



# **Markov decision processes**



Devika Subramanian

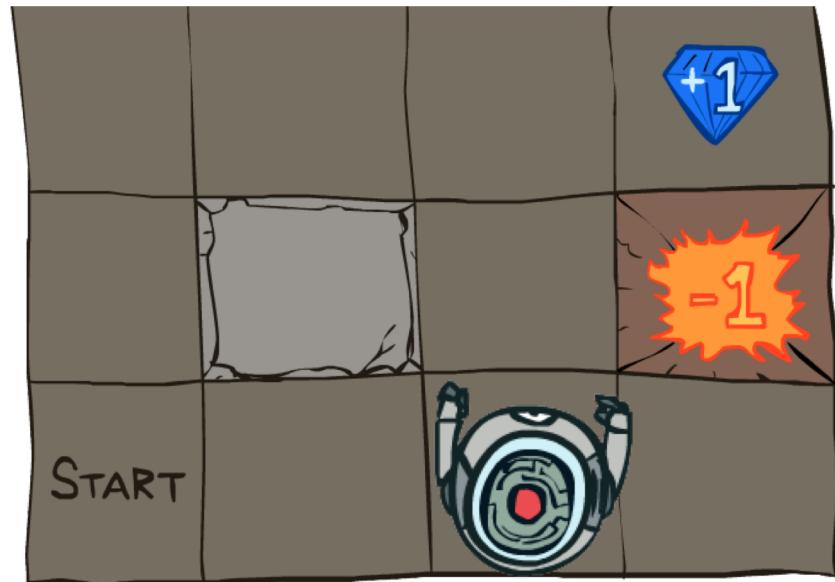
# Outline

---

- ▶ **Modeling stochastic worlds**
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes
- ▶ **A simple game**
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ **The value function and the optimal policy**
  - ▶ V, Q and Bellman's equations
- ▶ **Algorithms for solving MDPs**
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

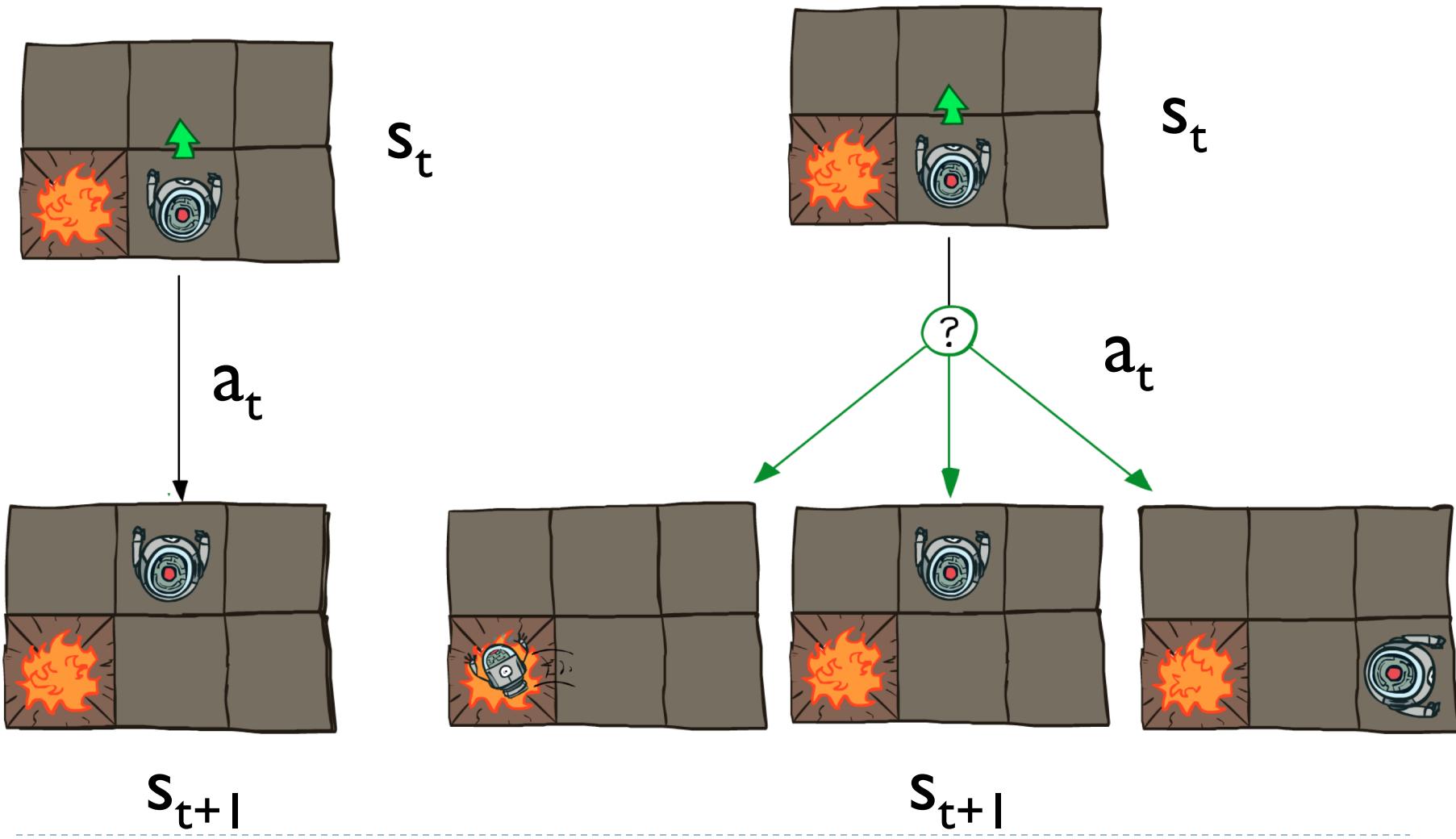
# Example: grid world (stochastic)

- Environment is a grid
  - Bounded by walls
  - Has internal obstacles
  - Has pit of Tartarus
  - Has an Elysian Fields
  - Agent knows where these are.
- Agent effectors are noisy
  - 80% of time agent moves in intended direction
  - 10% of time agent moves -90 degrees from intended direction.
  - 10% of the time agent moves +90 degrees from intended direction
- Agent sensors are perfect



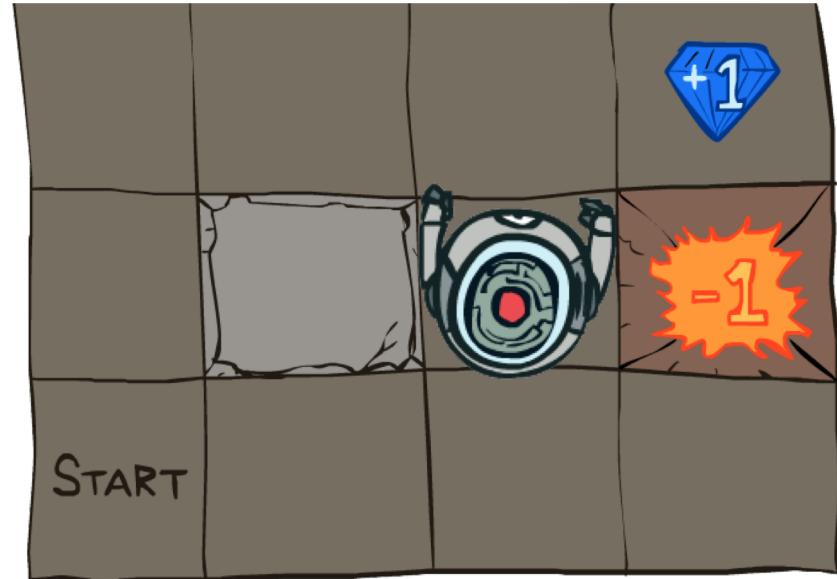
Goal: get to the Elysian field  
as quickly as possible

