

Reinforcement Learning (AI644)

Prepared by Dr. Ashraf Elsayed

Lecture 3





مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY

Reinforcement Learning

Lecture 3

Prepared by
Dr. Ashraf Elsayed

www.midocean.ae

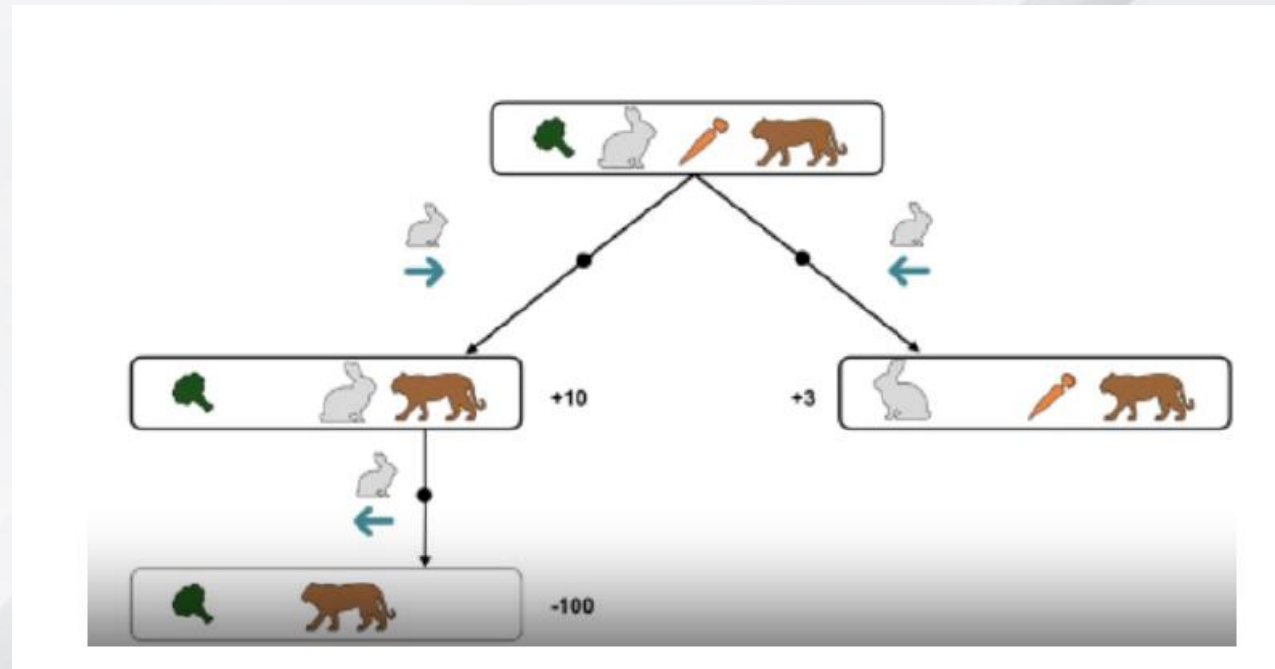
Markov Decision Processes

The objectives of this lecture is to:

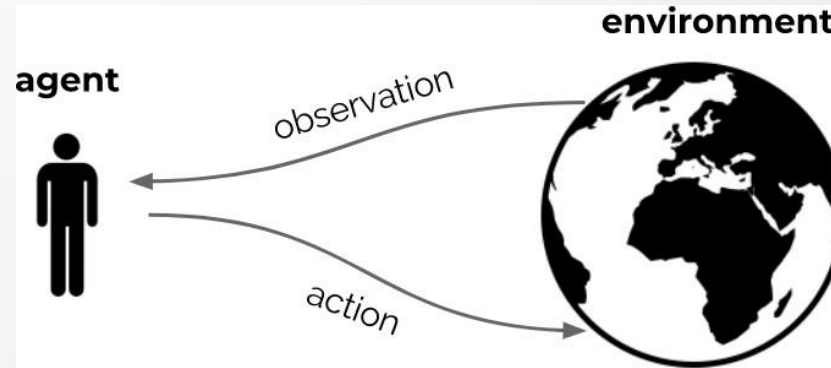
- Understand Markov Decision Processes
- Understand Episodic Tasks and Continuing Tasks
- Define and explain Deterministic Policy and Stochastics Policy

Introduction

- The k-Armed Bandit problem does not account for the fact that *different situations call for different actions*.
- A bandit rabbit would only be concerned about immediate reward and so it would go for the carrot. But a better decision can be made by considering the long-term impact of our decisions.
- Now, let's look at how the situation changes as the rabbit takes actions.
- We will call these situations *states*. In each state the rabbits selects an *action*.

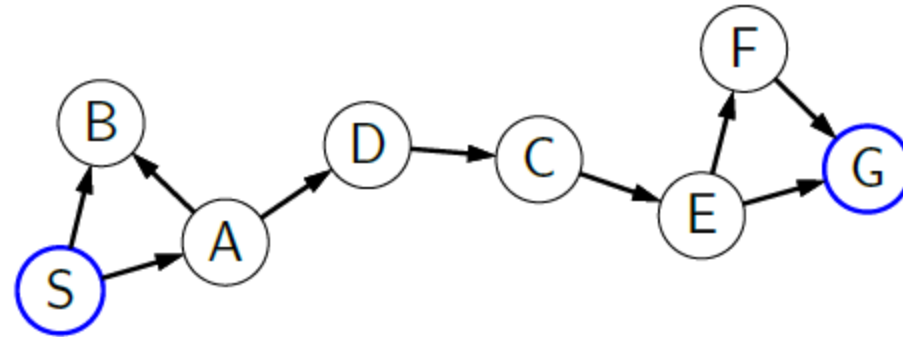


Markov Decision Processes



- We will discuss a mathematical formulation of the agent-environment interaction
- This is called a **Markov Decision Process (MDP)**
- Enables us to talk clearly about the **objective** and **how to achieve it**

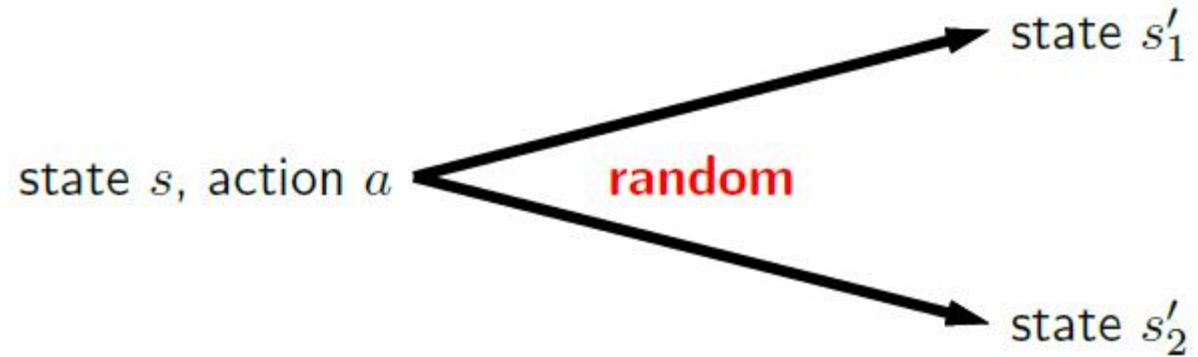
So far: search problems



state s , action a deterministic $\longrightarrow$ state $\text{Succ}(s, a)$



Uncertainty in the real world



Applications



Robotics: decide where to move, but actuators can fail, hit unseen obstacles, etc.



