UNIT 3 MARKOV PROCESS AND OPTIMALITY PROOF

Ami Munshi

Syllabus

3	**Markov Process and Optimality Proofs** Markov property, Markov chains, Markov reward process (MRP), Markov Decision Process (MDP) Modelling, Bellman equation, state and action value functions, Bellman optimality equation, Cauchy sequence & Green's equation	05

Course Outcomes

After completion of the course, students will be able to –

1. Apply the basics of Reinforcement Learning (RL) to compare with traditional control design
2. Correlate how RL relates and fits into the broader umbrella of machine learning, deep learning
3. Recommend value functions and appropriate algorithms for optimal decision-making
4. Design a dynamic programming approach to an industrial control problem

References

Text Books

1. Laura Graesser and Wah Loon Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, 1st Edition, Pearson India/Padmavati Publisher, 2022.
2. Richard Sutton and Andrew Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.
3. Abhishek Nandy and Manisha Biswas, *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*, 1st Edition, Apress Publisher, 2017.

Reference Books

1. Nimish Sanghi, *Deep Reinforcement Learning with Python: With PyTorch, TensorFlow and OpenAI*, 2nd edition, Apress Publisher, 2021.
2. Alexander Zai and Brandon Brown, *Deep Reinforcement Learning in Action*, 1st Edition, Manning Publisher, 2020.
3. Csaba Szepesvari, *Algorithms for Reinforcement Learning*, 3rd Edition, Morgan & Claypool Publisher, 2019.

Russel and Norwig "Artificial Intelligence a Modern Approach", 4th edition

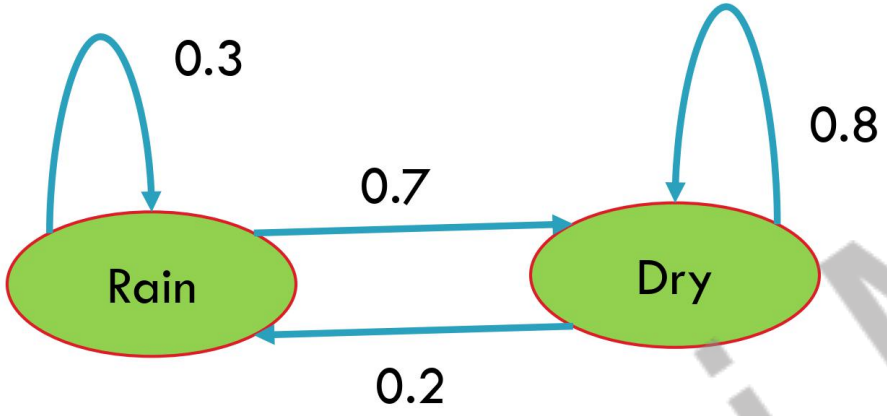
Basics from NLP

Markov Model

- Set of states
  - $\{S_1, S_2, \ldots S_N\}$
- Process moves from one state to another generating a sequence
  - $S_{i1}, S_{i2}, .. S_{ik}$
- Markov chain property
  - probability of each subsequent state depends only on what was the previous state
  - $P(S_{ik}|S_{i1}, S_{i2}, \ldots S_{i(k-1)}) = P(S_{ik}|S_{i(k-1)})$

Ref: https://www.youtube.com/watch?v=irRCDQQtTh8
https://www.youtube.com/watch?v=LmWWUnqiKYk&list=PLS5J_kYIArq70IFrO7hTSpGI97AosjFCI&index=7
https://cs.wmich.edu/elise/courses/cs626/s12/HiddenMarkovModels.pdf

Example of Markov Model



- Two states
  - Rain and dry
- Transition probability
  - $P(Rain|Rain) = 0.3$
  - $P(Rain|Dry) = 0.2$
  - $P(Dry|Dry) = 0.8$
  - $P(Dry|Rain) = 0.7$
- Initial Probability
  - P(Rain)=0.4
  - P(Dry)=0.6

Transition probability matrix

Current \ Future	$Rain$	Dry
$Rain$	0.3	0.7
Dry	0.2	0.8

Initial probability

P(Rain)=0.4
P(Dry)=0.6

Ref: https://cs.wmich.edu/elise/courses/cs626/s12/HiddenMarkovModels.pdf

Calculation of Sequence probability

- By Markov chain property, probability of state sequence can be found by the formula
- $P(s_{i1}, s_{i2}, \dots s_{ik})$
- $= P(s_{ik}|s_{i1}, s_{i2}, \dots s_{i(k-1)})P(s_{i1}, s_{i2}, .. s_{i(k-1)})$
- $= P(s_{ik}|s_{i(k-1)})P(s_{i1}, s_{i2}, .. s_{i(k-1)})$
- $= P(s_{ik}|s_{i(k-1)})P(s_{i(k-1)}|s_{i(k-2)}) \dots \dots P(s_{i2}|s_{i1})P(s_{i1})$

Ref: https://www.youtube.com/watch?v=irRCDQQtTh8
https://www.youtube.com/watch?v=LmWWUnqiKYk&list=PLS5J_kYIArq70IFrO7hTSpGI97AosjFCI&index=7
https://cs.wmich.edu/elise/courses/cs626/s12/HiddenMarkovModels.pdf

Calculation of Sequence probability

Transition probability matrix

Current \ Future	$Rain$	Dry
$Rain$	0.3	0.7
Dry	0.2	0.8

Initial probability

P(Rain)=0.4

P(Dry)=0.6

- By Markov chain property, probability of state sequence can be found by the formula
- $P(s_{i1}, s_{i2}, \dots s_{ik})$
- $= P(s_{ik}|s_{i(k-1)})P(s_{i(k-1)}|s_{i(k-2)}) \dots \dots P(s_{i2}|s_{i1})P(s_{i1})$
- Example
  - Find probability of {Dry, Dry, Rain, Rain}
  - $P(\{Dry, Dry, Rain, Rain\})$
  - $= P(Rain|Rain) * P(Rain|Dry) * P(Dry|Dry) * P(Dry)$
  - $= 0.3 * 0.2 * 0.8 * 0.6$

Ref: https://cs.wmich.edu/elise/courses/cs626/s12/HiddenMarkovModels.pdf

Recap of NLP ends here

Markov property

- A sequence of random variables $\{X_n\}$ is called a Markov chain if it has the Markov property

$$P\{X_k = i | X_{k-1} = j, X_{k-2}, \cdots, X_1\} = P\{X_k = i | X_{k-1} = j\}$$

- States are usually labelled {(0,)1, 2, ⋯}, and state space can be finite or infinite

Markov property

- First order Markov assumption (memoryless)

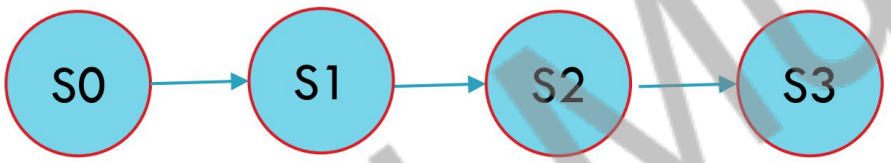
$$P(q_t = i \,|\, q_{t-1} = j, q_{t-2} = k, \cdots) = P(q_t = i \,|\, q_{t-1} = j)$$

- Homogeneity

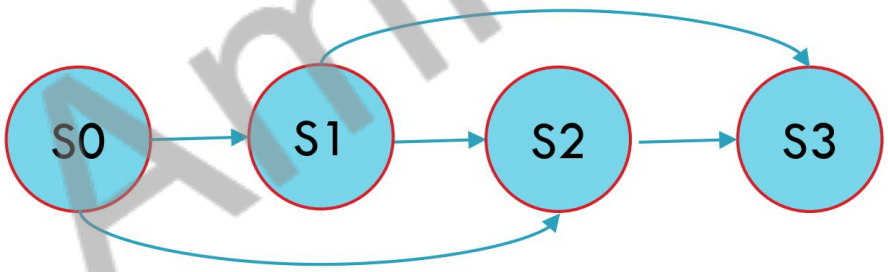
$$P(q_t = i \,|\, q_{t-1} = j) = P(q_{t+l} = i \,|\, q_{t+l-1} = j)$$

Markov's Assumption

- Current state S depends only on K previous states



$P(\{S_t|S_{t-1}\})$
First Order Markov



$P(\{S_t|S_{t-1}, S_{t-2}\})$
Second Order Markov with K=2

In case of First Order Markov, a state provides enough information to make the future conditionally independent of the past

Markov Chain or Markov Process

- Processes satisfying Markov assumption are called Markov processes or Markov chains

Transition Model and Sensor Model

- With the set of state and evidence variables for a given problem decided on, the next step is to specify
  - how the world evolves (the transition model) and
  - how the evidence variables get their values (the sensor model)
- Transition model specifies the probability distribution over the latest state variables, given the previous values, that is $P(\{S_t|S_{0:t-1}\})$
- When we include Markov's Assumption, the Transition Model becomes $P(\{S_t|S_{t-1}\})$

Transition Model

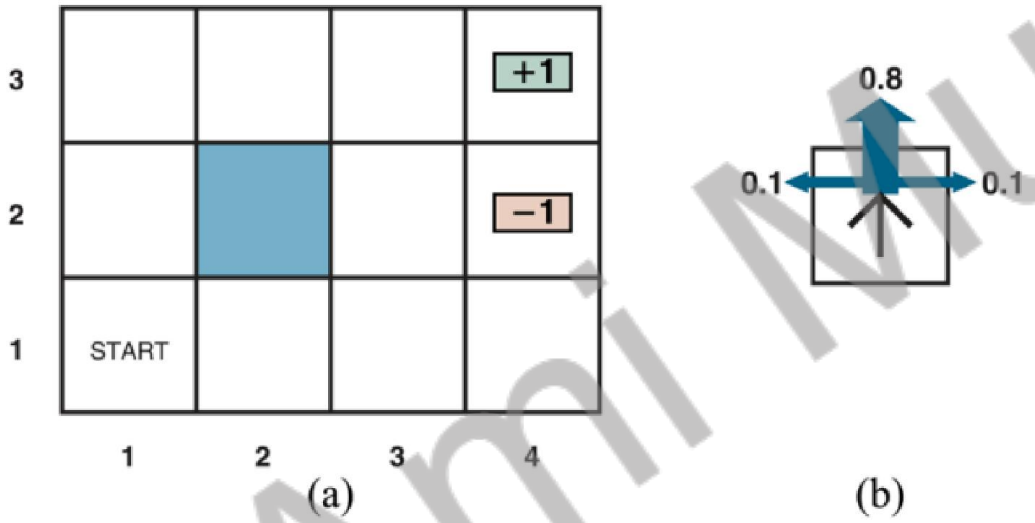
$$P(S_{t+k}|S_t)$$

$$P = \begin{bmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{bmatrix}$$

State transition probability

Sequential Decision Problems

Figure 17.1



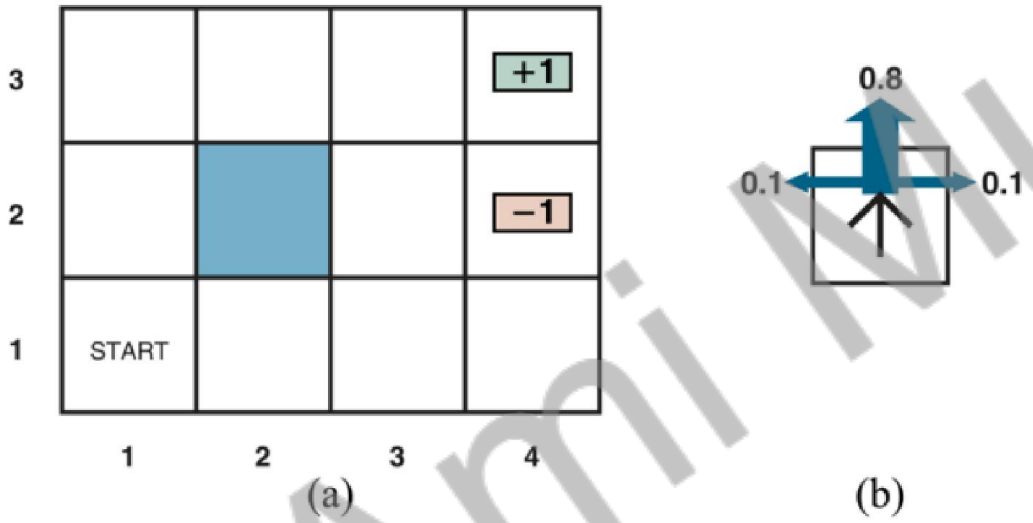
(a) A simple, stochastic 4 × 3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and −1, respectively, and all other transitions have a reward of −0.04.

- Suppose that an agent is situated in the environment
- Beginning in the start state, it must choose an action at each time step
- Interaction with the environment terminates when the agent reaches one of the goal states, marked +1or -1
- Actions in every state are Up, Down, Left, and Right
- We assume for now that the environment is fully observable, so that the agent always knows where it is

Note: Probability of intended action is 0.8
With probability 0.2, the agent moves at right angles to the intended action

Sequential Decision Problems

Figure 17.1

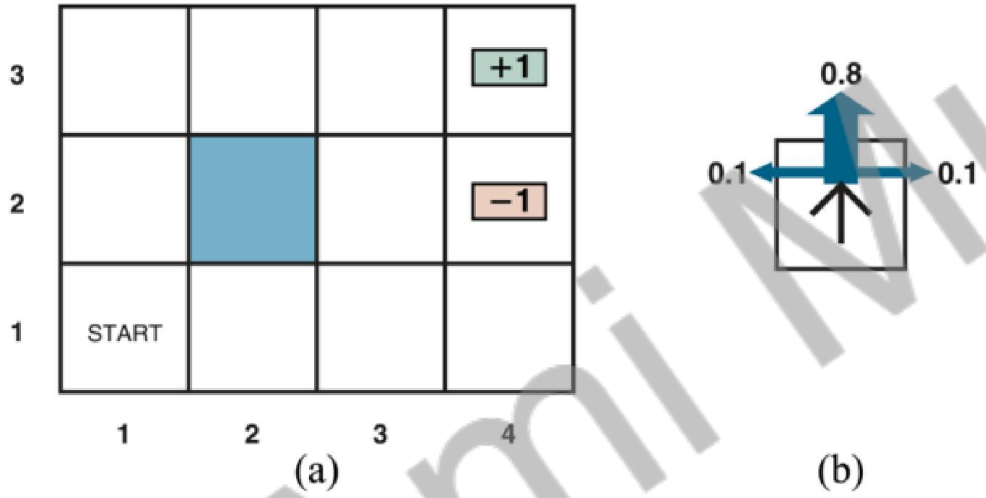


(a) A simple, stochastic 4 × 3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and −1, respectively, and all other transitions have a reward of −0.04.

- If the environment were deterministic, a solution would be easy: $[Up, Up, Right, Right, Right]$
- However in stochastic environment, the actions are unreliable
- Each action achieves the intended effect with probability 0.8, but the rest of the time, the action moves the agent at right angles to the intended direction
- Furthermore, if the agent bumps into a wall, it stays in the same square

Sequential Decision Problems

Figure 17.1

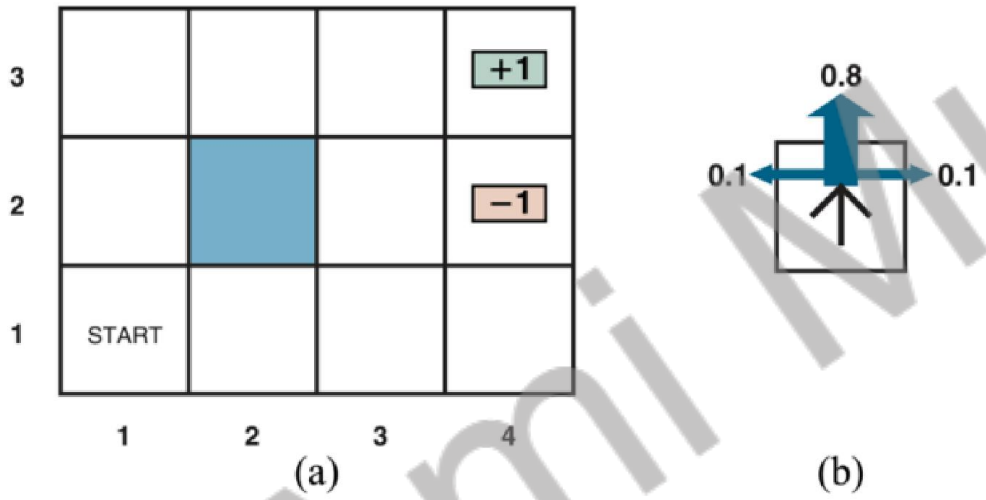


(a) A simple, stochastic 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward $+1$ and -1, respectively, and all other transitions have a reward of -0.04.