# Deterministic vs stochastic action models



# Example: grid world

- Environment is a grid
  - Bounded by walls
  - Has internal obstacles
  - Has pit(s) of Tarturus
  - Has an Elysian Fields
  - Agent knows where these are.
- Agent effectors are noisy
  - 80% of time agent moves in intended direction
  - 10% of time agent moves -90 degrees from intended direction.
  - 10% of the time agent moves +90 degrees from intended direction
- Agent sensors are perfect

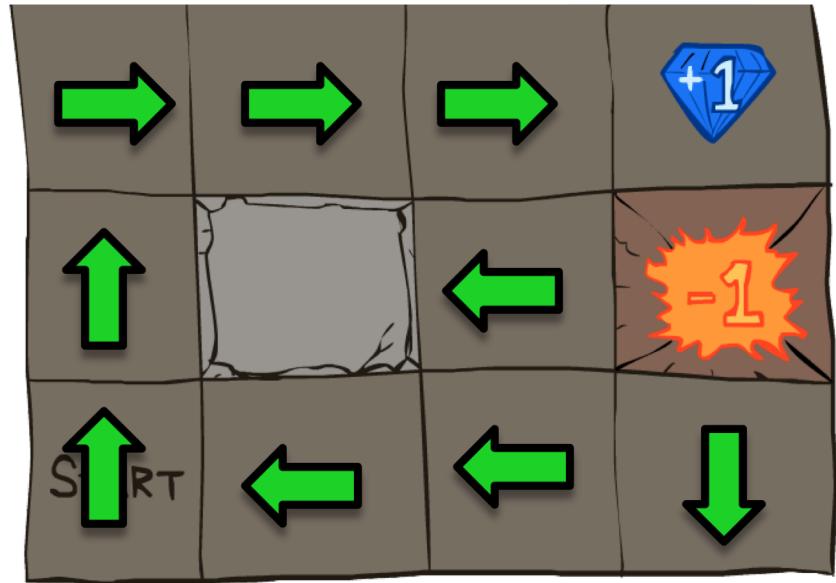


Goal: get to the Elysian field as quickly as possible

**Agent is at (3,2).  
Elysian field is at (4,3).  
What should agent do?**

# Policies versus plans

- ▶ A plan is a fixed length sequence of actions from a start state to a goal state.
  - ▶ In deterministic worlds, an optimal plan takes an agent from a specified start to a specified goal state.
- ▶ A policy is a mapping from states to actions.
  - ▶ In non-deterministic worlds, an agent follows an optimal policy which is defined as one that maximizes expected utility if followed.

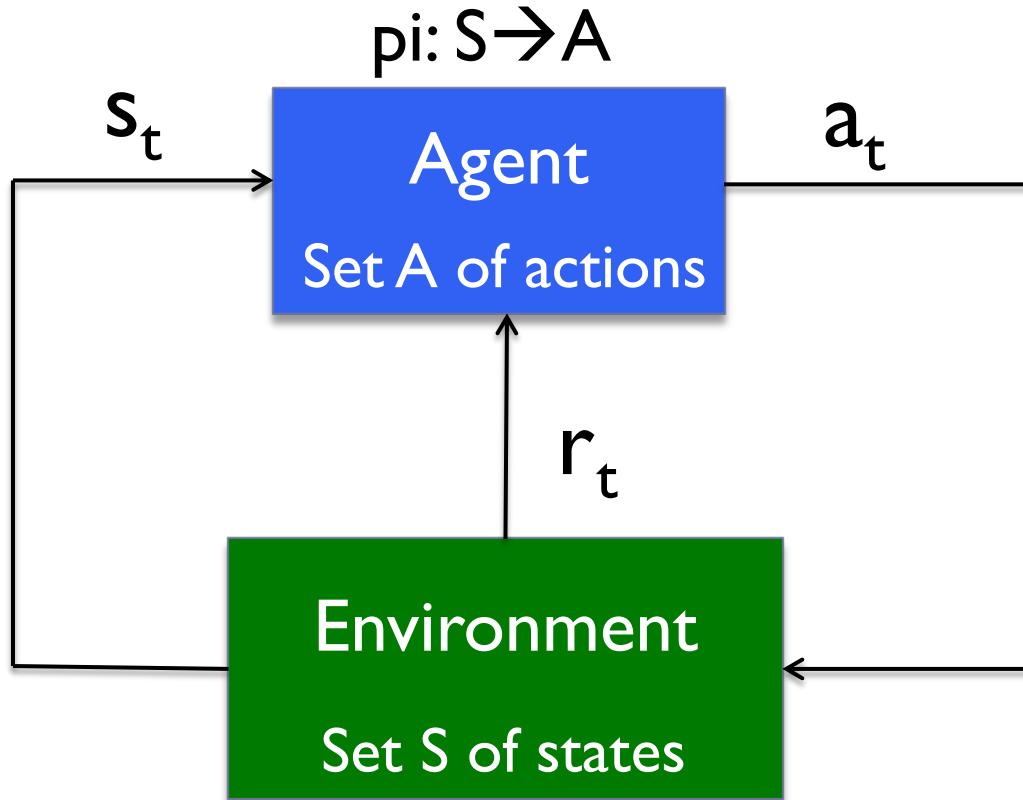


# Markov decision processes (MDP)

---

- ▶ An MDP consists of
  - ▶ A **set of states**  $S$ , a subset of which are **terminal states** ( $\text{isEnd}(s)=\text{True}$ ).
  - ▶ **Actions( $s$ )**: possible actions from state  $s$ 
    - ▶ No actions from terminal states
  - ▶ A **transition function**  $T(s,a,s')$  where  $s,s'$  in  $S$ ,  $a$  in  $\text{Actions}(s)$ 
    - ▶  $T(s,a,s')$  = probability of transitioning to state  $s'$  if action  $a$  is taken in state  $s$
  - ▶ A **reward function**  $r(s,a,s')$  where  $s,s'$  in  $S$ ,  $a$  in  $\text{Actions}(s)$ 
    - ▶  $r(s,a,s')$  = reward for taking action  $a$  in state  $s$  and ending up in state  $s'$
    - ▶ Can be sometimes of the form  $r(s,a)$  or  $r(s)$
- ▶ The grid world problem introduced before is an example of an MDP.
- ▶ What's Markov about an MDP is the state transition model  $T(s,a,s')$

# Markov decision process



$r(s,a,s')$  = reward for transitioning to  $s'$  from  $s$  via  $a$

$T(s,a,s')$  = probability of transitioning to  $s'$  from  $s$  via  $a$

# Transition function for robot navigation

- ▶ The next state is stochastically determined by the current state and the current action.
- ▶  $T(s_{t+1}=(x_2,y_2)|s_t=(x_1,y_1), a_t=a)$ 
  - ▶  $T(s_{t+1}=(1,2)|s_t=(1,1), a_t=N) = 0.8$
  - ▶  $T(s_{t+1}=(1,1)|s_t=(1,1), a_t=N) = 0.1$
  - ▶  $T(s_{t+1}=(2,1)|s_t=(1,1), a_t=N) = 0.1$
- ▶ This state transition model is a **first-order Markov model**, because the robot's position at time  $t+1$  is a function of position at time  $t$  and action at time  $t$ .



