Finite Markov Decision Processes (MDPs)

Mohammad **Dehghani**

Mechanical and Industrial Engineering Department

Introduction

k-Armed Bandit problem was limited:

- Doesn't include many aspects of real-world problems
- Agent is presented with the same situation and each time
- The same action is always optimal.

Real-world problems:

- Different situations call for different responses
- The actions we choose now affect the amount of reward we can get into the future

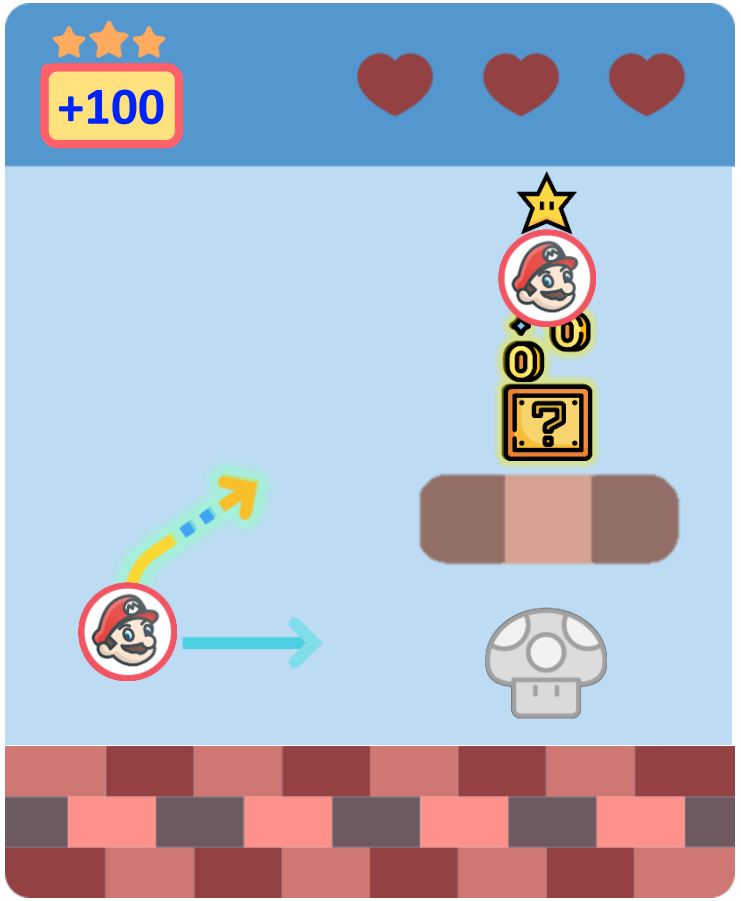
Markov Decision Process

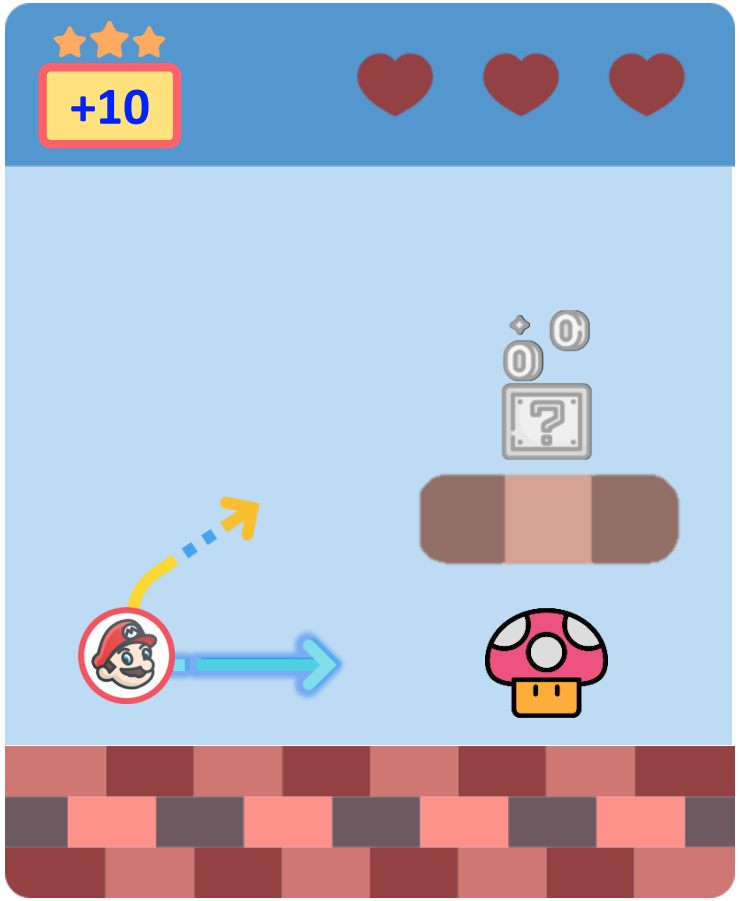
- Captures these two aspects of real-world problems:
  - Involves an associative aspect—choosing different actions in different situations
  - Actions influence not just immediate rewards, but also subsequent situations

states
actions
and
future rewards

Introduction

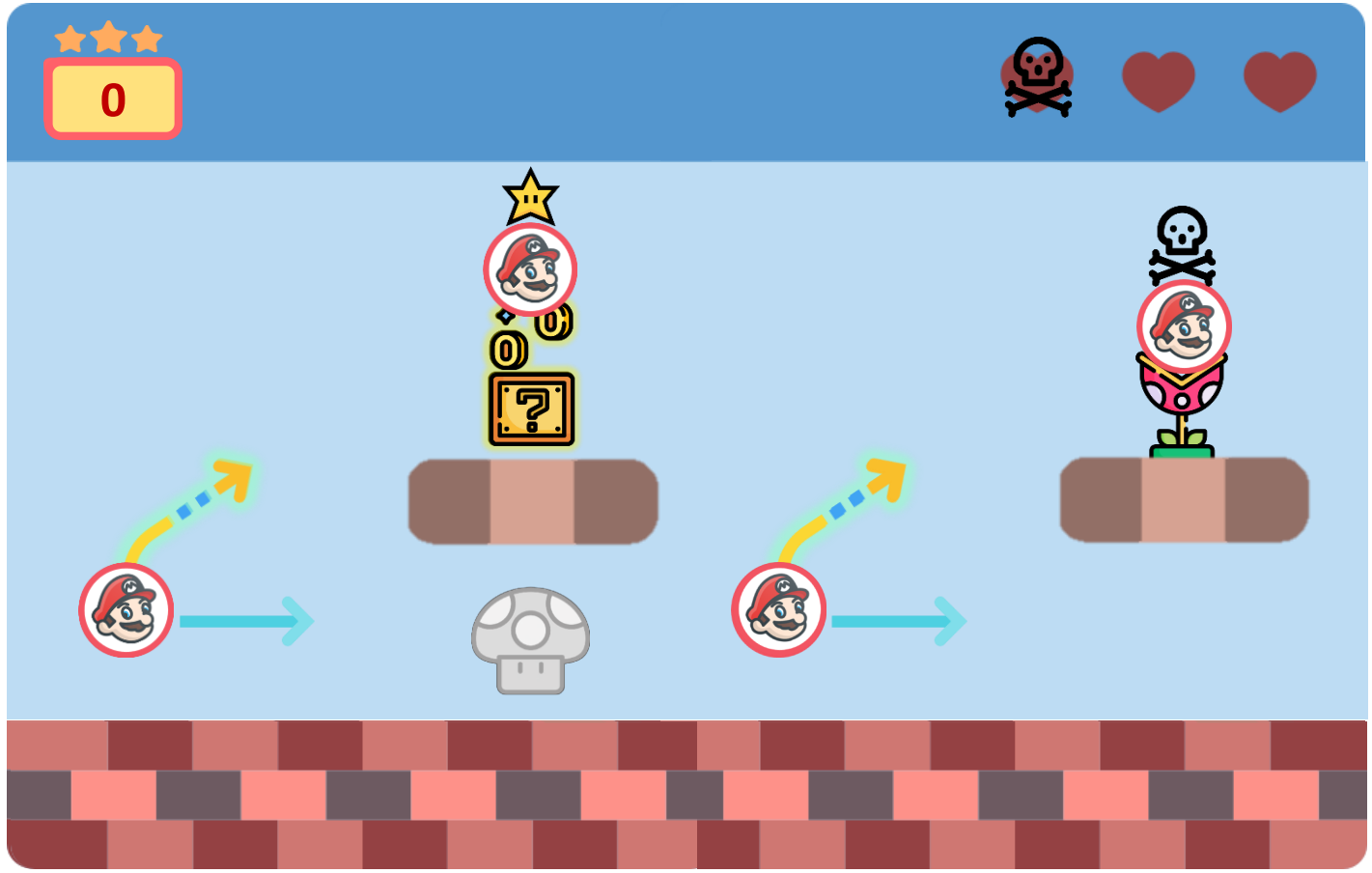
States, actions, and future rewards





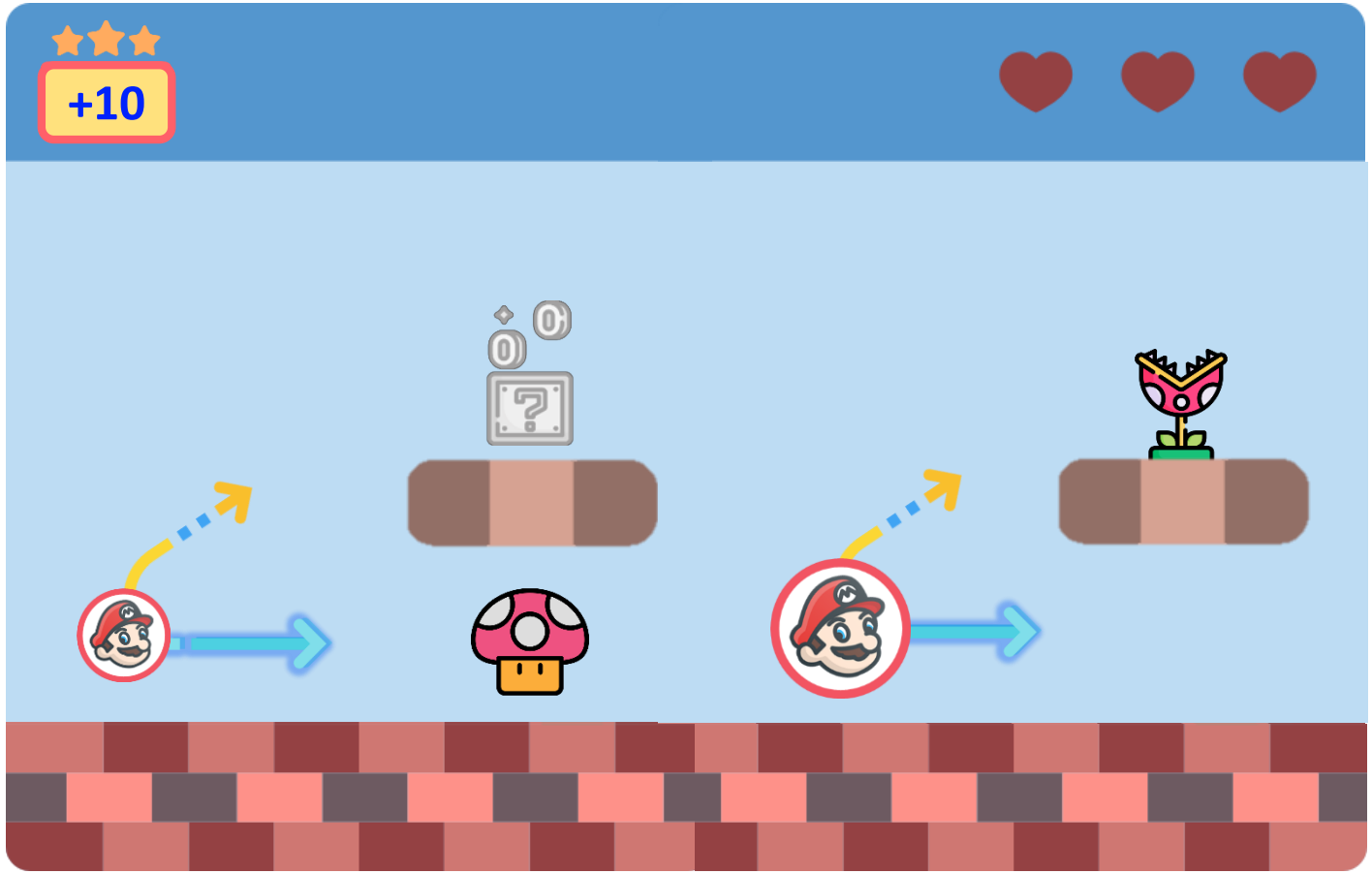
Introduction

States, actions, and future rewards



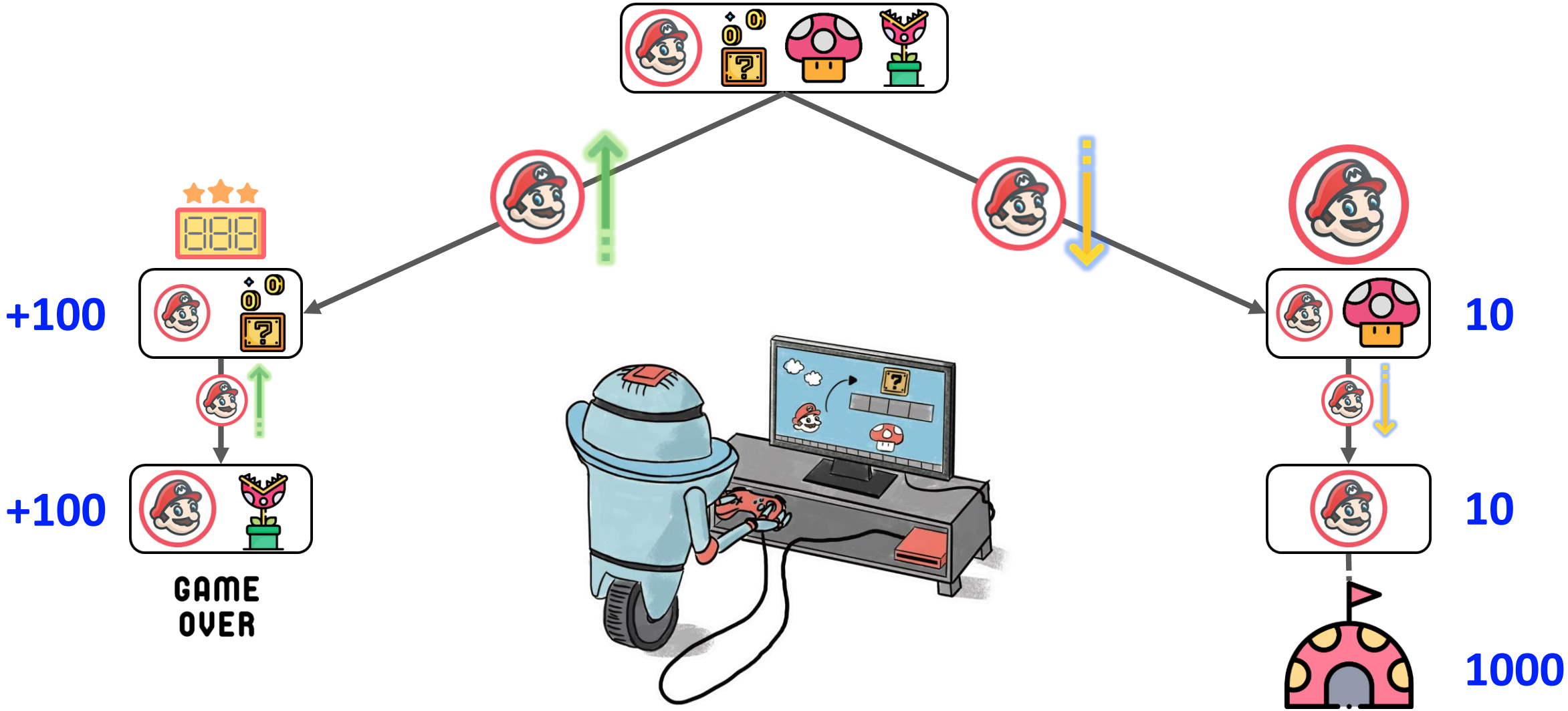
Introduction

States, actions, and future rewards



Introduction

States, actions, and future rewards

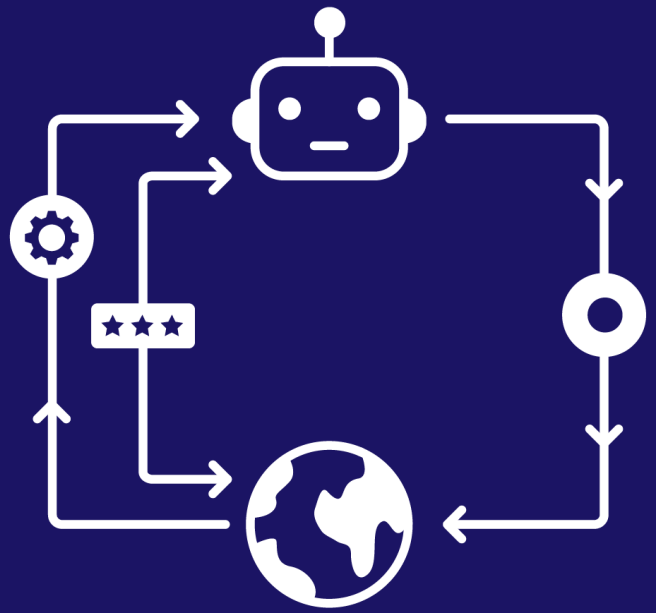


Summary

- **MDPs are a classical formalization of sequential decision making**
  - Actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.
- **MDP problem involves**
  - Evaluative feedback, as in bandits,
  - But is an associative aspect—choosing different actions in different situations.
- **Why MDPs**
  - MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made.

