Reminders

- Course project progress report 1 due 2/27 → 2/29
- HW3 due 2/29
- PRA4 due 3/14
- Come to OH for course project discussion!

Artificial Intelligence Methods for Social Good

Lecture 12:

Basics of Reinforcement Learning

17-537 (9-unit) and 17-737 (12-unit)

Fei Fang

feifang@cmu.edu

Outline

- Recap: Markov Decision Process (MDP)
- POMDP and Ranger Patrol Problem
- Reinforcement Learning (RL)
  - Q-Learning
  - Policy Gradient

Learning Objectives

- Understand the concept of
  - Reinforcement Learning
- Describe the following algorithms
  - Q-Learning
  - ϵ-Greedy
  - Policy Gradient
- For the adaptive ranger patrol problem, understand
  - Motivation
  - Task being solved
  - MDP Model for the problem

Recap: Markov Decision Process (MDP)

- Special case of sequential decision-making problems
- MDP = (S, A, T, R, γ)
  - S: set of states, $s_t \in S$ (where can agent be?)
  - A: set of actions, $a_t \in A$ (what can agent do?)
  - T: transition function $T(s_t, a_t, s_{t+1}) = \mathbb{P}(s_{t+1} | s_t, a_t)$ (what happens next?)

Next state only depends on the current state, not previous states! (Markovian)

  - R: reward function $r_t = R(s_t)$ or $R(s_t, a_t)$ or $R(s_t, a_t, s_{t+1})$ (what do I gain?)
  - $\gamma \in [0, 1]$ (discount factor)

Recap: Markov Decision Process (MDP)

- MDP = (S, A, T, R, γ)
- (Deterministic) Policy $\pi: S \rightarrow A$
  - Maps state to action, defines a *plan*
- Given a policy π, we can sample history $h = \{s_0, a_0, s_1, a_1, \ldots\}$
- Goal: find π to maximize utility
  - Expected
  - $\pi^* = \text{argmax}_\pi \mathbb{E}_{h\sim\pi}\left[\sum_t \gamma^t R\big(s_t, \pi(s_t)\big)\right]$
- Myopic strategy does not work: a_t affects future states

Recap: Value Function

- The value function of a given policy π describes the expected accumulated reward with discount starting from a state

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R\big(s_t, \pi(s_t)\big)]$$

Where $s_0 = s$

Sometimes called state value function or V-value function

Recap: Q-Value Function

- Similar to state value function $V^{\pi}(s)$, but defined on state-action pair
- $Q^{\pi}(s,a)$: expected total reward from state s onward if taking action a in state s, and follow policy π afterward