# Reward function for robot navigation

---

- ▶ The agent receives rewards at each time step.
  - ▶  $r(s,a,s') = -0.01$ , where  $s,s'$  in  $S$  and  $s'$  is not a terminal state,  $a$  in  $\text{Actions}(s)$ 
    - ▶ Small cost for each step
  - ▶  $r(s,a,(4,3)) = +1$ , where  $s$  in  $S$  and  $a$  in  $\text{Actions}(s)$  and  $T(s,a,(4,3)) > 0$
  - ▶  $r(s,a,(4,2)) = -1$ , where  $s$  in  $S$  and  $a$  in  $\text{Actions}(s)$  and  $T(s,a,(4,2)) > 0$ 
    - ▶ Big reward/cost at the end (Elysian fields or the pits of Tartarus)

# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function and the optimal policy
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# A simple game

---

- ▶ For each round  $r = 1, 2, \dots$ 
  - ▶ You choose to stay or quit.
  - ▶ If you quit, you get \$10 and we end the game.
  - ▶ If you stay, you get \$4 and then I roll a 6-sided dice.
    - ▶ If the dice roll is 1 or 2, we end the game.
    - ▶ Otherwise, continue to the next round.
- ▶ What should you do at each round?
  - ▶ quit or stay?

# Simple game MDP

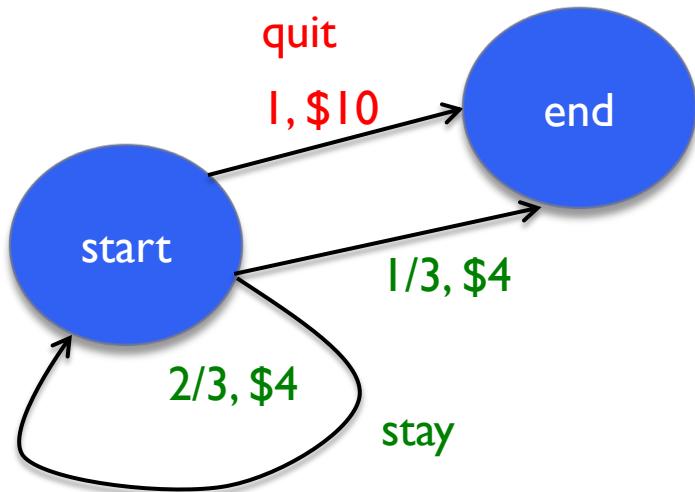
- ▶ States = {start,end}, end is terminal state
- ▶ Actions = {stay, quit}
  - ▶ Actions(start) = {stay,quit}
  - ▶ Actions(end) = {}
- ▶ Transitions  $T(s,a,s')$  and rewards  $r(s,a)$

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

s	a	$r(s,a)$
start	quit	10
start	stay	4

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

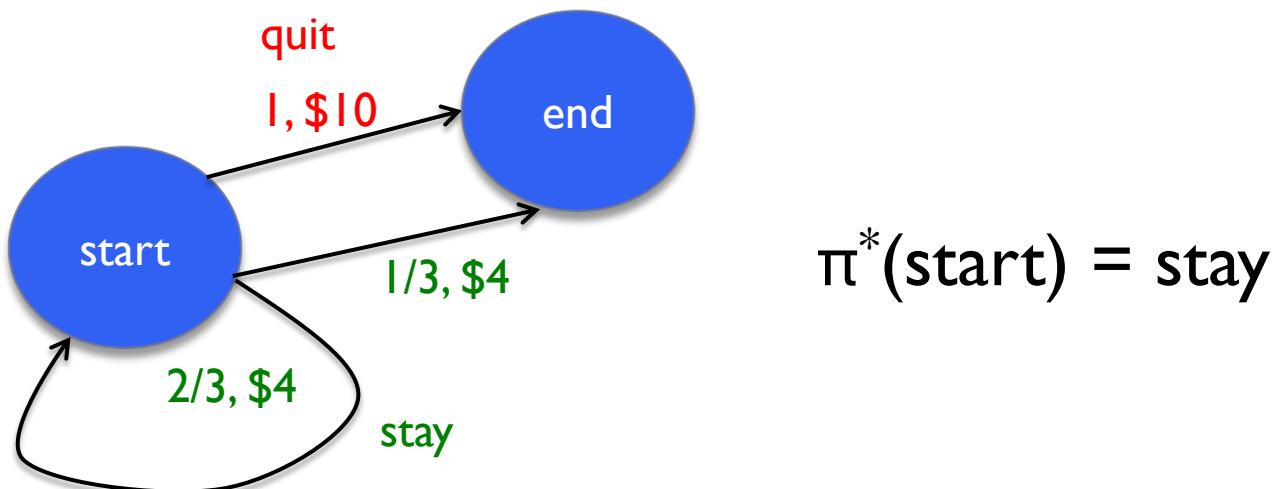
# Maximizing expected utility



- ▶ Maximize expected utility!
- ▶ Calculate expected utility of all available policies, and choose the one with the highest expected utility.
- ▶ Expected utility of “quit”
  - ▶ 10
- ▶ Expected utility of “stay”
  - ▶  $1/3 * 4 + 2/3 * 1/3 * 8 + 2/3 * 2/3 * 1/3 * 12 + \dots = 12$

# Solution for simple game

- ▶ A policy is a mapping from non-terminal states in the MDP to actions.
- ▶ The optimal policy is to choose “stay” in start state.

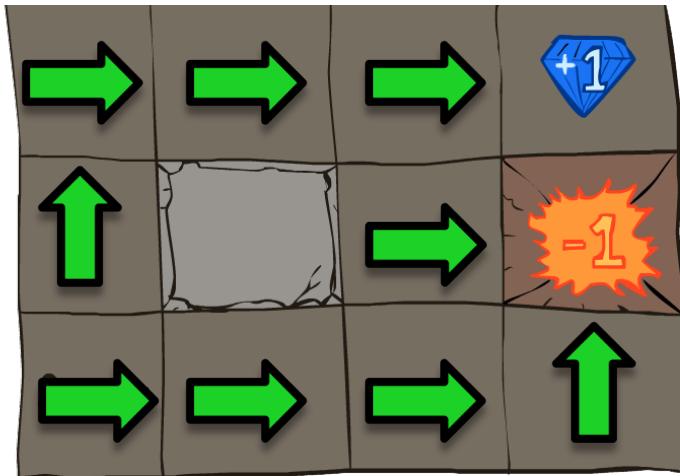


# How to calculate optimal policies

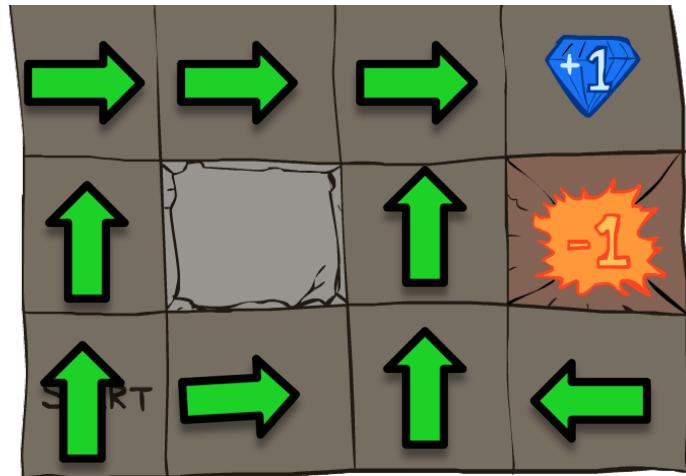
---

- ▶ Model the decision problem as an MDP
  - ▶ Define the states  $S$  and actions  $\text{Actions}(s)$  and identify the terminal states.
  - ▶ Construct the transition function  $T(s,a,s')$ , where  $s,s'$  in  $S$  and  $a$  in  $\text{Actions}(s)$ .
  - ▶ Construct the reward function  $r(s,a,s')$  where  $s,s'$  in  $S$  and  $a$  in  $\text{Actions}(s)$
- ▶ Design an algorithm that solves the MDP, i.e., constructs an optimal policy given a set of states, actions, the transition function and the reward function.