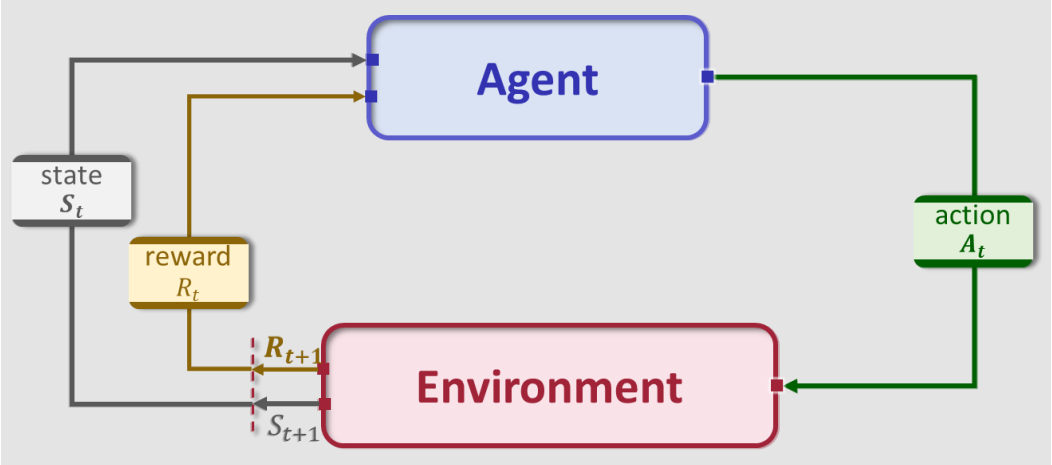
Outline

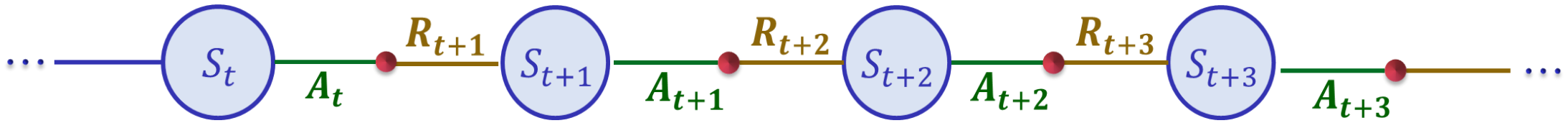
- **The Agent-Environment Interface**
  - Define Important Elements of an MDP model
  - Formulate some MDP examples
- **Goals, Rewards, and Returns**
  - MDPs involve delayed reward and the need to tradeoff immediate and delayed reward
  - Episodic vs Continuing Tasks
- **Policies and Value Functions**
  - Bellman Equations
- **Optimal Policies and Optimal Value Functions**



The Agent-Environment Interface

The Agent-Environment Interface



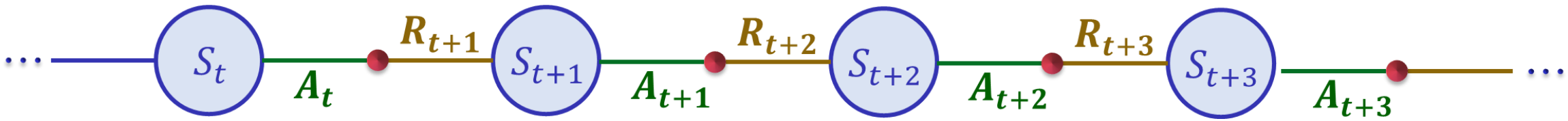


- **Agent and Environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$**
  - Agent observes state at step t: $S_t \in \mathcal{S}$
  - selects action at step t: $A_t \in \mathcal{A}(S_t)$
  - receives a numerical reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$
  - finds itself in a new state: $S_{t+1} \in \mathcal{S}^+$

Markov Decision Processes

- **The MDP and agent together thereby give rise to a sequence or trajectory that begins like this:**

$$S_o, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3$$



- Actions influence immediate rewards as well as future states and through those, future rewards.

Markov Decision Processes

Dynamics Function

The dynamics function p is an ordinary deterministic function of four arguments.

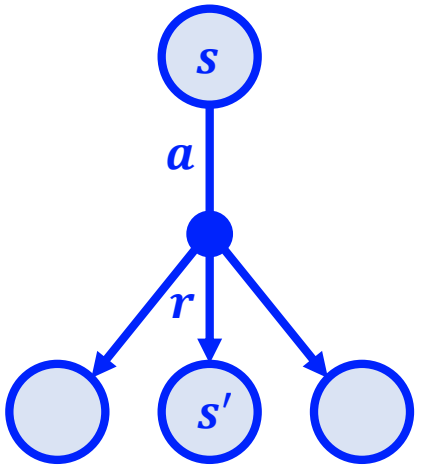
$$p\colon \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$$

where

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

and

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



- In a ***finite*** MDP, the sets of *states*, *actions*, and *rewards* ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements.

Markov Decision Processes

Markov Property

- In a ***Markov*** decision process , the probability of each possible value for S_t and R_t depends only on the ***immediately preceding*** state and action, S_{t-1} and A_{t-1}, and, given them, not at all on earlier states and actions.

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_t, A_t, \dots, S_1, A_1)$$

- The present state is ***sufficient*** and remembering earlier states would not improve predictions about the future.
- Given the present state, future states are ***independent*** of the past states

Markov Decision Processes

- **The *state-transition probabilities* with three-argument function p: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$**

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

which determine the probabilities of moving from one state to the next,

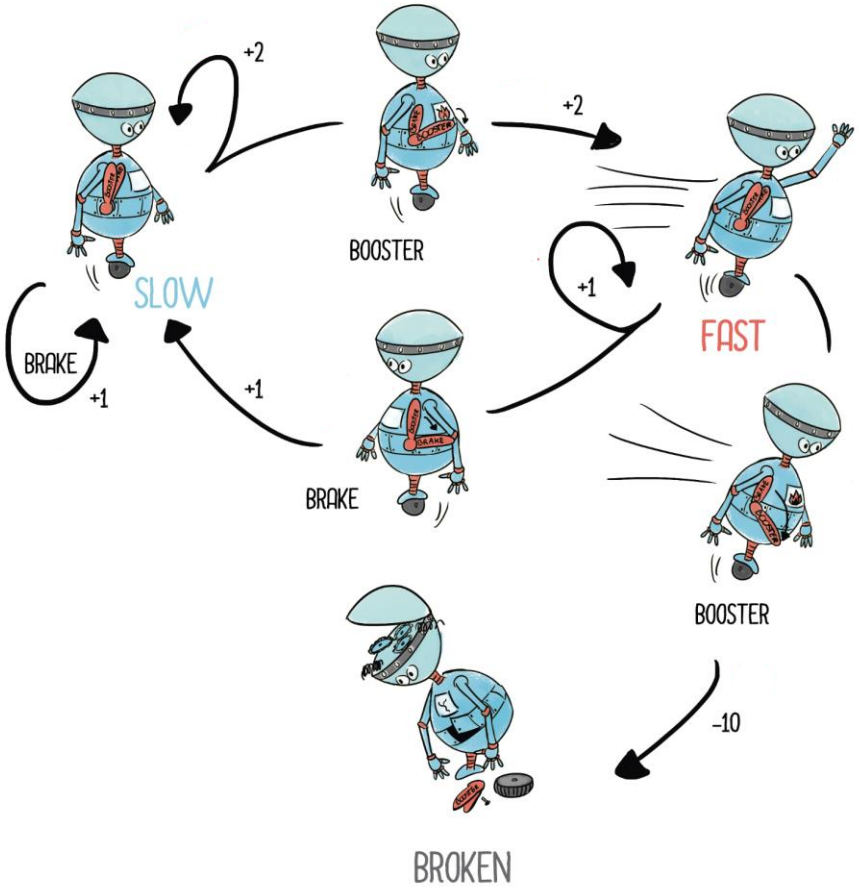
- **We can also compute the *expected rewards for state–action pairs* as a two-argument function r: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$**

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

- **The expected *rewards for state–action–next-state* triples as a three-argument function r: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$**

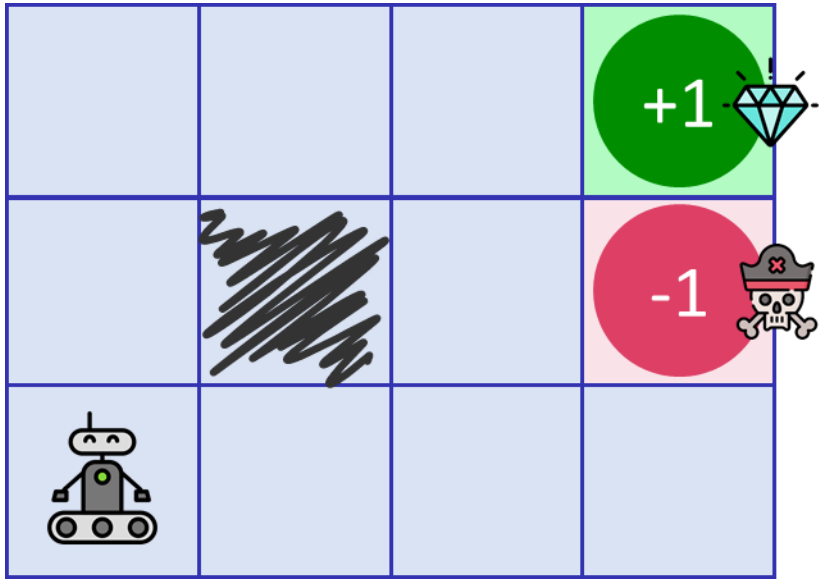
$$r(s, a, s') \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

MDP Examples



MDP Examples

The Grid World



- agent lives on grid
- always occupies a single cell
- can move left, right, up, down
- but movements are noisy/stochastic
- gets zero reward unless in “+1” or “-1” cells

MDP Examples

The Grid World

States and actions

- **State Set:**
  - $\mathcal{S} = \{S_1, S_2, \dots, S_{11}\}$
- **Action Set:**
  - $\mathcal{A} = \{Left, Right, Up, Down\}$

Reward function

- **Rewards:**
  - $R(S_{11}, .) = +1$
  - $R(S_7, .) = -1$
  - Otherwise: $R(S, .) = 0$
- **In General**
  - $R(s, a) = \mathbb{E}[r_{t+1} | s_t = s, a_t = a]$

MDP Examples

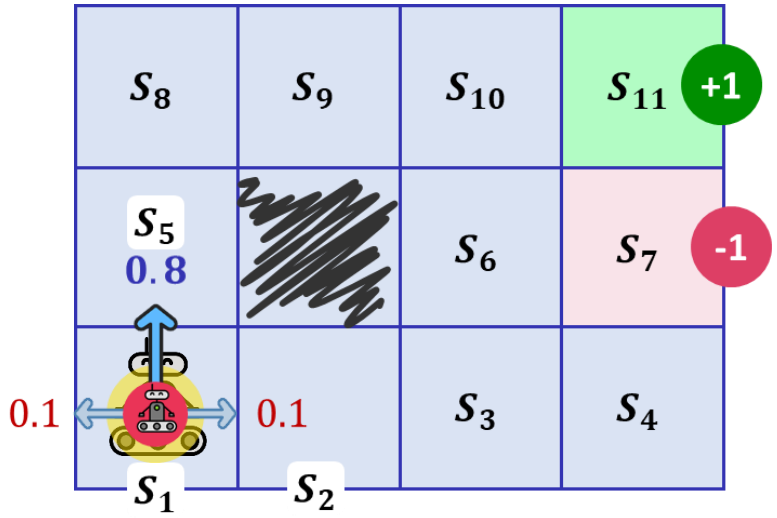
The Grid World

Transition function

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

- **Example**
  - State: S_1, Action: Up
    - $\Pr\{S_t = \mathbf{S_5} | S_{t-1} = S_1, A_{t-1} = Up\} = 0.8$
    - $\Pr\{S_t = \mathbf{S_2} | S_{t-1} = S_1, A_{t-1} = Up\} = 0.1$
    - $\Pr\{S_t = \mathbf{S_1} | S_{t-1} = S_1, A_{t-1} = Up\} = 0.1$

- **Tabular Transition Model**
  - This entire probability distribution can be written as a table over state, action, next state.



$S_t - 1$	A_{t-1}	S_t	$p(s'\|s,a)$
S_1	Up	S_5	0.8
S_1	Up	S_2	0.1
S_1	Up	S_1	0.1
…	…	…	…

MDP Examples

The Grid World

The MDP Model

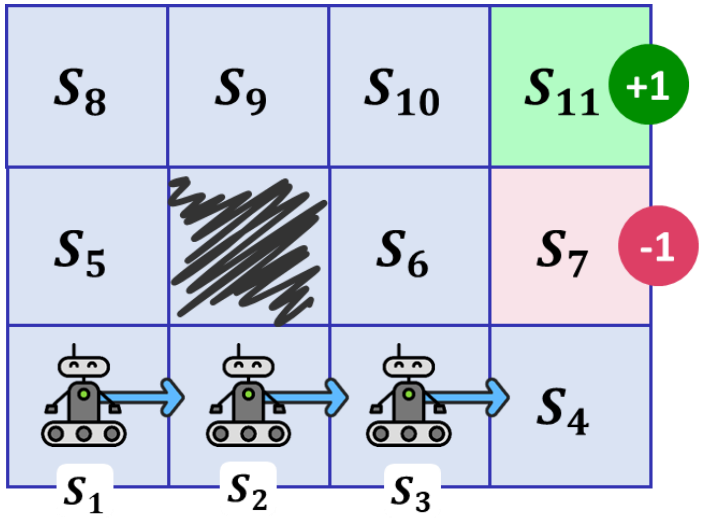
- **An MDP is a tuple:**
  - Model:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$$

  - **State Set:** $S = \{S_1, S_2, \dots, S_{11}\}$
  - **Action Set:** $A = \{Left, Right, Up, Down\}$
  - **Rewards:** $R(s,a) = \mathbb{E}[r_{t+1} | s_t = s, a_t = a]$
  - **Transition Model:** $p(s'|s,a) \doteq \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\}$

MDP Examples

The Grid World

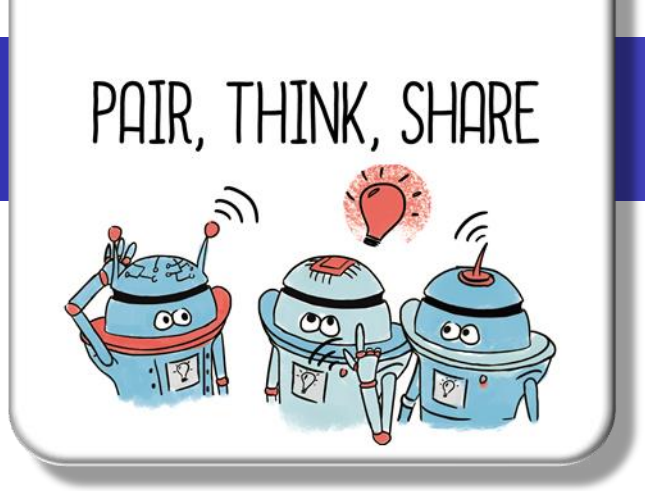


Markov Property

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_t, A_t, \dots, S_1, A_1)$$

- Suppose agent starts in S_1 and follows this path: S_1, S_2, S_3
- Notice that the probability of arriving in S_4 if agent executes $Right$ actions does not depends on path taken to S_3

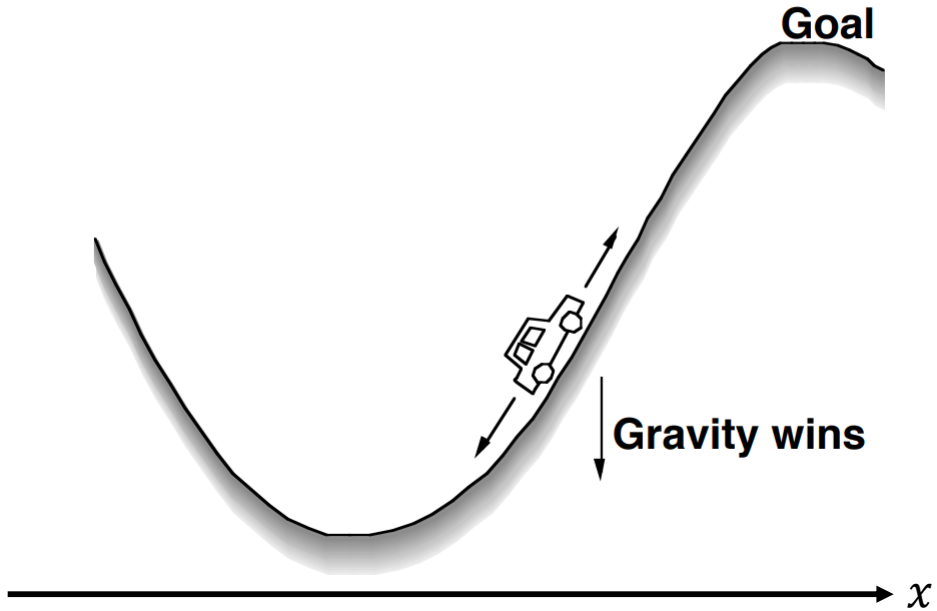
$$Pr(S_4|S_3, Right) = \Pr\{S_4|S_3, Right, S_2, Right, S_1, Right\}$$



- **The Mountain Car Problem:** A car in a valley
  - The goal of this domain is to have the underpowered car reach the top of the mountain (Goal) by building momentum using forward or reverse actions or using no action.