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

Obviously $V^{\pi}(s) = Q^{\pi}(s,\pi(s))$

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

Recap: Bellman Equation

- $V^*(s), Q^*(s,a)$ satisfy the following Bellman Equation

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^*(s') \right]$$

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

Outline

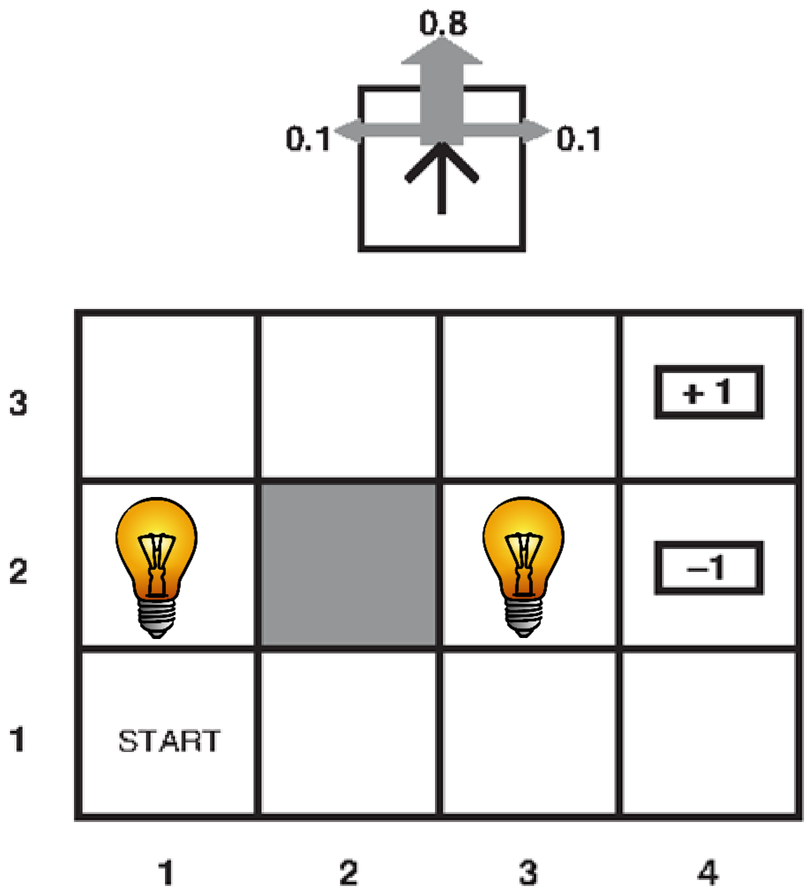
- Recap: Markov Decision Process (MDP)
- POMDP and Ranger Patrol Problem
- Reinforcement Learning (RL)
  - Q-Learning
  - Policy Gradient

Partially Observable MDP (POMDP)

- POMDP = $(S, A, T, R, \Omega, O, \gamma)$
  - Ω: A set of possible observations
  - O: Observation probabilities: $o_t \sim O(o|s_{t+1})$
  - Agent is at state s_t (unknown to agent), takes action a_t, gets reward r_t, ends up at state s_{t+1}, and observes o_t