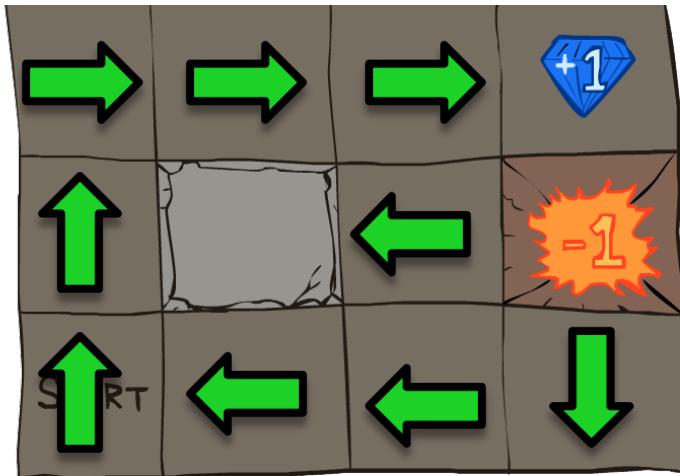
# Optimal policies for robot navigation



$$r(s) < -1.6284$$



$$-0.4278 < r(s) < -0.0850$$



$$-0.0221 < r(s) < 0$$

(c) Devika Subramanian, 2018

4/15/18

$$r(s) > 0$$

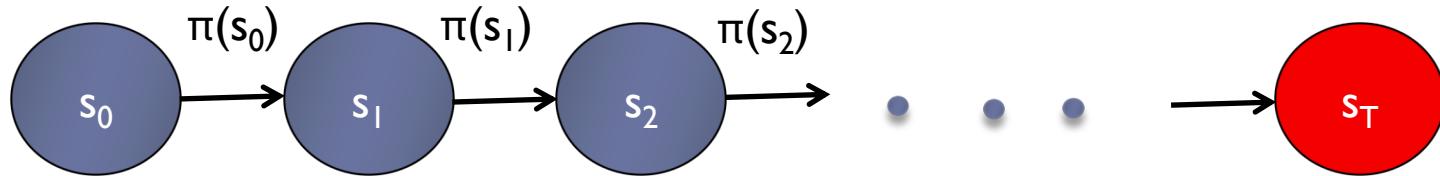
# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes versus classical search
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# Evaluating a policy $\pi$ : $V_\pi$

Following a policy yields a path through the set of states  $S$ , culminating in a terminal state.



$$V_\pi(s_0 s_1 \dots s_T) = \sum_{t=0}^T r(s_t, \pi(s_t), s_{t+1})$$

$r(s, a, s') = -0.01$  in non-terminal states  $s'$

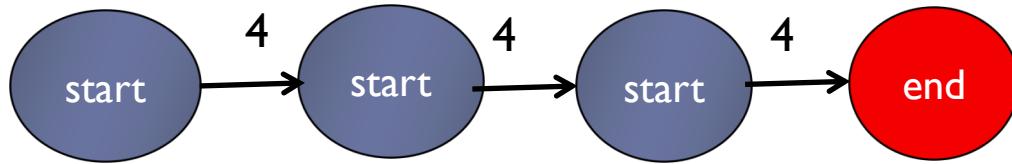
$r(s, a, s') = +1$  if  $s' = (4, 3)$

$r(s, a, s') = -1$  if  $s' = (4, 2)$

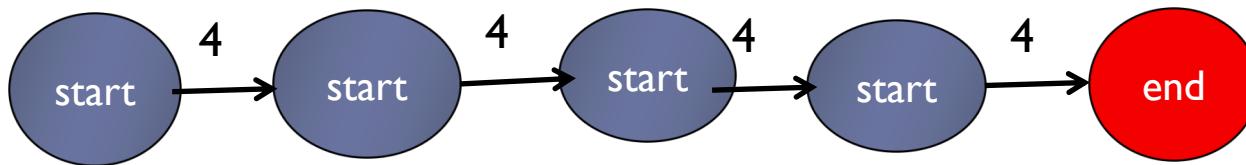
Following a policy yields a state sequence (with some associated probability).

The value of a state sequence is the sum of the rewards over each transition in the state sequence.

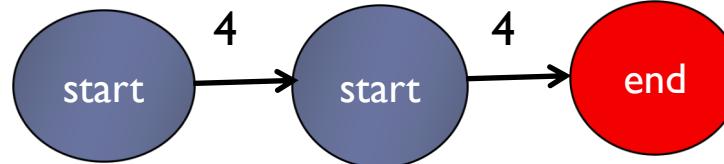
# Evaluating the policy “stay” in the simple game



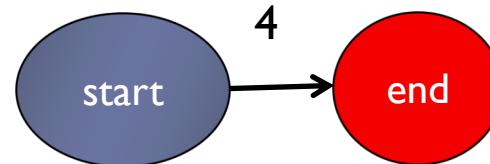
$$V = 12, p = \frac{2}{3} * \frac{2}{3} * \frac{1}{3}$$