Resource allocation: decide what to produce, don't know the customer demand for various products



Agriculture: decide what to plant, but don't know weather and thus crop yield

Dice game



Example: dice game

For each round $r = 1, 2, \dots$

- You choose **stay** or **quit**.
- If **quit**, you get \$10 and we end the game.
- If **stay**, you get \$4 and then I roll a 6-sided dice.
 - If the dice results in 1 or 2, we end the game.
 - Otherwise, continue to the next round.

Start

Stay

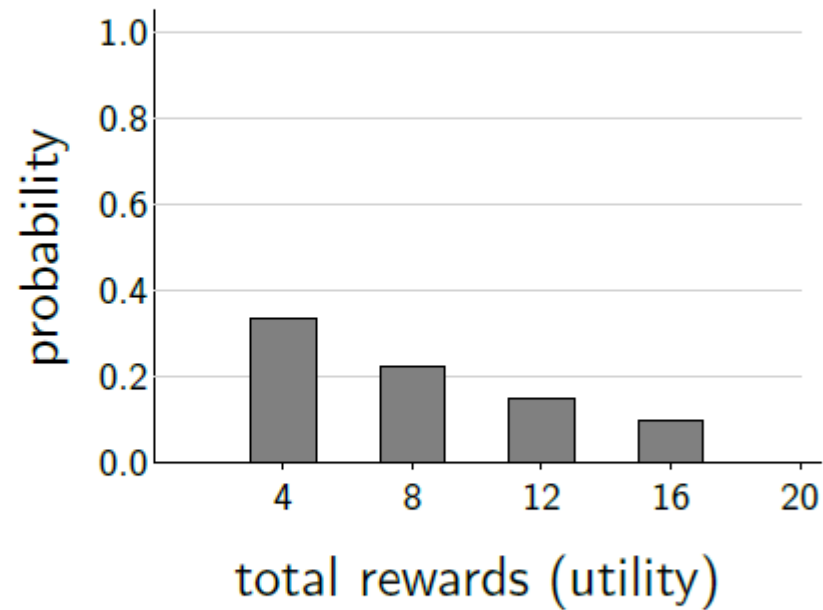
Quit

Dice:

Rewards:

Rewards

If follow policy "stay":

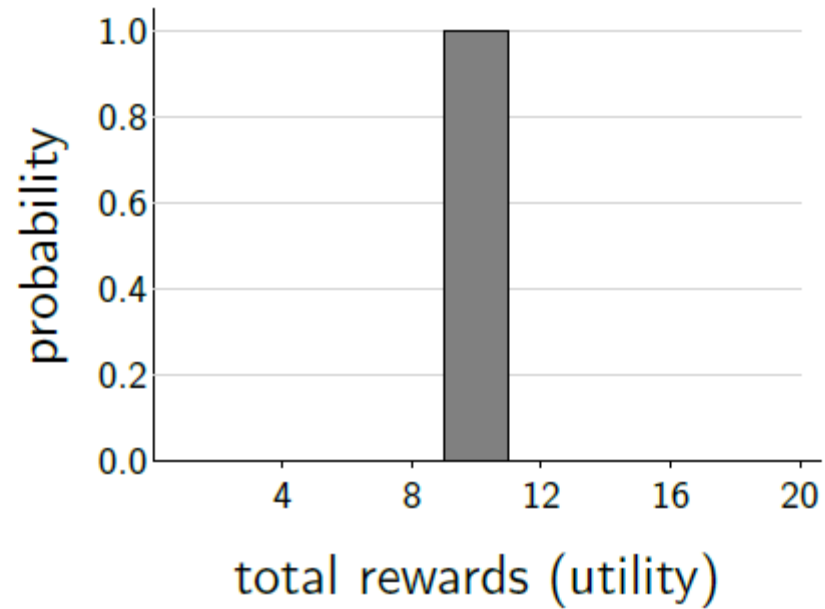


Expected utility:

$$\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \dots = 12$$

Rewards

If follow policy "quit":



Expected utility:

$$1(10) = 10$$

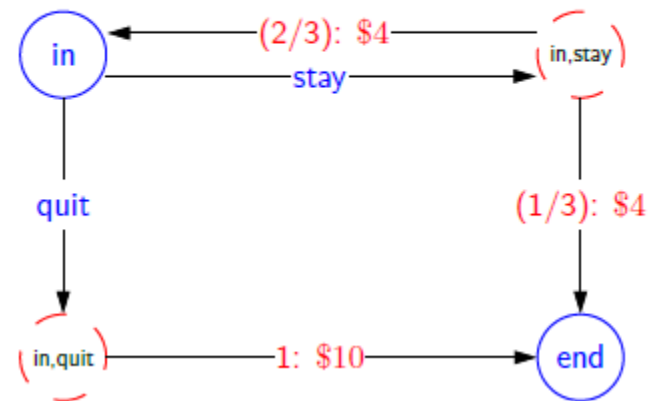
MDP for dice game



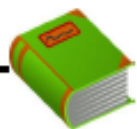
Example: dice game

For each round $r = 1, 2, \dots$

- You choose **stay** or **quit**.
- If **quit**, you get \$10 and we end the game.
- If **stay**, you get \$4 and then I roll a 6-sided dice.
 - If the dice results in 1 or 2, we end the game.
 - Otherwise, continue to the next round.



Markov decision process



Definition: Markov decision process

States: the set of states

$s_{\text{start}} \in \text{States}$: starting state

$\text{Actions}(s)$: possible actions from state s

$T(s, a, s')$: probability of s' if take action a in state s

$\text{Reward}(s, a, s')$: reward for the transition (s, a, s')

$\text{IsEnd}(s)$: whether at end of game

$0 \leq \gamma \leq 1$: discount factor (default: 1)

MDP: Definition

Definition (Markov Decision Process - *Sutton & Barto 2018*)

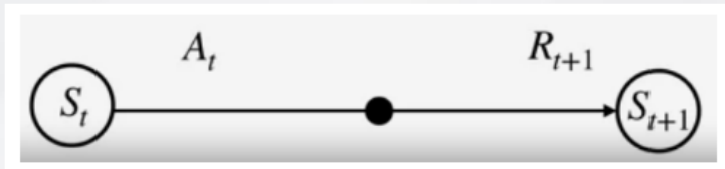
A **Markov Decision Process** is a tuple $(\mathcal{S}, \mathcal{A}, p, \gamma)$, where

- ▶ $\mathcal{S}$ is the set of all possible states
- ▶ $\mathcal{A}$ is the set of all possible actions (e.g., motor controls)
- ▶ $p(r, s' \mid s, a)$ is the joint probability of a reward r and next state s' , given a state s and action a
- ▶ $\gamma \in [0, 1]$ is a discount factor that trades off later rewards to earlier ones

- For now, assume the environment is **fully observable**:
⇒ the current **observation** contains all relevant information

Dynamics Function

- From a set of State-Action the agent receive a Reward and ended in state S_{t+1}
- The transition *dynamics function* p , formalizes this notion.
- Given a state S and action a , p tells us the joint probability of next state S' and reward R .
- We will typically assume that the set of states, actions, and rewards are finite.
- Since p is a probability distribution, it must be non-negative and it's sum over all possible next states and rewards must equal one.