- Example
- From the start square (1,1)
- the action Up moves the agent to (1,2) with probability 0.8
- but with probability 0.1, it moves right to (2,1)
- and with probability 0.1, it moves left, bumps into the wall, and stays in (1,1)
- In such an environment, the sequence $[Up, Up, Right, Right, Right]$ goes up around the barrier and reaches the goal state at (4,3) with probability
  - $0.8^5 = 0.32768$

Sequential Decision Problems

Figure 17.1

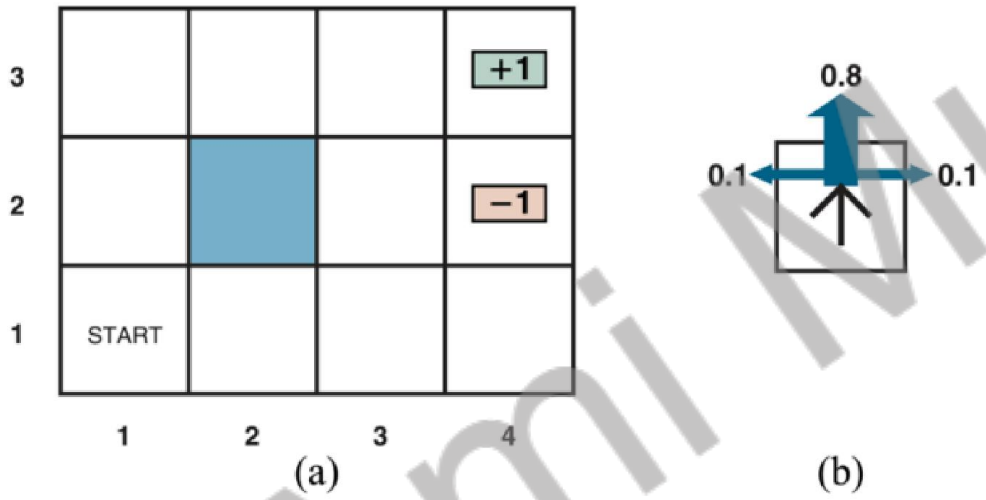


(a) A simple, stochastic 4 × 3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and −1, respectively, and all other transitions have a reward of −0.04.

- Transition model describes the outcome of each action in each state given by
  - $P(s'|(s,a))$
- For every transition from $s\ to\ s'$ to via action a , the agent receives a reward
  - $R(s,a,s')$
- Reward may be positive or negative, bounded by$\pm R_max$
- For this example, the reward is -0.04 for all transitions except those entering terminal states (which have rewards +1 and -1)
- Let us consider value to be the sum of all rewards
  - Example
  - If the agent reaches +1 state after 10 steps then the reward would be
    - $(9 \times 0.01) + 1 = 0.64$

Sequential Decision Problems

Figure 17.1



(a) A simple, stochastic 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward $+1$ and -1, respectively, and all other transitions have a reward of -0.04.

- Negative reward of -0.04 gives the agent an incentive to reach (4,3) quickly
- A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a Markov decision process

Markov Decision Process

- A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a Markov decision process, or MDP
- If consists of
  - a set of states (with an initial state)
  - a set ACTIONS(s) of actions in each state
  - a transition model
  - a reward functionType equation here.
- Methods for solving MDPs usually involve dynamic programming:
  - Dynamic Programming means simplifying a problem by recursively breaking it into smaller pieces and remembering the optimal solutions to the pieces

Ref: Russel and Norwig "Artificial Intelligence a Modern Approach", 4th edition

Markov Decision Process

- Classical Formalization of Sequential Decision Making
  - Actions influence not just immediate rewards, but also
    - subsequent situations, or states, and through those future rewards.
- MDPs involve delayed reward and the need to trade off immediate and delayed reward

How is MDP different from Bandit Problem?

- In bandit problems we estimated the value $q_*(a)$ of each action a
- In MDPs we estimate the value $q_*(s, a)$ of each action a in each state s
  - Or we estimate the value $v_*(s)$ of each state given optimal action selections

Definition of MDP

- We have states and actions in each state
- Reward $R_t = R(S_t, A_t)$
- Transition model $P(S_t|S_{t-1}, A_{t-1})$
- Discount factor γ
- Horizon (episode or time steps)
- Goal is to map state to action to maximize the reward (Optimal Policy)

Modelling MDP

- We have states and actions in each state
- Reward $R_t = R(S_t, A_t)$
- Transition model $P(S_t|S_{t-1}, A_{t-1})$
- Discount factor γ
- Horizon (episode or time steps)
- Goal is to map state to action to maximize the reward (Optimal Policy)

$$(S, A, P(S_t|S_{t-1}, A_{t-1}), R(S_t, A_t), \gamma)$$

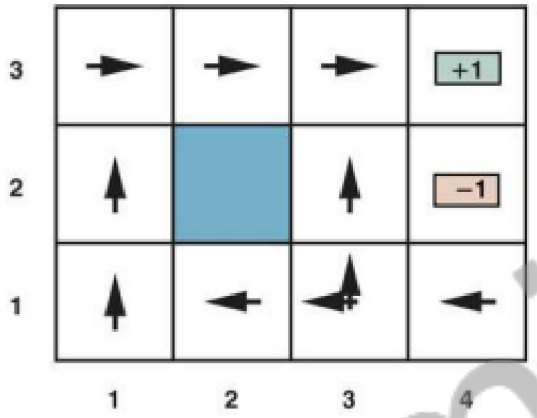
Policy

- What the agent should do for any state that the agent might reach is called a policy
- It is traditional to denote a policy by π, $\pi(s)$ is the action recommended by the policy for state
- No matter what the outcome of the action, the resulting state will be in the policy, and the agent will know what to do next

Policy

- Each time a given policy is executed starting from the initial state, the stochastic nature of the environment may lead to a different environment history
- The quality of a policy is therefore measured by the expected value of the possible environment histories generated by that policy
- An optimal policy (π_*) is a policy that yields the highest expected value

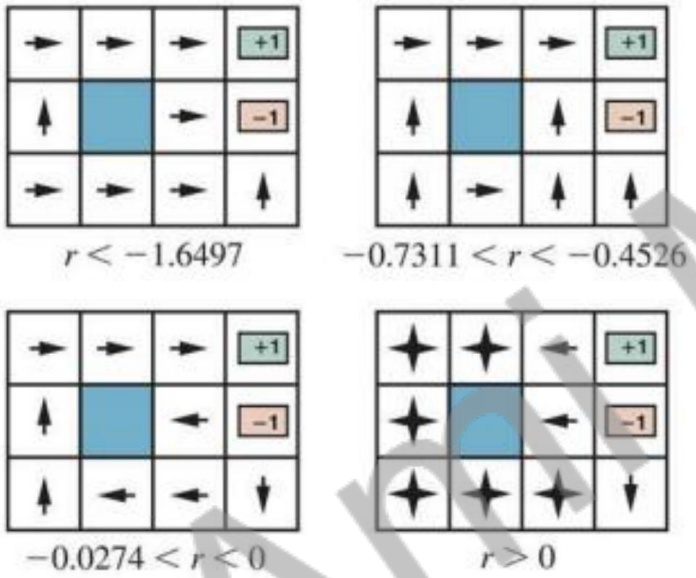
Example



- Balance of risk and reward changes depending on the value of $r = R(s, a, s')$ transitions between nonterminal states
- The policies shown in the given figure are optimal for

$$-0.0850 < r < -0.0273$$

Example

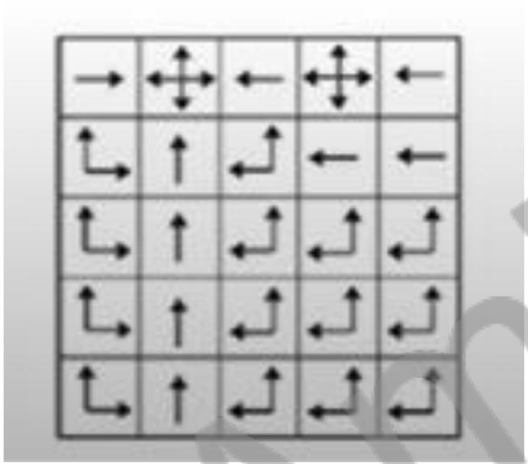


$r < -1.6497$ $\quad -0.7311 < r < -0.4526$

$-0.0274 < r < 0$ $\quad r > 0$

(b)

Goal of MDP



- The goal of MDP and RL is to find the best policy π
- Policy(π)
  - Action A_t taken in state S_t
  - Precisely π is a probability that action A_t is taken in S_t
- Under Markov assumption, this policy is given as
  - $A_t = \pi(\mathrm{S_t})$

Agent Environment Interface

- Frame a problem of learning from interaction in environment to achieve goal
- Agent
  - learner or decision maker
- Environment
  - Things the agent interacts with
  - Comprises of everything outside the agent
- Rewards
  - Special numeric values that the environment gives for every action
  - Agent seeks to maximize these rewards
  - It is a scalar
  - Outside direct control of agent
  - Bounded

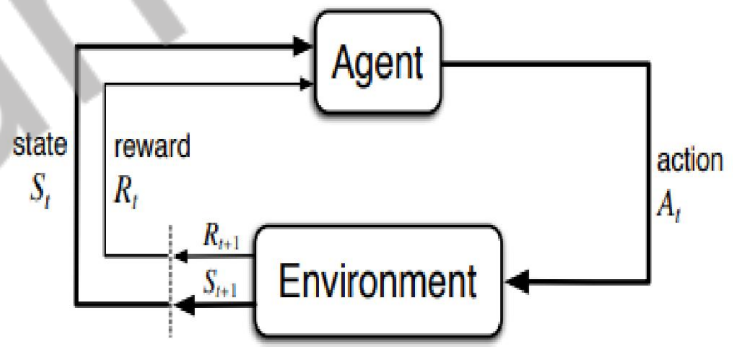


Figure 3.1: The agent–environment interaction in a Markov decision process.

Agent Environment Interface

- Agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$.
- At each time step t, the agent receives some representation of the environment's state, $S_t \epsilon S$, and on that basis selects an action, $A_t \epsilon A(s)$
- One time step later, in part as a consequence of its action, the agent receives a numerical reward, $R_{t+1} \epsilon R$, and finds itself in a new state, S_{t+1}
- MDP and agent together thereby give rise to a sequence or trajectory
  - $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$

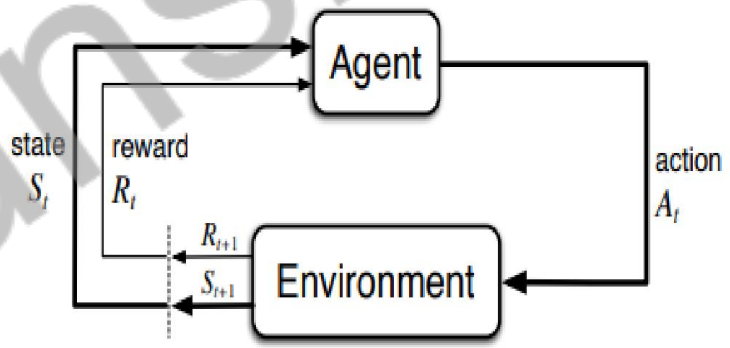
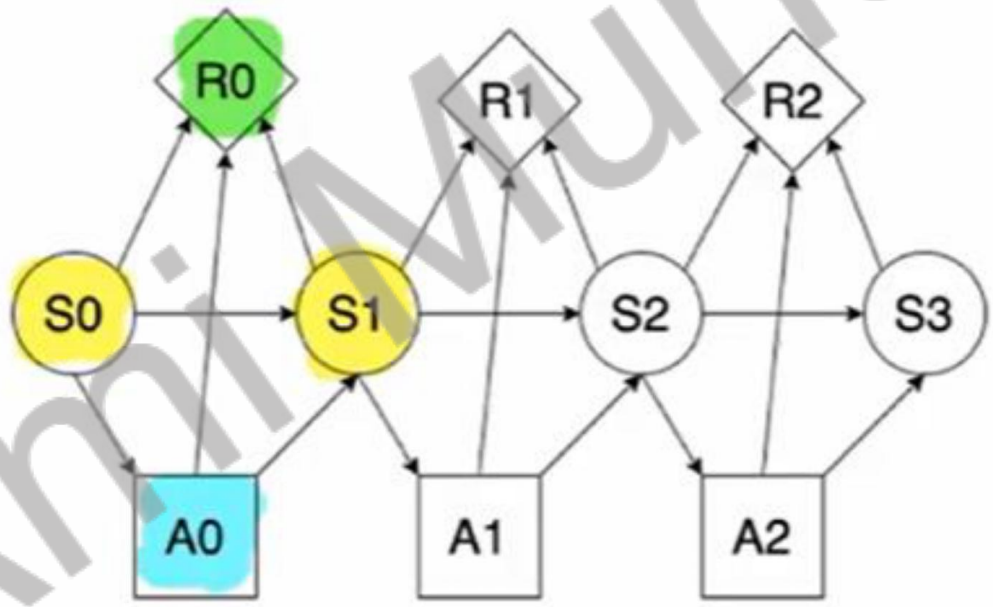


Figure 3.1: The agent–environment interaction in a Markov decision process.