$$V = 16, p = \frac{2}{3} * \frac{2}{3} * \frac{2}{3} * \frac{1}{3}$$



$$V = 8, p = \frac{2}{3} * \frac{1}{3}$$



$$V = 4, p = \frac{1}{3}$$

# The value function $V_\pi$

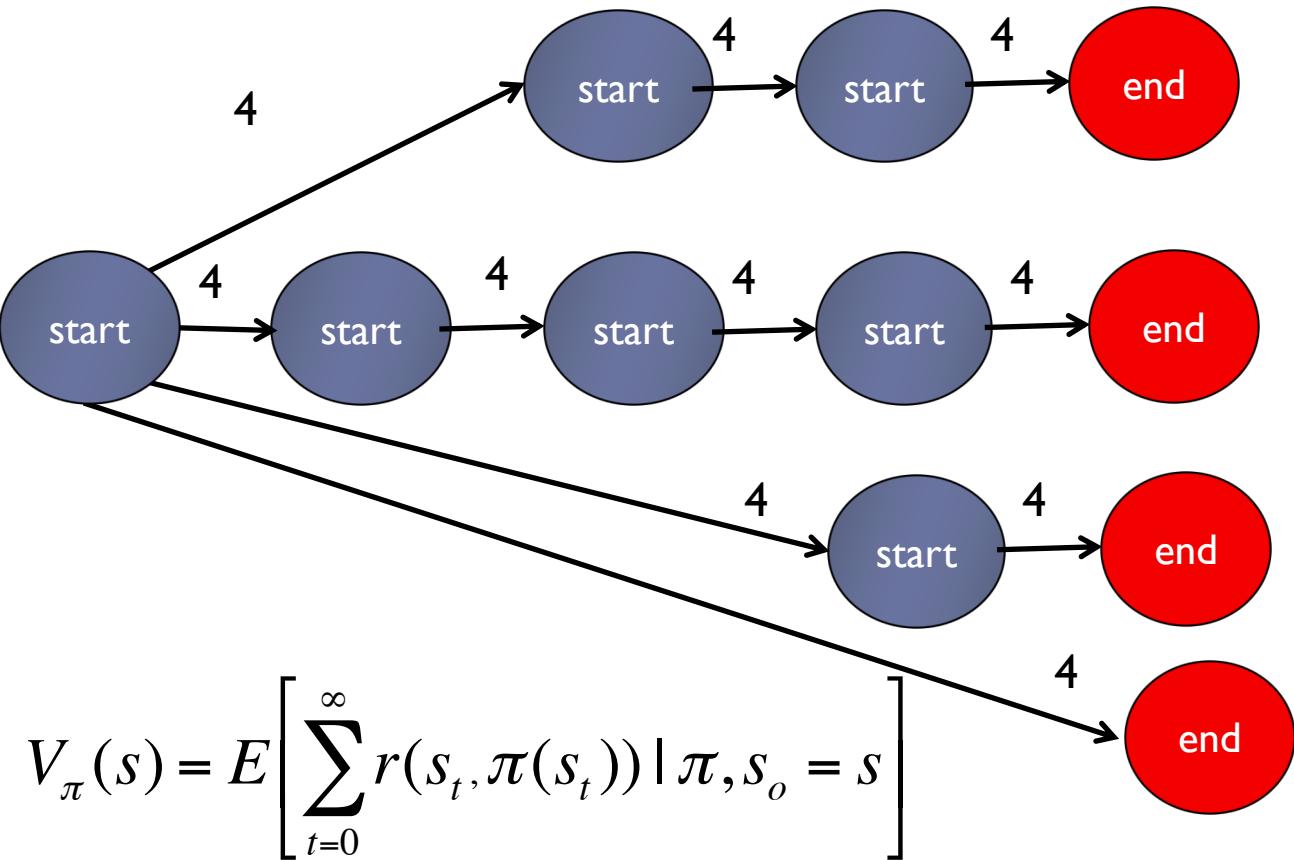
---

- ▶ Expected utility of a state  $s$  under policy  $\pi$

$$V_\pi(s) = E \left[ \sum_{t=0}^{\infty} r(s_t, \pi(s_t)) \mid \pi, s_o = s \right]$$

- ▶ Expectation is calculated over all sequences of states generated by the policy, that start with state  $s$ . It is the average sum of rewards along every path in the state space starting from  $s$ , generated by the policy, weighted by the probability of that path.

# The value function $V_{\pi}(s)$ for the simple game



# Recursive computation of $V_\pi$

- The value of a state  $s$  under the policy  $\pi$  is the expected reward associated with taking action  $\pi(s)$  plus the utility of following policy  $\pi$  from a successor state  $s'$ .

$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s')[r(s, \pi(s), s') + V_\pi(s')]$$

```
graph TD; s((s)) -- "Vπ(s)" --> s1((s')) -- "T(s, π(s), s')" --> s1((s')) -- "Vπ(s')" --> s2((s')) -- "Vπ(s')" --> s3((s')) -- "Vπ(s')"/>
```

# The Q value of a policy $\pi$

---

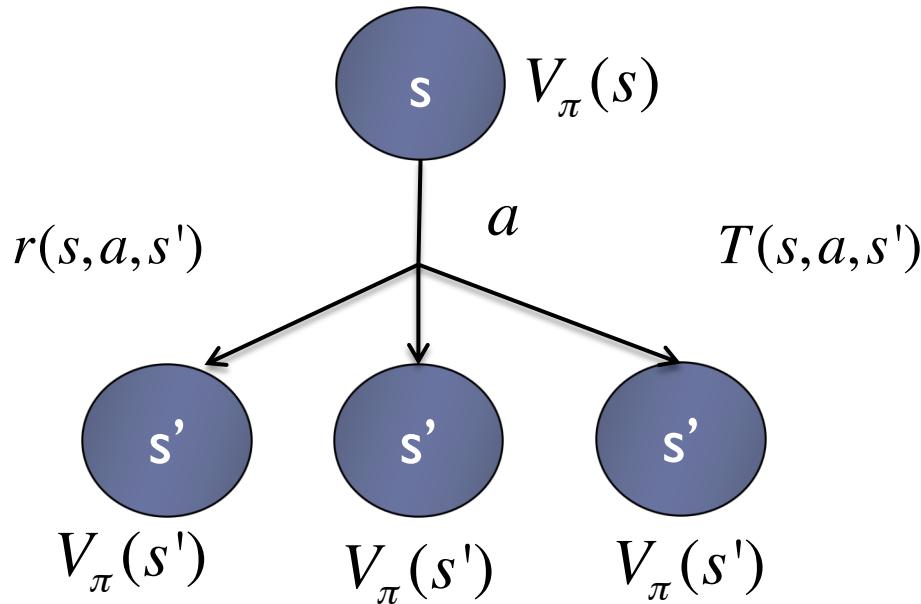
- ▶  $Q_\pi(s, a)$  is the expected value of taking action  $a$  from state  $s$ , and then following policy  $\pi$  thereafter.
- ▶  $V_\pi(s)$  is the expected value of following policy  $\pi$  from state  $s$ .

$$V_\pi(s) = \begin{cases} 0 & \text{if } s \text{ is a terminal state} \\ Q_\pi(s, \pi(s)) & \text{otherwise} \end{cases}$$

# Recursive computation of $Q_\pi$

- ▶  $Q_\pi(s,a)$  is the expected value of taking action  $a$  from state  $s$ , and then following policy  $\pi$  thereafter.

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



$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s')[r(s, \pi(s), s') + V_\pi(s')]$$

# Computing $V$ and $Q$ for the simple game

- States = {start, end}, end is terminal state
- Actions = {stay, quit}
  - Actions(start) = {stay, quit}
  - Actions(end) = {}
- Transitions  $T(s, a, s')$  and rewards  $r(s, a)$

Let  $\pi(\text{start}) = \text{stay}$ .

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

$$V_\pi(\text{start}) = \frac{1}{3}(4 + 0) + \frac{2}{3}(4 + V_\pi(\text{start}))$$

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

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

s	a	$r(s, a)$
start	quit	10
start	stay	4

System of linear equations which can be solved in closed form.

# Policy evaluation

---

- ▶ Calculation of  $V_\pi$  for a given policy  $\pi$
- ▶ **Approach 1:** set up system of linear equations and solve for  $V_\pi$  from the recurrences.
- ▶ **Approach 2:** (iterative improvement) repeatedly apply recurrences on  $V$  till convergence
  - ▶ Initialize  $V_\pi^{(0)}(s)$  to be 0 for all states  $s$ .
  - ▶ For  $i = 1, 2, \dots, T$ 
    - ▶ for each state  $s$

$$V_\pi^i(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[r(s, \pi(s), s') + V_\pi^{i-1}(s')]$$

# Convergence of iterative method

---

- ▶ Maintain  $V_{\pi}$  for  $i$  and for  $i-1$ . Check that the highest difference between the two vectors is less than or equal to a specific epsilon.

$$\max_s |V_{\pi}^i(s) - V_{\pi}^{i-1}(s)| \leq \varepsilon$$

- ▶ Each iteration of policy evaluation takes time  $O(|S|^2)$
- ▶ Complexity of policy evaluation is  $O(T|S|^2)$

# Optimal policy

- ▶ The optimal policy  $\pi^*$  has the highest expected utility among all policies. The associated value function is  $V^*$ .

$$V^*(s) = V_{\pi^*}(s) = \max_{\pi} V_{\pi}(s)$$

- ▶ We can use the recurrence on  $V$  to write the recurrence on  $V^*$ :

$$V_{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[r(s, \pi(s), s') + V_{\pi}(s')]$$

$$V^*(s) = \max_{a \in Actions(s)} \sum_{s'} T(s, a, s')[r(s, a, s') + V^*(s')]$$