- **Define the MDP model of the problem?**

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$$



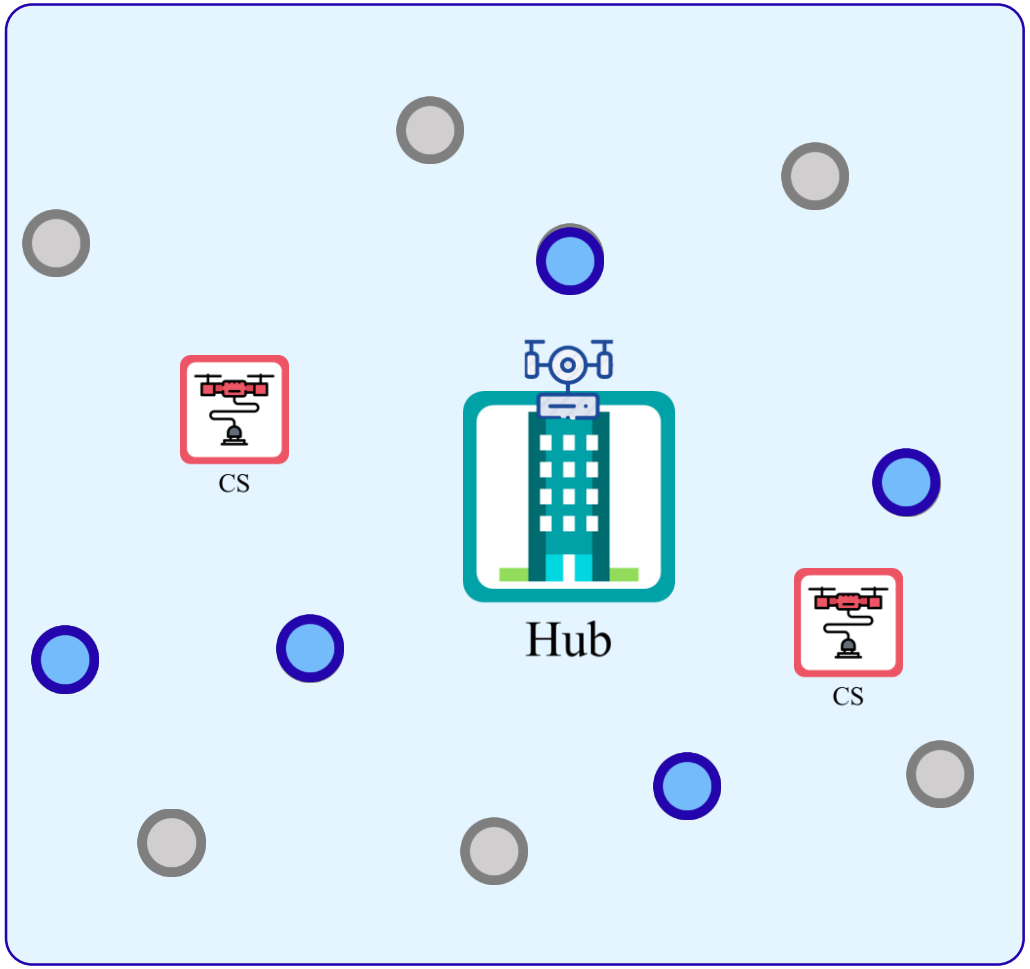
$$\mathbf{x} = [x, \dot{x}]$$

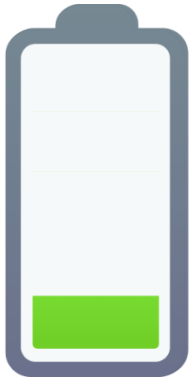
$$x' = x + \Delta t \dot{x}$$

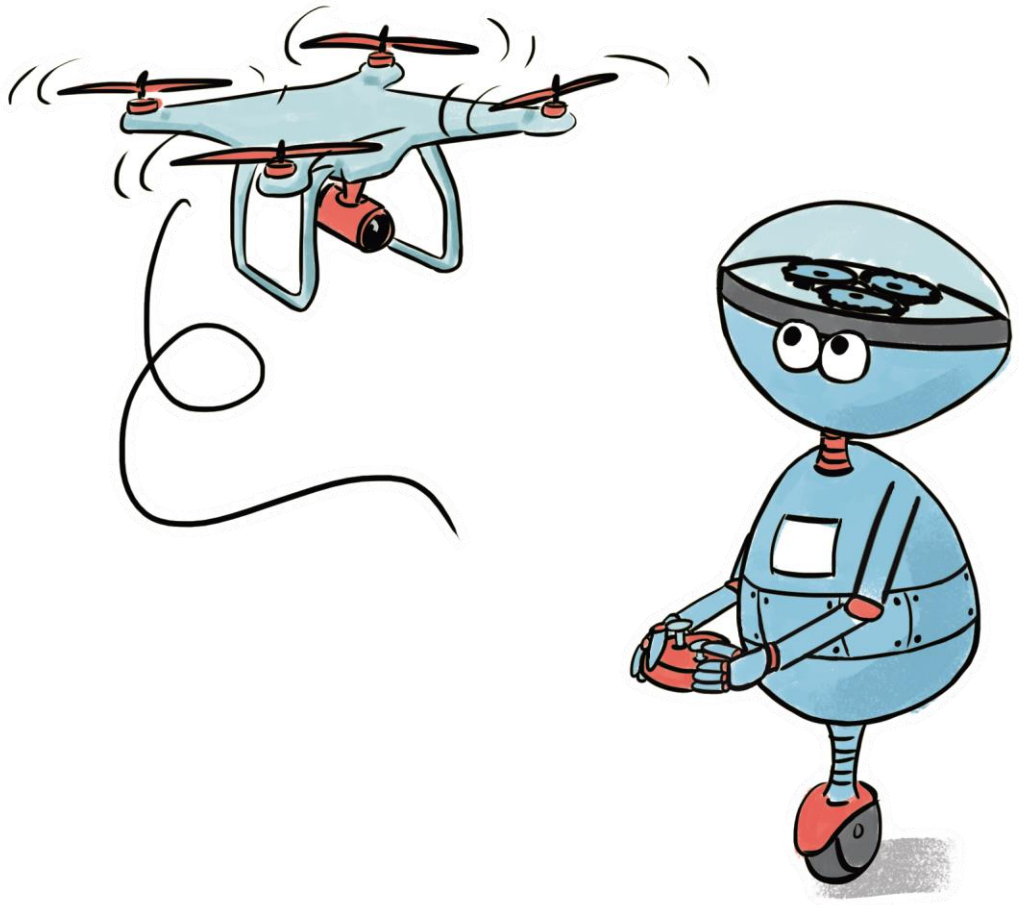
$$\dot{x}' = \dot{x} + \Delta t \left(-9.8\, m\, \cos(3x) + \frac{f}{m} a - \mu \dot{x} \right)$$

MDP Examples

Rechargeable Drone



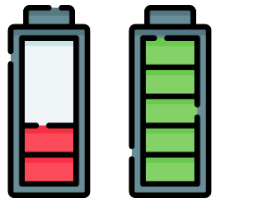




MDP Examples

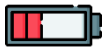
Rechargeable Drone

States



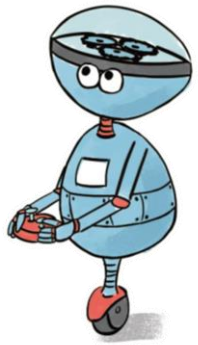
$\mathcal{S} = \{low, high\}$

Actions



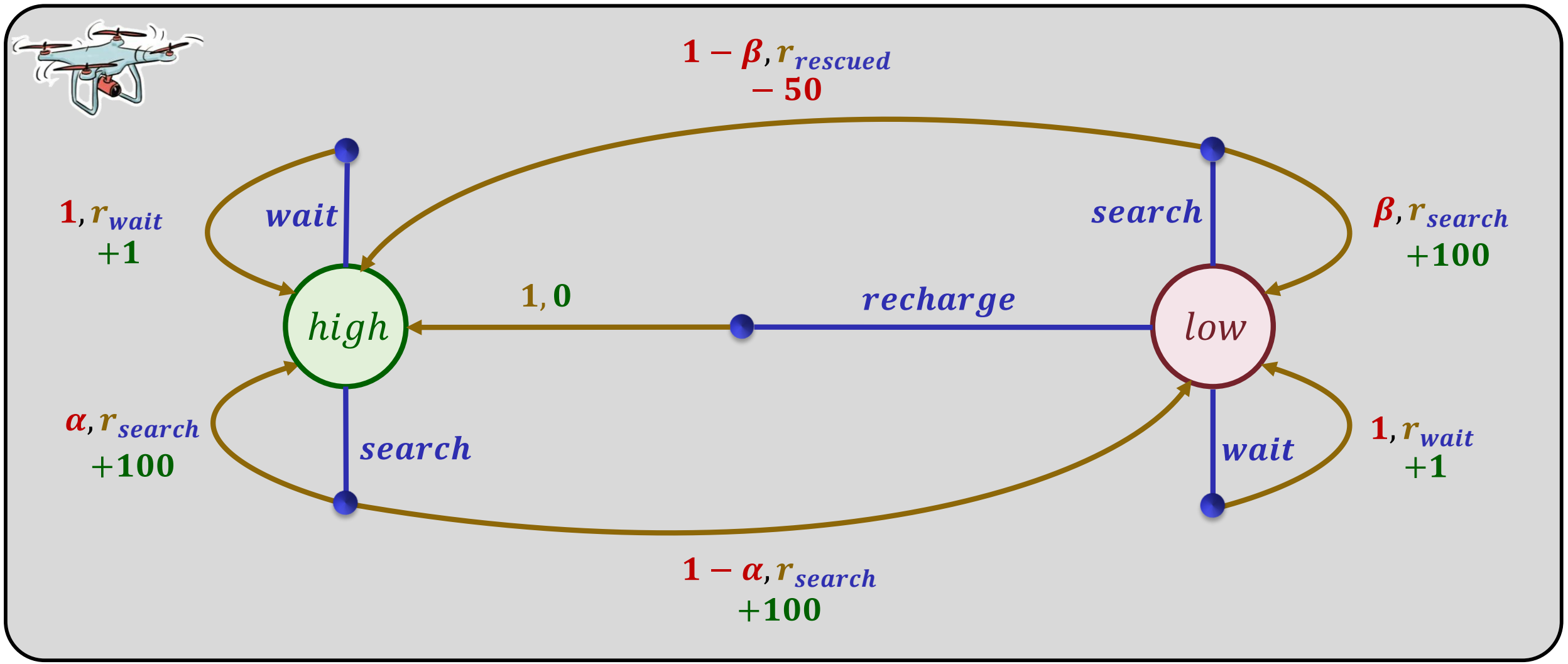
$\mathcal{A}(low) = \{search, wait, recharge\}$

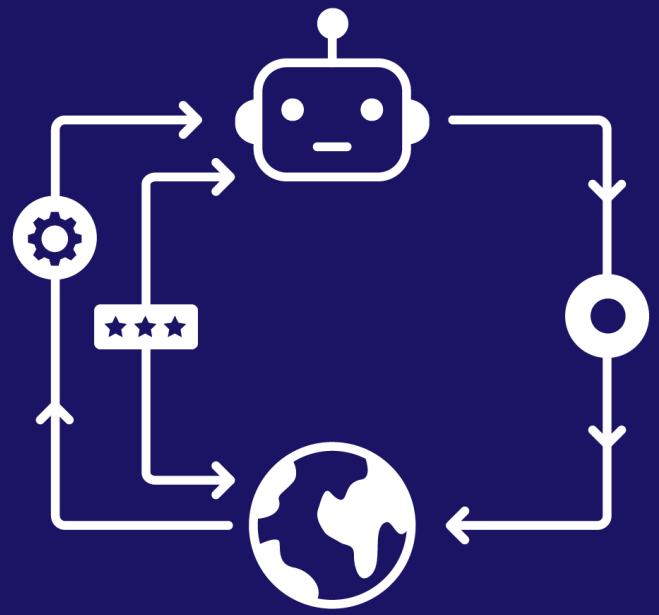
$\mathcal{A}(high) = \{search, wait\}$



MDP Examples

Rechargeable Drone

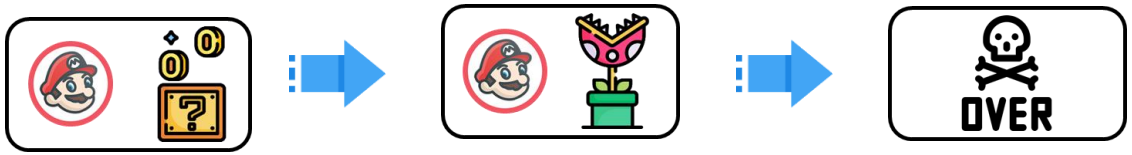




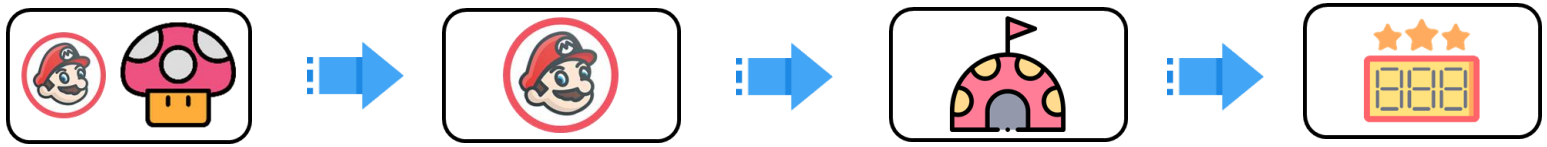
Goals
Rewards
and
Returns

Goals, Rewards, and Return

- **In reinforcement learning, the agent's objective is to maximize *future* reward.**
- **Two options:**
  - Maximizing the immediate reward (i.e. bandits)
    - Unfortunately this won't work in an MDP



  - Maximizing the total future reward



Goals, Rewards, and Return

$\mathbf{R_{t+1}}$: ***Reward signal*** is a scalar feedback signal emitted by the environment

- Indicates how well agent is performing when reaching step $t + 1$
- The agent's sole objective is to maximize the ***total reward*** it receives over the ***long run***.

Reward Hypothesis

All goals can be described by the maximization of the expected cumulative sum of a received scalar signal (called ***reward***)

- **Example:** Learning a Robot
  - **How to walk:** The reward on each time step can be defined proportional to the robot's forward motion.
  - **How to escape from a maze:** The reward is often −1 for every time step that passes prior to escape; this encourages the agent to escape as quickly as possible

Goals, Rewards, and Return

Expected Return

- In general, we seek to maximize the expected return, G_t, which in the simplest case is the sum of the rewards:

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

where T is the final step.