MDP



Finite MDP

- In a finite MDP, the sets of states, actions, and rewards $(S, A, and\ R)$ all have a finite number of elements
- Random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action
- That is, for particular values of these random variables, $s' \epsilon\ S$ and $r\ \in\ R$, there is a probability of those values occurring at time t, given particular values of the precedina state and action:

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\},$$

$$\text{for all } s', s \in \mathcal{S},\ r \in \mathcal{R},\ \text{and } a \in \mathcal{A}(s)$$

The function p defines the dynamics of the MDP

The dynamics function $p : S \times R \times S \times A \rightarrow [0, 1]$ is an ordinary deterministic function of four arguments

Finite MDP

p specifies a probability distribution for each choice of s and a

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s).$$

Finite MDP

- In a Markov decision process, the probabilities given by p completely characterize the environment's dynamics
- That is, the probability of each possible value for S_t and R_t depends on
  - the immediately preceding state and action, S_{t-1} and A_{t-1} and
  - not on earlier states and actions
- This is best viewed as a restriction not on the decision process, but on the state
- State must include information about all aspects of the past agent–environment interaction that make a difference for the future
- If it does, then the state is said to have the Markov property

Finite MDP

- Computation of state-transition probabilities
  - $p: S \times S \times A \rightarrow [0,1]$

$$p(s'|s,a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a).$$

Finite MDP

- Computation of expected rewards for state–action pairs as a two-argument function
  - $r: S \times A \rightarrow R$

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a),$$

Assumption: Rewards are stationary and does not change with time for same parameters of state

Finite MDP

- Computation of expected rewards for state–action–next-state triples as a three-argument function
  - $r: S \times A \times S \to R$

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a, S_t=s'] = \sum_{r\in\mathcal{R}} r \frac{p(s',r\mid s,a)}{p(s'\mid s,a)}.$$

Infinite MDP

- What if the process in MDP is infinite??
- Use discounted factor γ to discount the reward

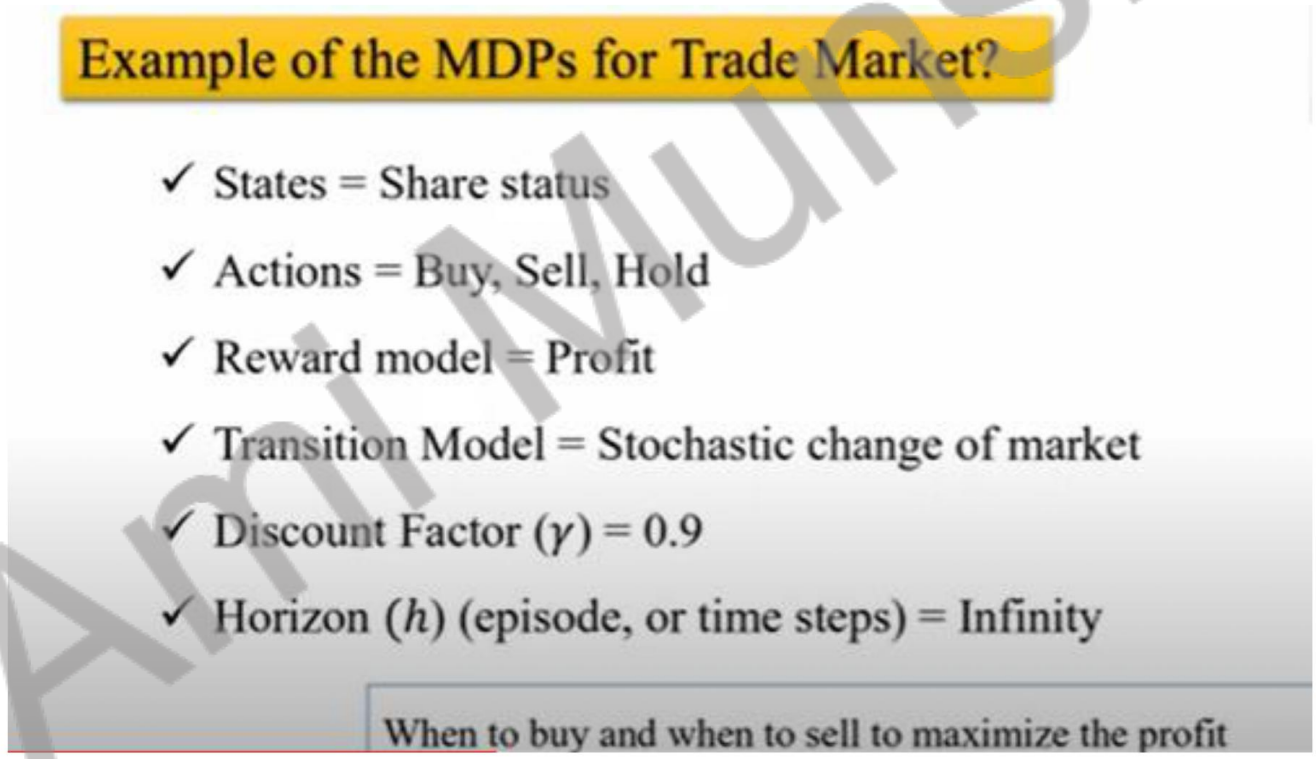
Reward

- Let $R(S)$ be the reward for entering a state S
- Total Reward
  - $R(S_0) + R(S_1) + R(S_2) + \cdots$
  - If sum is infinite we can not compare the two policies in MDP
- Average Reward
  - $\lim_{n\to\infty} \frac{1}{n}(R(S_0) + R(S_1) + R(S_2) + \cdots)$
  - Average is zero over infinite time, hence we can not compare the two policies in MDP
- Discounted Reward
  - $\sum \gamma^t R(S_t, A_t)$ *where* γ *is between* $[0,1]$
  - $R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \cdots$

Can be used for Finite MDP

Can be used for infinite MDP

Example of Infinite MDP

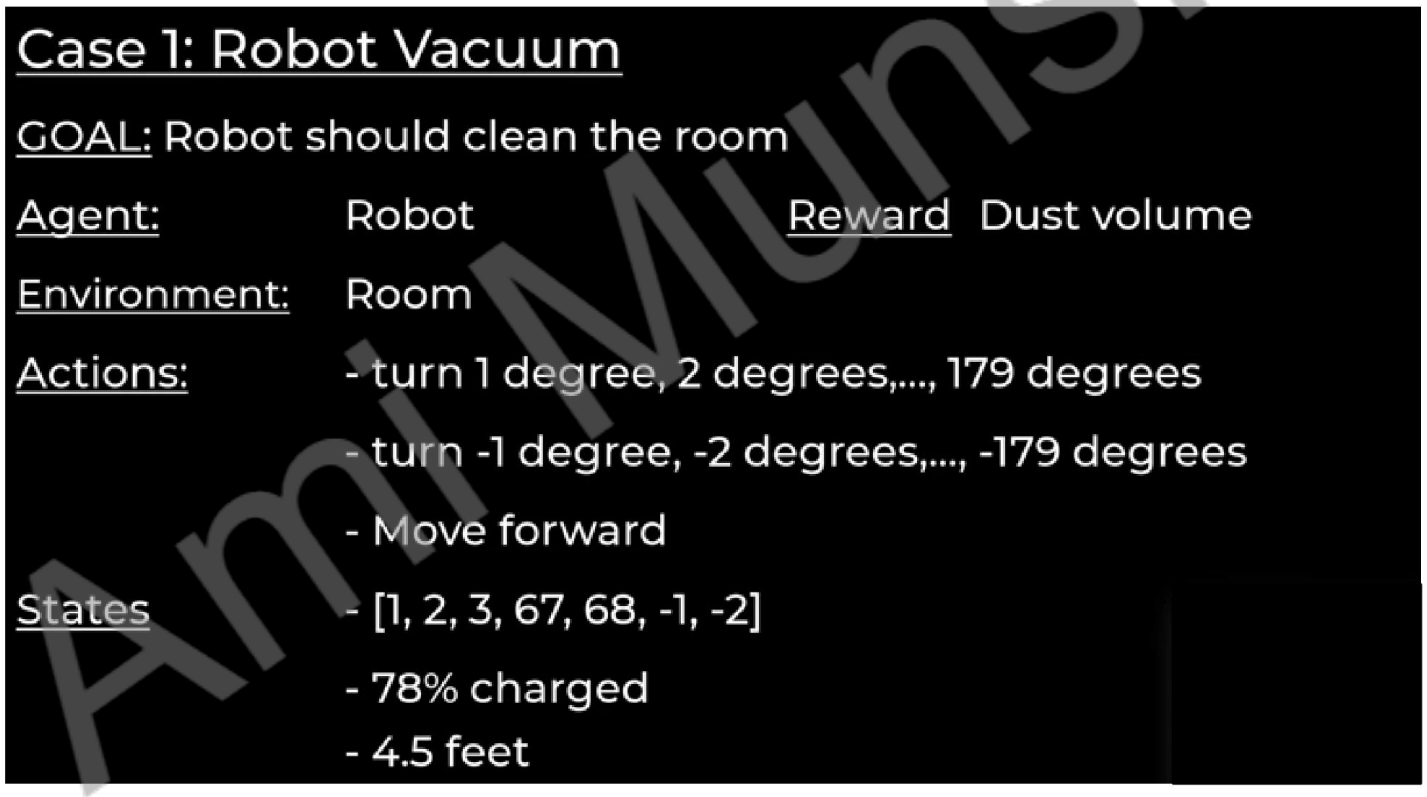


Ref: https://youtu.be/nMlRhsgcTpY?feature=shared

Assumption for MDP

- Fully observable and discrete environment
- We have complete model and transition dynamics of the environment
- Stochastic process (uncertainty)
- Action selection is sequential and can depend on previous actions

Example- Robot Vacuum Cleaner



Ref: https://www.youtube.com/watch?v=sWKSOY9DCiI

Example: Human riding a bike

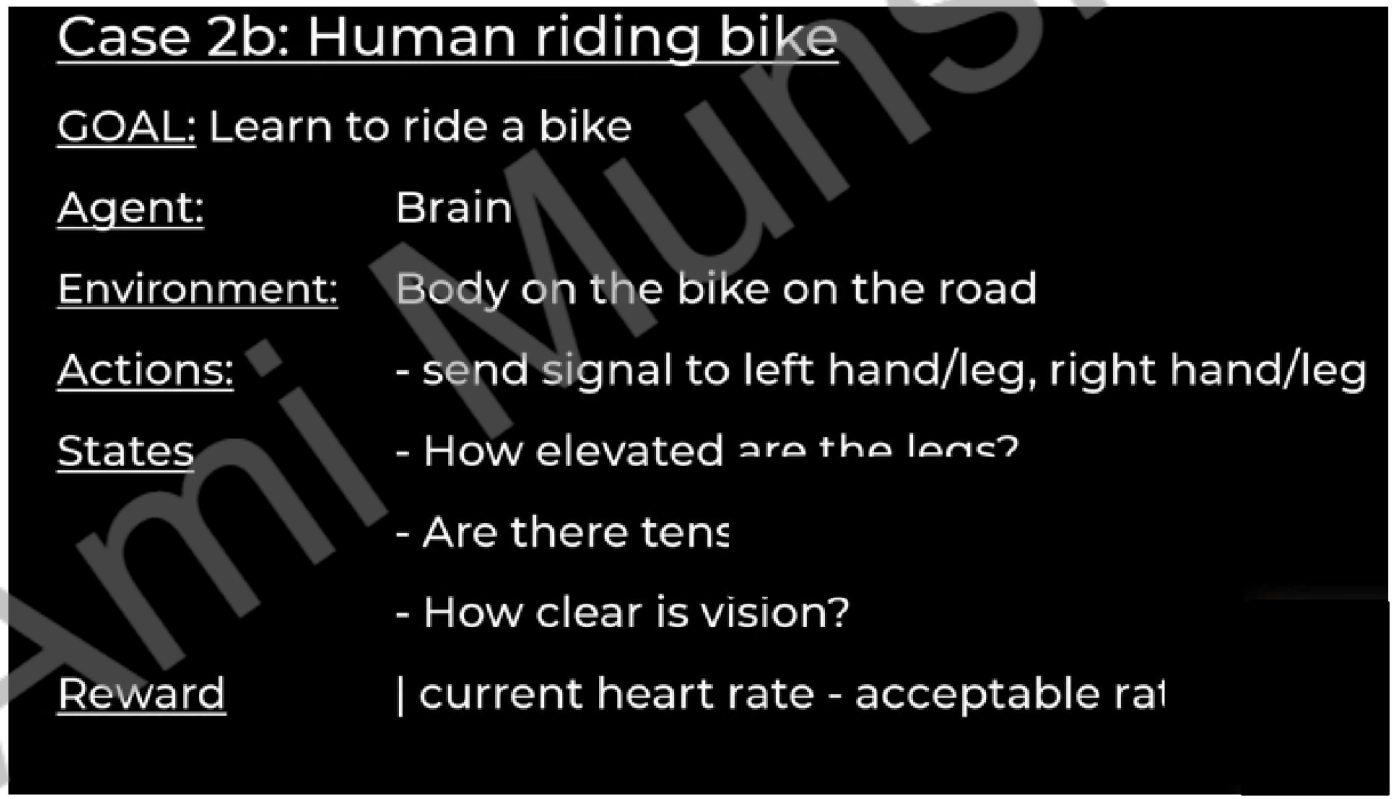
Case 2: Human riding bike

GOAL: Learn to ride a bike

Agent:	Human
Environment:	Bike on the road
Actions:	- peddle, brake, lean left, lean right, turn
States	- Is there an obstacle 12 feet in front?
	- Is there an obstacle to the left?
	- Is there an obstacle to the right?
Reward	0 when straight, -50 when slant
	-100 when crash or fallen down

Ref: https://www.youtube.com/watch?v=sWKSOY9DCiI

Example: Human riding a bike



Ref: https://www.youtube.com/watch?v=sWKSOY9DCiI

Example Bioreactor

Example 3.1: Bioreactor Suppose reinforcement learning is being applied to determine moment-by-moment temperatures and stirring rates for a bioreactor (a large vat of nutrients and bacteria used to produce useful chemicals). The actions in such an application might be target temperatures and target stirring rates that are passed to lower-level control systems that, in turn, directly activate heating elements and motors to attain the targets. The states are likely to be thermocouple and other sensory readings, perhaps filtered and delayed, plus symbolic inputs representing the ingredients in the vat and the target chemical. The rewards might be moment-by-moment measures of the rate at which the useful chemical is produced by the bioreactor. Notice that here each state is a list, or vector, of sensor readings and symbolic inputs, and each action is a vector consisting of a target temperature and a stirring rate. It is typical of reinforcement learning tasks to have states and actions with such structured representations. Rewards, on the other hand, are always single numbers. ■

Identify Environment, State, Action and Reward

Example- Bioreactor

- RL is applied to maximize the production of a useful chemical by controlling
  - bioreactor's temperature and
  - stirring rate
- States include sensor readings, such as
  - Thermocouple readings (temperature sensors)
  - Other sensory readings (e.g., pH levels, oxygen concentration)
  - Symbolic inputs representing the ingredients in the vat and the target chemical

Example- Bioreactor

- Actions
  - Target temperature for bioreactor
  - Target stirring rates
- Each action is a vector containing chosen temperature and stirring rate

Example- Bioreactor

- Rewards
  - Reward might be a moment-by-moment measure of the rate at which the useful chemical is produced
  - This means that the faster or more efficiently the chemical is produced, the higher the reward

Example- Bioreactor

- Environment-Entire bioreactor system, including
  - The vat of nutrients and bacteria
  - The sensors providing readings
  - The heating elements and motors controlled by the lower-level systems
- Environment receives the actions (target temperature and stirring rate) from the agent and updates the state of the bioreactor accordingly, also providing the corresponding reward based on the chemical production rate.

Example- Pick and Place Robot

Example 3.2: Pick-and-Place Robot Consider using reinforcement learning to control the motion of a robot arm in a repetitive pick-and-place task. If we want to learn movements that are fast and smooth, the learning agent will have to control the motors directly and have low-latency information about the current positions and velocities of the mechanical linkages. The actions in this case might be the voltages applied to each motor at each joint, and the states might be the latest readings of joint angles and velocities. The reward might be +1 for each object successfully picked up and placed. To encourage smooth movements, on each time step a small, negative reward could be given as a function of the moment-to-moment jerkiness of the motion. ■

Identify Environment, State, Action and Reward

Devise Examples

Exercise 3.1 Devise three example tasks of your own that fit into the MDP framework, identifying for each its states, actions, and rewards. Make the three examples as *different* from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples. □

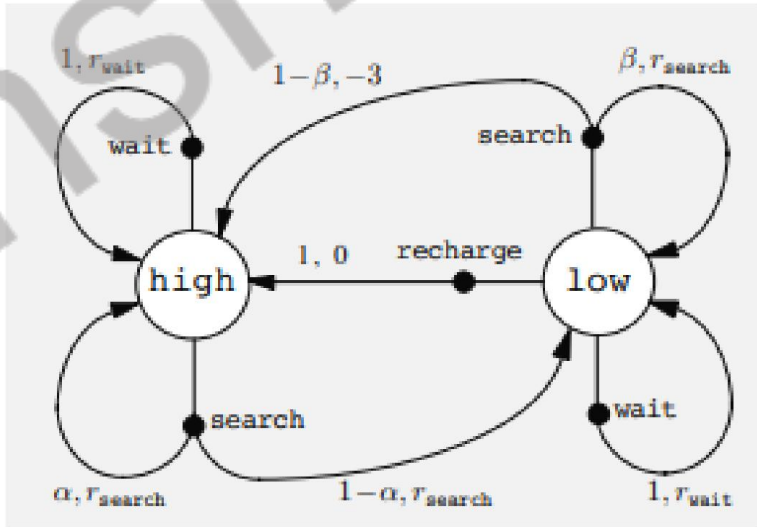
- Hint:
  - Chess game
  - Tic Tac Toe
  - Investment Portfolio
  - Golf
  - Shoe tying robot

Draw state diagram and state table

A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. To make a simple example, we assume that only two charge levels can be distinguished, comprising a small state set $\mathcal{S} = \{\texttt{high}, \texttt{low}\}$. In each state, the agent can decide whether to (1) actively **search** for a can for a certain period of time, (2) remain stationary and **wait** for someone to bring it a can, or (3) head back to its home base to **recharge** its battery. When the energy level is **high**, recharging would always be foolish, so we do not include it in the action set for this state. The action sets are then $\mathcal{A}(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$ and $\mathcal{A}(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$.