Bellman's equation

# Relationship between $V^*$ and $Q^*$

---

- ▶ Given  $V^*$ , we can compute  $Q^*$

$$Q^*(s, a) = Q_{\pi^*}(s, a) = \sum_{s'} T(s, a, s')[r(s, a, s') + V^*(s')]$$

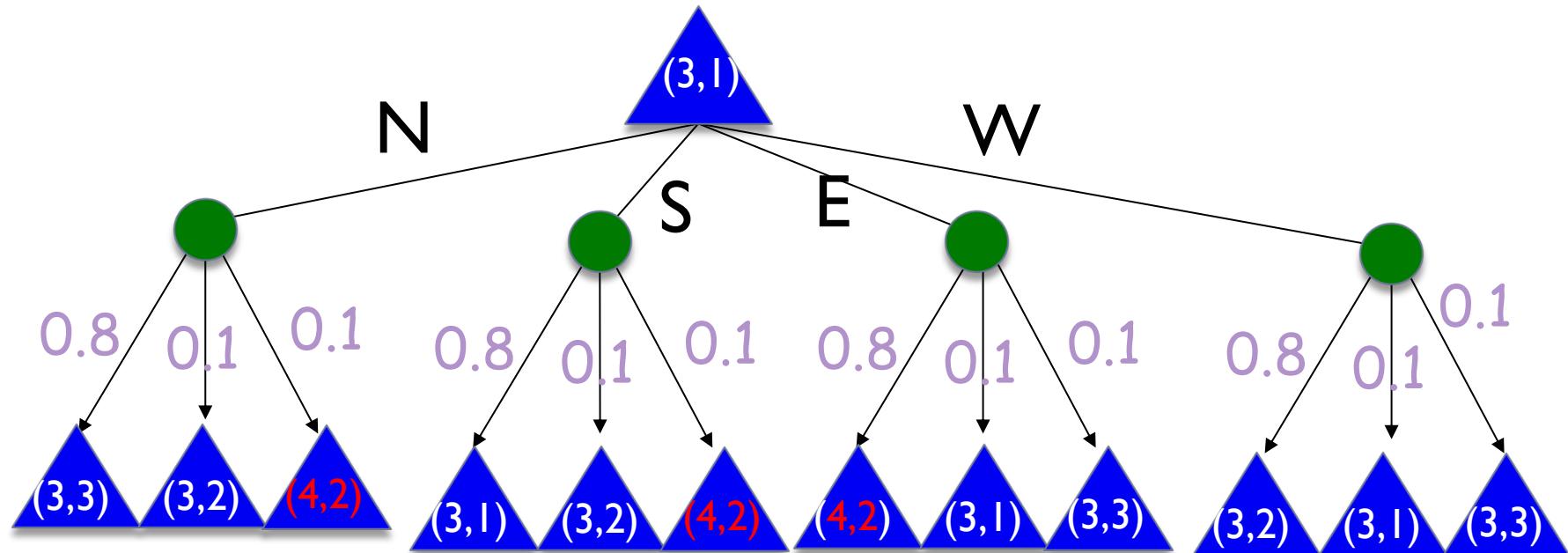
- ▶ Given  $Q^*$ , we can compute  $V^*$

$$V^*(s) = \max_{a \in Actions(s)} Q^*(s, a)$$

# Calculating $\pi^*$ from $V^*$ (policy extraction)

Principle of maximizing expected utility

$$\pi^*(s) = \operatorname{argmax}_{a \in \text{Actions}(s)} \sum_{s'} T(s, a, s') [r(s, a, s') + V^*(s')] = \operatorname{argmax}_{a \in \text{Actions}(s)} Q^*(s, a)$$



# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes versus classical search
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function and the optimal policy
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# Policy improvement

---

- ▶ Once we calculate  $V_\pi$  we can try to improve the policy to something better.
- ▶ Algorithm:
  - ▶ Compute  $Q_\pi(s,a)$  from  $V_\pi(s)$

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

- ▶ Update  $\pi$

$$\pi_{new}(s) = \arg \max_{a \in Actions(s)} Q(s,a)$$

- ▶ Take the action prescribed by the computed  $Q(s,a)$  – the action that has the highest expected value from state  $s$ .

# Example of policy improvement

---

- ▶ Suppose  $\pi(\text{start}) = \text{quit}$  in the simple game.
- ▶  $V_\pi(\text{start}) = 10$  and  $V_\pi(\text{end}) = 0$  under this policy

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

- ▶  $Q_\pi(\text{start}, \text{quit}) = 10$
- ▶  $Q_\pi(\text{start}, \text{stay}) = 1/3*[4+0] + 2/3[4+10] = 10.67$
- ▶ Policy improvement
  - ▶  $\pi_{\text{new}}(\text{start}) = \text{stay}$

# Policy improvement properties

---

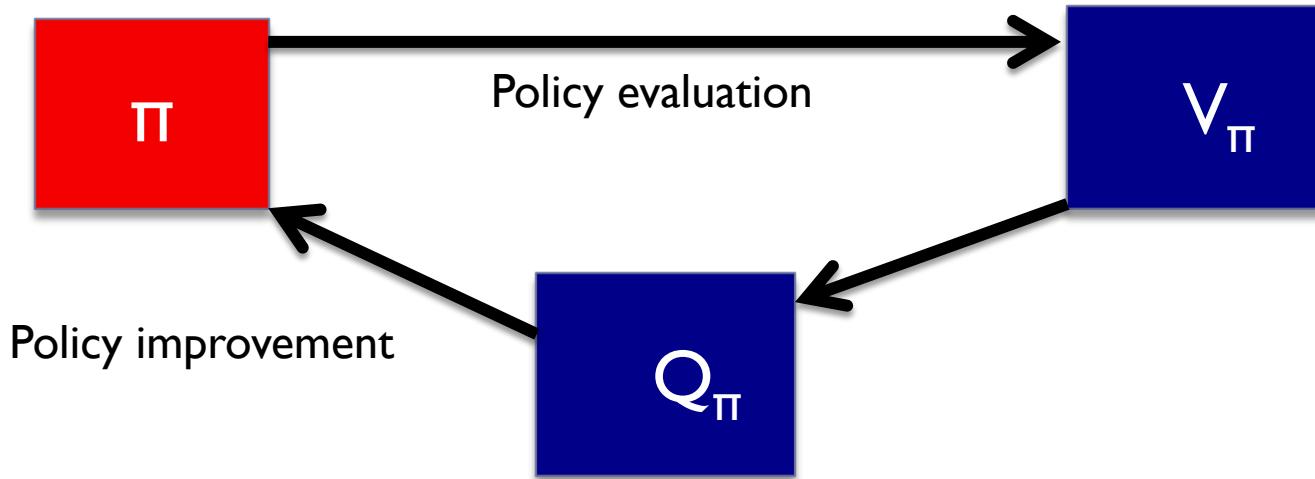
- ▶ The value of the new policy is at least as large as the old one.

$$V_{\pi_{new}}(s) \geq V_{\pi}(s) \text{ for all states } s$$

- ▶ Complexity of policy improvement:  $O(|A||S|^2 + |A||S|)$

# Policy iteration

- ▶ Iterative process
  - ▶ Start with a (random) policy  $\pi$
  - ▶ For  $i = 1, 2, \dots, T$ 
    - ▶ Policy evaluation: Compute  $V_\pi$  corresponding to policy  $\pi$
    - ▶ Policy improvement: Update policy based on  $Q_\pi$  calculated from  $V_\pi$



# Policy iteration algorithm

---

- ▶ Start with a random policy  $\pi$
- ▶ Loop
  - ▶ Compute value of policy  $\pi$