- The return (G_t) is a random variable because the dynamics of the MDP can be stochastic. This is why we maximize the Expected Return

$$\mathbb{E}[G_t] \doteq \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T]$$

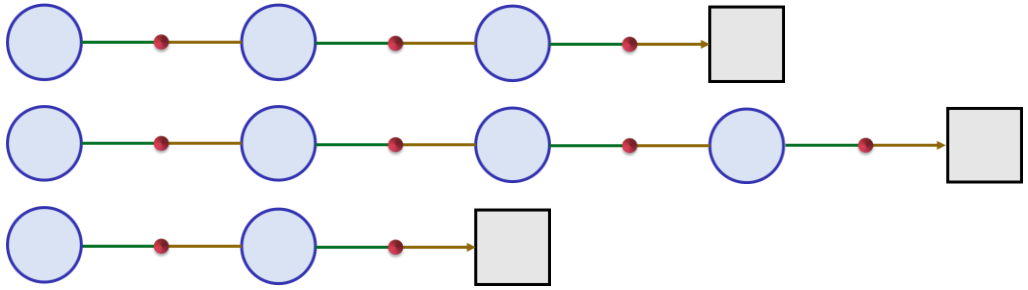
Goals, Rewards, and Return

Rewards

- **Reward is random Variable**
  - could have positive reward at goal, zero reward elsewhere
  - could have negative reward on every time step
  - could have an arbitrary reward function or distribution

Episodic Tasks

- In episodic tasks, the agent-environment interaction breaks into subsequences called ***episodes***.



- Each episode begins independently of how the previous one ended. At termination, the agent is reset to a start state.
- Every episode has a final state which we call the terminal state followed by a reset to a standard starting state.

Goals, Rewards, and Return

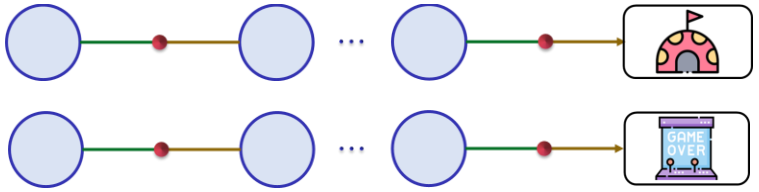
Episodic vs Continuing Tasks

- **Episodic Tasks Examples**
  - **Chess:**
    - Checkmate
    - Draw
    - Resignation

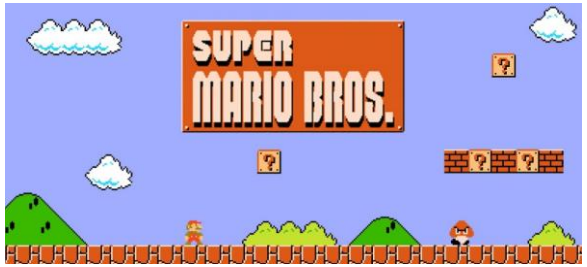
  - A single game of chess would constitute an episode
  - Each game starts from the same start state with all the pieces reset.

  - **Super Mario**
    - Win
    - Lose

  - A single game of Super Mario is a an episode



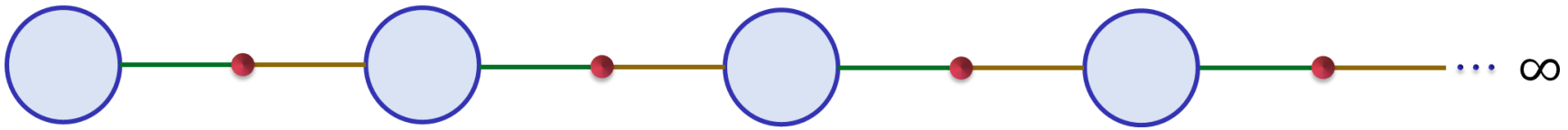
  - Each game starts from the same start state



Continuing Tasks

- Continuing tasks refer to tasks where the agent-environment interaction does not break into episodes and continues without limit.

- Unlike Episodic tasks, continuing tasks cannot be broken up into independent episodes.

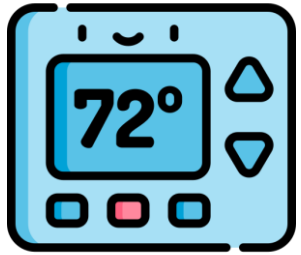


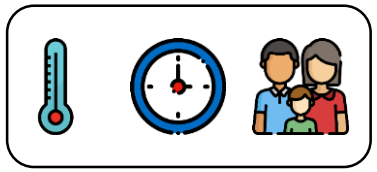
- The agent environment interaction goes on indefinitely.

Goals, Rewards, and Return

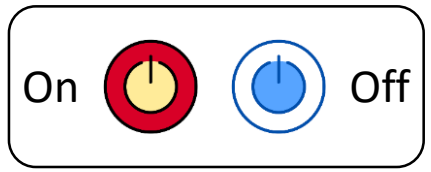
Episodic vs Continuing Tasks

- **Continuing Tasks Examples**
  - **Thermostat:** Never stops interacting with the environment.

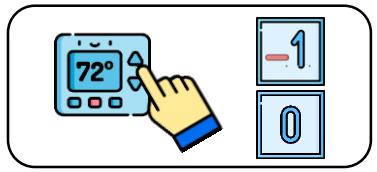




States



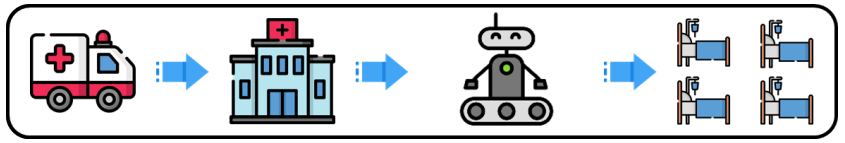
Actions

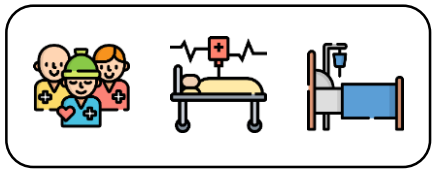


Reward

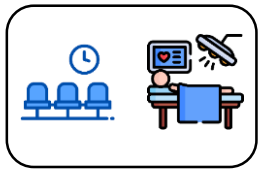
  - The interaction goes on continually. There are no terminal states.

  - **Emergency Department Triage**
    - Assess patients' severity of injury or illness within a short time after their arrival
    - assign priorities, and transfer each patient to the appropriate place for treatment





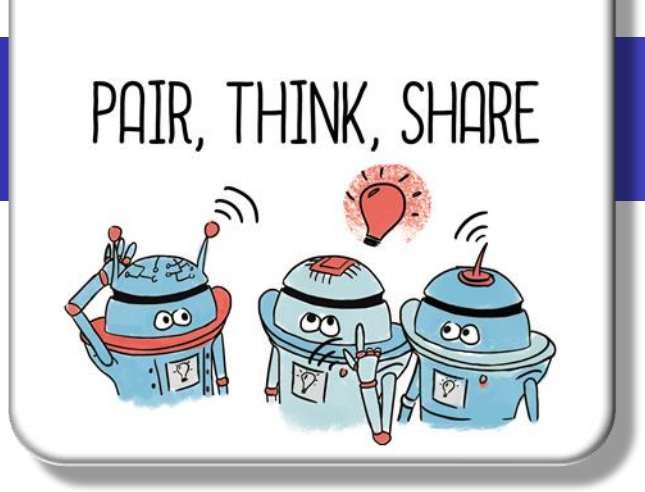
States

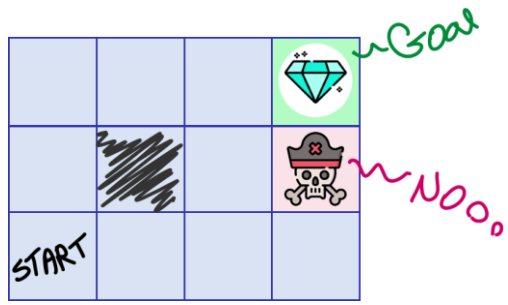


Actions

Reward

  - This an Continuing Task: Agent needs to take an action whenever a new patient arrives or discharges

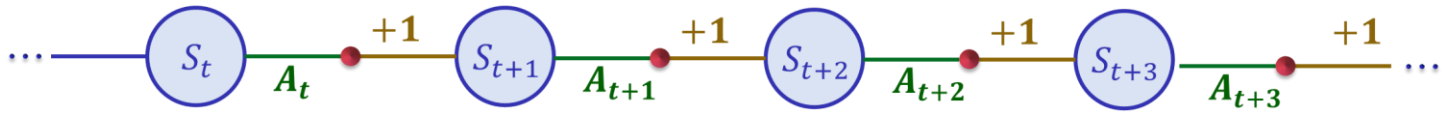


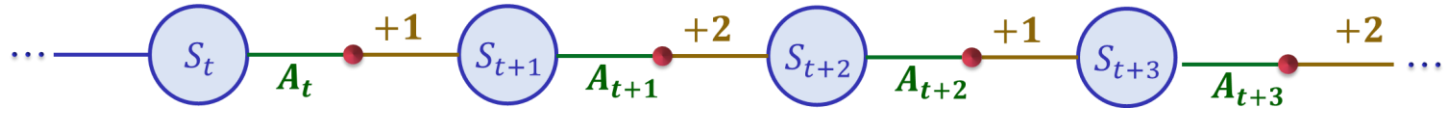


- In a MDP model with **Infinite** Horizons, the ***Utility Function*** for visiting some states is defined as follows:

$$U(S_t, S_{t+1}, S_{t+2}, S_{t+3}, \dots) = \sum_{i=t}^{\infty} R(S_i),$$

- Which of the following sequences has the largest/better Utility?

1. 

2. 

Discounted Return

- **According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.**

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

where γ is a parameter, $0 \leq \gamma \leq 1$, called the ***discount factor***.

- **Discount factor guaranties the return G_t to be finite with the upper bound of:**

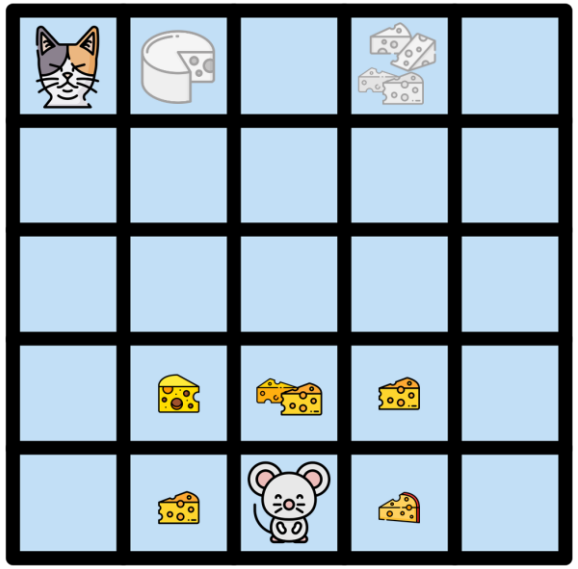
$$G_t \leq R_{max} \times \frac{1}{1-\gamma}$$

where R_{max} is the maximum reward the agent can receive.

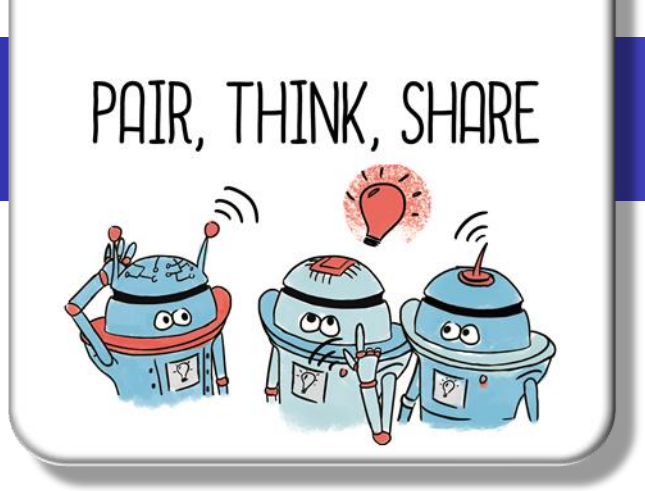
Discounted Return

- **According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.**

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



Net Present Value



Discounted Return

- According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.

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

where γ is a parameter, $0 \leq \gamma \leq 1$, called the ***discount*** factor.

Discounted Return: Effect of γ on Agent's behavior

1. **How does the agent behave if we set $\gamma = 0$**

1. **How does the agent behave if we set $\gamma = 1$**

Recursive Nature of Returns

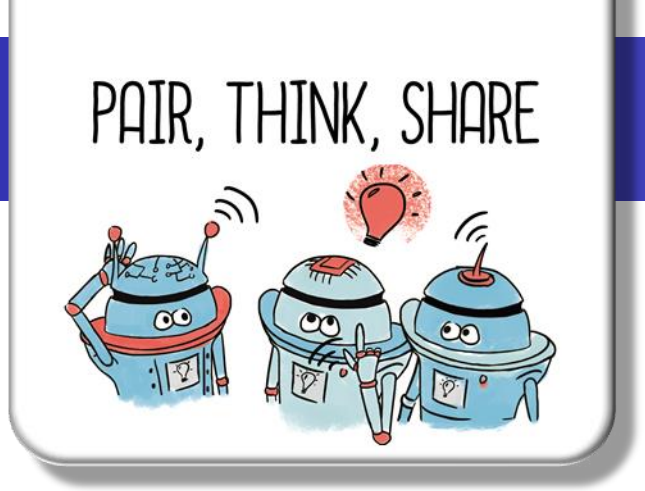
- **We can find the Recursive format of G_t as follows:**

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

$$= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right)$$

$$\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) = G_{t+1}$$

$$G_t = R_{t+1} + \gamma G_{t+1}$$



$$G_t = R_{t+1} + \gamma G_{t+1}$$

- **Suppose $\gamma = 0.5$ and the reward sequence is**
  - $R_1 = 1, R_2 = 6, R_3 = -12, R_4 = 16$, and then zeros for R_5, and later

1. **What are the following?**

   $G_4 =$ $G_3 =$ $G_2 =$ $G_1 =$

2. **Suppose $\gamma = 0.5$ and the reward sequence is all 1s**

   $G =$

3. **Suppose $\gamma = 0.5$ and the reward sequence is**
   - $R_1 = 1, R_2 = 13, R_3 = 13, R_4 = 13$, and so on, all 13s

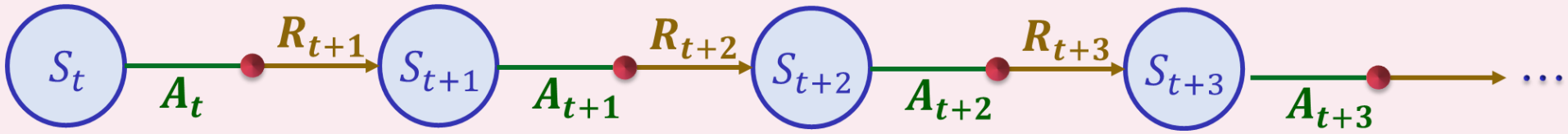
   $G_2 =$ $G_1 =$ $G_0 =$

Episodic vs Continuing Tasks - Summary

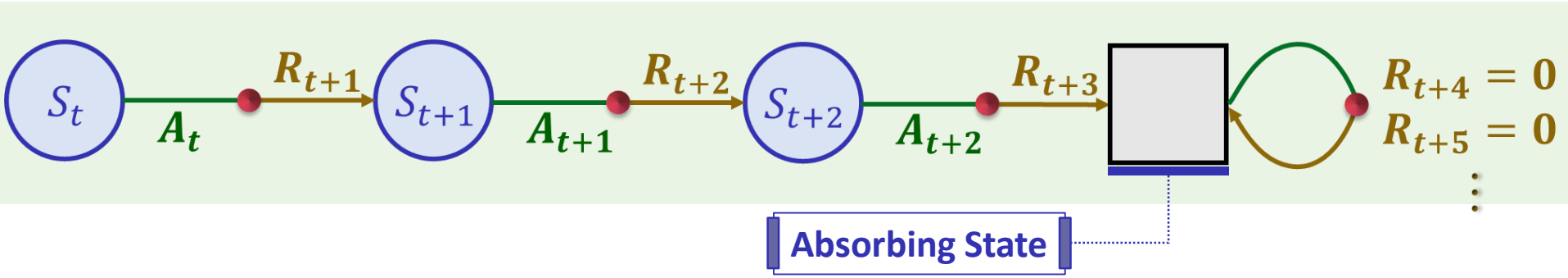
	Episodic Tasks	Continuing Tasks
Interaction	Break naturally into ***episodes***	Goes on ***continually***
Termination	Terminates at ***Terminal State***	Goes on ***indefinitely***
Episodes	Independent	Single episode
Unify Notations	$S_{t,i}$ the state representation at time t of episode i	S_t

Episodic vs Continuing Tasks - Summary

Continuing Tasks



Episodic Tasks



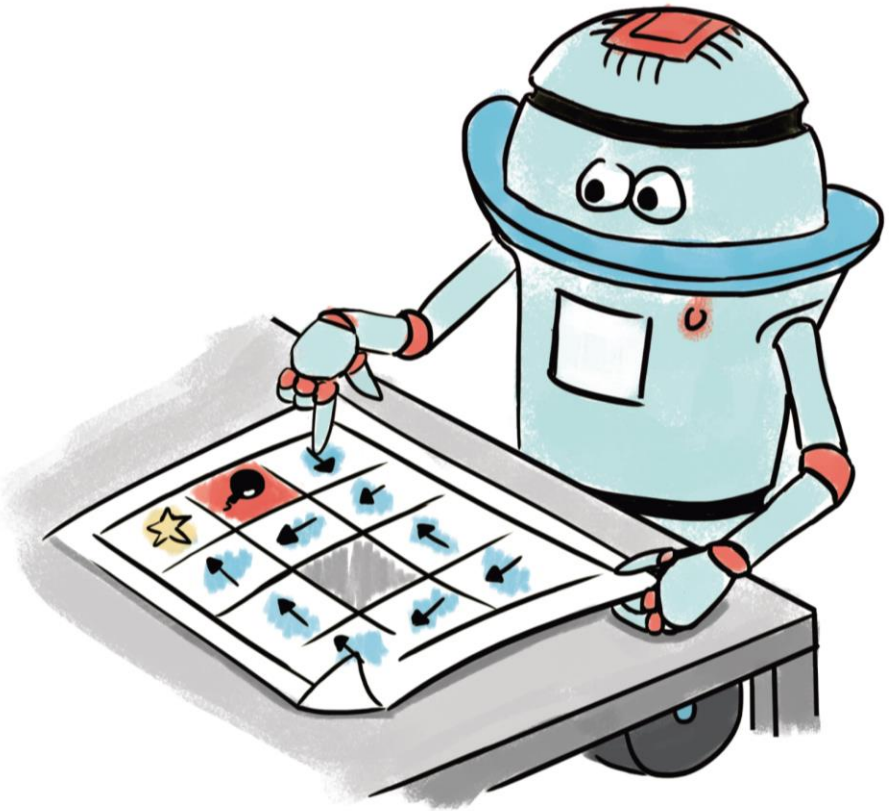
Discounted Return

- According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.