The rewards are zero most of the time, but become positive when the robot secures an empty can, or large and negative if the battery runs all the way down. The best way to find cans is to actively search for them, but this runs down the robot's battery, whereas waiting does not. Whenever the robot is searching, the possibility exists that its battery will become depleted. In this case the robot must shut down and wait to be rescued (producing a low reward). If the energy level is `high`, then a period of active search can always be completed without risk of depleting the battery. A period of searching that begins with a `high` energy level leaves the energy level `high` with probability α and reduces it to `low` with probability $1 - \alpha$. On the other hand, a period of searching undertaken when the energy level is `low` leaves it `low` with probability β and depletes the battery with probability $1 - \beta$. In the latter case, the robot must be rescued, and the battery is then recharged back to `high`. Each can collected by the robot counts as a unit reward, whereas a reward of -3 results whenever the robot has to be rescued. Let $r_{\texttt{search}}$ and $r_{\texttt{wait}}$, with $r_{\texttt{search}} > r_{\texttt{wait}}$, denote the expected number of cans the robot will collect (and hence the expected reward) while searching and while waiting respectively. Finally, suppose that no cans can be collected during a run home for recharging, and that no cans can be collected on a step in which the battery is depleted. This system is then a finite MDP, and we can write down the transition probabilities and the expected rewards, with dynamics as indicated in the table on the left:

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-



Note that there is a row in the table for each possible combination of current state, s, action, $a \in \mathcal{A}(s)$, and next state, s'. Some transitions have zero probability of occurring, so no expected reward is specified for them. Shown on the right is another useful way of

summarizing the dynamics of a finite MDP, as a *transition graph*. There are two kinds of nodes: *state nodes* and *action nodes*. There is a state node for each possible state (a large open circle labeled by the name of the state), and an action node for each state–action pair (a small solid circle labeled by the name of the action and connected by a line to the state node). Starting in state s and taking action a moves you along the line from state node s to action node (s, a). Then the environment responds with a transition to the next state's node via one of the arrows leaving action node (s, a). Each arrow corresponds to a triple (s, s', a), where s' is the next state, and we label the arrow with the transition probability, $p(s'|s, a)$, and the expected reward for that transition, $r(s, a, s')$. Note that the transition probabilities labeling the arrows leaving an action node always sum to 1.

Goal, Rewards, Returns

Goals and rewards

- For each step taken by the agent there is a reward $R \epsilon \mathbb{R}$
- Agent's goal is to maximize the total amount of reward it receives
- This means maximizing not immediate reward, but cumulative reward in the long run
- We can clearly state this informal idea as the reward hypothesis
  - **That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)**
- Use of a reward signal to formalize the idea of a goal is one of the most distinctive features of reinforcement learning

Examples of formulating goals in terms of rewards

- To make a robot learn to walk
  - reward on each time step proportional to the robot's forward motion
- In making a robot learn how to escape from a maze
  - the reward is often -1 every time step that passes prior to escape
- To make a robot learn to find and collect empty soda cans for recycling
  - reward of zero most of the time
  - a reward of +1 for each can collected
  - negative rewards when it bumps into things or when somebody yells at it
- To learn to play checkers or chess
  - the natural rewards are +1 for winning
  - -1 for losing
  - 0 for drawing
  - and for all non terminal positions
- **In the above examples, agent always learns to maximize its reward**

Formulating goals through rewards

- Reward signal is your way of communicating to the agent what you want achieved, not how you want it achieved- Why???
- For answer, read Chapter3, Page 34, Sutton

Rewards and Episodes

- Agent's goal is to maximize the cumulative reward it receives in the long run
- How to define this formally??
- If the sequence of rewards received after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$, then what precise aspect of this sequence do we wish to maximize?
- We seek to maximize the expected return
- Return, denoted G_t, is defined as some specific function of the reward sequence
- The simplest form of G_t is $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$
- Here T is the final time step

Rewards and Episodes

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

- This approach makes sense in applications in which
  - there is a natural notion of final time step, that is
  - when the agent–environment interaction breaks naturally into subsequences, which we call episodes
- Example
  - plays of a game
  - trips through a maze
  - any sort of repeated interaction

Episodic task

- Each episode ends in a special state called the terminal state
- This is followed by a reset to a standard starting state or to a sample from a standard distribution of starting states
- Even if you think of episodes as ending in different ways, such as winning and losing a game, the next episode begins independently of how the previous one ended
- Thus the episodes can all be considered to end in the same terminal state, with different rewards for the different outcomes
- Tasks with episodes of this kind are called episodic tasks

Continuing Task

- In many cases the agent–environment interaction does not break naturally into identifiable episodes, but goes on continually without limit
- For example,
  - formulate an on-going process-control task, or an application to a robot with a long life span
- We call these continuing tasks
- In this case the return formation is difficult as $T=\infty$
- Return that we are trying to maximize is also **infinite**

Discount Rate

- When T=∞, we use the concept of discounting
- According to this approach,
  - Agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized
  - In particular, it chooses A_t to maximize the expected discounted return as follows

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

- Here $0 \leq \gamma \leq 1$ is the discount rate
- Discount rate determines the present value of future rewards:
  - a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately

Discount Rate

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

- If $\gamma < 1$, the infinite sum has finite value
- If $\gamma = 0$, the agent is being **myopic-** being concerned only with maximizing immediate reward
  - its objective in this case is to learn how to choose A_t so as to maximize only R_{t+1}
- But in general, acting to maximize immediate reward can reduce access to future rewards so that the return is reduced
- As γ approaches 1, the return objective takes future rewards into account more strongly; the agent becomes more **farsighted**

Returns

- Returns at successive time steps are related to each other in a way that is important for the theory and algorithms of reinforcement learning

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

- This works for all the time steps $t < T$, even if termination occurs at $t + 1$, provided we make $G_T = 0$

Returns

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

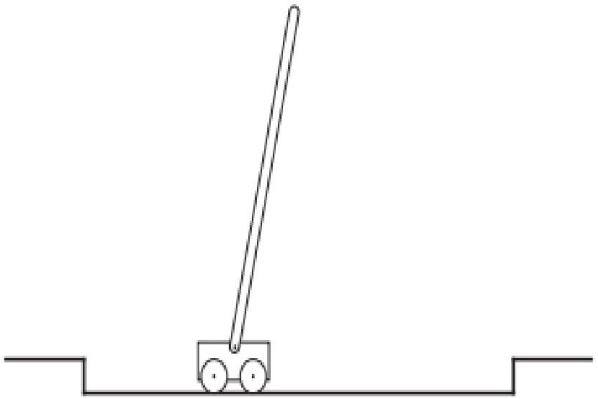
Here if the reward is constant +1 and $\gamma < 1$ then the return is nonzero and constant given by

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}.$$

Here if the reward is constant R_k and $\gamma < 1$ then the return is nonzero and constant given by

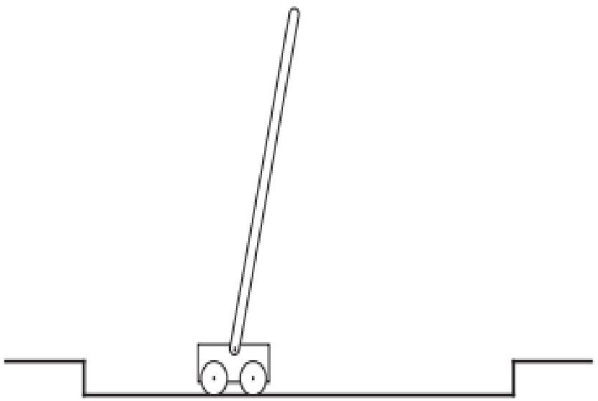
$$G_t = \sum_{k=0}^{\infty} \gamma^k = \left(\frac{1}{1-\gamma}\right) * R_k$$

Example: Pole balancing



- Goal: Avoid Failure
  - Failure is said to occur if
    - the pole falls past a given angle from vertical or
    - if the cart runs off the track

Example: Pole balancing



- Episodic task ends upon failure
  - reward = +1 for each step before failure
  - return = number of steps before failure
- Continuing task with discounted returns
  - reward=-1upon failure
  - return $=-\gamma^k$, for k steps before failure

In either case, return is maximized by avoiding failure for as long as possible

How to calculate return at time t?

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \qquad \gamma \epsilon [0,1)$$

γ	Reward sequence R_k	Return
0.5 (or any between [0,1))	1,0, 0, 0, 0	1
0.5	1,0,2,0,0,0,0	$1 + 0 + 0.5^2 * 2 = 1.5$
0.9	1,0,2,0,0,0,0	$1 + 0 + 0.9^2 * 2 = 2.62$

Example

Exercise 3.8 Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are G_0, G_1, ..., G_5? Hint: Work backwards. □

Exercise 3.8 Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are $G_0, G_1, \ldots, G_5$? Hint: Work backwards. □

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

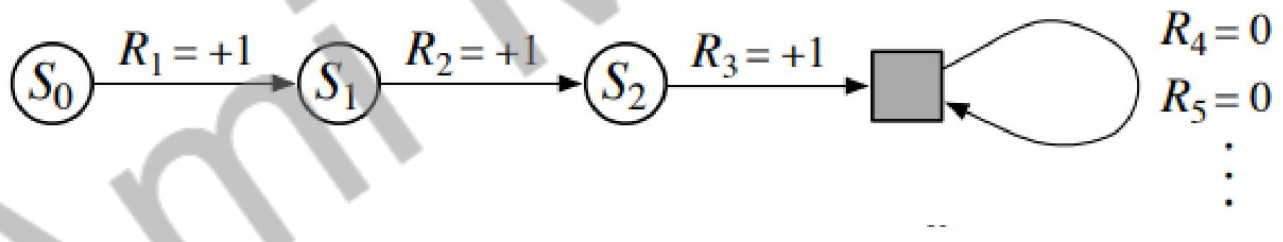
$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

- $G_5 = R_6 + \gamma R_7 + \cdots = 0$
- $G_4 = R_5 + \gamma R_6 + \cdots = 2$
- $G_3 = R_4 + \gamma R_5 + \gamma^2 R_6 + \cdots = 3 + 0.5 * 2 = 4$
- $G_2 = 6 + 0.5 * 3 + 0.5^2 * 2 = 8$
- $G_1 = 2 + 0.5 * 6 + 0.5^2 * 3 + 0.5^3 * 2$
- $G_0 = -1 + 0.5 * 2 + +0.5^2 * 6 + 0.5^3 * 3 + 0.5^4 * 2$

- $G_5 = R_6 + \gamma G_6 = 0$
- $G_4 = R_5 + \gamma G_5 = 2 + 0 = 2$
- $G_3 = R_4 + \gamma G_4 = 3 + 0.5 * 2 = 4$
- $G_2 = R_3 + \gamma G_3 = 6 + 0.5 * 4 = 8$
- $G_1 = R_2 + \gamma G_2 = 2 + 0.5 * 8 = 6$
- $G_0 = R_1 + \gamma G_1 = -1 + 0.5 * 6 = 2$

Unified Notation for Episodic and Continuing Tasks

- In episodic tasks, we number the time steps of each episode starting from zero
- We usually do not have to distinguish between episodes, so instead of writing $S_{t,j}$ for states in episode j, we write just S_t
- Think of each episode as ending in an absorbing state that always produces reward of zero:



We can cover all cases by writing $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k,$

It includes possibility that $T = \infty$ and $\gamma = 1$ $(but\ not\ both)$

Why???

Example

- Suppose if $\gamma = 0.5$ and reward sequence are all 1s for $T = \infty$. Calculate the return.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

- If reward is constant and $\gamma < 1$ *the* $G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$.

- $G = \frac{1}{1-\gamma} R_k = \frac{1}{1-0.5} = 2$

Example

Suppose $\gamma = 0.5$ and the reward sequence is

$$R_1 = 1, R_2 = 13, R_3 = 13, R_4 = 13, \text{ and so on, all 13s}$$

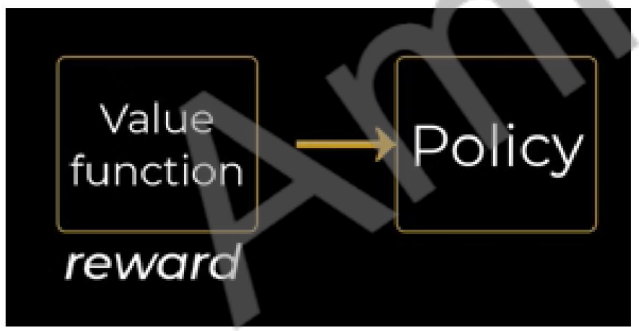
$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k,$$

- $G_3 = \sum_{k=3+1}^{\infty} \gamma^{k-3-1} \times 13 = \sum_{k=4}^{\infty} \gamma^{k-4} \times 13 = \left(\frac{1}{1-0.5}\right) * 13 = 26$
- $G_2 = \sum_{k=2+1}^{\infty} \gamma^{k-2-1} \times 13 = \sum_{k=3}^{\infty} \gamma^{k-3} \times 13 = \left(\frac{1}{1-0.5}\right) * 13 = 26$
- $G_1 = R_2 + \gamma G_2 = 13 + 0.5 * 26 = 26$
- $G_0 = R_1 + \gamma G_1 = 1 + 0.5 * 26 = 14$

RL Algorithms

Value based

- Policy is how the agent behaves in given situation
- It quantifies reward and determines policy based on the value



Policy based

- Determines optimal policy

Ref: https://www.youtube.com/watch?v=9JZID-h6ZJ0

Policy and Value Function

- RL algorithms involve estimating value functions—functions of states (or of state–action pairs)
  - that estimate how good it is for the agent to be in a given state
  - or how good it is to perform a given action in a given state
- Notion of "how good" here is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return
- Rewards the agent can expect to receive in the future depend on what actions it will take
- Accordingly, value functions are defined with respect to particular ways of acting, called policies

Policy and Value Function

- Policy is a mapping from states to probabilities of selecting each possible action
- Deterministic Stationary Policy
  - Can be defined by a mapping π, which maps states to actions, $a = \pi(s)$
- Stochastic Stationary Policy
  - Defined by the probability distribution of selecting each possible action, denoted as $\pi(a \mid s)$
- If the agent is following policy π at time t, then
  - $\pi(a|s)$ is the probability that action $A_t = a$ is selected if $S_t = s$
- In RL, agent's policies can change based on experience that the agent gets

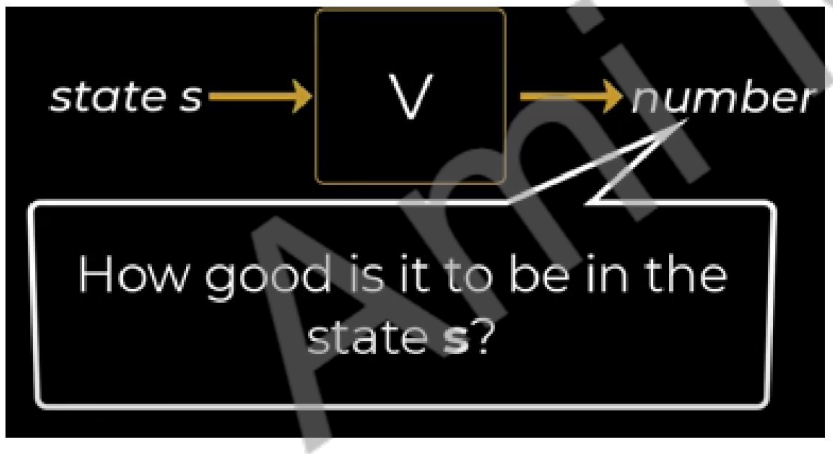
Value based functions

Ref: https://www.youtube.com/watch?v=9JZID-h6ZJ0

Value based functions

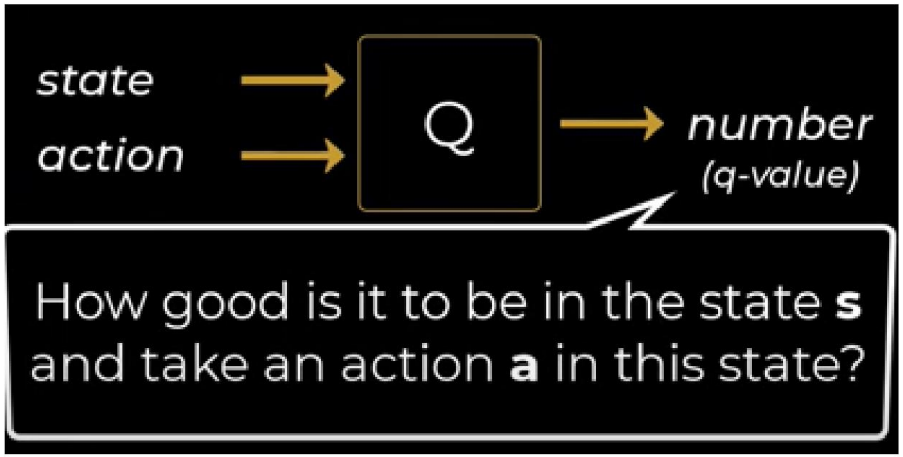
State Value Function V(s)