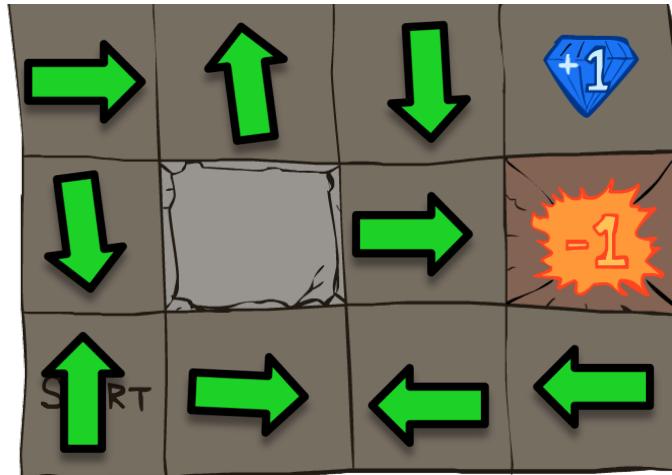
$$V_\pi(s) = \sum_{s'} T(s, \pi(s), s')[r(s, \pi(s), s') + V_\pi(s')]$$

- ▶ Improve the policy at each state, if possible
- ▶ Until no change to policy occurs

$$\pi'(s) = \arg \max_a \sum_{s'} T(s, a, s')[r(s, a, s') + V_\pi(s')]$$

# Example of policy iteration

- ▶ Step 0: Make a random initial policy



- ▶ Step 1: Evaluate policy
  - ▶  $V(1,1) = -0.01 + 0.8*V(2,1) + 0.1*V(1,1) + 0.1*V(2,1)$
  - ▶ ...
  - ▶  $V(3,3) = -0.01 + 0.8*V(2,3) + 0.1*V(3,2) + 0.1*V(3,3)$
  - ▶ 9 linear equations in nine unknowns (Gaussian elimination) or by iterative refinement of  $V$ .

# Example of policy iteration

Value of policy from step 0

3	-1.03	-0.97	-0.82	+1
2	-1.41	obstacle	-1.01	-1
1	-1.40	-1.22	-1.21	-1.19

# Example of policy iteration

---

- ▶ Step 2: Policy improvement
  - ▶ For every state calculate best action based on  $V$  computed in previous step

$$Q(s, a) = \sum_{s'} T(s, a, s')[r(s, a, s') + V(s')]$$

$$\pi(s) = \operatorname{argmax}_{a \in \text{Actions}(s)} Q(s, a)$$

- ▶ If there are any changes in policy, go back to policy evaluation (Step 1)

# Convergence of policy iteration

---

- ▶ Algorithm terminates when the policy improvement step yields no change.
- ▶ There are only finitely many policies for a finite state space, and each iteration can be shown to yield a better policy.
- ▶ So policy iteration must terminate.
- ▶ Can be shown to terminate on the optimal policy because  $V$  has a single fixed point (Howard, 1980).

# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes versus classical search
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function and the optimal policy
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# Computing policy from Bellman's equations directly

---

- ▶ We can write equations for all 9 non-terminal states in the grid world problem.

$$\begin{aligned} V(1,1) = & -0.01 + \max \{ 0.8V(1,2) + 0.1V(2,1) + 0.1V(1,1), \\ & 0.9V(1,1) + 0.1V(1,2), \\ & 0.9V(1,1) + 0.1V(2,1), \\ & 0.8V(2,1) + 0.1V(1,2) + 0.1V(1,1) \} \end{aligned}$$

- ▶ Unique solutions exist to this system of equations (Bellman, 1957).

# Value iteration

---

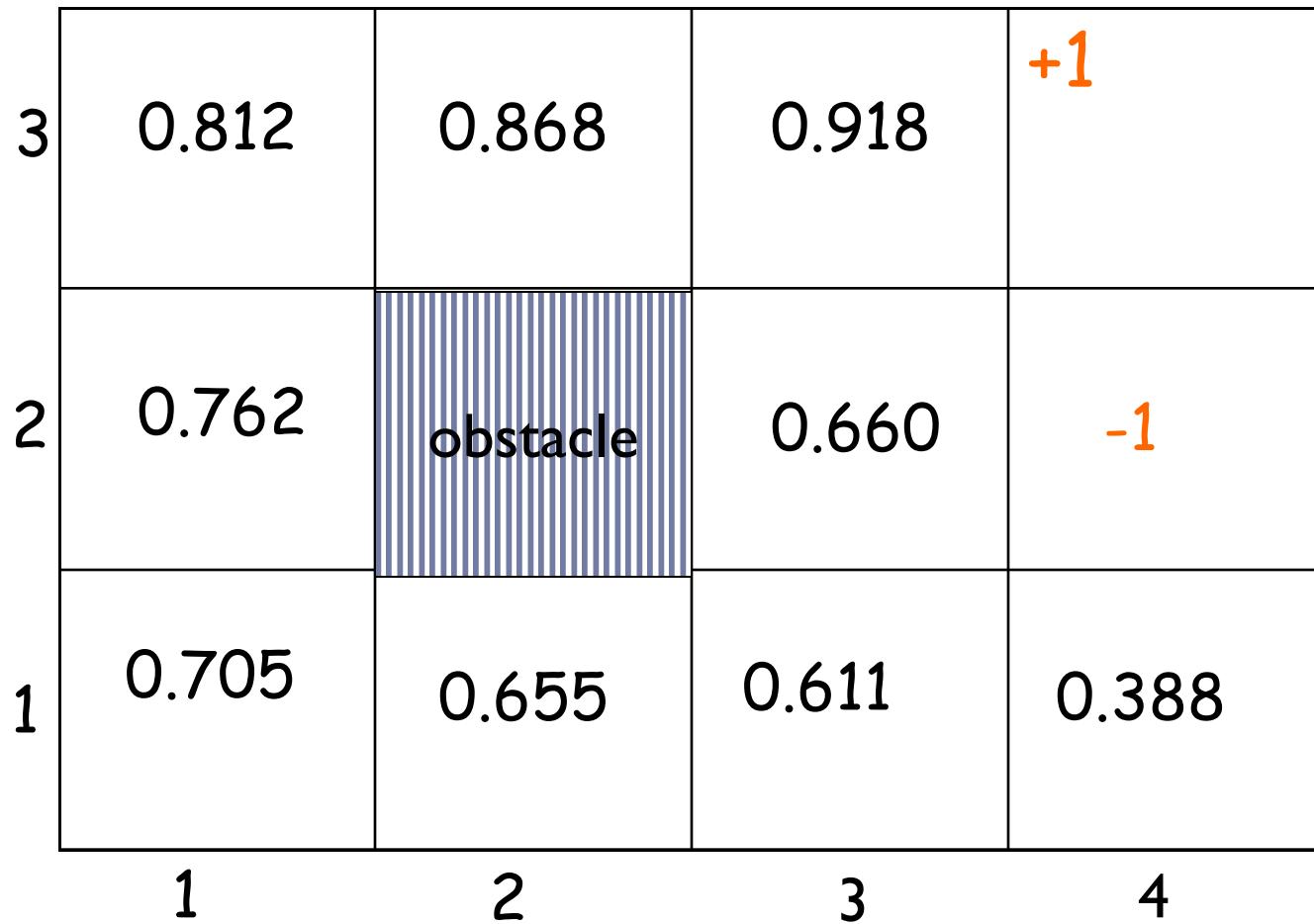
- ▶ Define  $V^{(i)}(s)$  = optimal value of state  $s$  if we reach a terminal state in  $i$  more steps (maximum sum of rewards starting from state  $s$  and ending in  $i$  steps)
- ▶ Start with  $V^{(0)}(s) = 0$  for all states  $s$  in  $S$
- ▶ Calculate  $V^{(i+1)}(s)$  from  $V^{(i)}(s)$  using Bellman update

$$V^{(i+1)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s, a, s')[r(s, a, s') + V^{(i)}(s')]$$

- ▶ Repeat until convergence (successive values of  $V$  are close enough)

# Computation of $V^*$

$$r(s) = -0.04$$



# Value iteration and Bellman's equations

---

- ▶ Bellman's equations characterize the optimal values  $V^*$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s')[r(s, a, s') + V^*(s')]$$

- ▶ Value iteration computes the optimal values  $V^*$

$$V^{(i+1)}(s) \leftarrow \max_a \sum_{s'} T(s, a, s')[r(s, a, s') + V^{(i)}(s')]$$

- ▶ Value iteration is a fixed point solution method.