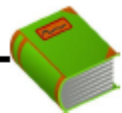


$$p(s', r | s, a)$$

$$p : S \times R \times S \times A \rightarrow [0, 1]$$

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

Transitions



Definition: transition probabilities

The **transition probabilities** $T(s, a, s')$ specify the probability of ending up in state s' if taken action a in state s .



Example: transition probabilities

s	a	s'	$T(s, a, s')$
in	quit	end	1
:	:	:	:
in	stay	in	$2/3$
in	stay	end	$1/3$

Probabilities sum to one



Example: transition probabilities

s	a	s'	$T(s, a, s')$
in	quit	end	1
⋮	⋮	⋮	⋮
in	stay	in	$2/3$
in	stay	end	$1/3$

For each state s and action a :

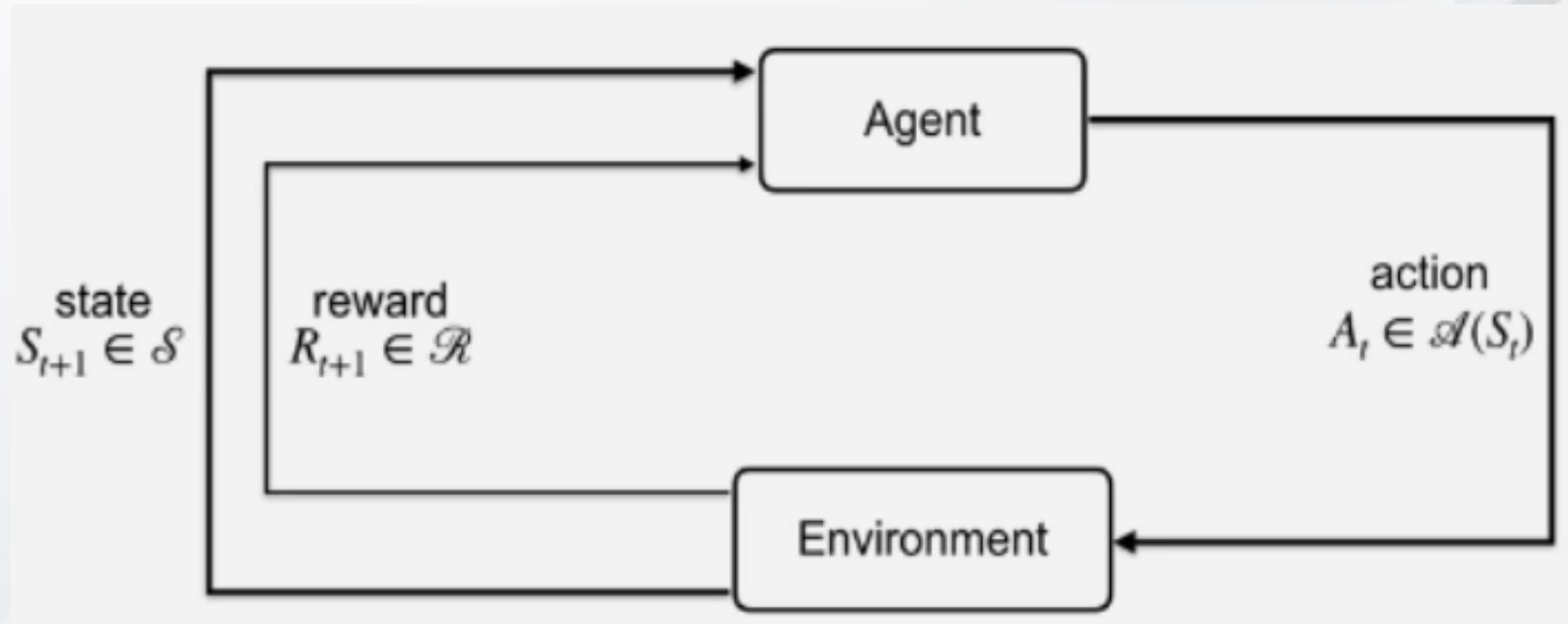
$$\sum_{s' \in \text{States}} T(s, a, s') = 1$$

Successors: s' such that $T(s, a, s') > 0$

Markov Property

- Note that future state and reward only depends on the current state and action.
- This is called the *Markov property*.
- It means that *the present state is sufficient* and remembering earlier states would not improve predictions about the future.