Infinite Horizon

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

Policies



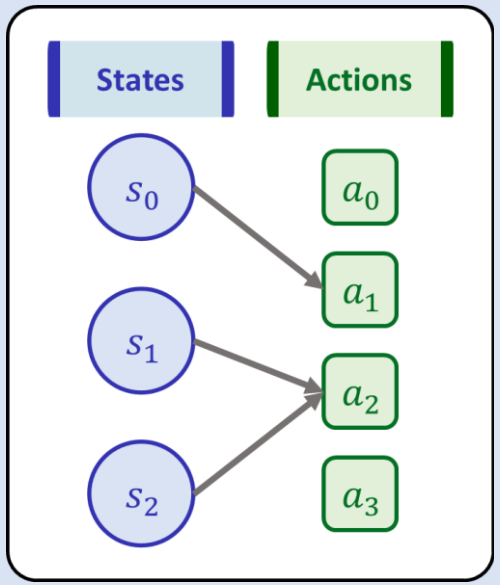
Policies

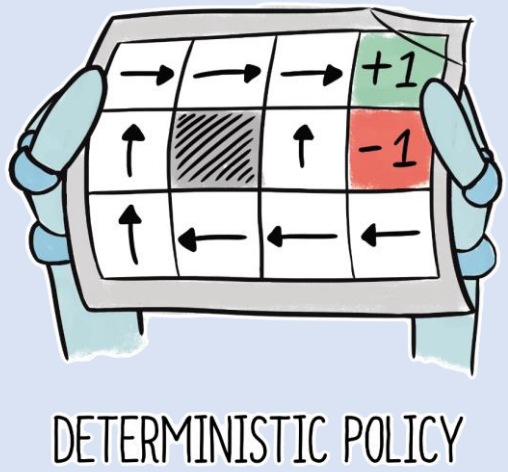
- **Agent's role in this interaction is to choose an *action* on each time *step*.**
- **Almost all RL algorithms involve estimating value functions**
  - Functions of states
  - Function of state–action pairs
    - These functions estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state)
- **The choice of action has an immediate impact on both the *immediate reward*, and the next state.**
  - How to maximize the future rewards that can be expected (expected return)
- **Value functions are defined with respect to particular ways of acting, called *policies***

Determinist Policy

- A policy is a mapping from states to probabilities of selecting each possible action.

$$\pi(s) = a$$



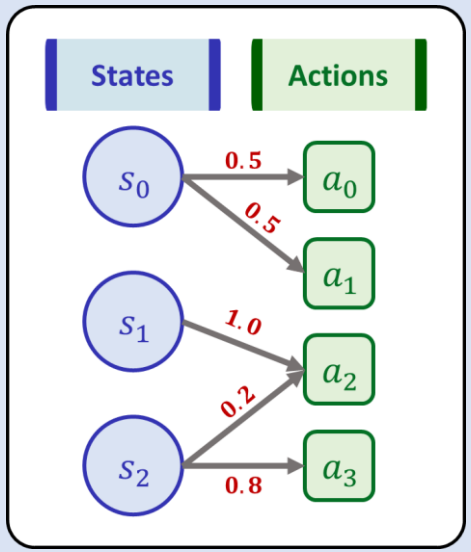


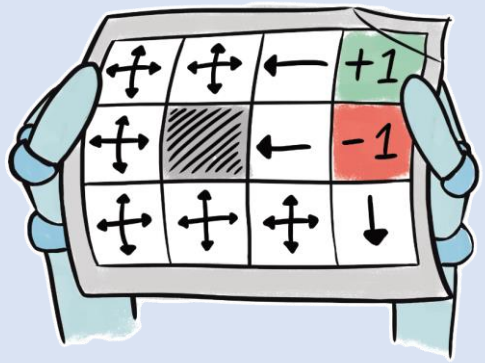
State	Action
S_0	a_1
S_1	a_2
S_2	a_2

Stochastic Policy

- A policy is a mapping from states to ***probabilities*** of selecting each possible action.

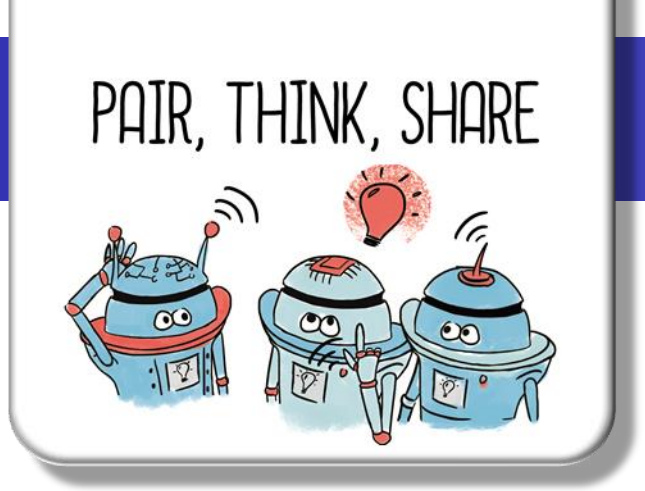
1. $\pi(a|s) = a$
2. $\sum_{a \epsilon \mathcal{A}(s)} \pi(a|s) = 1$
3. $\pi(a|s) \geq 0$





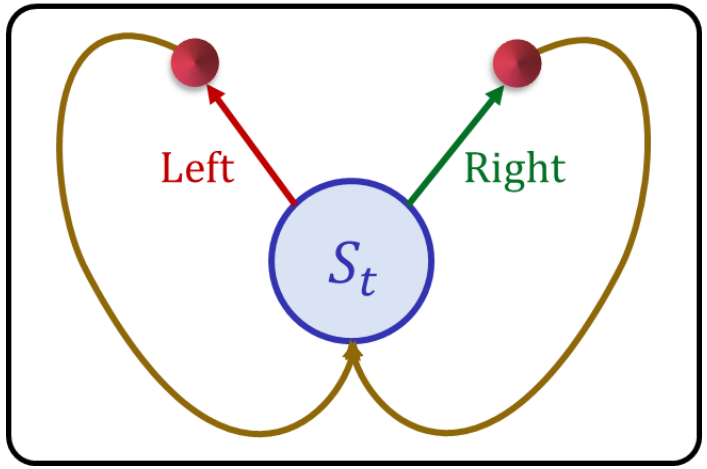
STOCHASTIC POLICY

State	Action	$\pi(a\|s)$
s_0	a_0	0.5
	a_1	0.5
s_1	a_2	1.0
s_2	a_2	0.2
	a_3	0.8



Valid and Invalid Policies

- It's important that policies depend only on the ***current state***, not on other things like ***time*** or ***previous states***
- Assume there is policy that chooses to go either left or right with equal probability.
- Which of the following examples is a valid Policy, and which is not? Why?



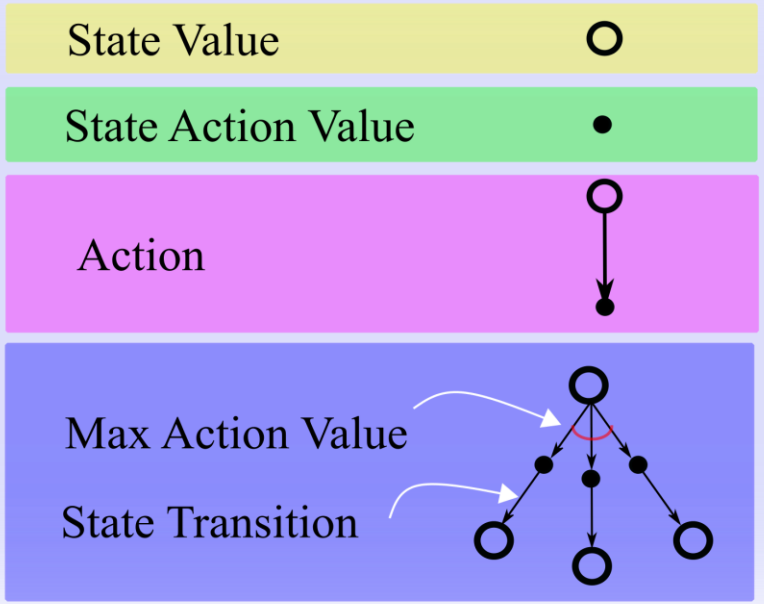
1. **Left** with 50% prob. and **Right** with 50% prob.

   L L R L R R L R L L L R R

2. Alternate **Left** and **Right**

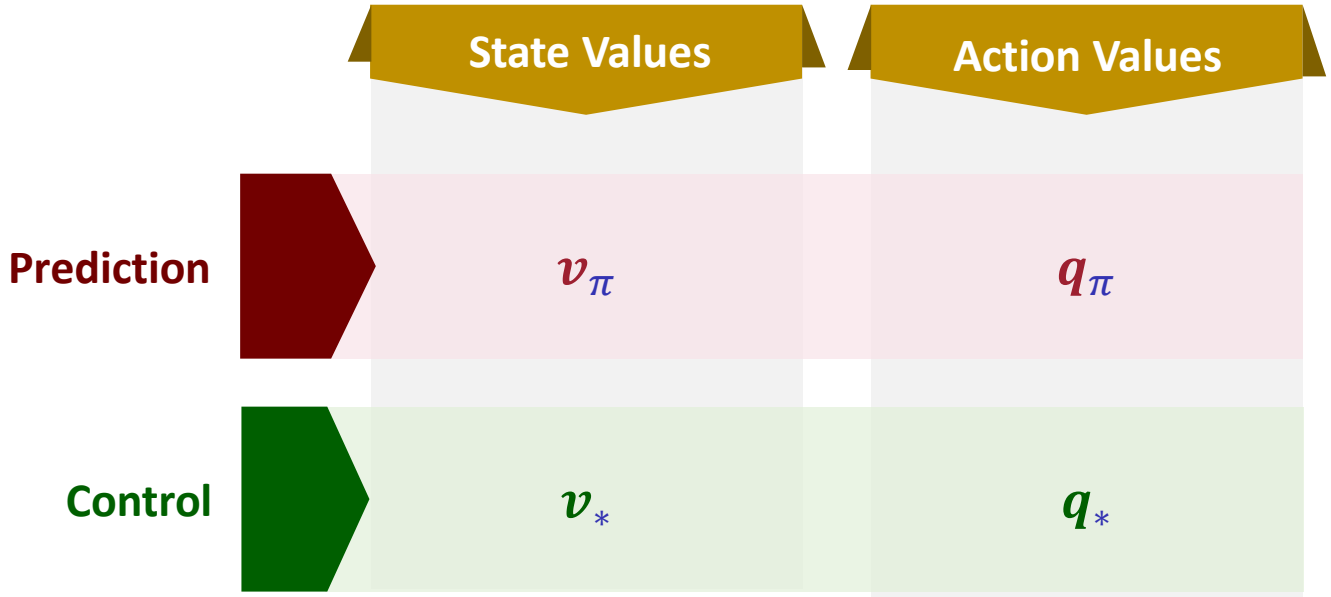
   L R L R L R L R L R L R L

Value Functions



Value Functions

- In reinforcement learning, reward captures the notion of short-term gain.
- The objective however, is to learn a policy that achieves the most reward in the long run.
- Value functions formalize what this means.
- Value functions estimate how good it is for the agent to be in a given state
  - It determines how good it is to perform a given action in a given state



Value Functions

State-value function

state-value function for policy π

- A ***state value function*** is the future award an agent can expect to receive starting from a particular state.
- Value of State S when acting according to policy π:

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

where

- $E[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π
- t is any time step

» Value of a state == expected return from that state if agent follows policy «

Value Functions

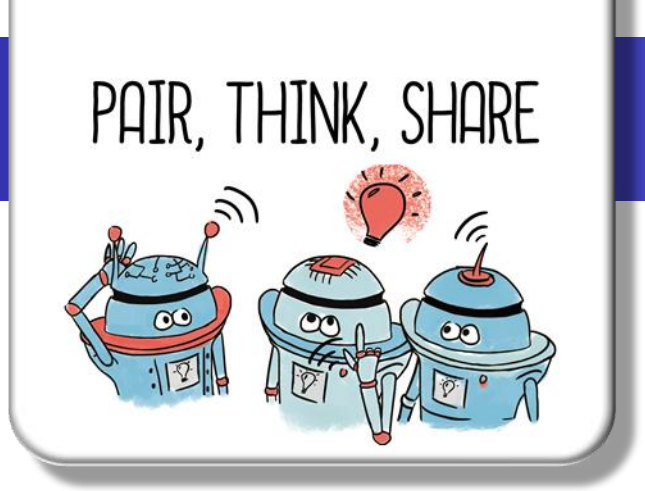
Action-value function

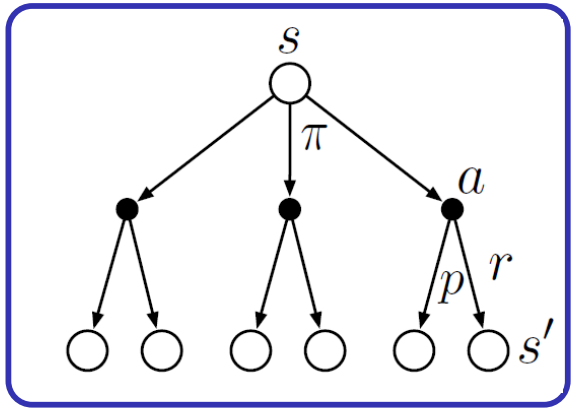
action-value function for policy π

- The ***action value*** of a state is the expected return if the agent selects action A and then follows policy Pi.

