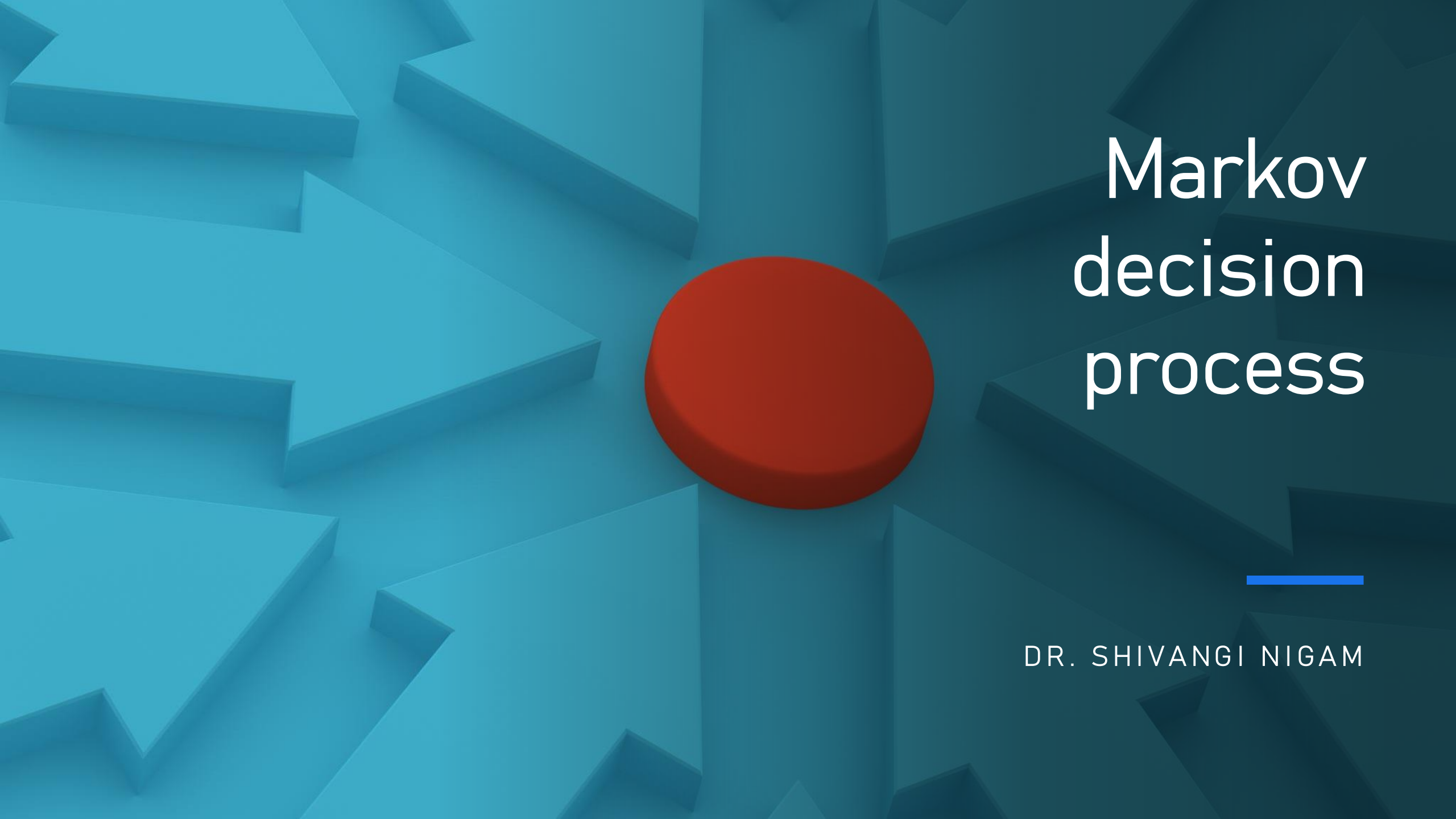




Markov decision process

DR. SHIVANGI NIGAM

How would you get to Mountain View on Friday night in the least amount of time?