Markov Property



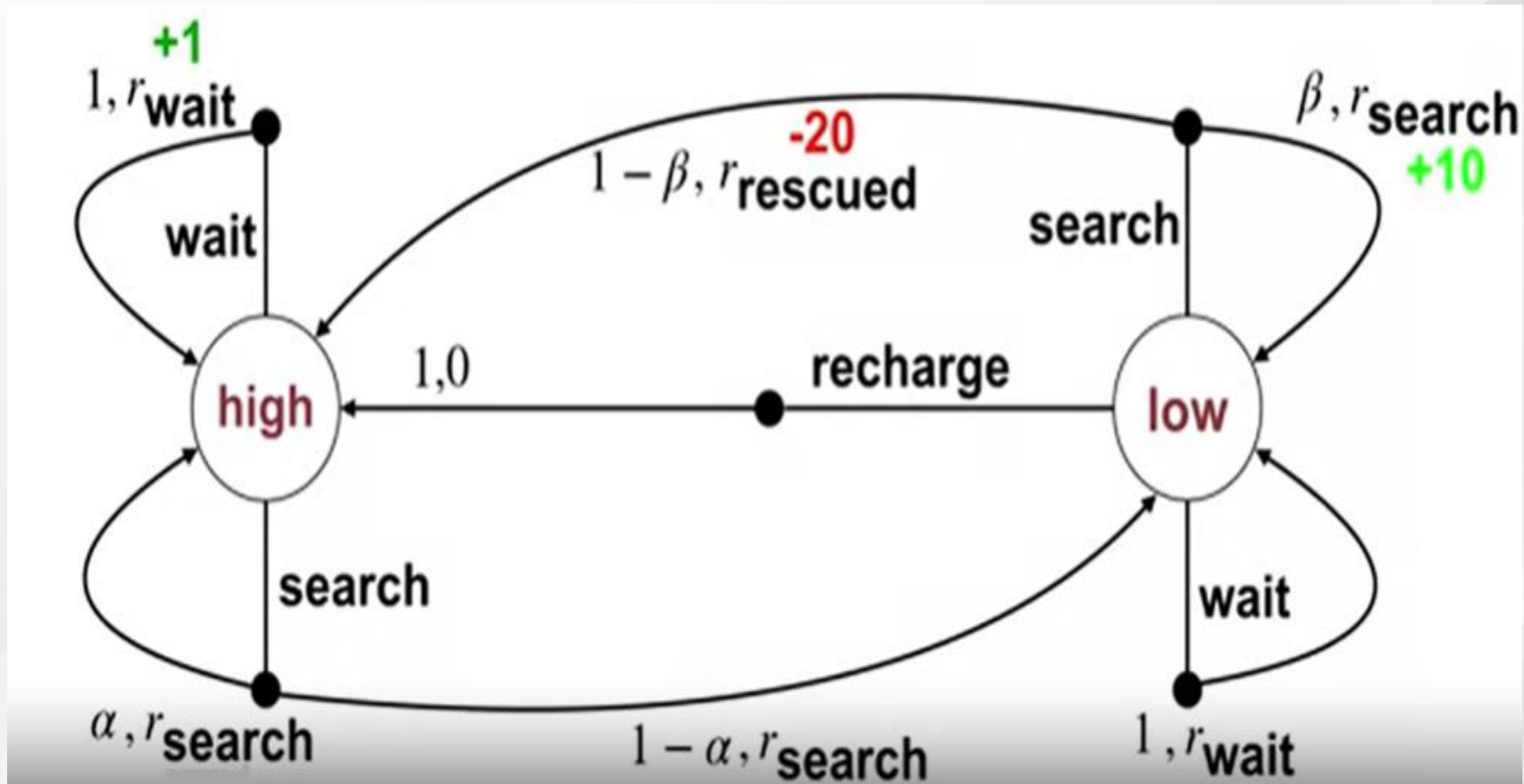
Example of MDP: Cleaning Robot

- Consider a robot that cleans soda cans
- Two states: *high* battery charge or *low* battery charge
- Actions: {wait, search} in high, {wait, search, recharge} in low
- Dynamics may be stochastic
 - $p(S_{t+1} = \text{high} \mid S_t = \text{high}, A_t = \text{search}) = \alpha$
 - $p(S_{t+1} = \text{low} \mid S_t = \text{high}, A_t = \text{search}) = 1 - \alpha$
- Reward could be expected number of collected cans (stochastic), or actual number of collected cans (deterministic)

Example: robot MDP

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

Example: robot MDP

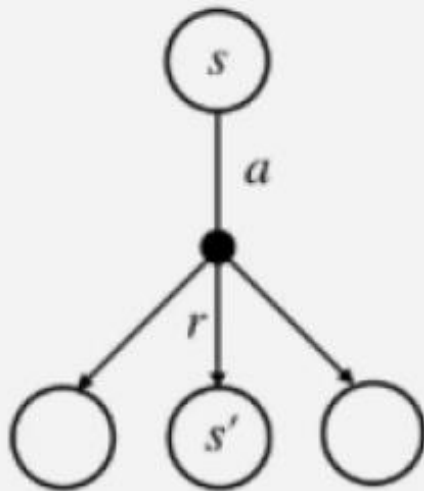


The Goal of Reinforcement Learning

- In reinforcement learning, the agent's *objective* is to *maximize future reward*.
- What looked good in the short-term, might not be the best in the long-term.
- The return at time step t , is simply *the sum of rewards* obtained *after time step t* .

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

- The *return G* is a random variable because the dynamics of the MDP can be stochastic.



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

We need to
maximize it

Finite

Final time
step reward

Episodic Tasks

- In the simplest case, the interaction naturally breaks into chunks 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*.
- We call these tasks *episodic tasks*.

Episodic Tasks

Example : Episodic Task



Every game of chess
has a final move

Episodic Tasks

episode



Continuing Tasks

- There is a problem when implementing the Return representation of episodic tasks into continuing tasks.

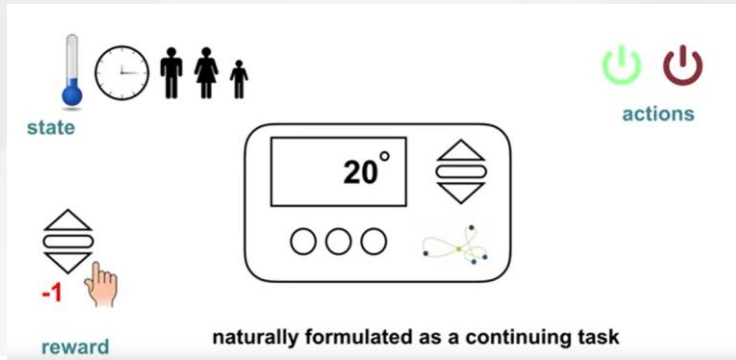
- **Episodic Tasks**

- Interaction breaks naturally into episodes
- Each episode ends in a terminal state
- Episodes are independent

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

- **Continuing Tasks**

- Interaction goes on actually
- No terminal state



$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \dots = \infty?$$

Discounting

- The return formulation can then be modified to include *discounting*.
- The effect of discounting on the return is simple, *immediate rewards contribute more to the sum*.
- Rewards far into the future contribute less because they are multiplied by Gamma raised to successively larger powers of k.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{k-1} R_{t+k} + \dots$$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The last expression will be finite as long as $0 \leq \gamma < 1$. We have to guarantee the discounting with γ .

Why continuing tasks G_t is finite?

- Assume R_{max} is the maximum reward the agent can receive

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \sum_{k=0}^{\infty} \gamma^k R_{max} = R_{max} \sum_{k=0}^{\infty} \gamma^k \rightarrow \text{An upper bound of } G_t$$

Converging $\sum_{k=0}^{\infty} \gamma^k$ geometric series when $\gamma < 1$

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

The expression $R_{max} \times \frac{1}{1-\gamma}$ is finite and then G_t is finite.

Effect of γ on agent behavior

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{k-1} R_{t+k} + \dots$$

with $\gamma = 0$

$$= R_{t+1} + 0R_{t+2} + 0^2 R_{t+3} + \dots + 0^{k-1} R_{t+k} + \dots$$

$$= R_{t+1}$$

- With $\gamma = 0$, it is evident that the agent only cares about the immediate reward. This is called a *Short-sighted agent*.
- With $\gamma \rightarrow 1$, agent takes future rewards into account more strongly. This is called a *Far-sighted agent*.