# Complexity of value iteration

---

- ▶ Each iteration is  $O(S^2A)$ . Why?
- ▶ We will prove convergence of value iteration and place a bound on the number of iterations needed for value iteration to converge.

# Termination criteria for value iteration

---

- ▶ RMS error of the utility values.
- ▶ Policy loss: stop when policies on subsequent iterations are the same.
  - ▶ Practically, policy convergence occurs much before value convergence.

# Optimizing value iteration

---

- ▶ It is slow: each iteration takes  $O(S^2A)$  time
- ▶ Many optimizations to speed it up
  - ▶ Early termination: policy convergence
  - ▶ Selective updating: Updating  $V$  values of important states rather than all states in each iteration, by taking advantage of state space structure

# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes versus classical search
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function and the optimal policy
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# A problem with sum of rewards utility

---

- ▶ For infinite horizon problems, with no terminal states, the sum of rewards becomes infinity.

$$V(s_0 s_1 \dots) = \sum_{t=0}^{\infty} r(s_t)$$

- ▶ There is no way to compare state histories and define an optimal policy!

# Discounting

---

- ▶ Discounting makes the sum of rewards finite, so we can compare histories.

$$V(s_0 s_1 \dots) = \sum_{t=0}^{\infty} \gamma^t r(s_t)$$

- ▶ Discount factor  $\gamma$

$$[1, 1, 1, \dots] \rightarrow [1, \gamma, \gamma^2, \dots]$$

- ▶ Discounting is a good model of human and animal preferences over time.

# Value iteration with discounting

---

- ▶ Define  $V^{(i)}(s)$  = optimal value of state  $s$  if we reach a terminal state in  $i$  more steps
- ▶ Start with  $V^{(0)}(s) = 0$  for all states  $s$  in  $S$
- ▶ Calculate  $V^{(i+1)}(s)$  from  $V^{(i)}(s)$  using Bellman update

$$V^{(i+1)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s, a, s') [r(s, a, s') + \gamma V^{(i)}(s')]$$

- ▶ Repeat until convergence (successive values of  $V_t$  are close enough) (converged  $V_i = V^*$ )

# Outline

---

- ▶ Modeling stochastic worlds
  - ▶ Stochastic versus deterministic worlds
  - ▶ Policies versus plans
  - ▶ Markov decision processes versus classical search
- ▶ A simple game
  - ▶ Modeling game as MDP
  - ▶ Solving for optimal policy using the principle of maximizing expected utility
- ▶ The value function and the optimal policy
  - ▶ V, Q and Bellman's equations
- ▶ Algorithms for solving MDPs
  - ▶ Policy iteration
    - ▶ Properties and proof of convergence
  - ▶ Value iteration
  - ▶ Discounting
  - ▶ Proof of convergence of value iteration

# Value iteration : initialization

Discount factor = 0.9, step cost = 0, 0.8/0.1/0.1 action model

3	0	0	0	+
2	0	obstacle	0	-
1	0	0	0	0

# Value iteration : iteration 1

Discount factor = 0.9, step cost = 0, 0.8/0.1/0.1 action model

3	0	0	A	+
2	0	obstacle	0	-
1	0	0	0	0

# Value iteration : iteration 2

Discount factor = 0.9, step cost = 0, 0.8/0.1/0.1 action model

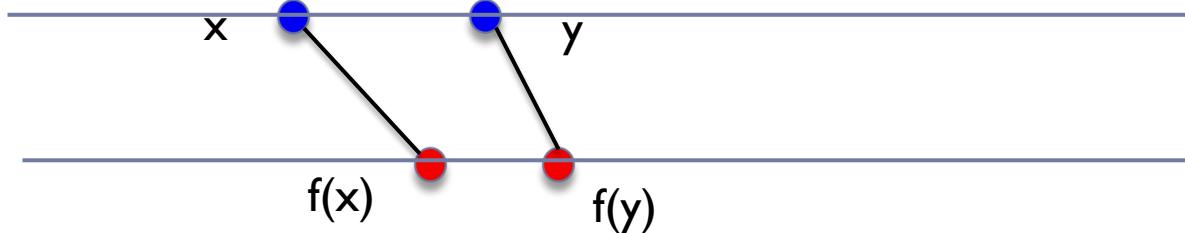
3	0	C	B	+
2	0	obstacle	D	-
1	0	0	0	0

# Contraction mapping

- ▶ A contraction is a function  $f: \mathbb{R} \rightarrow \mathbb{R}$  such that

$$|f(x) - f(y)| \leq c |x - y| \text{ where } 0 \leq c < 1$$

- ▶ where  $|x-y|$  denotes the distance between  $x, y$  in  $\mathbb{R}$ .



# Contraction mapping principle

---

- ▶ Consider a contraction mapping  $f:R \rightarrow R$  on the reals with contraction coefficient  $\gamma$
- ▶ Take the sequence  $\{x_1, x_2, \dots, \}$ 
  - ▶ where  $x_{i+1} = f(x_i)$
- ▶ The sequence has a unique fixed point  $a$  and

$$|a - x_n| \leq \gamma^n |a - x_1|$$

- ▶ We can calculate the number  $n$  of applications of the contraction mapping to get to within a specified  $\varepsilon$  of the fixed point  $a$ .

# Application to value iteration convergence

- ▶ We construct the sequence  $V_0, V_1, \dots, V_n$ 
  - ▶ where  $V_{i+1} = BV_i$
  - ▶  $B$  is the Bellman update (which is a contraction by factor  $\gamma$ )
- ▶ The fixed point of the sequence is  $V^*$

$$|V^* - V_n| \leq \gamma^n |V^* - V_0| \quad (-r_{\max}, r_{\max}) \text{ is the highest range in the reward function}$$

- ▶ We will take the max norm of a vector as the distance function  $|.|$
- ▶ Then, we want  $|V^* - V_n| \leq \epsilon$
- ▶ We know  $V_0 = \text{all zeros}$  and the biggest distance between  $V^*$  and  $V_0$  can only be  $\frac{2r_{\max}}{(1-\gamma)}$

# Number of iterations

---

- ▶ If  $V^*$  were the true utility vector, then

$$\|BV - BV^*\| \leq \gamma \|V - V^*\|$$

- ▶ To get to the desired error bound  $\epsilon$ , run  $N$  iterations
- ▶  $\epsilon \leq \gamma^N \frac{2r}{(1-\gamma)}$

# Types of MDPs

---

- ▶ **Finite horizon MDPs:** A decision problem in which the agent has to maximize its expected utility over a specific time period.
  - ▶ The optimal policy for finite horizon problems could be non-stationary (be a function of the length of the horizon).
  - ▶ We do not study finite horizon MDPs in this class.
- ▶ **Indefinite and infinite horizon MDPs** (i.e., no fixed deadline)
  - ▶ Optimal policy is **stationary**; the optimal action in a given state is a function of that state and not the time at which the agent is at that state.
  - ▶ We have covered this family of MDPs in this class.