bike

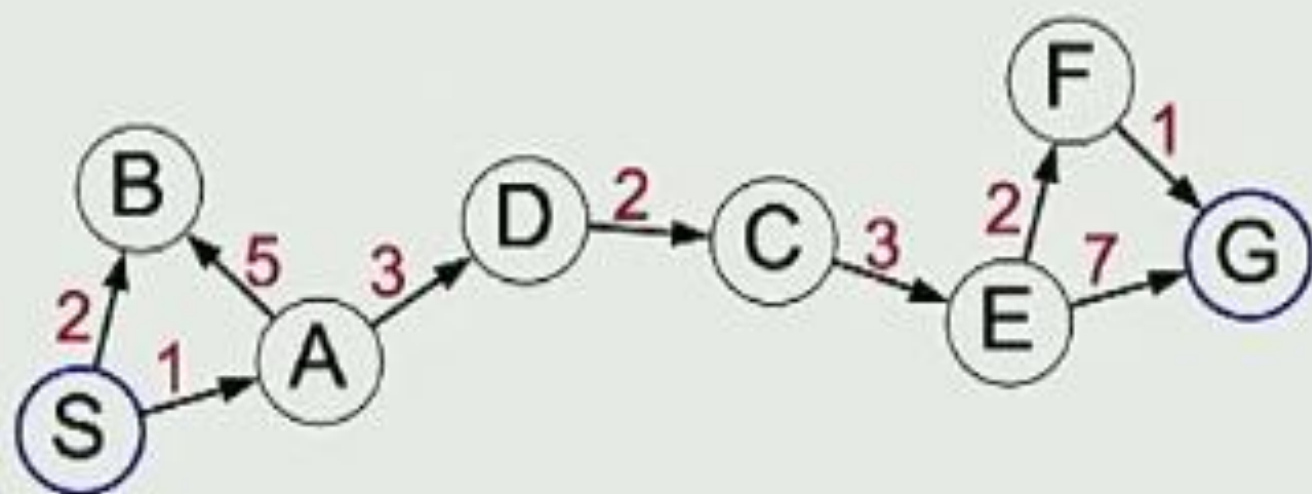
drive

Caltrain

Uber/Lyft

fly

So far: search problems



state s , action a $\xrightarrow{\text{deterministic}}$ 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

Volcano crossing



```
// Model
moveReward = 0 // For every action you take
passReward = 20 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0 // If slip, go in random direction
discount = 1 // How much to value the future

// CHANGE THIS to 10
numIters = 0 // # iterations of value iteration
```

		-50	20
		-50	
2			

Run (or press ctrl-enter)

Volcano crossing



```
// Model
moveReward = 0 // For every action you take
passReward = 20 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0.1 // If slip, go in random direction
discount = 1 // How much to value the future

// CHANGE THIS to 10
numIters = 10 // # iterations of value iteration
```

13.4 ↓	12.3 ↓	-50	20
13.7 →	14.1 ↓	-50	18.2 ↑
2 ↑	15.9 →	16.3 →	18.1 ↑

Value: 13.68

Run (or press ctrl-enter)



Roadmap

Markov decision process

Policy evaluation

Value iteration

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.

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:

0

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:

10

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: 6

Rewards: 4

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: 2

Rewards: 8

Rewards

If follow policy "stay":



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:

$$1(10) = 10$$

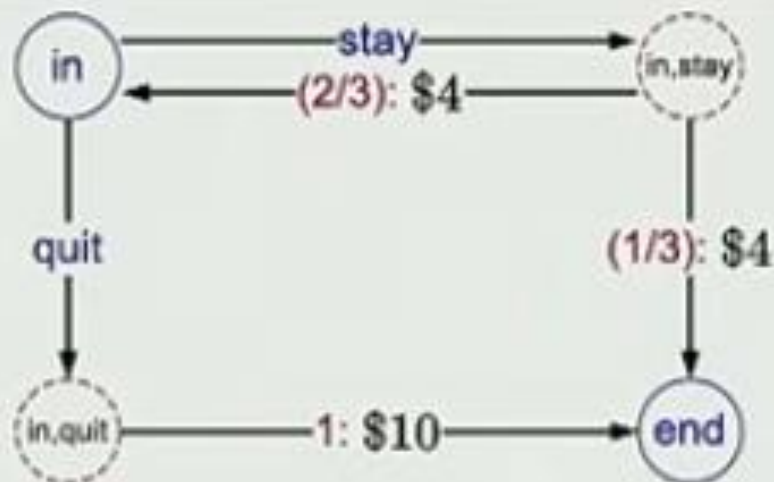
MDP for dice game



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)

Search problems



Definition: search problem

States: the set of states

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

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

$\text{Succ}(s, a)$: where we end up if take action a in state s

$\text{Cost}(s, a)$: cost for taking action a in state s

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

Search problems



Definition: search problem

States: the set of states

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

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

$\text{Succ}(s, a)$: where we end up if take action a in state s

$\text{Cost}(s, a)$: cost for taking action a in state s

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

- $\text{Succ}(s, a) \Rightarrow T(s, a, s')$
- $\text{Cost}(s, a) \Rightarrow \text{Reward}(s, a, s')$

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 $T(s, a, s') > 0$



Transportation example



Example: transportation

Street with blocks numbered 1 to n .