Recursive nature of returns

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

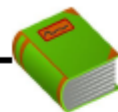
$$G_t = R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots)$$

Recall that $(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots)$ is G_{t+1}

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

Remember that by definition G_{t+1} is equal to zero in the final episode

Discounting



Definition: utility

Path: $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \dots$ (action, reward, new state).

The **utility** with discount γ is

$$u_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots$$

Discount $\gamma = 1$ (save for the future):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + 4 + 4 + 4 = 16$$

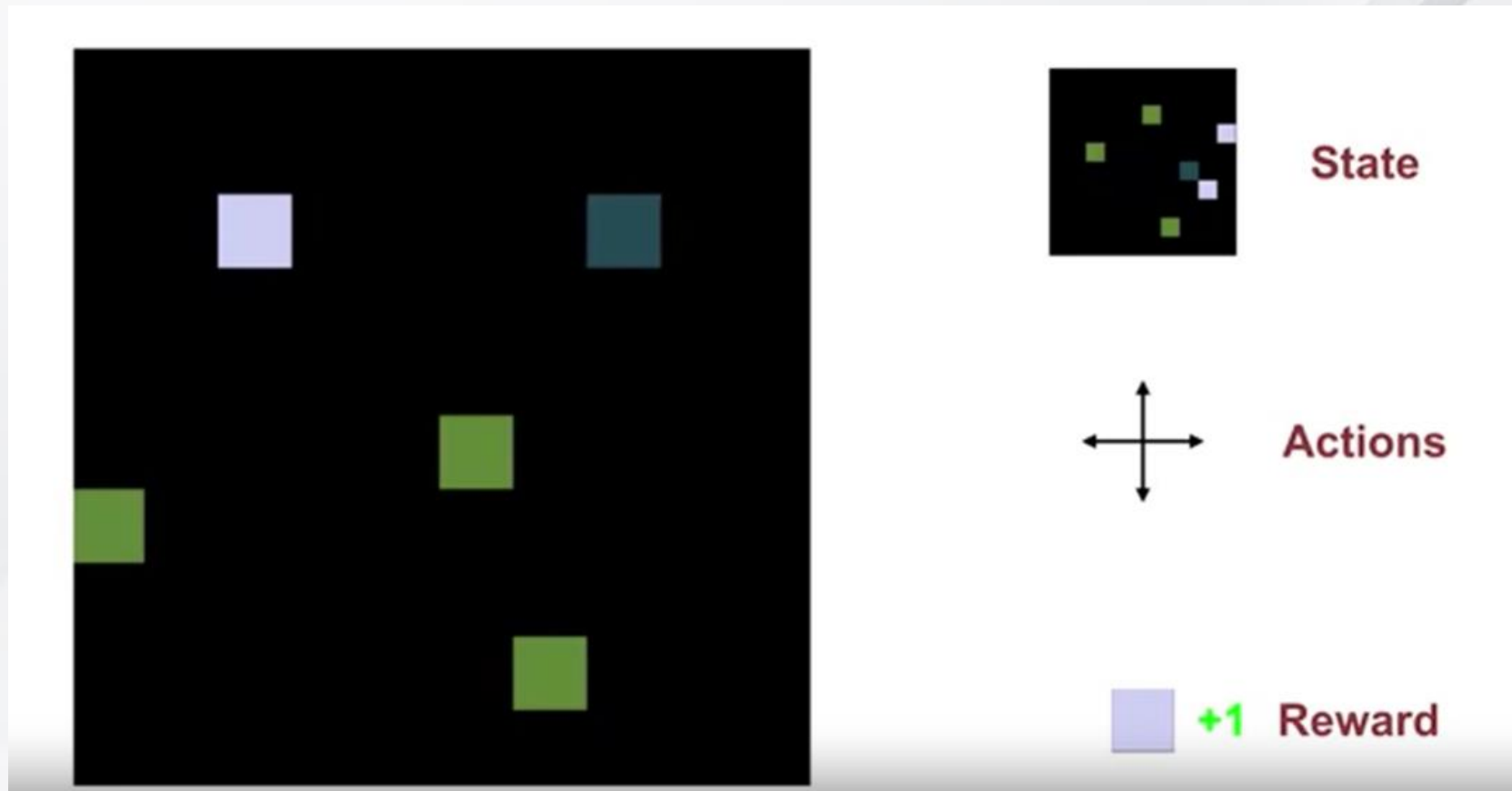
Discount $\gamma = 0$ (live in the moment):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + 0 \cdot (4 + \dots) = 4$$

Discount $\gamma = 0.5$ (balanced life):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + \frac{1}{2} \cdot 4 + \frac{1}{4} \cdot 4 + \frac{1}{8} \cdot 4 = 7.5$$

Example of Episodic Task



Example of Continuing Task (Scheduler)

Example Scheduler

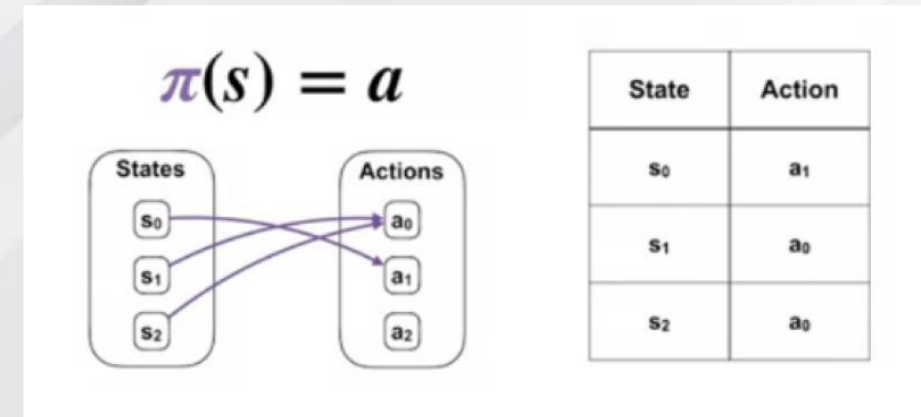


States: { Priority,
Free Servers }

Actions: { Accept, Reject }



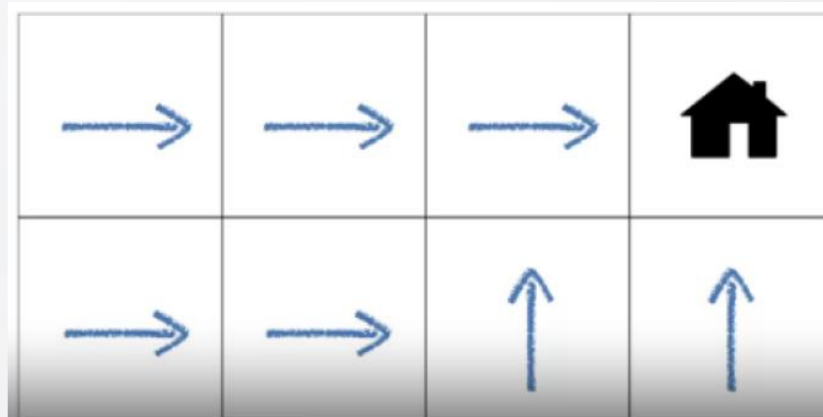
- Policies are how an agent selects actions.
- In the simplest case, a policy maps each state to a single action. This kind of policy is called the *deterministic policy*.
- We will use the fancy Greek letter π to denote a policy.
- $\pi(S)$ represents the action selected in state S by the policy π .
- In this example, π selects the action A_1 in state S_0 and action A_0 in states S_1 and S_2 . We can visualize a deterministic policy with a table.