- State is snapshot of an environment



State Action Value function Q(s,a)

- Action is decision taken by the environment



Ref: https://www.youtube.com/watch?v=9JZID-h6ZJ0

Value function

- Value function of state s under policy π, is denoted by $v_\pi(s)$
  - It is the expected return when starting in state s, and following policy π there after
- For MDP, $v_\pi(s)$ is formally defined as

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

State Value function

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

- Note:
  - Value of the terminal state, if any, is always zero
  - We call the function v_π the state-value function for policy π

State Value Function

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

Ref: https://youtu.be/CTPHADvQxSs?feature=shared

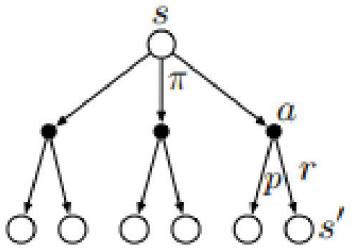
State Action Value function

- Similarly, we define the value of taking action a in state s under a policy π, denoted $q_\pi(s,a)$, as the expected return starting from s, taking the action a, and thereafter following policy π
- We call q_π the action-value function for policy π

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t=s, A_t=a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \,\middle|\, S_t=s, A_t=a\right]$$

Bellman Equation for v_π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

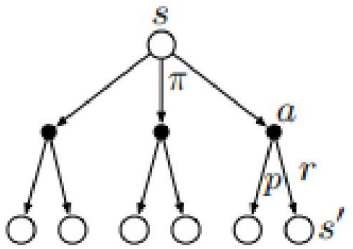


Backup diagram for v_π

- It expresses a relationship between the value of a state and the values of its successor states

Bellman Equation for v_π

- It expresses a relationship between the value of a state and the values of its successor states
- Think of looking ahead from a state to its possible successor states, as suggested by the diagram to the right
- Each open circle represents a state and each solid circle represents a state–action pair
- Starting from state s, the root node at the top, the agent could take any of some set of actions— three are shown in the diagram—based on its policy π
- From each of these, the environment could respond with one of several next states, s_0, along with a reward, r, depending on its dynamics given by the function p
- Bellman equation averages over all the possibilities, weighting each by its probability of occurring
- It states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way



Backup diagram for v_π

Summary of all the equations

1) State-value function is defined as the expectation value of future total return given a current state s and a policy π.

$$v_\pi(s) \equiv \mathbb{E}_\pi[G_t|S_t = s], \text{ for all } s \in \mathcal{S}$$

2) Action-value function is defined as the expectation value of future total return give a current state s and action, plus a policy π.

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

3) Bellman equations (recursive relationship). With the aid of backup diagram, we can easily calculate the state-value function and the action-value function for a given policy π, in a recursive way:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$$

Optimal Policies and Optimal Value Functions

- A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states
- In other words, $\pi \geq \pi'$ if and only if
  - $v_\pi(s) \geq v'_\pi(s) for\ all\ s \epsilon S$
- This is an **optimal policy**

Optimal Policies and Optimal Value Functions

- Optimal state value function for optimal policy is given by

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

for all $s \in \mathcal{S}$.

- Optimal state action value function for optimal policy is given by

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a),$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a].$$

Bellman optimality equation for v_π

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big].
\end{aligned}
$$

- Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state

Bellman optimality equation for q_*

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\
&= \sum_{s',r} p(s', r \,|\, s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big].
\end{aligned}
$$

Backup diagrams for v_* and q_*

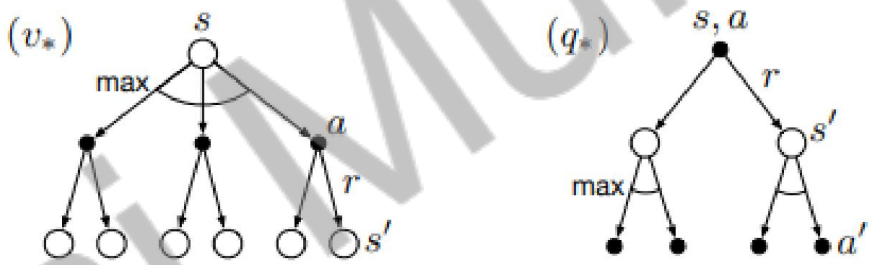


Figure 3.4: Backup diagrams for v_* and q_*

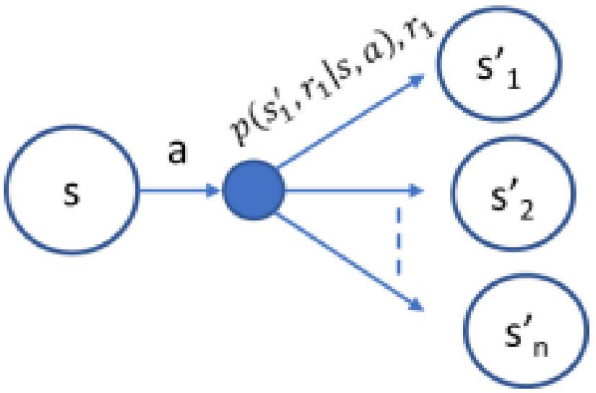
Bellman Equation for solving problems

- $V(s) = \max_{a \in A} \{ R(s) + \gamma \sum P(s' | (s, a)) \; V(s') \}$

Example- Dice Game

- At any time, the game is specified by two states: IN and END. When the game in the state IN, the agent can choose an action from the set $\mathcal{A} = \{STAY, QUIT\}$. If QUIT is selected, then the game will go to state END and the agent receive a reward \$10 with a probability of 1. If STAY is selected, the game will stay at IN state and the agent receive a reward \$4 with a probability of $2/3$, or transition to state END and the agent receive a reward \$4 with a probability of $1/3$.

Ref: Weidong Kuang, Fundamentals of RL, University of Texas, 2021



(b) general diagram for MDP

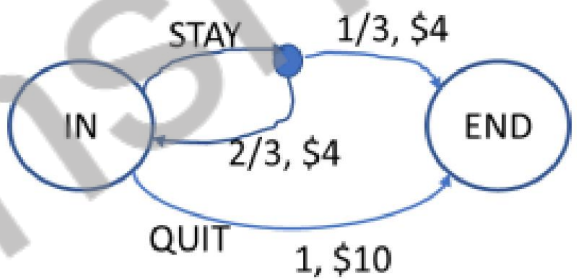


Fig.2 (a) an MDP

Example

$$p(IN, 4|IN, STAY) = 2/3.$$

$$p(IN, 10|IN, QUIT) = 0.$$

If you play the game, will you choose QUIT or STAY to get the maximal return in average

Agent use a random policy (uniform) on selecting the action: STAY (S) or QUIT (Q)
Calculate State Value Function

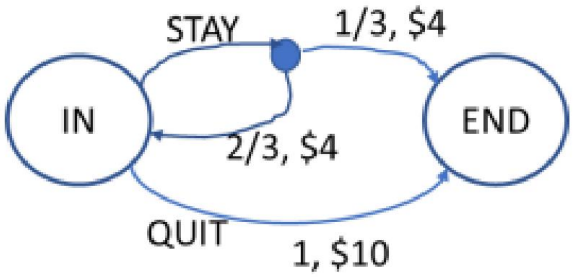


Fig.2 (a) an MDP

$$\pi(a|s) = \frac{1}{2}, \quad for\ a = STAY\ or\ QUIT, s = IN$$

The Bellman equations for state-value function can be written as

$$\begin{cases} v(END) = 0 \\ v(IN) = \frac{1}{2}\big(10 + v(END)\big) + \frac{1}{2}\left(\frac{1}{3}\big(4 + v(END)\big) + \frac{2}{3}\big(4 + v(IN)\big)\right) \end{cases}$$

Using the equation given below

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

Agent use a random policy (uniform) on selecting the action: STAY (S) or QUIT (Q)

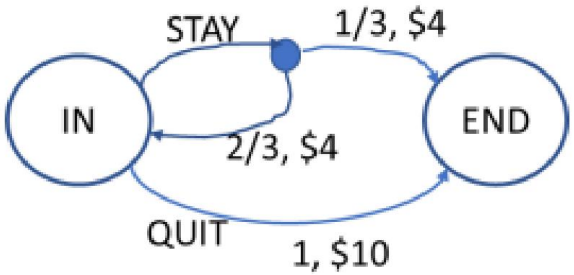


Fig.2 (a) an MDP

The Bellman equations for state-value function can be written as

$$\begin{cases} v(END) = 0 \\ v(IN) = \frac{1}{2}(10 + v(END)) + \frac{1}{2}\left(\frac{1}{3}(4 + v(END)) + \frac{2}{3}(4 + v(IN))\right) \end{cases}$$

By solving the equations, we get the state-value function for a random policy,

$$\begin{cases} v(END) = 0 \\ v(IN) = 10.5 \end{cases}$$

The result tells us that the player will get $10.5 in average if selecting STAY or QUIT with an equal probability. This random policy is obviously better than just selecting QUIT deterministically that results in a $10 return

Agent adopts a policy of always selecting STAY

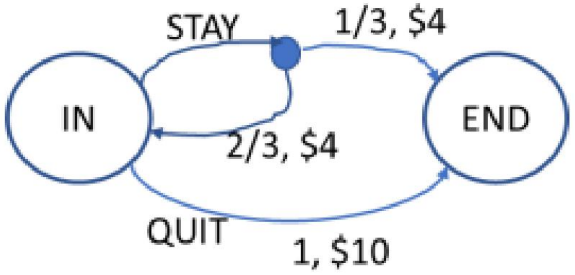


Fig.2 (a) an MDP

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

$$\begin{cases} v(END) = 0 \\ v(IN) = 0\big(10 + v(END)\big) + 1\left(\frac{1}{3}\big(4 + v(END)\big) + \frac{2}{3}\big(4 + v(IN)\big)\right) \end{cases}$$

$$\begin{cases} v(END) = 0 \\ v(IN) = 12 \end{cases}$$

Since this policy (i.e. always selecting STAY) results in a larger state-value function, it is better than the random policy

Calculate the action-value function for a random policy

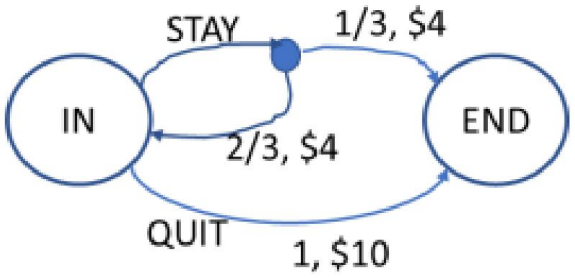


Fig.2 (a) an MDP

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\right]$$

$$\begin{cases} q(IN,QUIT) = 10 \\ q(IN,STAY) = \frac{1}{3}(4+0) + \frac{2}{3}\left(4 + \frac{1}{2}q(IN,QUIT) + \frac{1}{2}q(IN,STAY)\right) \\ q(END,STAY) = 0 \\ q(END,QUIT) = 0 \end{cases}$$

By solving the equations, we have the action-value function for the random policy,

$$\begin{cases} q(IN,QUIT) = 10 \\ q(IN,STAY) = 11 \\ q(END,STAY) = 0 \\ q(END,QUIT) = 0 \end{cases}$$

Example –Volcano Crossing

As shown in Fig, the land is divided into 3x4 grid cells: (x,y), x=1,2,3 and y=1,2,3,4. The volcano is located in the cells (1,3) and (2,3). There are two island cells (3,1) and (1,4). An agent travels on the land. The agent can move in four directions: east (E), south (S), west (W), and north (N). If the agent tries to move off the grid, it remains in the same cell. The number in each cell represents the reward when the agent moves to it. The rewards are zero for the cell without a number. For example, if the agent moves to any volcano cell (either (1,3) or (2,3)), it receives -50 reward. If the agent moves to the island cell (1,4), a reward of 20 is received. Although the agent can select the moving direction, the agent slips with a probability p when moving. If it slips, it will move randomly in one of the four directions. If it does not slip (with a probability of 1-p), it will go to the cell it intends for. Whenever the agent reaches any island or volcano cell, the travel ends.

Assume the agent starts at cell (1,1) and discount rate $\gamma = 1$

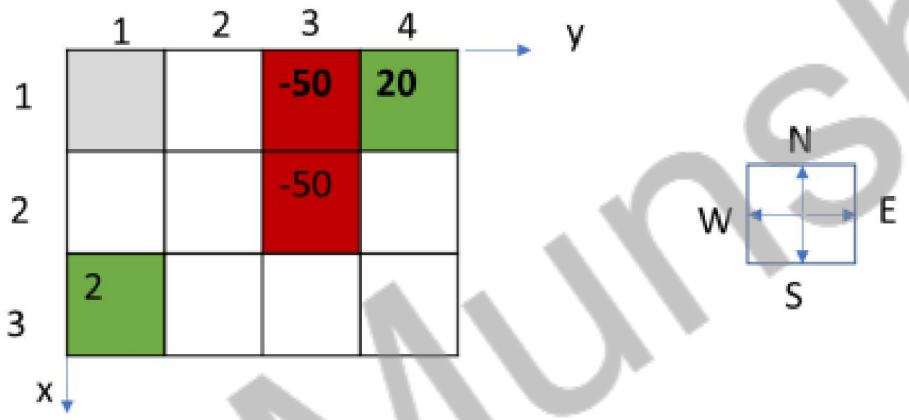


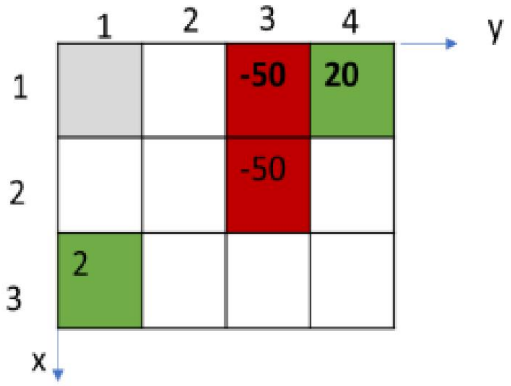
Fig. 3 Volcano crossing: an MDP

The question is: how should the agent select a moving direction given its current location so that it can get a maximal return for the travel in average sense?

A further question is: should the agent try to move to the island (3,1) for a reward of 2 or to move to the island (1,4) for a larger reward of 20 but with a bigger risk of slipping into volcano?

Describe Volcano in terms of MDP

- State is the location of the agent (x,y)
- Action set $\mathcal{A} = \{E, S, W, N\}$
- MDP is defined by
  - $p(s', r|s, a) \equiv \Pr\{St = s', Rt = r|S_{t-1} = s, A_{t-1} = a\}$
- Assume slipping probability p=0.1



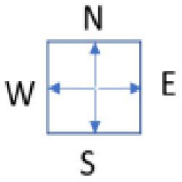


Fig. 3 Volcano crossing: an MDP

- Evaluate
  - $p((1,1), 0 \mid (1,1), E)$
  - $= 0.1 \times \left(\frac{1}{4} + \frac{1}{4}\right) = 0.05$
- Interpretation??