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

» Value of a state/action pair
==
expected return when taking action a from state s and following policy after that «



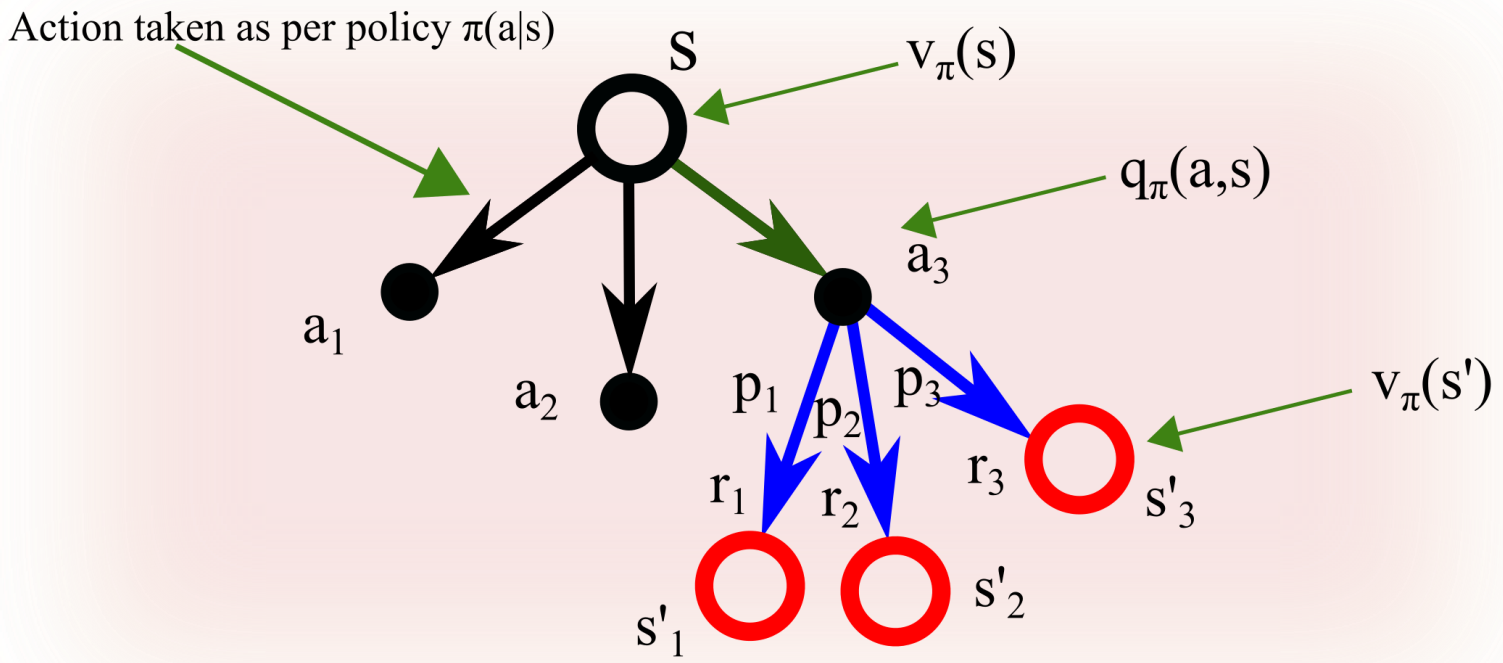


- Give an equation for v_π, in terms of q_π and π

- Give an equation for q_π, in terms of v_π and the four argument $p(s', r|s, a)$

Value Functions

Backup diagram



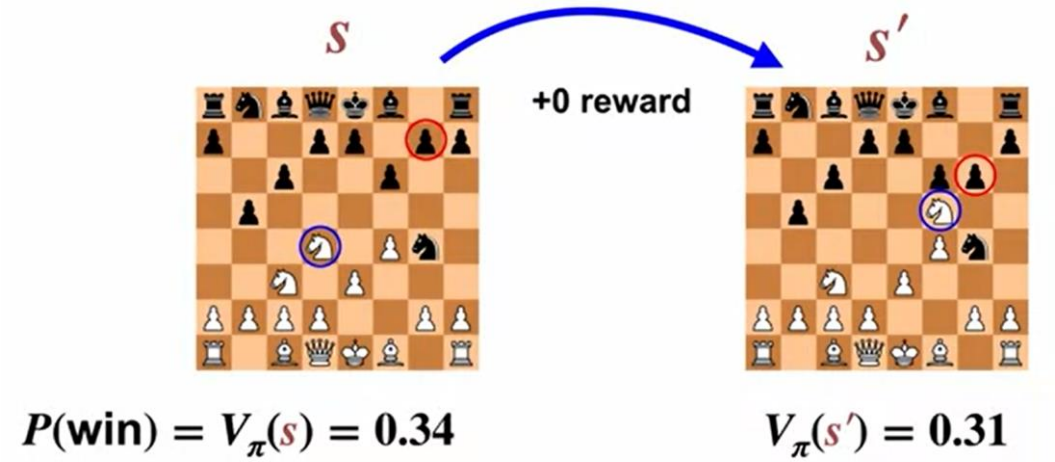
Backup Diagram for State Value $v_\pi(s)$
and Action Value $q_\pi(s,a)$ with all components

Value Functions

- **Value functions allows an agent to query the quality of its current situation instead of waiting to observe the long-term outcome.**

  - **Benefits:**
    - First: the return is not immediately available
    - Second: the return may be random due to stochasticity in both the policy and environment dynamics.

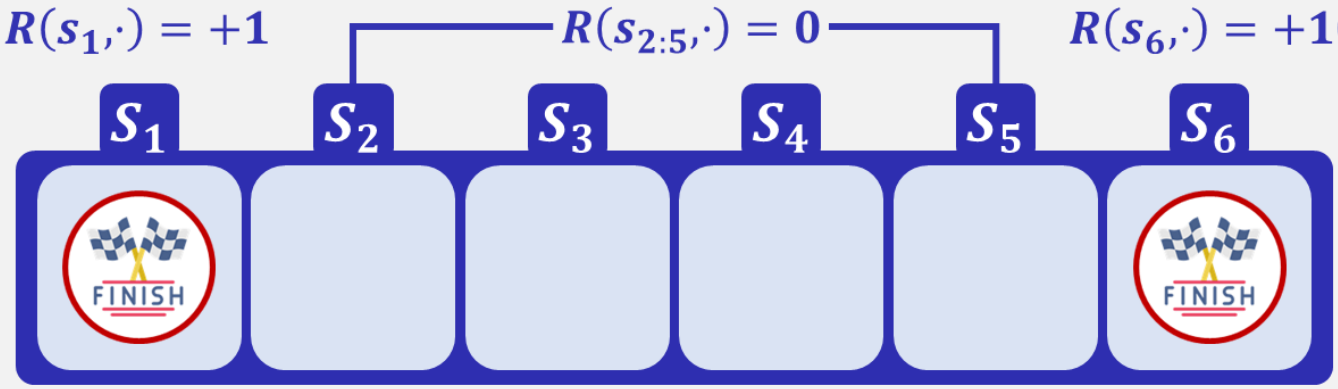
- **The value function summarizes all the possible futures by averaging over returns, enable us to judge the quality of different policies**



Value Functions

Example 1

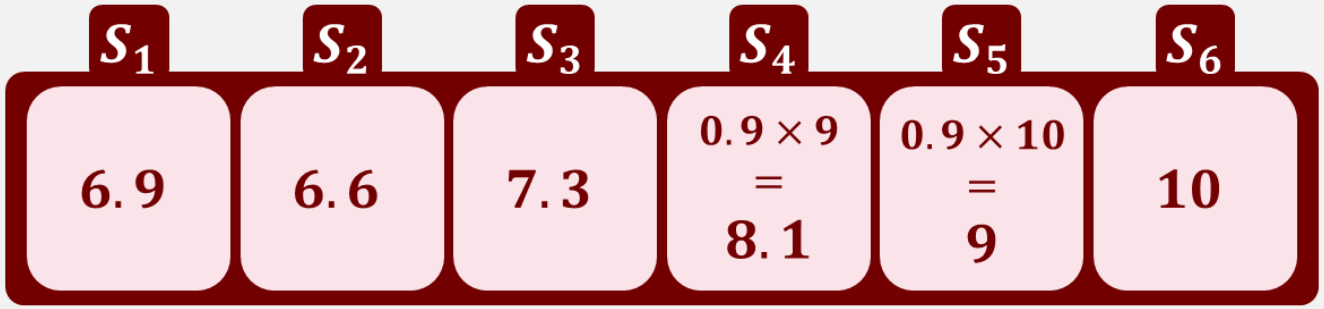
States



Terminal State

Policy

Value fn

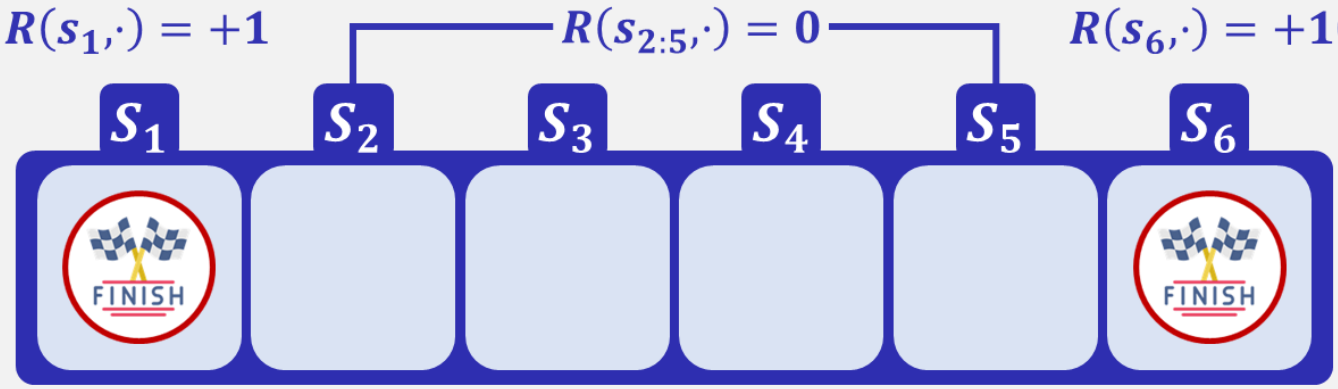


$\gamma = 0.9$

Value Functions

Example 2

States



Terminal State

Policy

Value fn

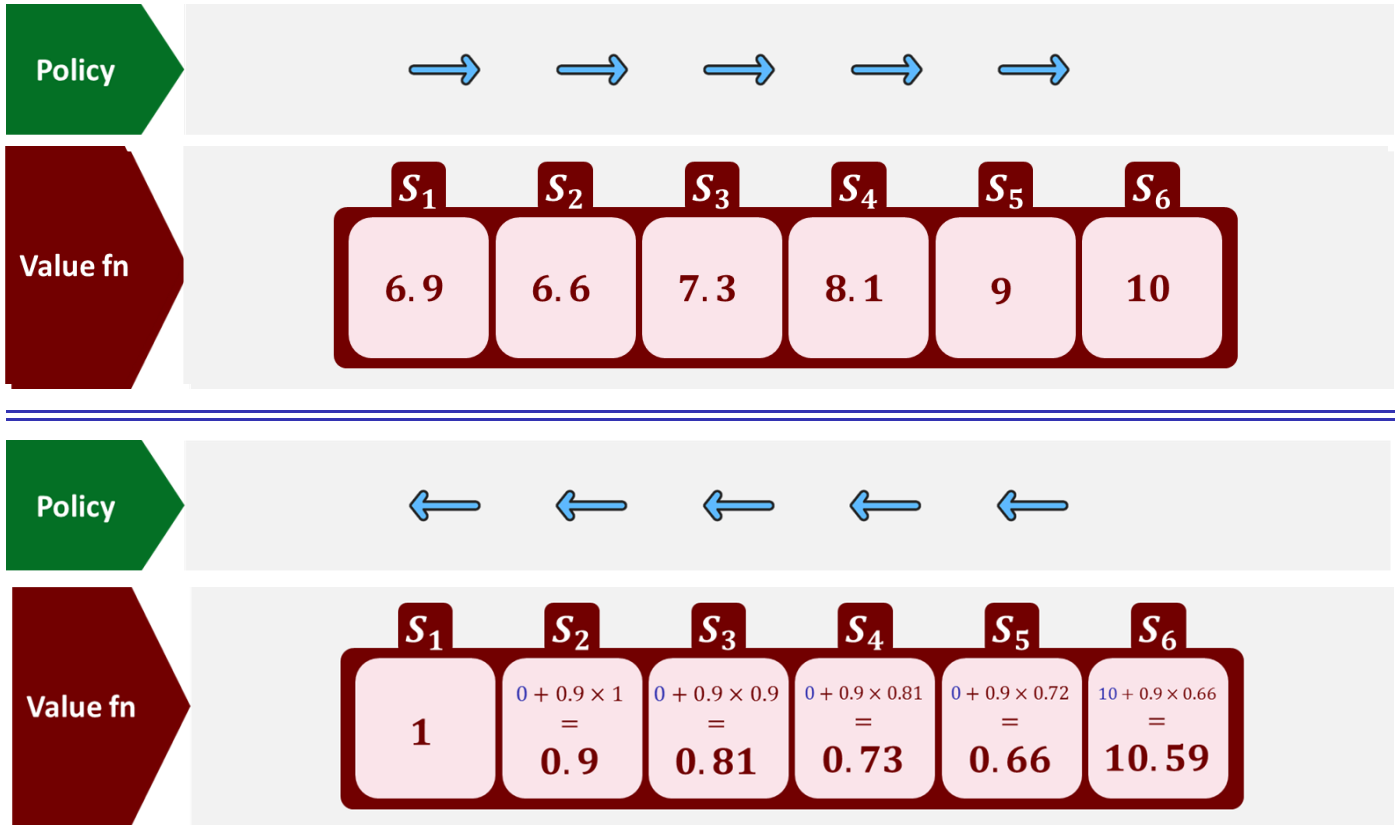
S_1	S_2	S_3	S_4	S_5	S_6
1	$0 + 0.9 \times 1 = 0.9$	$0 + 0.9 \times 0.9 = 0.81$	$0 + 0.9 \times 0.81 = 0.73$	$0 + 0.9 \times 0.72 = 0.66$	$10 + 0.9 \times 0.66 = 10.59$

$\gamma = 0.9$

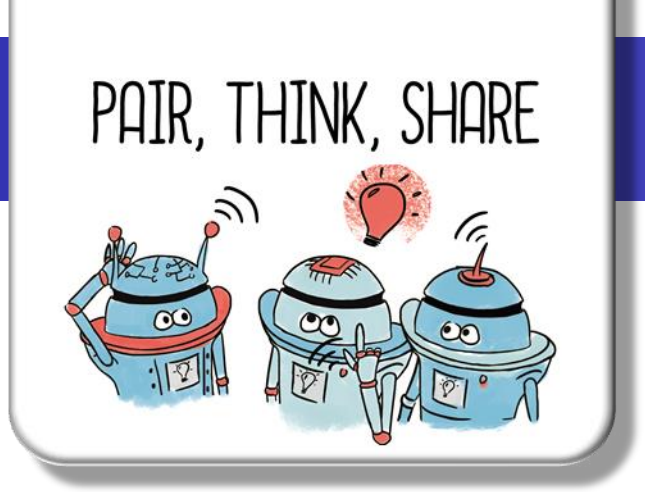
Value Functions

Compare policies: Example 1 vs Example 2

- **Value function can help us compare two different policies**



$\gamma = 0.9$



- Find the Value Functions based on the following model and policy

States

$R(s_1,\cdot) = +1$ $R(s_{2:5},\cdot) = -1$ $R(s_6,\cdot) = +10$

S_1 S_2 S_3 S_4 S_5 S_6

FINISH FINISH

Terminal State

Policy

→ → → → →

Value fn

S_1 S_2 S_3 S_4 S_5 S_6

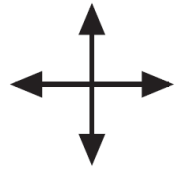
? ? ? ? ? ?

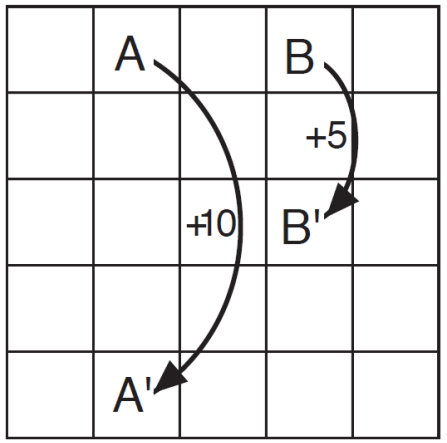
$\gamma = 1.0$

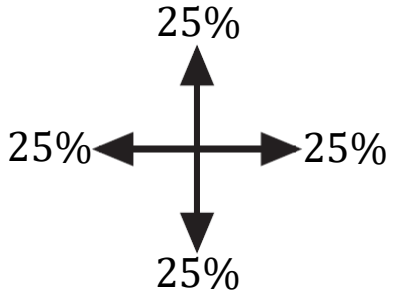
Value Functions

Example 3: The Grid World

- **Actions:**
  - ***north***, ***south***, ***east***, and ***west***
- ***Rewards***
  - *From state A, all four actions yield a reward of $+10$ and take the agent to A'*
  - *From state B, all actions yield a reward of $+5$ and take the agent to B'.*
  - *Actions that cause the agent bum to the wall yield a reward of -1 (location unchanged)*
  - *Other actions result in a reward of 0*
- ***Policy***
  - *Agent Selects four actions with equal probability*

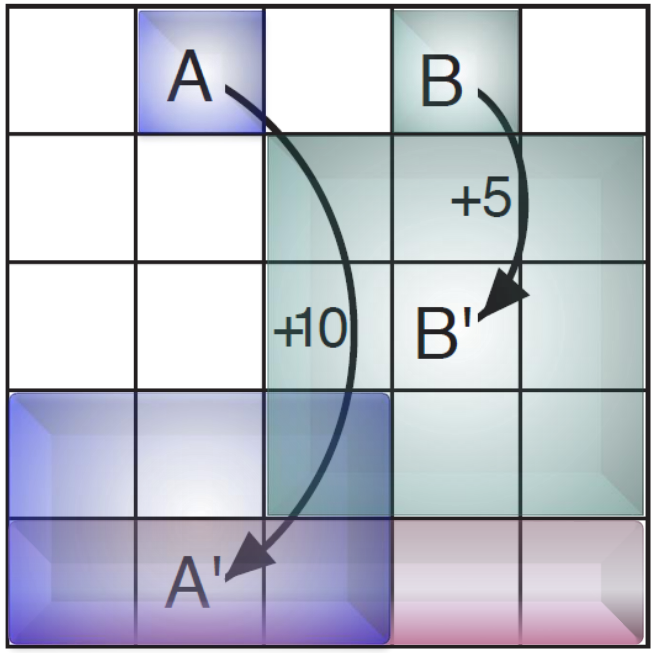




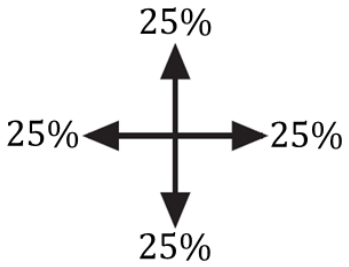


Value Functions

Example 3: The Grid World



$\gamma = 0.9$



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

Value Functions

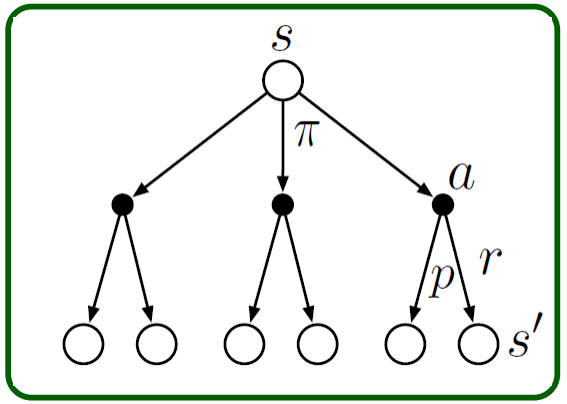
State-value function (Recursive Format)

state-value function for policy π

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

$$G_t = R_{t+1} + \gamma G_{t+1}$$

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



Bellman Equation for State Value

- Bellman equation allows us to express values of states as values of other states.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')], \text{ for all } s \in \mathcal{S}$$

- The Bellman equation for the state value function defines a relationship between the value of a state and the value of his possible successor states
  - It allows us to relate the value of the current state to the value of future states without waiting to observe all the future rewards.
- The 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.