- Notice that the agent can select the same action in multiple states, and some actions might not be selected in any state

Deterministic Policy

- The arrows in this example describe one possible policy, which moves the agent towards its house.
- Each arrow tells the agent which direction to move in each state.

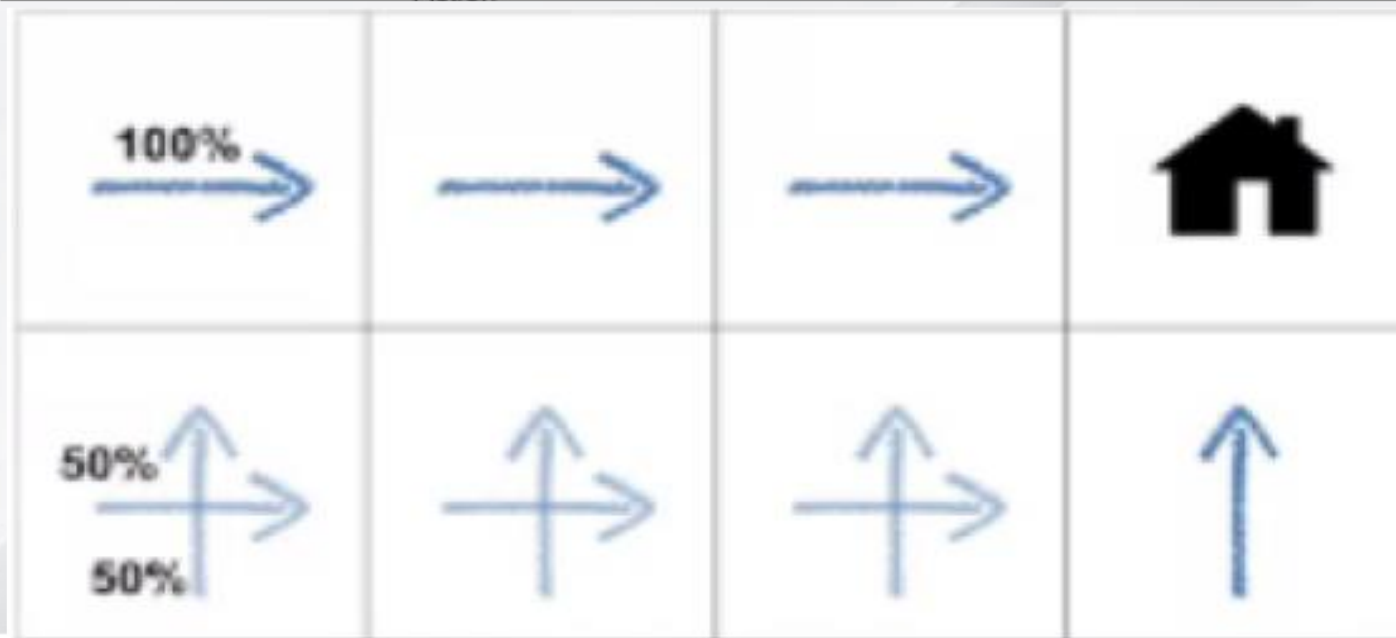
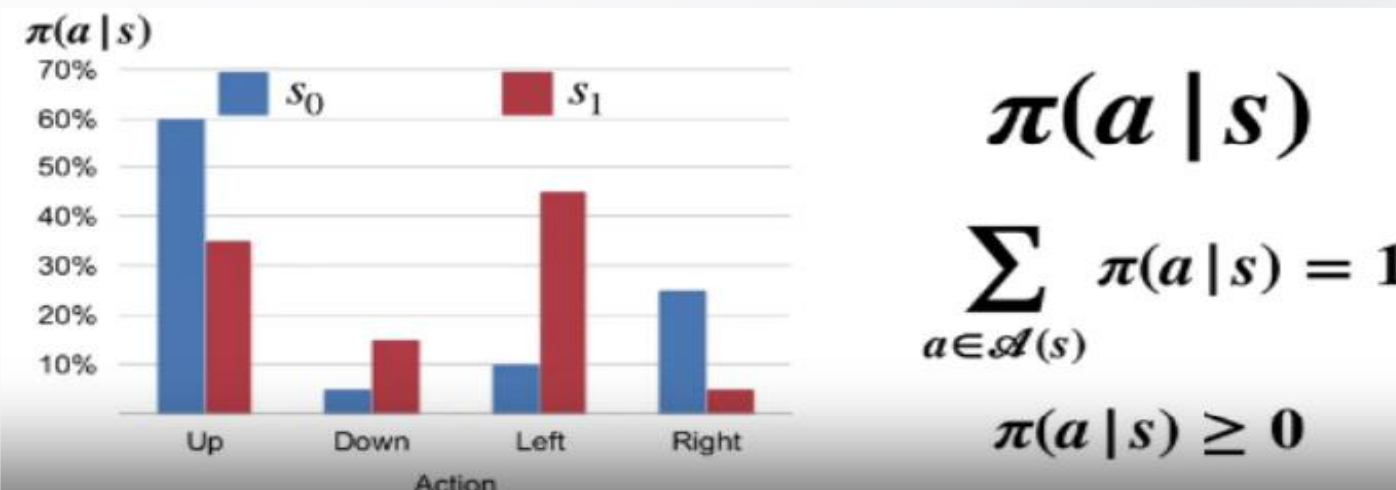


- In general, a policy assigns probabilities to each action in each state.

Stochastic Policy

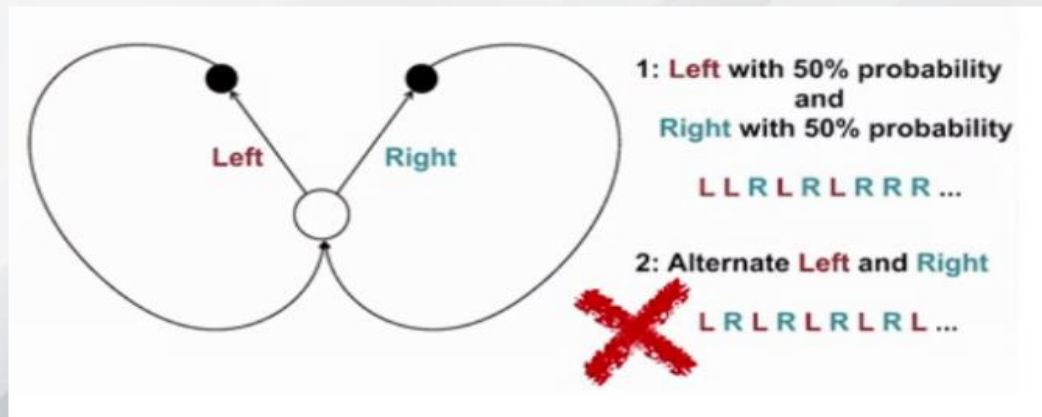
- We use the notation $\pi(a|s)$, to represent the probability of selecting action A in a state S .
- A *stochastic policy* is one where multiple actions may be selected with non-zero probability.