Walking from s to $s + 1$ takes 1 minute.

Taking a magic tram from s to $2s$ takes 2 minutes.

How to travel from 1 to n in the least time?



Transportation example



Example: transportation

Street with blocks numbered 1 to n .

Walking from s to $s + 1$ takes 1 minute.

Taking a magic tram from s to $2s$ takes 2 minutes.

How to travel from 1 to n in the least time?

Tram fails with probability 0.5.

What is a solution?

Search problem: path (sequence of actions)

MDP:



Definition: policy

A **policy** π is a mapping from each state $s \in \text{States}$ to an action $a \in \text{Actions}(s)$.

What is a solution?

Search problem: path (sequence of actions)

MDP:



Definition: policy

A **policy** π is a mapping from each state $s \in \text{States}$ to an action $a \in \text{Actions}(s)$.



Example: volcano crossing

s	$\pi(s)$
(1,1)	S
(2,1)	E
(3,1)	N
...	...



Roadmap

Markov decision process

Similar to a search problem, it's just probabilistic/stochastic

Policy evaluation

How good a policy would be?

Value iteration

Evaluating a policy



Definition: utility

Following a policy yields a **random path**.

The **utility** of a policy is the (discounted) sum of the rewards on the path (this is a random quantity).

Evaluating a policy



Definition: utility

Following a policy yields a **random path**.

The **utility** of a policy is the (discounted) sum of the rewards on the path (this is a random quantity).

	Path	Utility	
Random paths	[in; stay, 4, end]	4	Random variable
	[in; stay, 4, in; stay, 4, in; stay, 4, end]	12	
	[in; stay, 4, in; stay, 4, end]	8	
	[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]	16	
	...	...	



Definition: value (expected utility)

The **value** of a policy is the **expected utility**.

Not
random
anymore

We do value
iteration here
to see where
we lead to

Value: 12

Evaluating a policy: volcano crossing

```
// Model
moveReward = -0.1 // For every action you take
passReward = 40 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0.3 // If slip, go in random direction
discount = 0.9 // How much to value the future

// Run algorithms on model
numIters = 10 // # iterations of value iteration
numEpisodes = 1 // # simulations
```

Run (or press ctrl-enter)

2.4 ↓	-0.5 ↓	-50	40	$a \quad r \quad s$ (2,1)
3.7 →	5 ↓	-50	31 ↑	E -0.1 (2,2)
2	12.6 →	16.3 →	26.2 ↑	S -0.1 (3,2)
				E -0.1 (3,3)
				E -50.1 (2,3)

Value: 3.73
Utility: -36.79

Value is the expected utility

Utility value changes but the Value remains around same

Evaluating a policy: volcano crossing

```
// Model
moveReward = -0.1 // For every action you take
passReward = 40 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0.3 // If slip, go in random direction
discount = 0.9 // How much to value the future

// Run algorithms on model
numIters = 10 // # iterations of value iteration
numEpisodes = 1 // # simulations
```

Run (or press ctrl-enter)

2.4	-0.5	-50	40
3.7	5	-50	31
2	12.6	16.3	26.2

α r s

E -0.1 (2,1)

S -0.1 (2,2)

E -0.1 (3,2)

E -0.1 (3,3)

E -0.1 (3,4)

N -0.1 (2,4)

N 39.9 (1,4)

Value: 3.73

Utility: 23.15

Value is the expected utility

Utility value changes but the Value remains around same

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):

[stay, stay, stay, stay]: $4 + 4 + 4 + 4 = 16$

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):

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

Policy evaluation



Definition: value of a policy

Let $V_{\pi}(s)$ be the expected utility received by following policy π from state s .