This probability means that when the agent selects an action E at the state (1,1), the agent will stay at the same place (1,1) with a reward of 0 with a probability of 0.05. Note that even though it selects E, there is a chance for slipping to occur. When slipping occurs, it could move E,S,W,N with a probability of ¼ for each direction, and W and N will lead the agent to stay

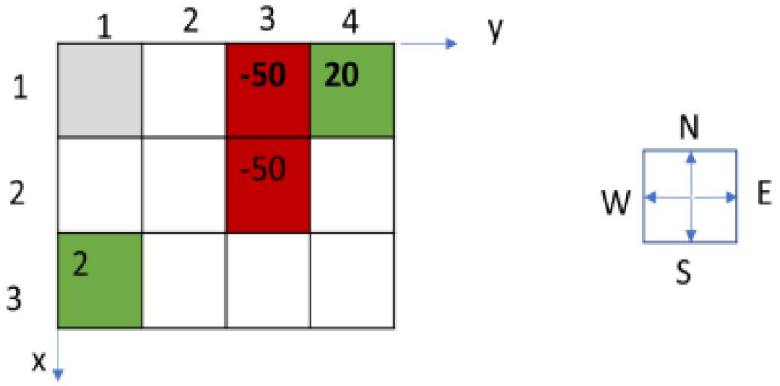
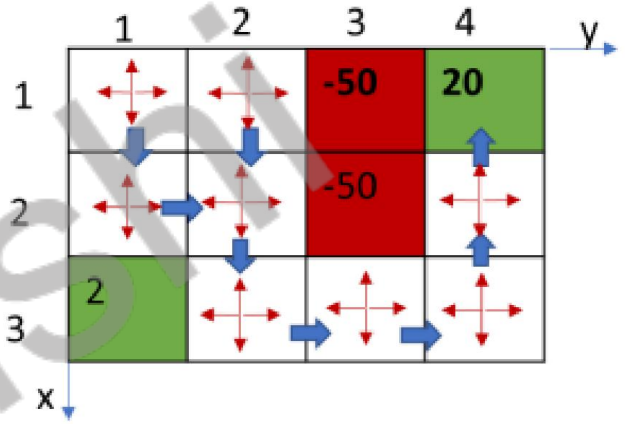


Fig. 3 Volcano crossing: an MDP

- Verify
  - $p((1,4), 20|(2,4), N)$
  - $= 0.9 + 0.1 \times \frac{1}{4} = 0.925$

Suppose that the agent selects the path toward the island with a reward of 20, indicated by the blue arrows in Fig. However, even if the agent selects one moving direction, it may slip with a probability of p, instead of moving in the selected direction. When a slip occurs, the agent will randomly move into its adjacent cell with equal probability (1/4), indicated by the red line arrows. Thus, the real policy $\pi(a|s)$ can be obtained based on the selected path and the location of the agent.



(a) pre-defined path and slip

consider s=(1,1), we have the policy for this state (location)

$$\pi\big(N|(1,1)\big) = p * \frac{1}{4} \quad \text{(slip to North)}$$

$$\pi\big(W|(1,1)\big) = p * \frac{1}{4} \quad \text{(slip to West)}$$

$$\pi\big(E|(1,1)\big) = p * \frac{1}{4} \quad \text{(slip to East)}$$

$$\pi\big(S|(1,1)\big) = 1 - p + p * \frac{1}{4} \quad \text{(intend for South and slip to South)}$$

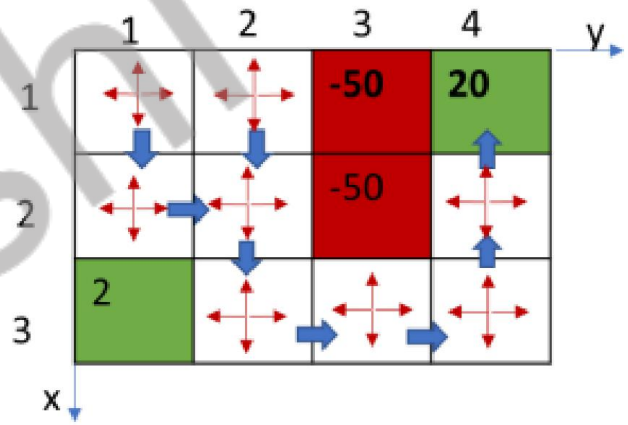
For the state (3,2), the policy is

$$\pi(N|(1,1)) = p * \frac{1}{4} \quad \text{(slip to North)}$$

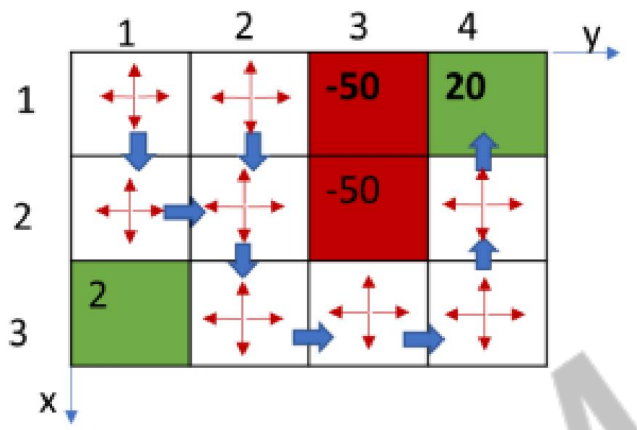
$$\pi(W|(1,1)) = p * \frac{1}{4} \quad \text{(slip to West)}$$

$$\pi(E|(1,1)) = 1 - p + p * \frac{1}{4} \quad \text{(intend for East and slip to East)}$$

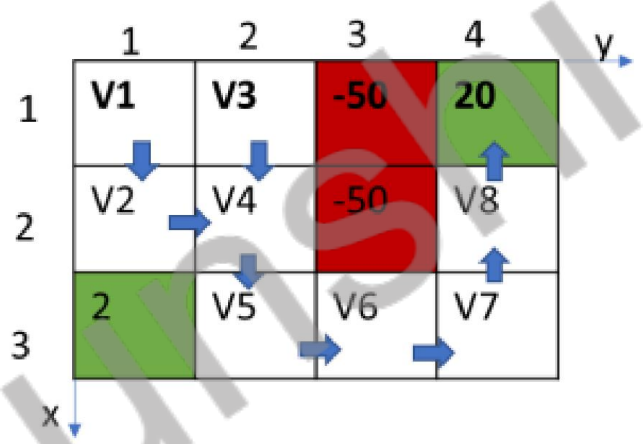
$$\pi(S|(1,1)) = p * \frac{1}{4} \quad \text{(slip to South)}$$



(a) pre-defined path and slip



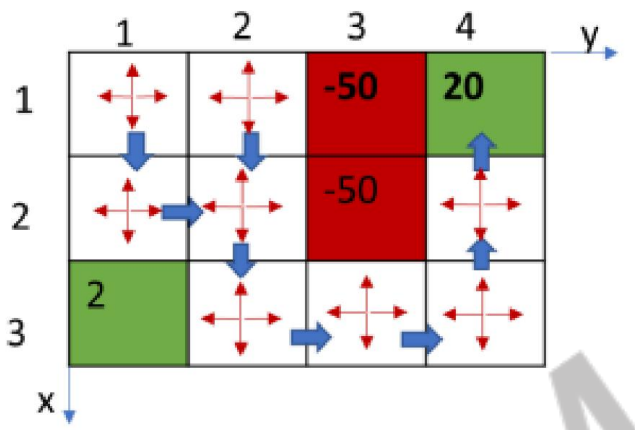
(a) pre-defined path and slip



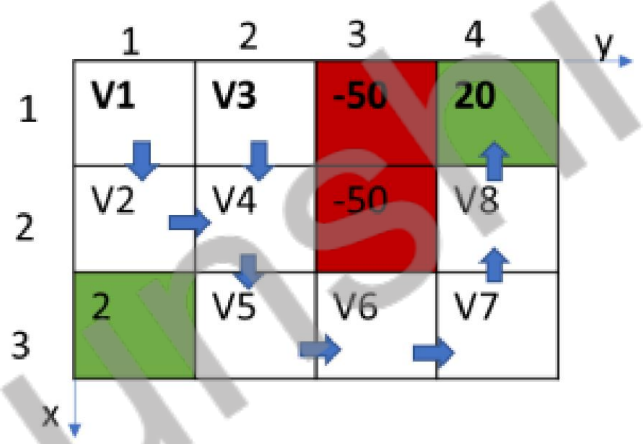
(b) state-value variable

To find the state-value function, we first define the state values for the states as V1, V2, …., V8, in Fig. (b), and then write one Bellman equation for each state. For instance, the Bellman equation for state (1,1) is

$$V1 = \underbrace{(1-p) \times V2}_{\text{Move to V2}} + \underbrace{p \times \frac{1}{4} \times (V1 + V1 + V2 + V3)}_{\text{slip}}$$



(a) pre-defined path and slip



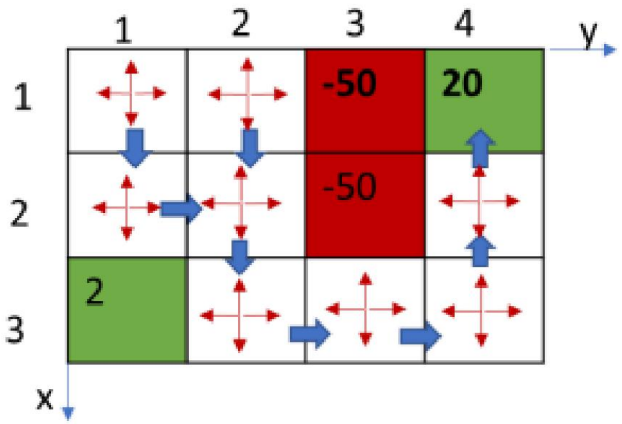
(b) state-value variable

Bellman equation for state (2,4) is

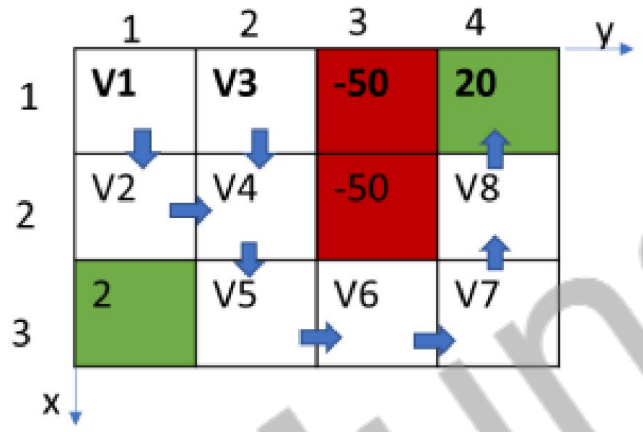
$$V8 = (1-p) \times 20 + p \times \frac{1}{4} \times (20 - 50 + V7 + V8)$$

Move to island

slip



(a) pre-defined path and slip



(b) state-value variable

0	0	-50	20
0	0	-50	0
2	0	0	0

Let initial values of V1 to V8=0

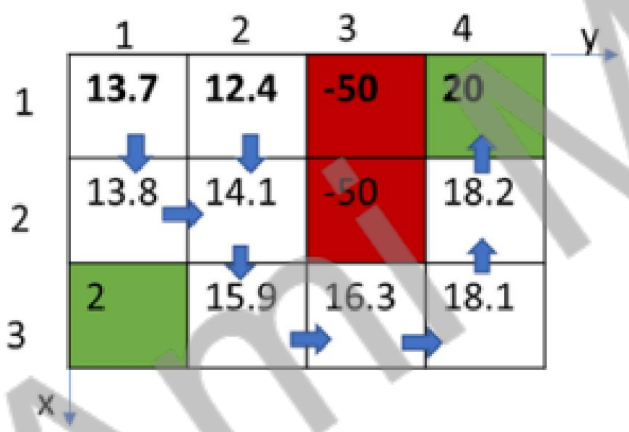
$$V8 = \underbrace{(1-p) \times 20}_{\text{Move to island}} + \underbrace{p \times \frac{1}{4} \times (20 - 50 + V7 + V8)}_{\text{slip}}$$

Take p=0.1

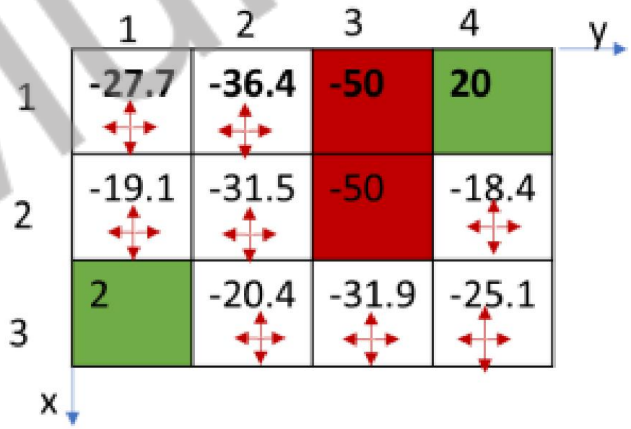
$$V_8 = (1 - 0.1) \times 20 + 0.1 \times \left(\frac{1}{4}\right) \times (20 - 50 + 0 + 0) = 17.25$$

In total, we can have totally 8 equations for the 8 states and get the state-value function by solving these 8 linear equations

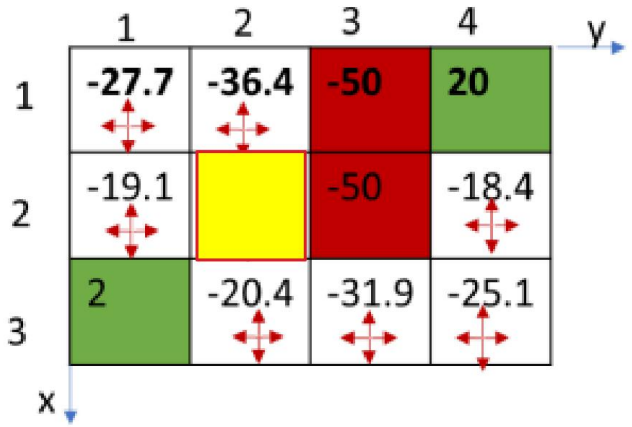
The solutions are shown in Fig.(c) for p=0.1.
Fig.(d) shows the state values for a policy of complete random move at all states (i.e., p=1)



(c) State-value function with p=0.1

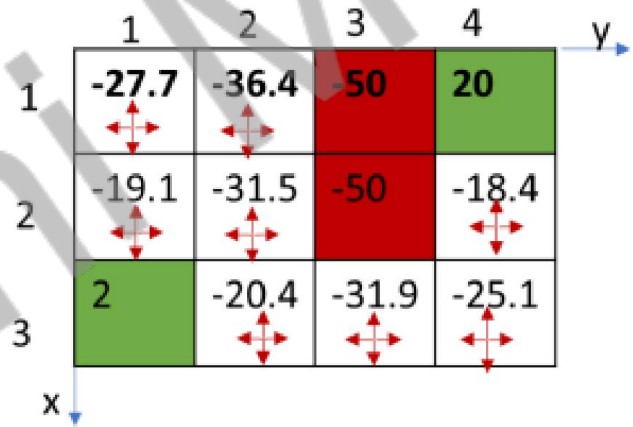


(d) state values for random path (i.e. p=1)



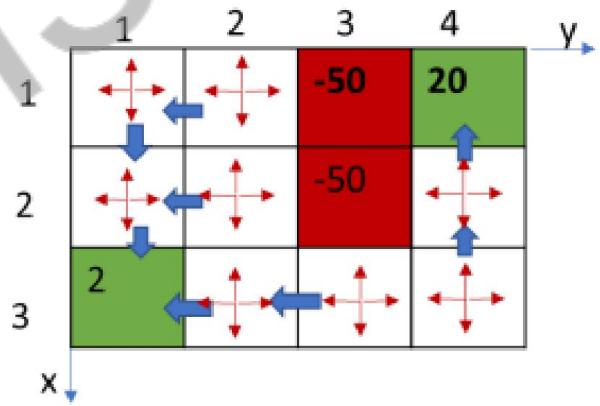
If p=1, find the value at cell highlighted in yellow

$$V(2,2) = \frac{1}{4}(-19.1 - 36.4 - 20.4 - 50) = -31.47$$



Timid Policy

- Consider a coward (timid) policy
- By following this policy, the agent tries to avoid slipping into volcano as possible as it can, regardless of a big reward 20 in the island