Stochastic Policy



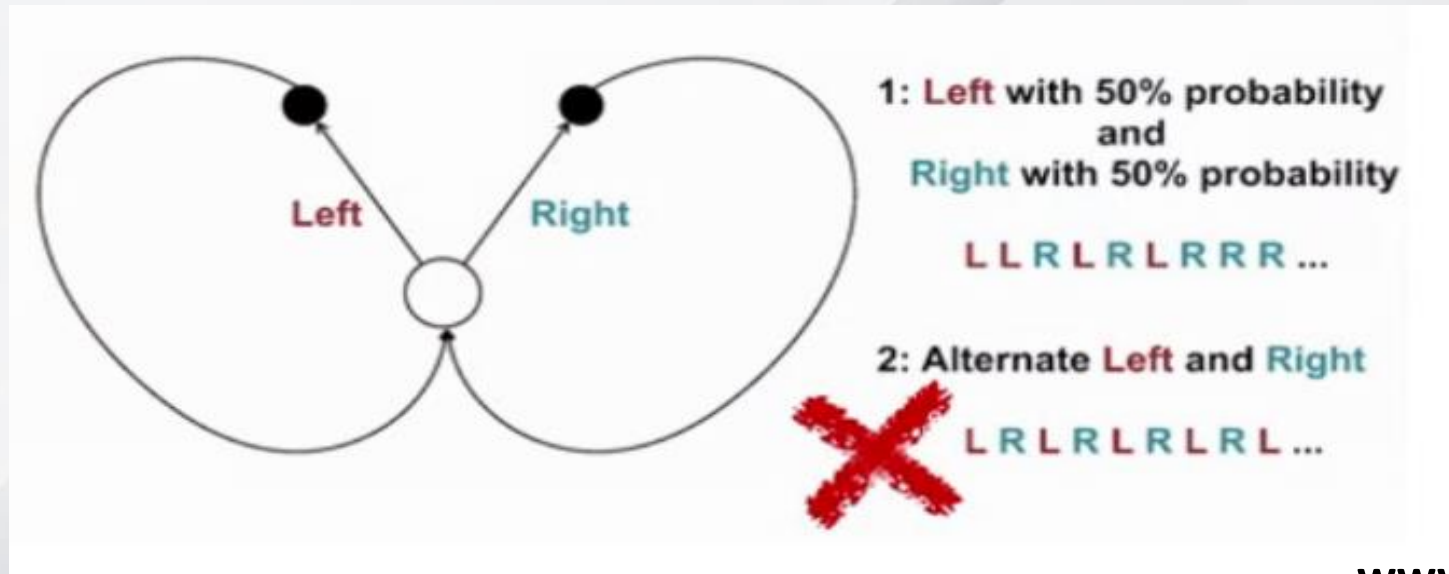
Valid and invalid Policies

- It is important that policies depend only on the current state, not on things like time or previous states.
- The *state* defines all the information used to select the current action.
- In this MDP, we can define a policy that chooses to go either left or right with equal probability.
- We might also want to define a policy that chooses the opposite of what it did last, alternating between left and right actions.
- However, that *would not be a valid policy* because this is conditional on the last action.



Valid and invalid Policies

- In MDPs, we assume that the state includes all the information required for decision-making.
- If alternating between left and right would yield a higher return, then the last action should be included in the state.



Value Functions

- *Value functions* are crucial in reinforcement learning.
- They allow an agent to query the quality of its current situation instead of waiting to observe the long-term outcome.
- The value function summarizes all the possible futures by averaging over *returns*.
- Value function enable us to judge the quality of different policies.
- Value functions predict rewards into the future.
- *State value function* refers to the expected return from a given state under a specific policy.
- *Action value function* refers to the expected return from a given state after selecting a particular action and then following a given policy.

State-Value Function

- The objective of RL is to learn a policy that achieves the most reward in the long run.
- *Value functions* formalize what this means.
- A *state value function* is the future award an agent can expect to receive starting from a particular state.
- The definition of *state-value functions* can be seen in the next equation

$$v(s) \doteq \mathbb{E} [G_t | S_t = s]$$

Recall that $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

- The definition of state-value functions with respect to a policy π is

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi} [G_t | S_t = s]$$

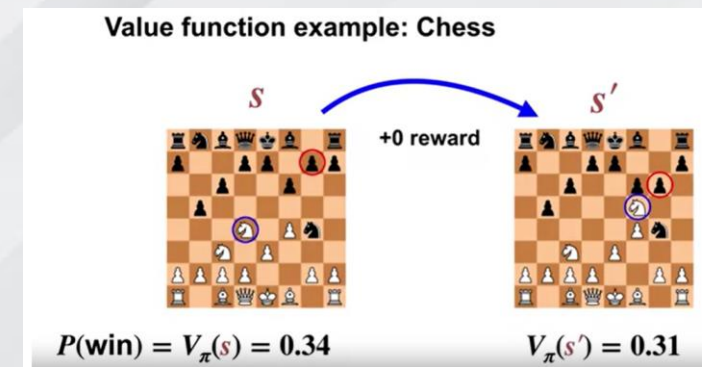
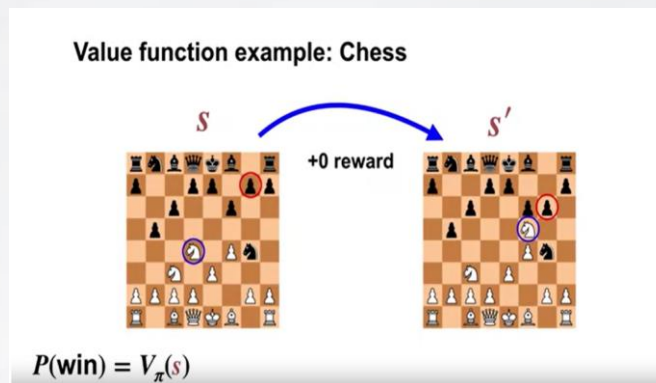
Action-Value Function

- An *action-value function* describes what happens when the agent first selects a particular action.
- For example, in a chess game, it allows us to assess the probability of winning for each possible move (action) given we follow some policy π .
- More formally, the action value of a state is the expected return if the agent selects action A and then follows policy π .