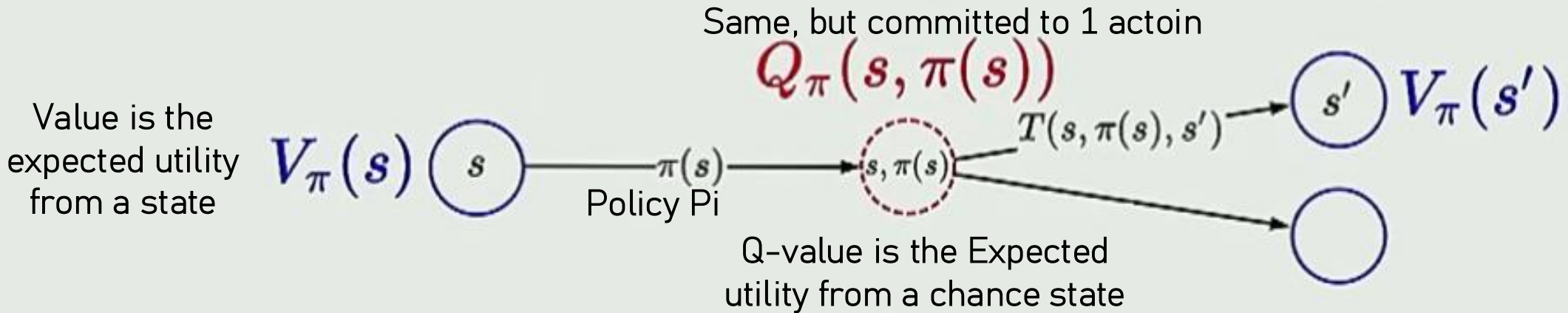


Definition: Q-value of a policy

Let $Q_{\pi}(s, a)$ be the expected utility of taking action a from state s , and then following policy π .

Policy evaluation

Plan: define recurrences relating value and Q-value



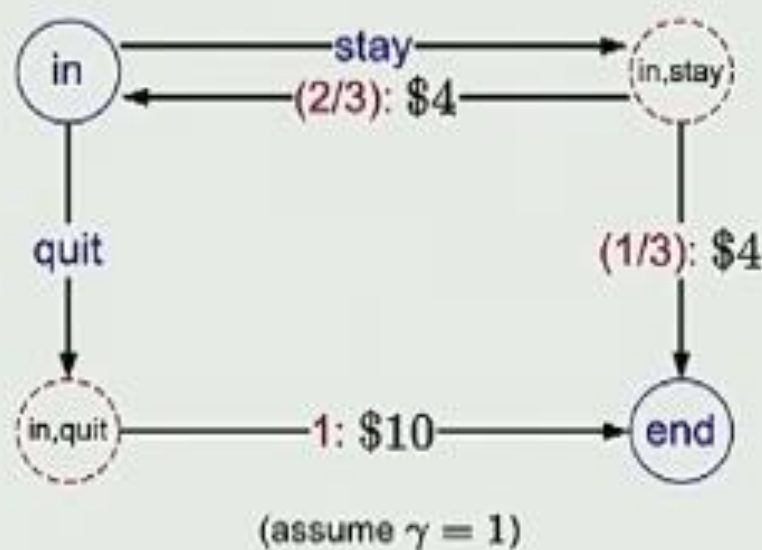
$$V_{\pi}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_{\pi}(s, \pi(s)) & \text{otherwise.} \end{cases}$$

$$Q_{\pi}(s, a) = \sum_{s'} T(s, a, s') [\text{Reward}(s, a, s') + \gamma V_{\pi}(s')]$$

Transition
Rewards
Discounted
Probabilities

Rewards

Dice game

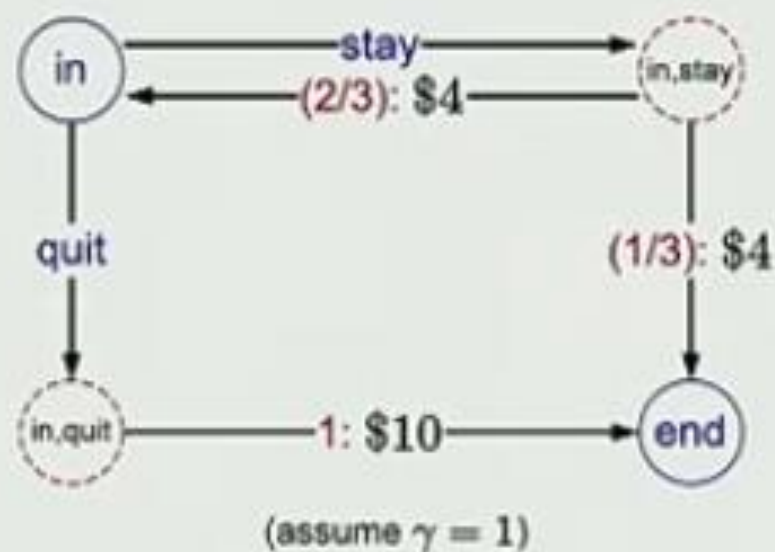


Let π be the "stay" policy: $\pi(\text{in}) = \text{stay}$.