(a) A timid policy

Comparison of Timid, Random and Good Policy

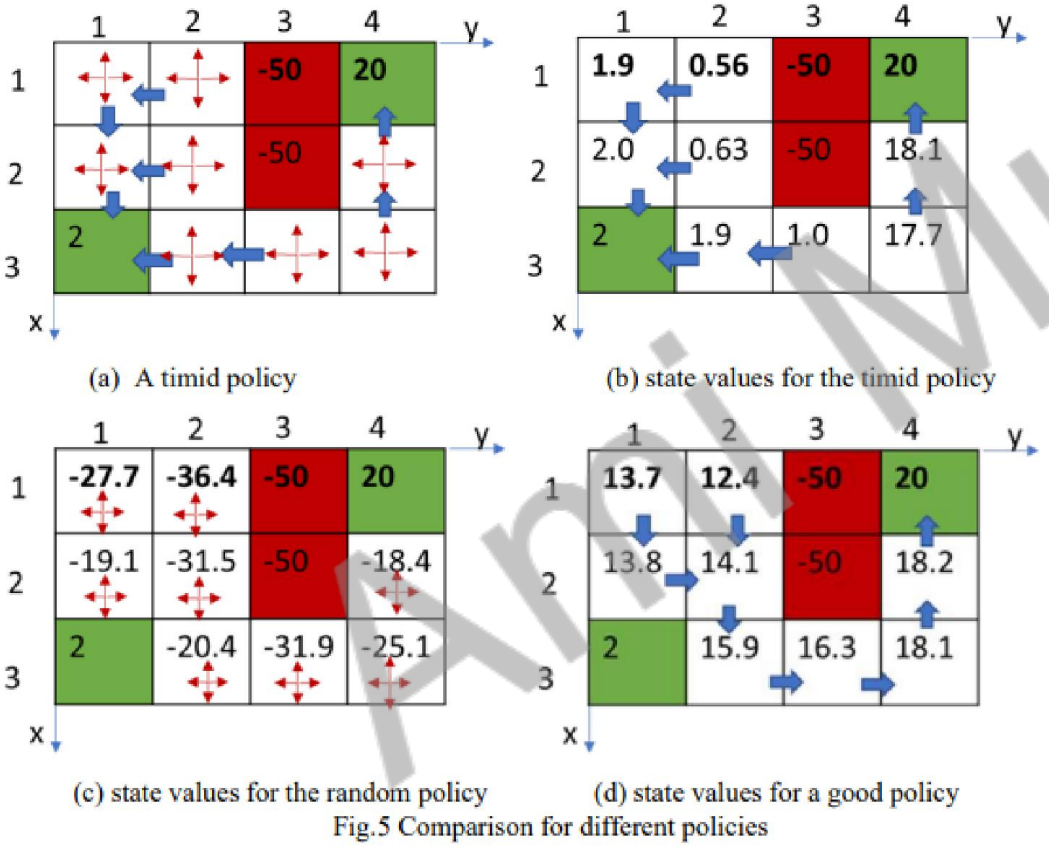
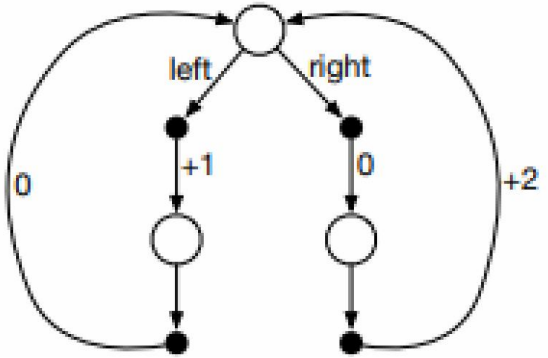


Fig.5 Comparison for different policies

Random policy is the worst one

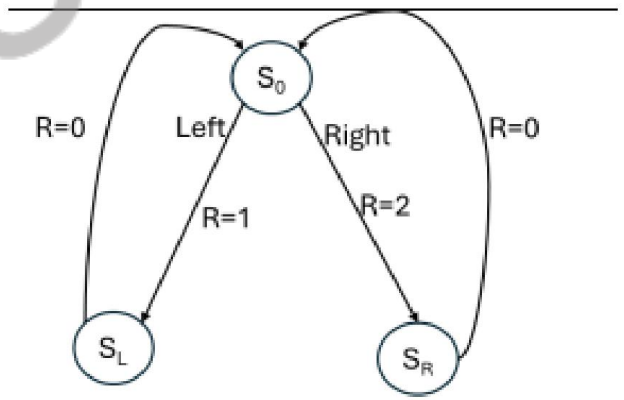
Example

Exercise 3.22 Consider the continuing MDP shown to the right. The only decision to be made is that in the top state, where two actions are available, left and right. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. What policy is optimal if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$? □

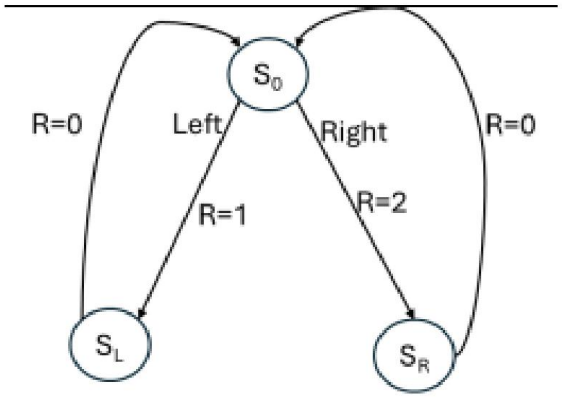


Problem similar to above just reframed

Consider the continuing Markov Decision Process shown above. The only decision to be made is in the top state (S_0), where two actions are available, left and right. The numbers (i.e. R=0, R=1, R=2) show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. Calculate and show which policy will be the optimal if:
(i) $\gamma = 0$; (ii) $\gamma = 0.9$; (iii) $\gamma = 0.5$.



Solution



In an infinite-horizon MDP, the expected return from state s following a deterministic policy π is given as,

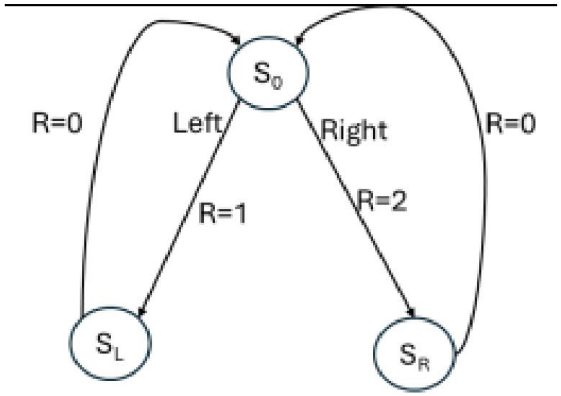
$$\mathbb{E}_{\pi}[G_t \mid s_t = s] = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

Here, for two deterministic policy $\pi_{\texttt{left}}$ and $\pi_{\texttt{right}}$, the expected returns from state s_0 are:

$$\mathbb{E}_{\pi_{\texttt{left}}}[G_0 \mid s_0] = 1 + \gamma \cdot 0 + \gamma^2 \cdot 1 + \gamma^3 \cdot 0 + \cdots = \sum_{k=0}^{\infty} \gamma^{2k}(1 + \gamma \cdot 0) = \frac{1}{1-\gamma^2}$$

$$\mathbb{E}_{\pi_{\texttt{right}}}[G_0 \mid s_0] = 0 + \gamma \cdot 2 + \gamma^2 \cdot 0 + \gamma^3 \cdot 2 + \cdots = \sum_{k=0}^{\infty} \gamma^{2k}(0 + \gamma \cdot 2) = \frac{2\gamma}{1-\gamma^2}$$

Solution



Putting the values of γ, we get

$$\mathbb{E}_{\pi_{\texttt{left}}}[G_0 \mid s_0] = \begin{cases} 1, & \text{if } \gamma = 0 \\ 5.26, & \text{if } \gamma = 0.9 \\ 1.33, & \text{if } \gamma = 0.5 \end{cases} \quad \text{and} \quad \mathbb{E}_{\pi_{\texttt{right}}}[G_0 \mid s_0] = \begin{cases} 0, & \text{if } \gamma = 0 \\ 9.47, & \text{if } \gamma = 0.9 \\ 1.33, & \text{if } \gamma = 0.5 \end{cases}$$

Hence, the optimal policies are:

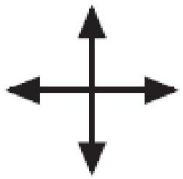
(i) $\pi_{\texttt{left}}$ for $\gamma = 0$,

(ii) $\pi_{\texttt{right}}$ for $\gamma = 0.9$,

(iii) Both $\pi_{\texttt{left}}$ and $\pi_{\texttt{right}}$ for $\gamma = 0.5$.

Example

- Consider the grid world shown. All four actions (right, left, up and down) are equiprobable. In the figure above, the value function, V^{π} at each state in the grid is provided, for the case where discount factor $\gamma = 0.9$.
- Show numerically that the Bellman equation given below applies for the center state valued at 0.7, with respect to its four neighboring states, 2.3, 0.4, -0.4 and 0.7. Assume that R(s) at all states is zero

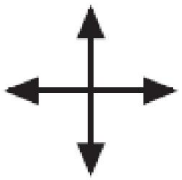


Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s')$$

Solution



Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s')$$

- 0.9 x [(0.25 x 2.3) + (0.25 x 0.4) + (0.25 x -0.4) + (0.25 x 0.7)] = 0.675 ≅ 0.7

Example

- Consider a 4×4 Grid-world problem where the goal is to reach either the top-left corner or the bottom-right corner (refer to Figure below). The agent can choose from four actions up, down, left, right, which deterministically cause the corresponding state transitions, except that actions that would take the agent off the grid leave the state unchanged. We model this as an undiscounted, episodic task, where the reward is -1 for all transitions. Suppose that the agent follows the equiprobable random policy, π
- Given above is the partial value function for this problem. Calculate respectively, the missing values in the first and second row?
- $V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s')$

Table 1: 4×4 Grid-world

0		-20	-22
-14	-18	-20	
	-20	-18	-14
-22	-20		0

Solution

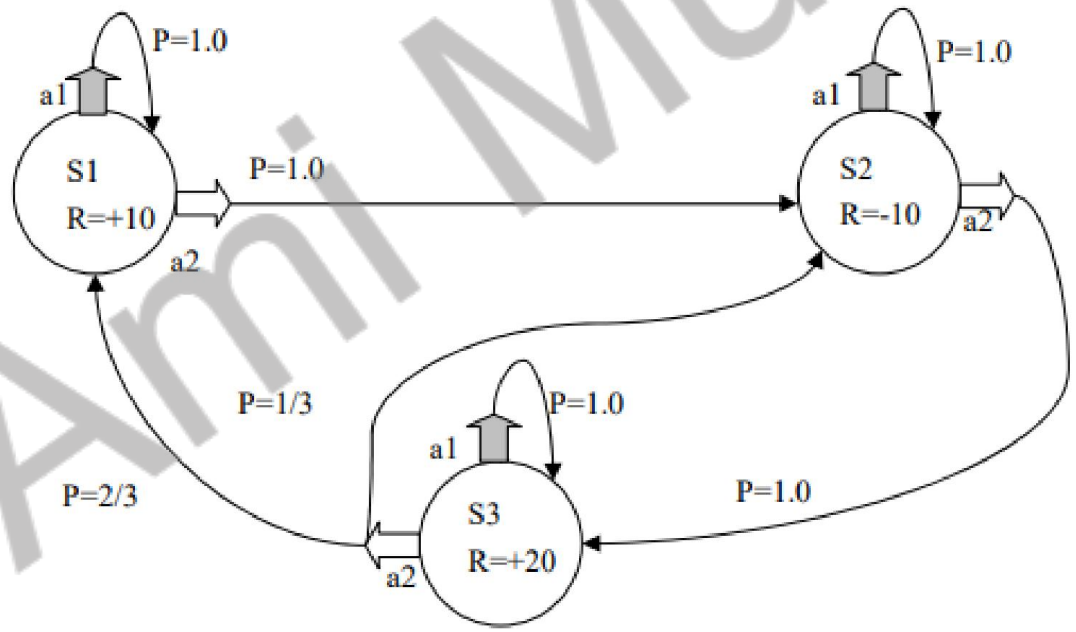
- Let the state value in the grid at first row second column be v_1.
- Then, applying Bellman Expectation Equation, we have:
- $v_1 = 0.25 \times [(-1 + 0) + (-1 - 20) + (-1 - 18) + (-1 + v_1)]$
- $= 0.25 \times [-42 + v1]$
- $\Rightarrow v1 = -14$

Solution

- Let the state value in the grid at second row fourth column be v2. Then, applying Bellman Expectation Equation, we have:
- $v2 = 0.25 \times [(-1 - 20) + (-1 - 22) + (-1 - 14) + (-1 + v2)]$
- $= 0.25 \times [-60 + v2]$
- $\Rightarrow v2 = -20$

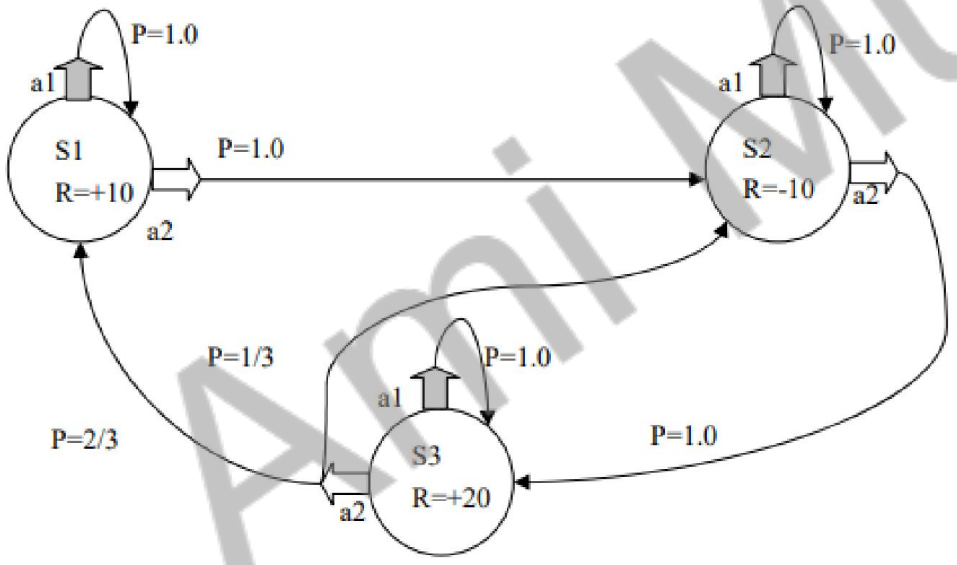
Example on MDP

In the following Markov Decision Process, there are three states S_1, S_2 and S_3. The rewards for each state and all of the state transitions are marked in the given figure. There are two actions. Action a_1 causes you to stay in the same state, and action a_2 causes you to move to other states. The discount factor is $\gamma = 0.9$.



Example on MDP

In the following Markov Decision Process, there are three states S_1, S_2 and S_3. The rewards for each state and all of the state transitions are marked in the given figure. There are two actions. Action a_1 causes you to stay in the same state, and action a_2 causes you to move to other states. The discount factor is $\gamma = 0.9$.



How many distinct policies are there in the above MDP?

