i+1}(s) = \max_a \sum_{s'} P(s' \mid a, s) \{R(s, a, s') + \gamma V_i(s')\}$$
}

Once the values converge, recover the best policy from the current value function estimate:

$$\arg\max_a \sum_{s'} P(s' \mid a, s) \left\{R(s, a, s') + \gamma \hat{V}(s')\right\}$$

Grid world



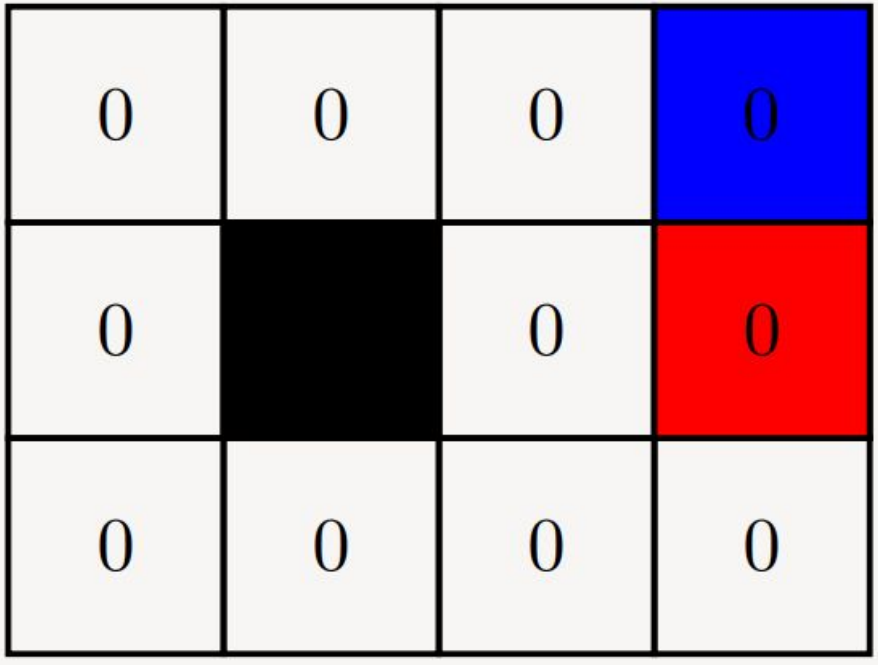
$\gamma = 0.9$

Agent can move up, down, left or right.

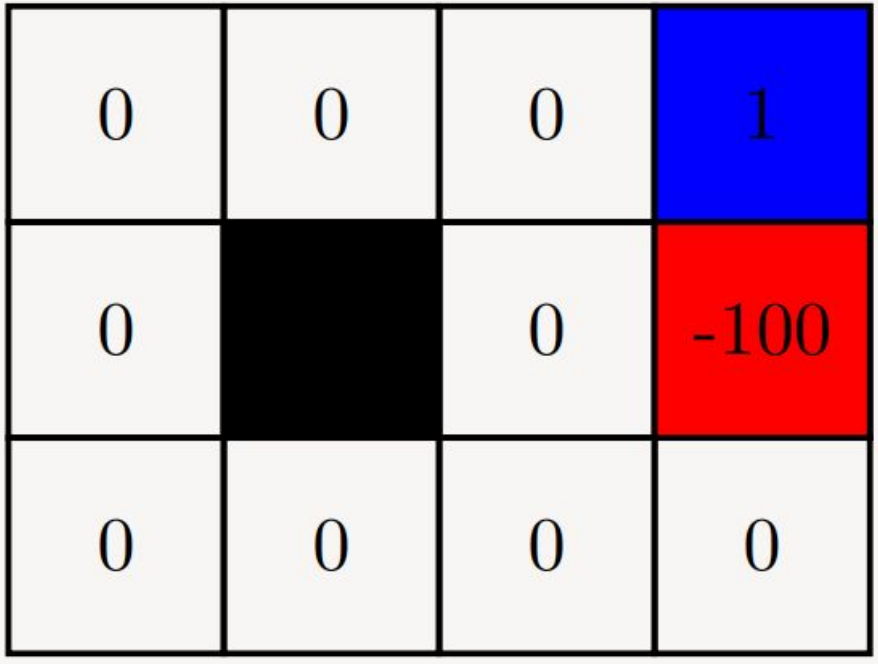
Moves successfully with $p = 0.8$, or perpendicular to direction with $p = 0.1$, each direction.

Hit a wall and the agent stays where it is.