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

Example: Chess

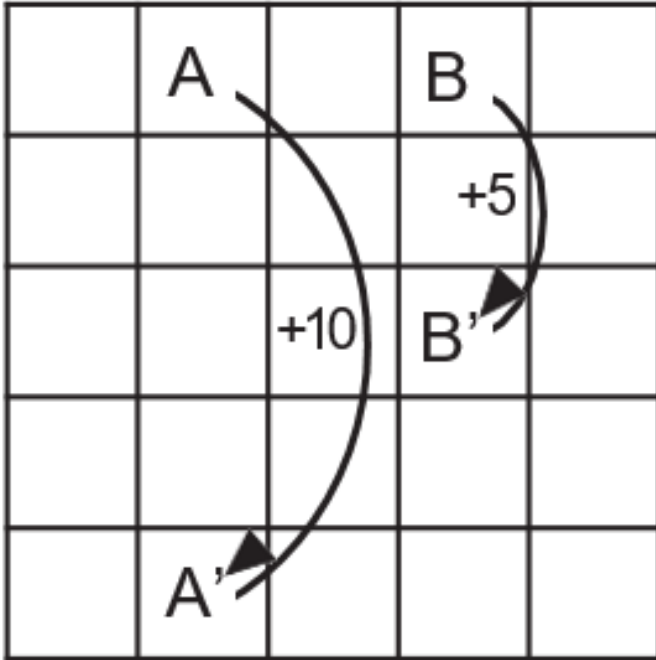
- In a chess game, the *state-value* is equal to the expected sum of future rewards.
- Since the only possible non-zero reward is plus one for winning, the state value is simply the probability of winning if we follow the current policy π .
- In this two player game, the opponent's move is part of the state transition. For example, the environment moves both the agents piece, circled in blue, and the opponent's piece, circled in red.
- This puts the board into a new state, S' .
- Since the value of state S' is lower than the value of state S . This means we are less likely to win the game from this new state assuming we continue following policy π .



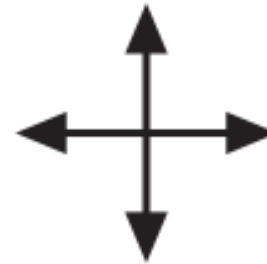
Example: Grid World

- In the Grid World, the states are defined by the locations on the grid, the actions move the agent up, down, left, or right.
- The agent cannot move off the grid and bumping generates a reward of -1. Most other actions yield no reward.
- There are two special states however, these special states are labeled A and B.
- Every action in state A yields plus 10 reward and plus 5 reward in state B.
- Every action in state A and B transitions the agents to states A' and B' respectively.

Example: Grid World



(a)



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

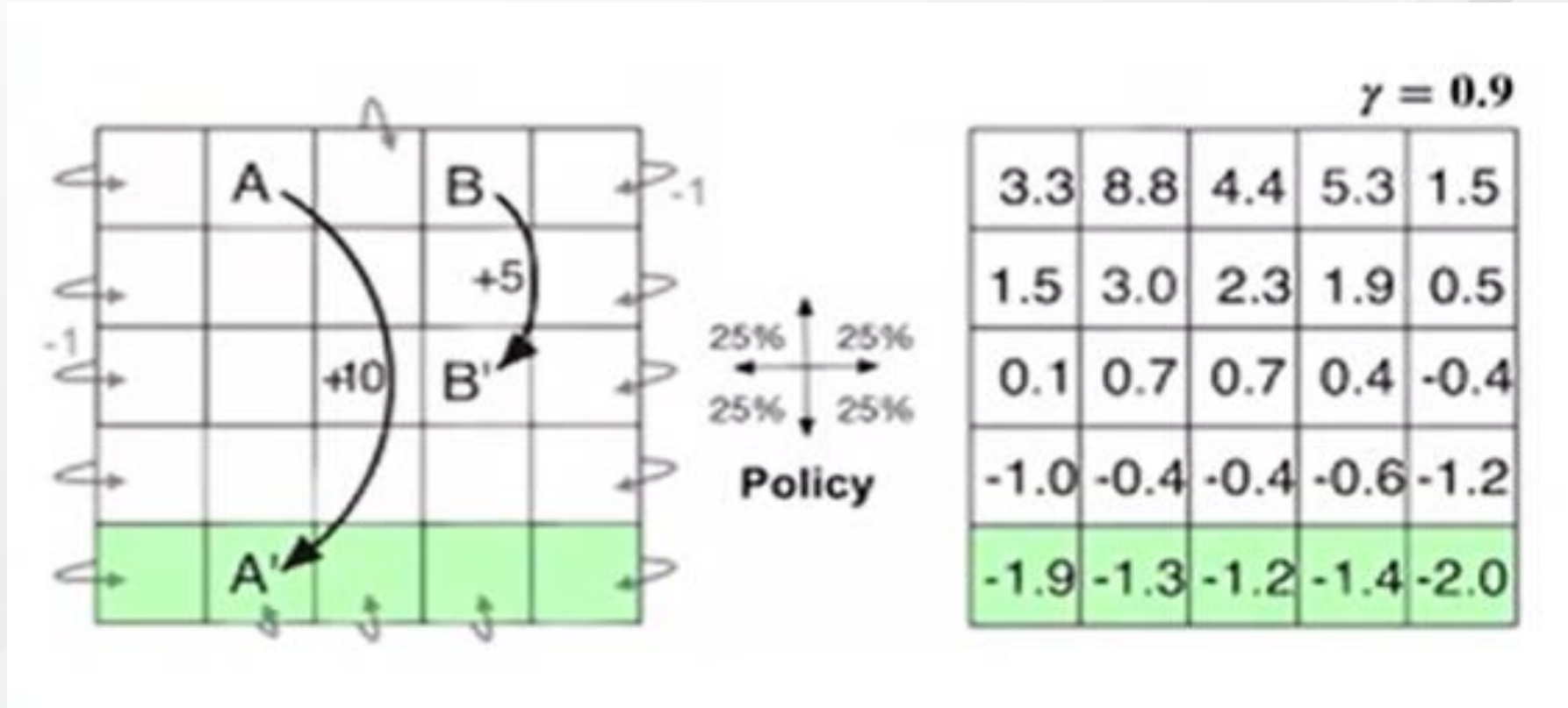
(b)

Reward is -1 when bumping into a wall, $\gamma = 0.9$

Example: Grid World

- First, notice the negative values near the bottom, these values are low because the agent is likely to bump into the wall before reaching the distance states *A* and *B*.
- Remember, *A* and *B* are both the only sources of positive reward in this MDP.
- State *A* has the highest value, notice that the value is less than 10 even though every action from state *A* generates a reward of plus 10, why?
 - *Because every transition from A moves the agent close to the lower wall and near the lower wall, the random policy is likely to bump and get negative reward.*

Example: Grid World



- Now why the value of state B is slightly greater than 5 ?



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY



Thank You