Value Functions

Bellman Equation : The Grid World Example

- **The Bellman equation must hold for each state for the value function.**
  - Show numerically that this equation holds for the center state, valued at $+0.7$, with respect to its four neighboring states valued at $+2.3$, $+0.4$, -0.4, and $+0.7$. $\gamma = 0.9$

1. $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]$
2. the centre tile transition probability $p(s', r|s, a) = 1$
3. $v_\pi(s) = \sum_a \pi(a|s) \gamma \left(\sum_{s'} v_\pi(s')\right)$
4. $v_\pi(s) = \sum_a \pi(a|s) \gamma \times 3$
5. $v_\pi(s) = \sum_a \pi(a|s) \times 2.7$

$$v_\pi(centre) = \frac{2.7}{4} \approx +0.7.$$

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

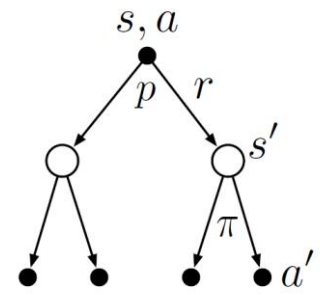
Value Functions

Bellman Equation: action-value function (Recursive Format)

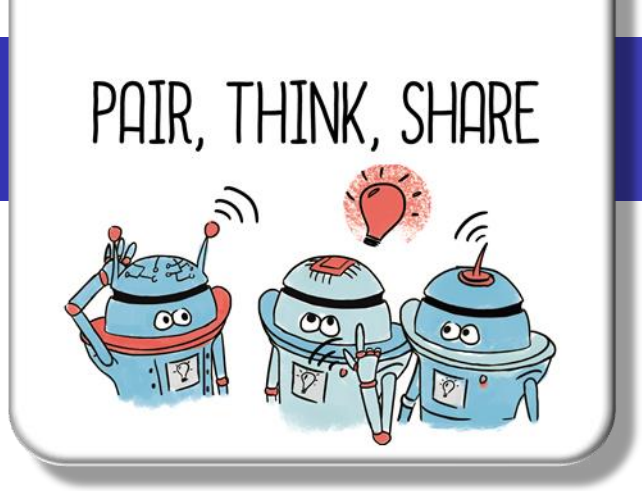
Bellman Equation for Action Value

- Based on the Bellman Equation, we can define the ***action value*** as a recursive equation for the value of a state action pair in terms of its possible successors state action pairs.

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



q_π backup diagram



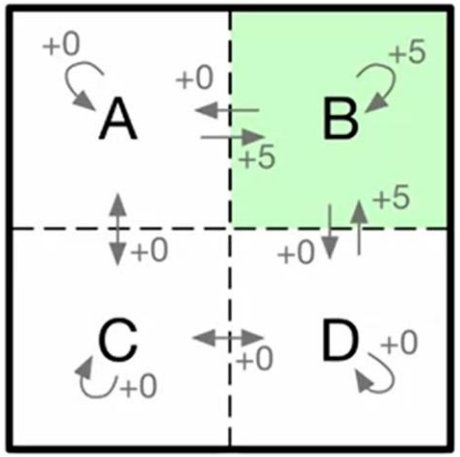
Bellman Equation for State Value

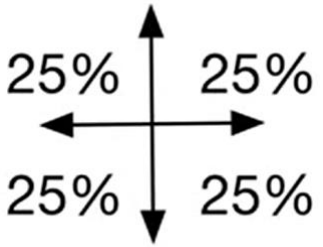
- Bellman equation allows us to express values of states as values of other states.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')], \text{ for all } s \in \mathcal{S}$$

Consider the following World Grid with four States

- **Actions:**
  - ***up***, ***down***, ***laft***, and ***right***
- ***Rewards***
  - *Any Action that yields to land in state B will receive (+5)*
    - *Including starting in state B and hitting a wall to remain there*
  - *Other actions result in a reward of* 0
  - $\gamma = 0.7$
- ***Policy***
  - *Agent Selects four actions with equal probability*





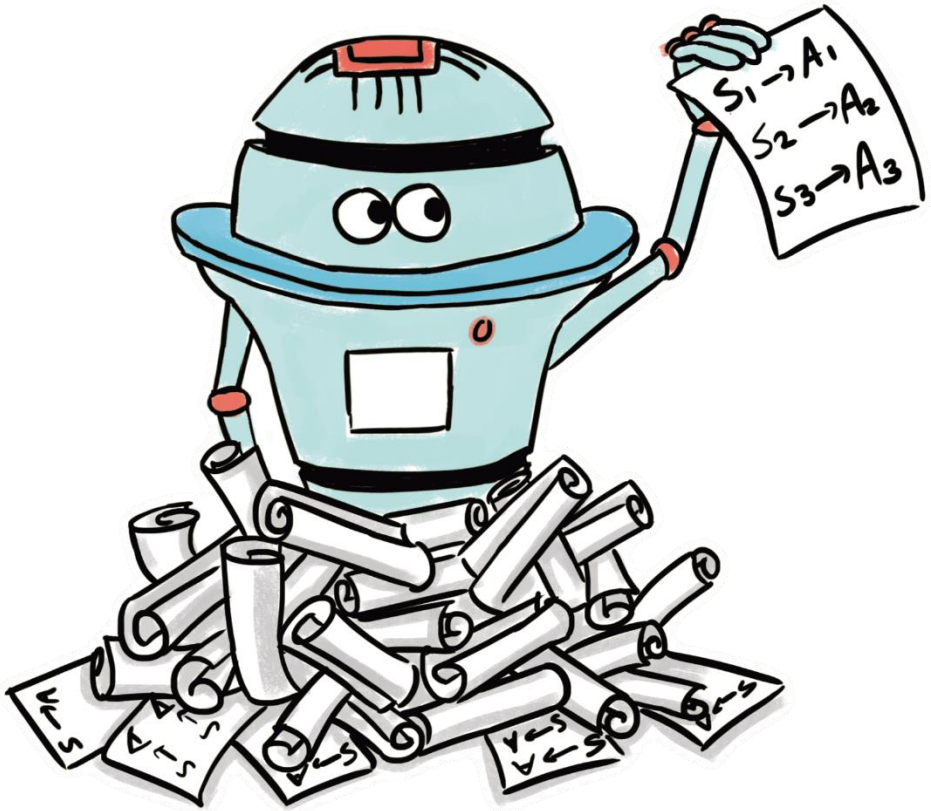
Use the Bellman Equation and write State Value function for State A $[v_\pi(A)]$

Why Bellman Equation?

- **We can directly solve Small MDPS**

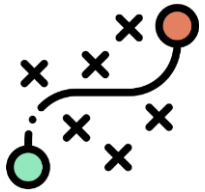
- **However, in more complex problems, this may not be practical**

Optimal Policies and Optimal Value Fn



Optimal Policy

Introduction



The policy states to actions and specifies how an agent to behave.

- Given this way of behaving, we then aim to find the value functions.

The ultimate goal of RL is to find a policy that obtains the max reward in the long run

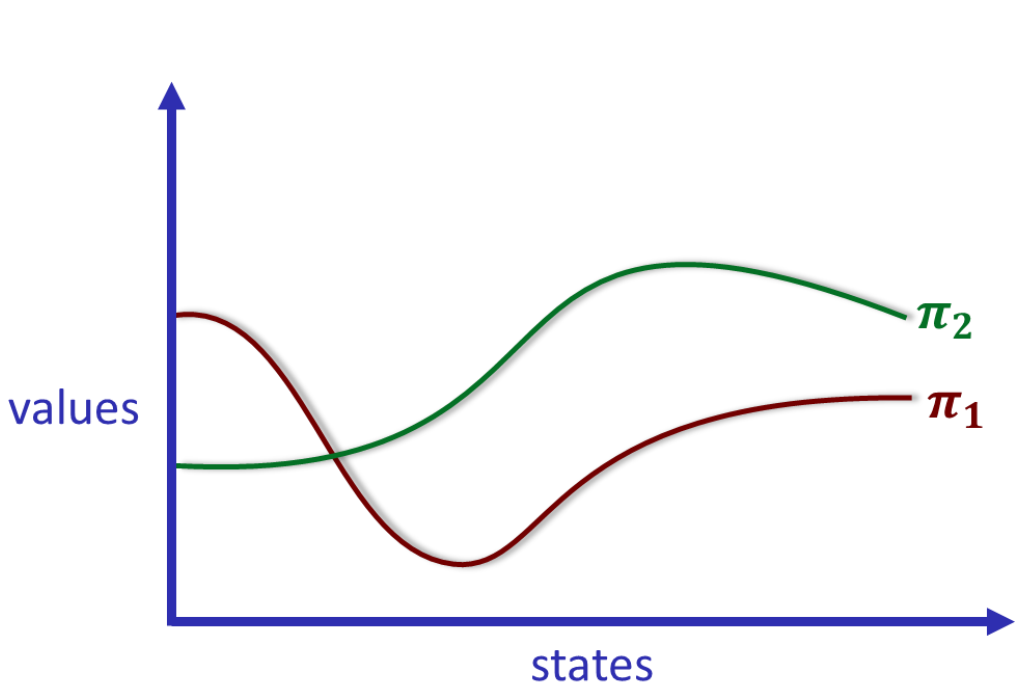
- Simply, evaluating the effectives of every single possible policy is not really practical

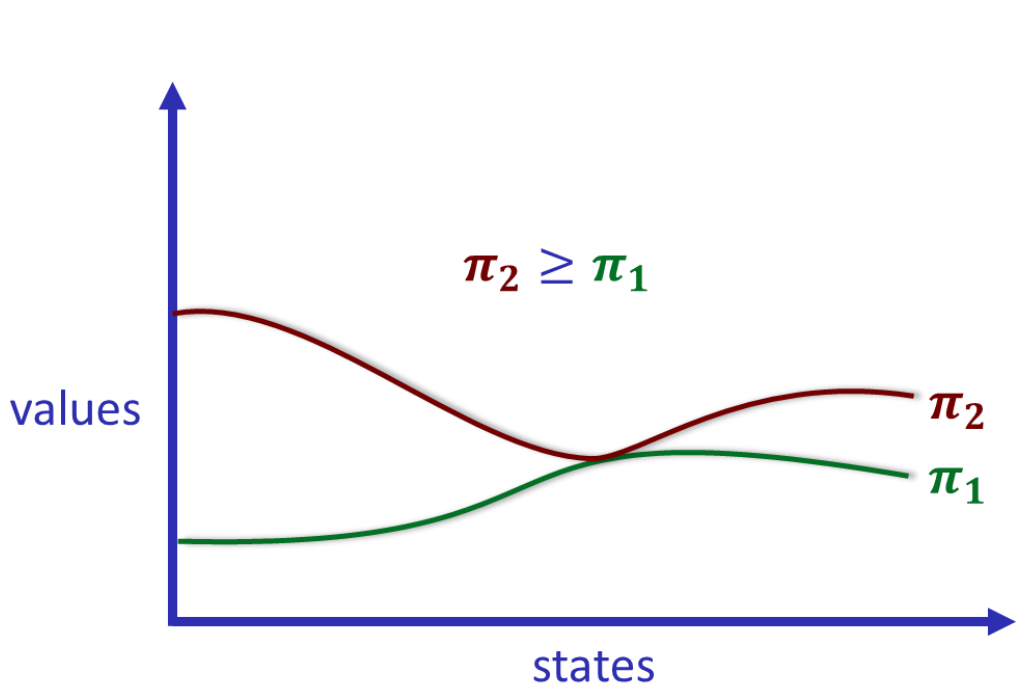
We need to define an approach that learns from some arbitrarily policies and then converges to the optimal policy

- We need to understand what it means for one policy to be better than another

Optimal Policy

Optimal Policy illustration

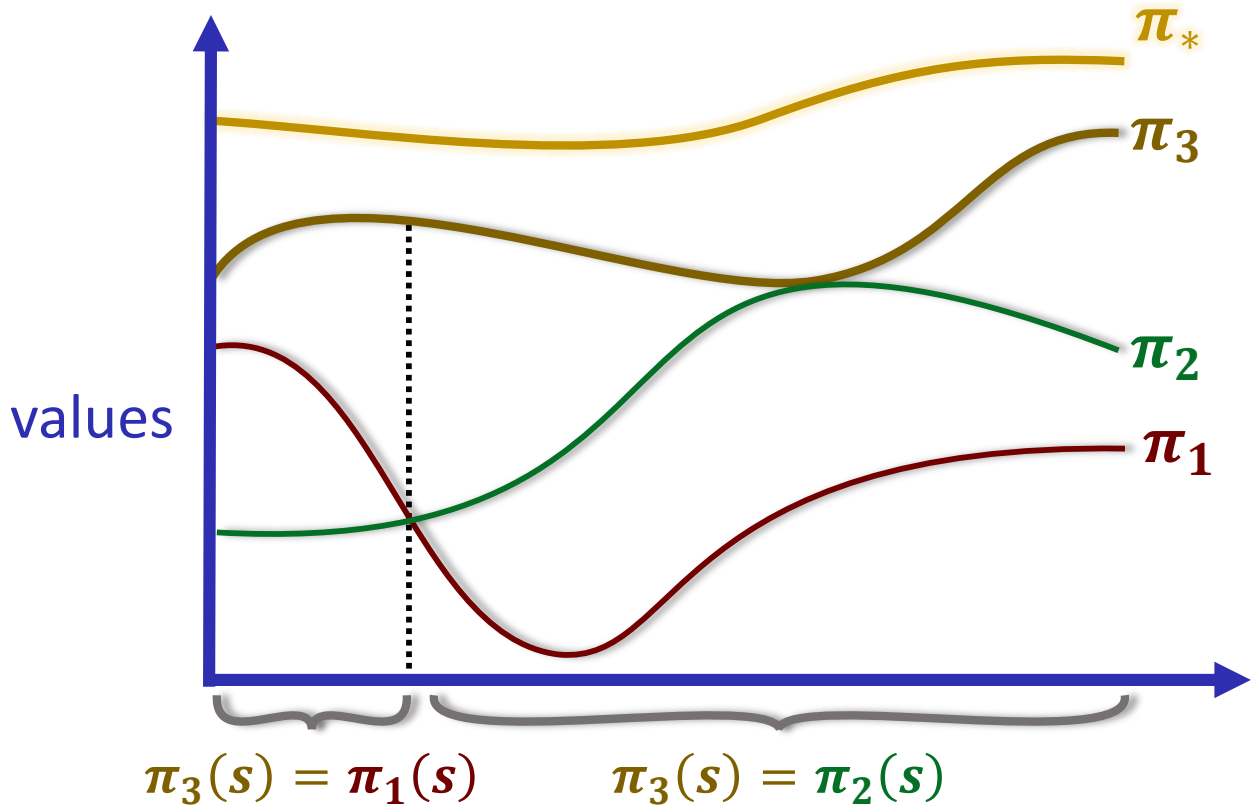




- Policy π_2 is *as good as or better than policy* π_1, if and only if, the value under π_2 is greater than or equal to the value under π_1 for every state

Optimal Policy

Optimal Policy illustration



- **there always exists some policy which is best in every state**

Optimal Policy

Definition

Optimal Policy

- 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 $s \in \mathcal{S}$.
- We denote all the optimal policies by π_*

- The optimal policy is ***as good as*** or ***better*** than all other policies
- There's always at least ***one*** optimal policy, but there may be more than one.
- We can proof that there must always exist at least one optimal deterministic policy.