$$V_{\pi}(\text{end}) = 0$$

$$V_{\pi}(\text{in}) = Q_{\pi}(\text{in}, \text{stay})$$

Dice game

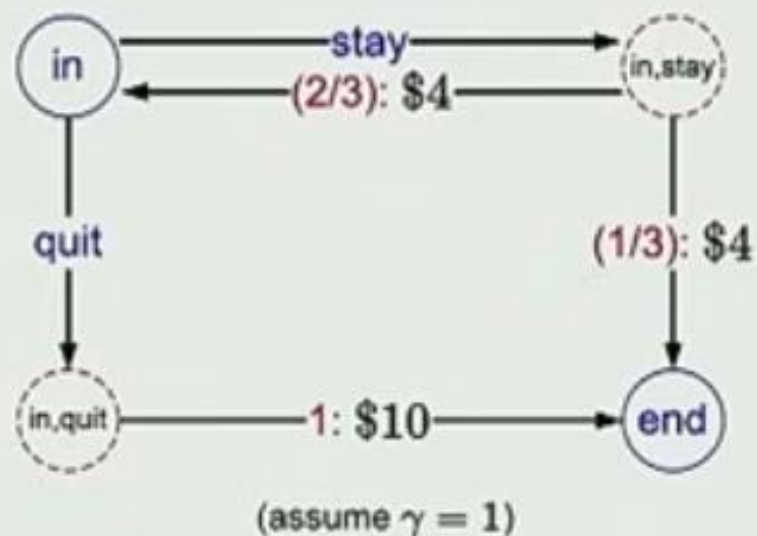


Let π be the "stay" policy: $\pi(\text{in}) = \text{stay}$.

$$V_{\pi}(\text{end}) = 0$$

$$V_{\pi}(\text{in}) = \frac{1}{3} (4 + V_{\pi}(\text{end})) + \frac{2}{3} (4 + V_{\pi}(\text{in}))$$

Dice game



Let π be the "stay" policy: $\pi(\text{in}) = \text{stay}$.

$$V_{\pi}(\text{end}) = 0$$

$$V_{\pi}(\text{in}) = \frac{1}{3} (4 + V_{\pi}(\text{end})) + \frac{2}{3} (4 + V_{\pi}(\text{in}))$$

In this case, can solve in closed form:

$$V_{\pi}(\text{in}) = 12$$

Policy evaluation



Key idea: iterative algorithm

Start with arbitrary policy values and repeatedly apply recurrences to converge to true values.

Policy evaluation



Key idea: iterative algorithm

Start with arbitrary policy values and repeatedly apply recurrences to converge to true values.



Algorithm: policy evaluation

Initialize $V_{\pi}^{(0)}(s) \leftarrow 0$ for all states s .

For iteration $t = 1, \dots, t_{PE}$:

For each state s :

$$V_{\pi}^{(t)}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [\underbrace{\text{Reward}(s, \pi(s), s') + \gamma V_{\pi}^{(t-1)}(s')}_{Q^{(t-1)}(s, \pi(s))}]$$

Policy evaluation implementation

How many iterations (t_{PE})? Repeat until values don't change much:

$$\max_{s \in \text{States}} |V_{\pi}^{(t)}(s) - V_{\pi}^{(t-1)}(s)| \leq \epsilon$$

Complexity



Algorithm: policy evaluation

Initialize $V_{\pi}^{(0)}(s) \leftarrow 0$ for all states s .

For iteration $t = 1, \dots, t_{PE}$:

For each state s :

$$V_{\pi}^{(t)}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [\underbrace{\text{Reward}(s, \pi(s), s') + \gamma V_{\pi}^{(t-1)}(s')}_{Q^{(t-1)}(s, \pi(s))}]$$

MDP complexity

S states

A actions per state

S' successors (number of s' with $T(s, a, s') > 0$)

It doesn't depend on actions as the policy is already given and we just need to follow that

Time: $O(t_{PE} S S')$



Summary so far

- **MDP**: graph with states, chance nodes, transition probabilities, rewards
- **Policy**: mapping from state to action (solution to MDP)
- **Value of policy**: expected utility over random paths
- **Policy evaluation**: iterative algorithm to compute value of policy