$V_{\pi_0}(i)$ is the expected sum of discounted rewards if we start at state i and follow the policy π_0. The initial policy π_0 is the one that assigns a_1 to every state
Write down the numerical values of the expected discounted rewards of each of the following states in the MDP

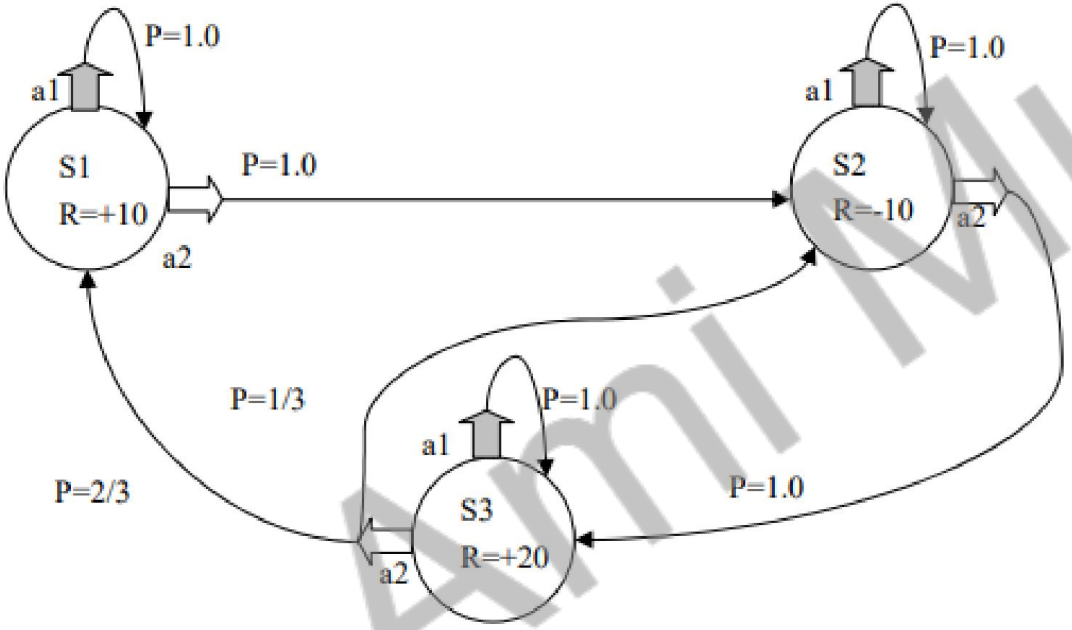


Figure 3: Markov Decision Process

Continuing from question above, suppose we run policy iteration with π_0 as the initial policy. Define π_1 as the updated policy after one iteration of policy iteration, and write down the updated policy for each of the states

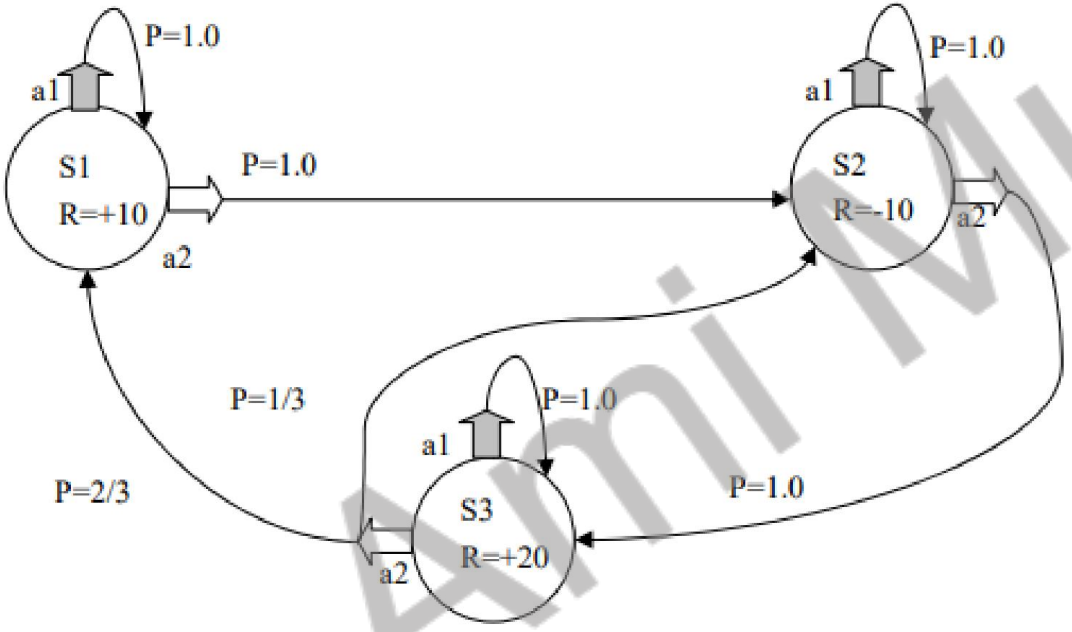


Figure 3: Markov Decision Process

Suppose we run value iteration, $V^k(i)$ is the expected sum of discounted rewards if we start at state i after $(k - 1)$ step of value iterations. Starting from the initial value of $V^1(i)$ which is the reward at state i, please write down the updated $V^2(i)$ after one value iteration

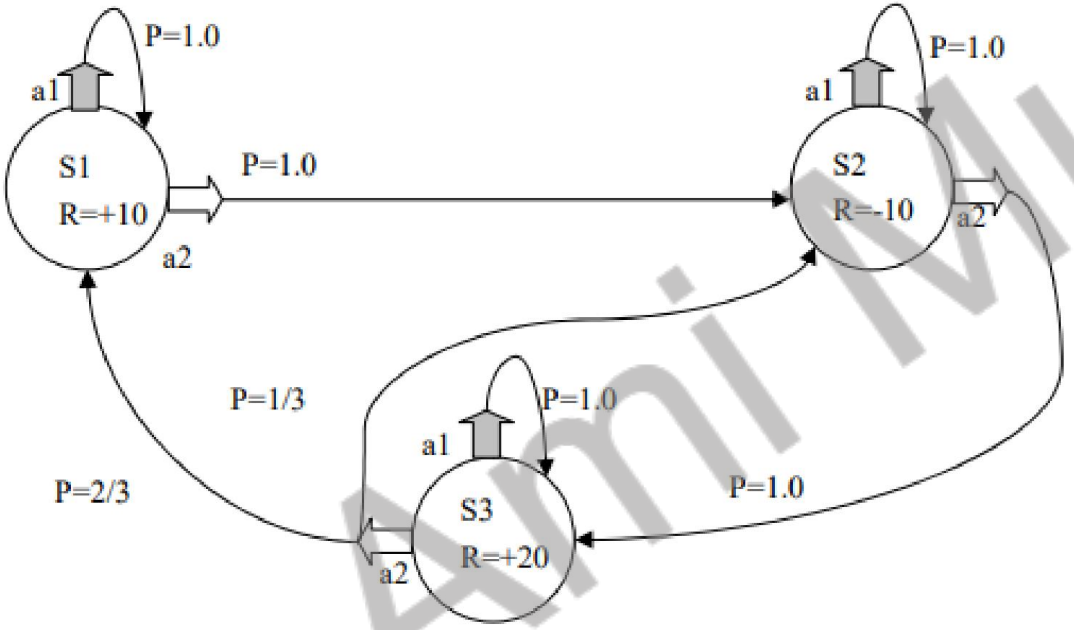
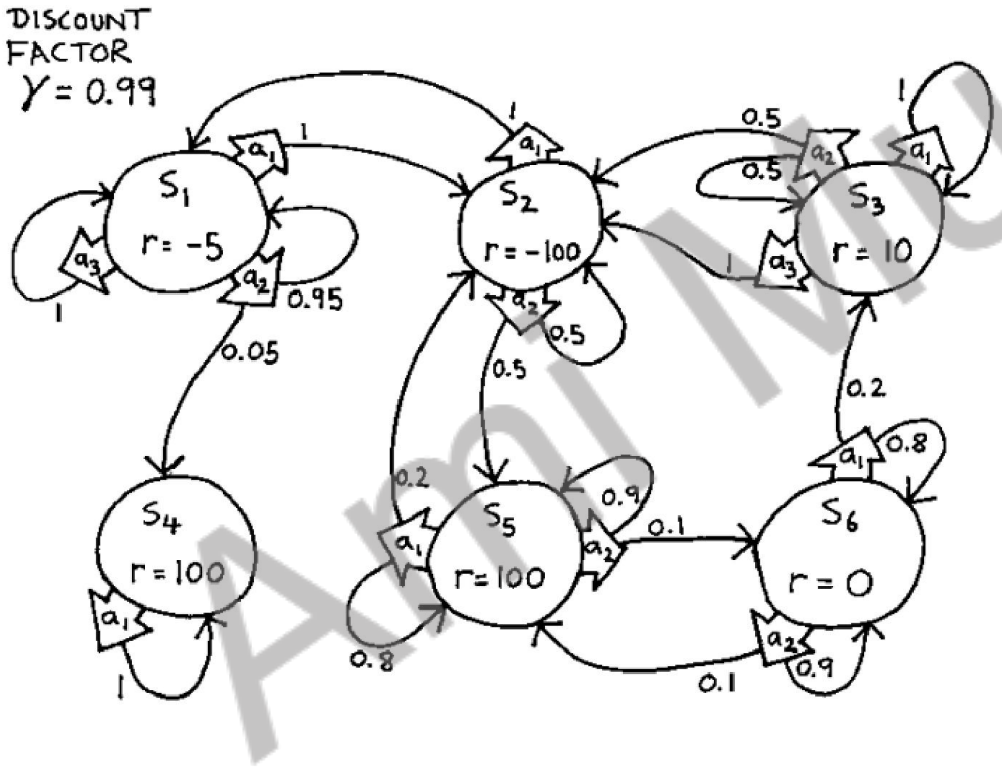


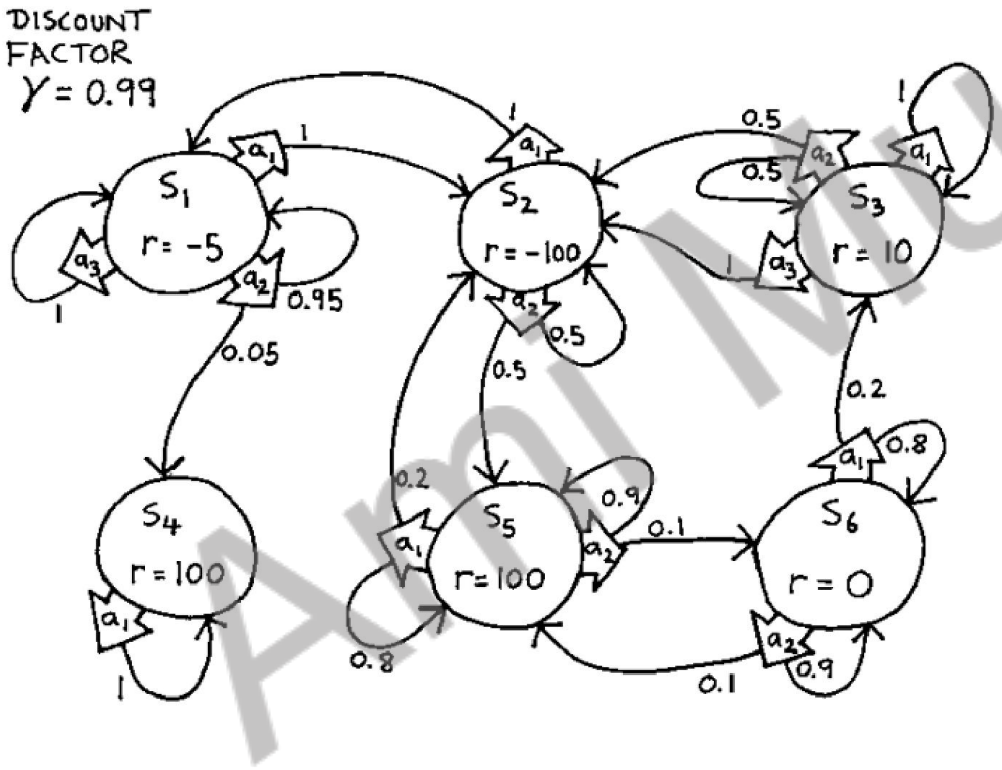
Figure 3: Markov Decision Process

Example on MDP



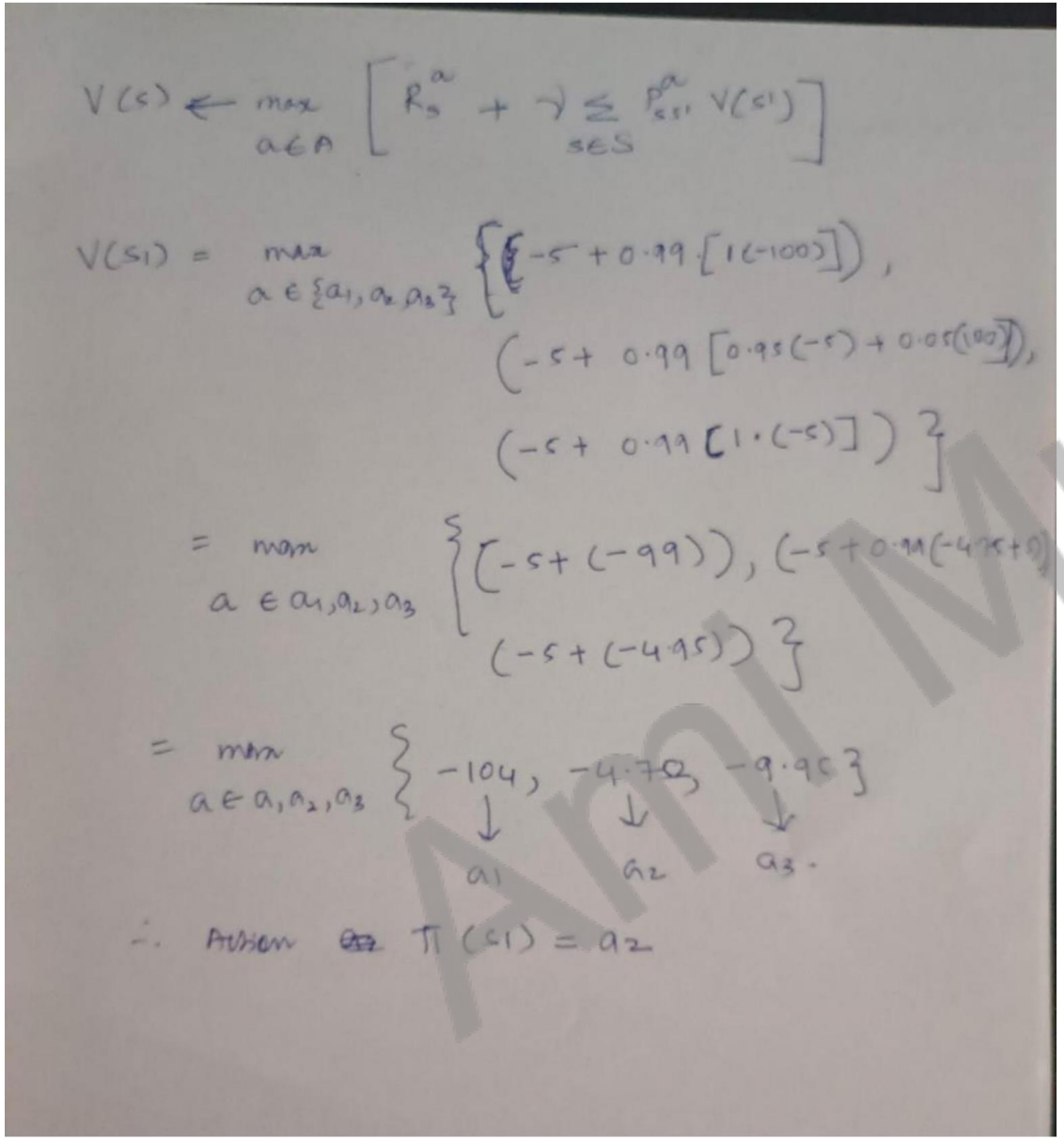
- Determine Optimal Policy for the given MDP intuitively without using value iteration algorithm

Example on MDP



- Optimal Policy

$$\pi(S_1) = a_2$$
$$\pi(S_2) = a_1$$
$$\pi(S_3) = a_3$$
$$\pi(S_4) = a_1$$
$$\pi(S_5) = a_1$$
$$\pi(S_6) = a_1$$



Using Bellman Equation to calculate optimal policy when in state S1