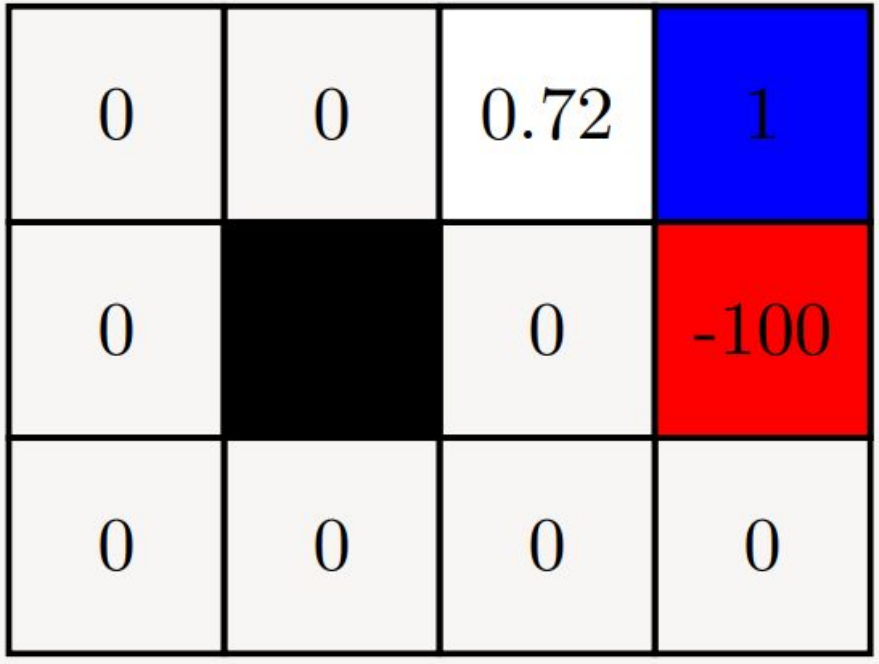
VI — initialisation



VI — 1 iteration



VI — 2 iterations



VI — 3 iterations

0	0.5184	0.7848	1
0		0.0648	-100
0	0	0	0

Asynchronous Value Iteration

- The agent doesn't need to sweep through all the states, but can update the value functions for each state individually.
- Do update assignments to the states in any order we want, even random
- This converges to the optimal value functions, if each state and action is visited infinitely often in the limit.
- It can either store $V[s]$ or $Q[s, a]$.

Asynchronous VI

Storing $V[s]$

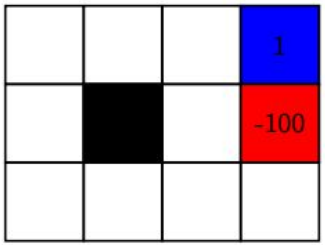
- Repeat forever:
  1. Select state s
  2. $V[s] \leftarrow \max_a \sum_{s'} P(s' \mid s, a) \left(R(s, a, s') + \gamma V[s']\right)$

Storing $Q[s, a]$

- Repeat forever:
  1. Select state s, action a
  2. $Q[s, a] \leftarrow \sum_{s'} P(s' \mid s, a) \left(R(s, a, s') + \gamma \max_{a'} Q[s', a']\right)$

Example Gridworld domain

Consider the gridworld below:



States in this environment are the positions on the tiles. The world is bounded by a boundary wall, and there is one obstacle, at [1,1] (using python or zero indexing starting from the bottom left corner). In addition, there are two terminal states, indicated by the coloured squares.

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. $T(exited, a, exited) = 1$ for all a).

Actions and Transitions: In this world, an agent can generally choose to move in four directions — *up*, *down*, *left* and *right* (which we might sometimes refer to as ^, v,< and >, respectively). 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, $R([3, 2], exit) = 1$, and $R([3, 1], exit) = -100$. All other states have 0 reward.

Discount factor: $\gamma = 0.9$.

Exercise 7.2.

a) Implement the `attempt_move` and `get_transition_probabilities` methods in the `Grid` class in `grid_world_starter.py` according to their docstrings. 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!).

b) What is the one-step probability of arriving in each state s' when starting from [0,0] for each a, i.e what is $P(s'|a, [0, 0]) \; \forall \; a, s'$?

c) What is the one-step probability of arriving in state [1,0] from each state s after taking action a, i.e what is $P([1, 0]|a, s) \; \forall \; (a, s)$?

Exercise 7.3. Implement VI for this problem, using: $V[s] \leftarrow \max_a \sum_{s'} P(s' \mid s, a)\,(R(s, a, s') + \gamma V[s'])$.

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 [3,1] and [3,2], respectively.

a) What is the value function estimate after 4 iterations? What is the policy according to the value function estimate after 4 iterations?

b) What is the value function estimate after 10 iterations? What is the policy according to the value function estimate after 10 iterations?

c) Run value iteration until the maximum change in values is less than `EPSILON` (i.e. the values have 'converged'). At this point, what is the value function estimate? What is the policy according to the value function?

d) On each iteration, print the iteration number and the largest difference in the value function estimate for any state. What do you observe in the results?