Optimal Policy

A simple Example

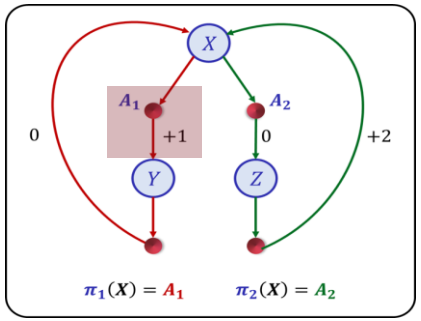
- **Which of the following policies is optimal?**
  - $\pi_1(X) = A_1$
  - $\pi_2(X) = A_2$

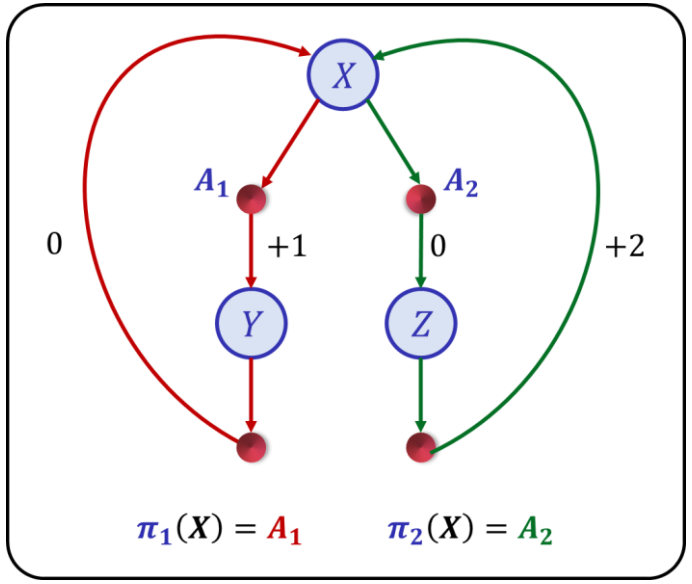
immediate reward

$\gamma = 0$

$\pi_1(X) = A_1$

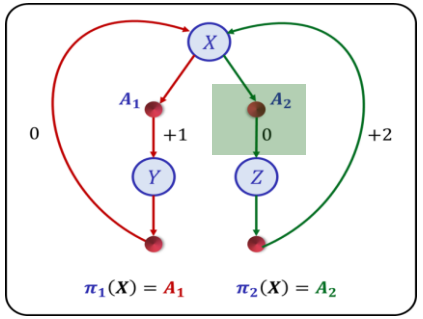
$\pi_1(X) = 1$ ✓





$\pi_2(X) = A_2$

$\pi_1(X) = 0$



Optimal Policy

A simple Example

- **Which of the following policies is optimal?**
  - $\pi_1(X) = A_1$
  - $\pi_2(X) = A_2$

immediate/delayed reward

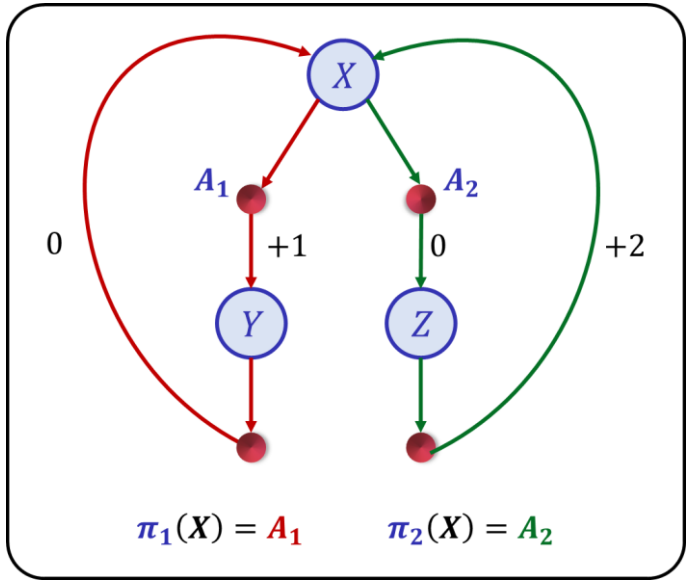
$\gamma = 0.9$

$\pi_1(X) = A_1$

$\pi_1(X) = 1 + 0.9 \times 0 + 0.9^2 \times 1 + \cdots$

$= 1 \times \sum_{k=0}^{\infty} (0.9)^{2k}$

$= \frac{1}{1-(0.9)^2} \approx 5.3$



$\pi_2(X) = A_2$

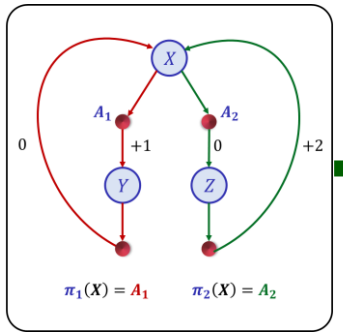
$\pi_1(X) = 0 + 0.9 \times 2 + 0.9^2 \times 0 + \cdots$

$= 2 \times \sum_{k=0}^{\infty} (0.9)^{2k+1}$

$= 2 \times \frac{0.9}{1-(0.9)^2} \approx 9.5$ ✓

Optimal Policy

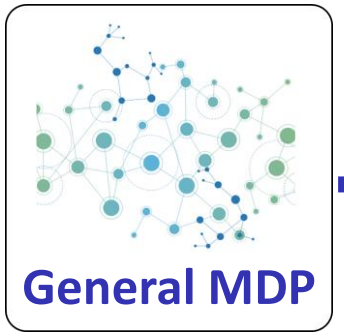
This Example



2
Deterministic Polices

Brute-Force
Search

In General



General MDP

$|\mathcal{A}|^{\mathcal{S}}$
Deterministic Policies

intractable

Brute-Force
Search

Optimal Value Functions

Bellman optimality equation

Definition

- **The value function for the optimal policy has the greatest value possible in every state (v^*)**

$$v_{\pi^*}(s) \doteq \mathbb{E}_{\pi^*}[G_t \mid S_t = s] = \max_{\pi} v_{\pi}(s) \text{ for all } s \in \mathcal{S}$$

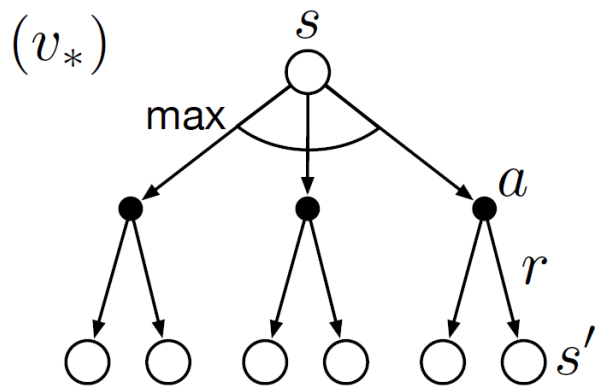
- **Optimal policies also has the optimal action-value function, which is the maximum possible for every state action pair (q^*)**

$$q_{\pi^*}(s) \doteq \mathbb{E}_{\pi^*}[G_t \mid S_t = s, A_t = a] = \max_{\pi} q_{\pi}(s, a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

Bellman optimality equation

- The value of a state under an optimal policy must equal the expected return for the best action from that state

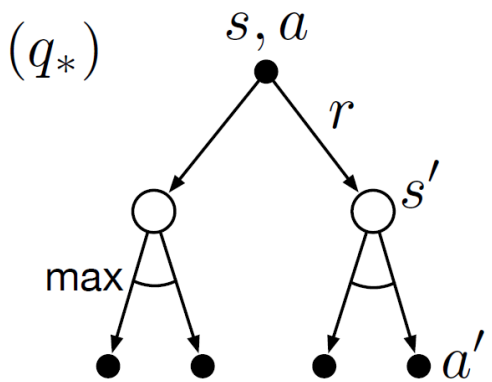
$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma\, v_*(s')], \text{ for all } s \in \mathcal{S}$$



Bellman optimality equation

- The state-action pair value under an optimal policy must equal the expected return for the best action from that state

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

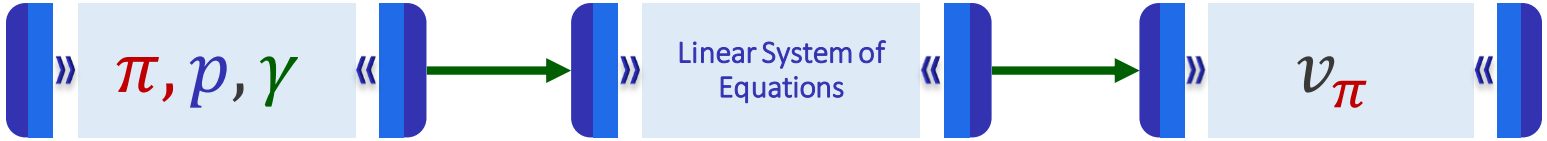


Optimal Value Functions

Bellman Equation vs Bellman optimality equation

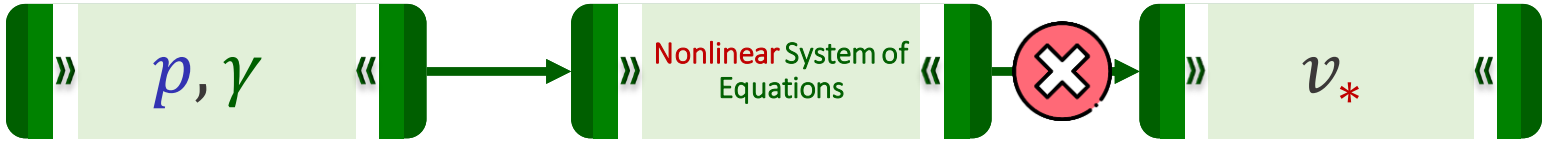
Bellman equation

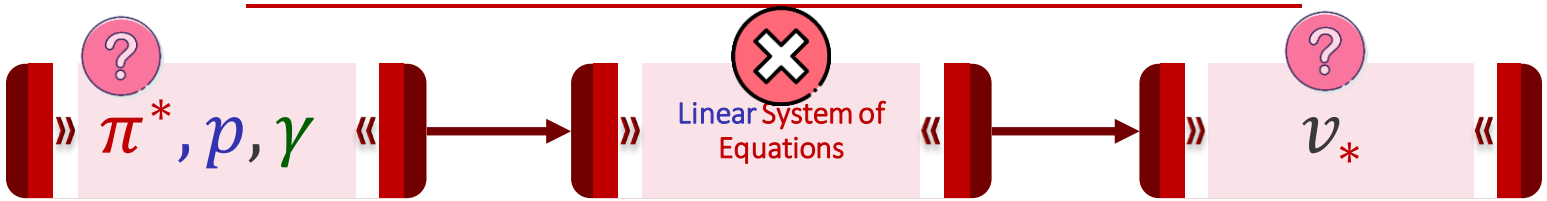
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')], \text{ for all } s \in \mathcal{S}$$

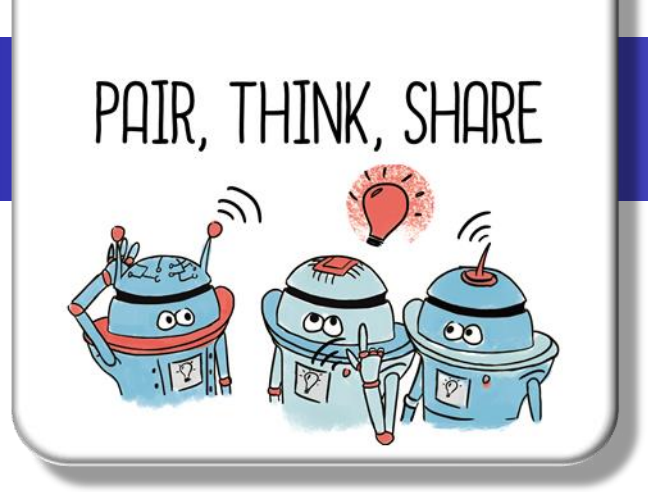


Bellman optimality equation

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma\, v_*(s')]$$

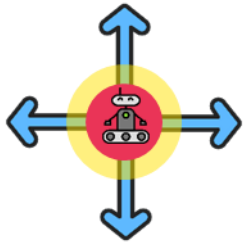


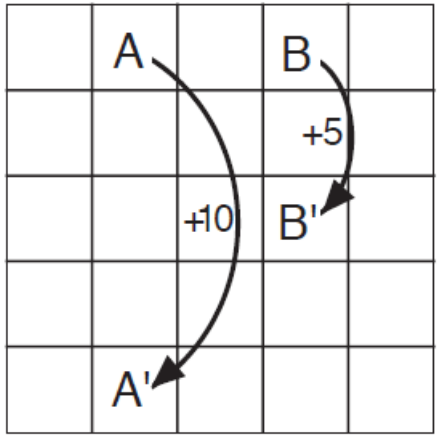




- The provided table shows ***optimal state values*** for the Gridworld example.

- Find the optimal action for the highlighted state?





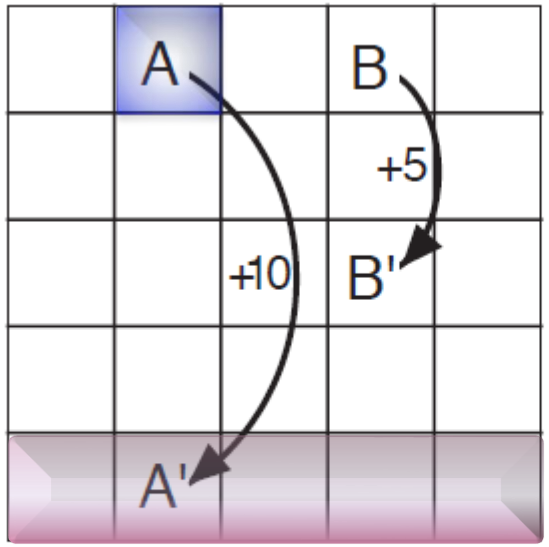
Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*

Optimal Policy

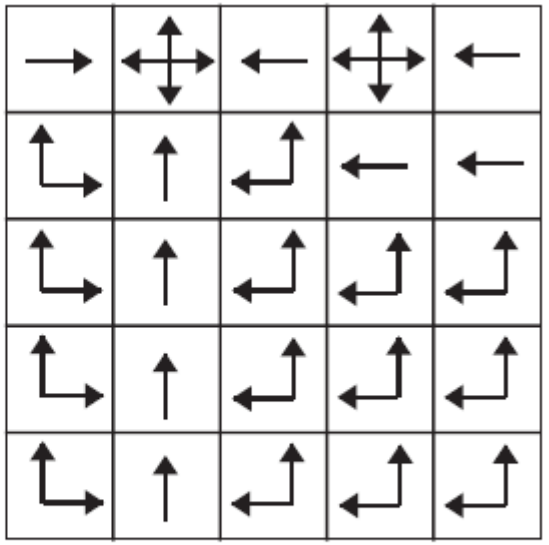
The Grid World Example



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

$$v_*(s) \doteq \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')],$$

$$\pi_*(s) \doteq \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')],$$

Summary

- **MDP**
  - MDP is a classical formalization of sequential decision making
  - MDP is the standard reinforcement learning query oracle where the agent can only interact with the MDP by choosing actions and observe the next state and the reward

- **MDP Tasks**
  - **Episodic Tasks:** the agent-environment interaction breaks into subsequences called episodes.
  - **Continuing tasks:** unlike Episodic tasks, tasks cannot be broken up into independent episodes.

- **Discounted Return**
  - The agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized
  - Discount factor guaranties the return G_t to be finite

Summary

- **Policy:**
  - A policy is a mapping from states to probabilities of selecting each possible action.

- **Function Values**
  - These functions estimate how good it is for the agent to be in a given state
  - Value functions are defined with respect to particular ways of acting indicated by the policy

- **Optimal policy**
  - The policy with the highest value in all states.
  - At least one optimal policy always exists but there may be more than one.
  - In General MDP, searching for the optimal policy by brute force intractable.

References

- [1] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
- [2] C. Gatti, Design of experiments for reinforcement learning. Springer, 2014.
- [3] A. Masadeh, Z. Wang, and A. E. Kamal, “An Actor-Critic Reinforcement Learning Approach for Energy Harvesting Communications Systems,” in 2019 28th International Conference on Computer Communication and Networks (ICCCN), 2019, pp. 1–6.
- [4] G. Tesauro, “Temporal difference learning and TD-Gammon,” Communications of the ACM, vol. 38, no. 3, pp. 58–68, 1995.