Example

MDP = $(S, A, T, R, \Omega, O, \gamma)$

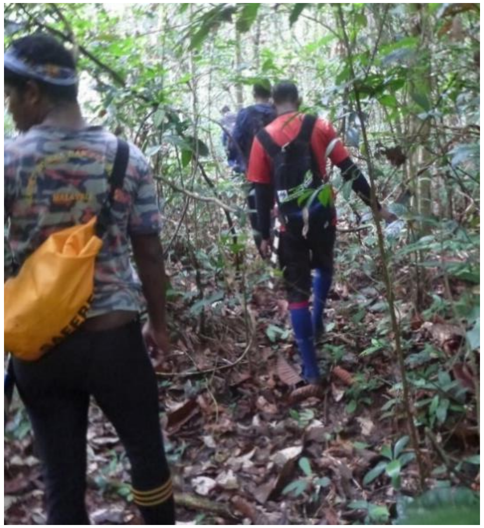


Ω ={Dark, Light}

Actions = {Up, Down, Left, Right}

Example: Ranger Patrol with Real-Time Information

Example: Ranger Patrol with Real-Time Information



Animal Footprint

Tiger Sign

Lighter

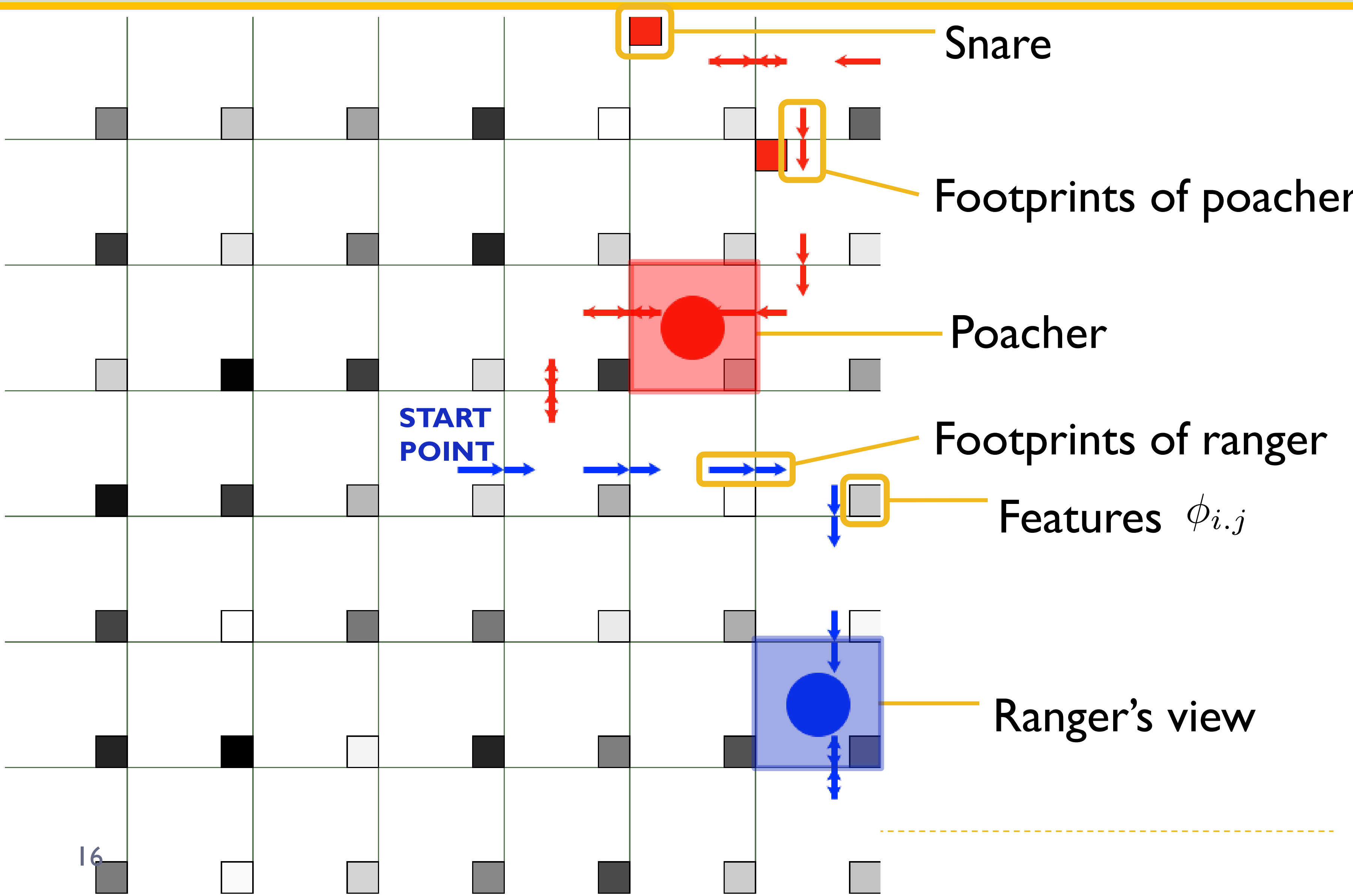
Example: Ranger Patrol with Real-Time Information

- Poachers move in the protected area and place snares (poaching tool)
- Rangers patrol in protected areas to combat poaching
- How should rangers react to real-time information?

Footprints

Deep Reinforcement Learning for Green Security Games with Real-Time Information Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, Fei Fang In AAAI-19

POMDP Model for Ranger Patrol



POMDP Model for Ranger Patrol

POMDP = $(S, A, T, R, \Omega, O, \gamma)$

- Assume the heuristic strategy used by poacher is
  - Randomly move towards one of the 4 neighboring cells
  - Place snare in each cell he visits with probability 0.3
- Assume the ranger gets +3 if catches poacher, +1 if removes snare, and -0.1 for every step regardless
- Ranger can choose to stand still

- Discussion: How to formulate the problem as a POMDP?

POMDP Model for Ranger Patrol

- State:
- Action:
- Transition:
- Reward:
- Observation:
- Observation probability:

POMDP Model for Ranger Patrol

- State: Location of poacher, ranger, snares, footprints in all locations
- Action: Up, Down, Left, Right, Still
- Transition: Determined by poacher's heuristic strategy
- Reward: Determined by the reward rule
- Observation: Location of ranger, footprint, snare
- Observation probability: With prob. 1, observe the ranger location and the footprint/snare in the ranger location

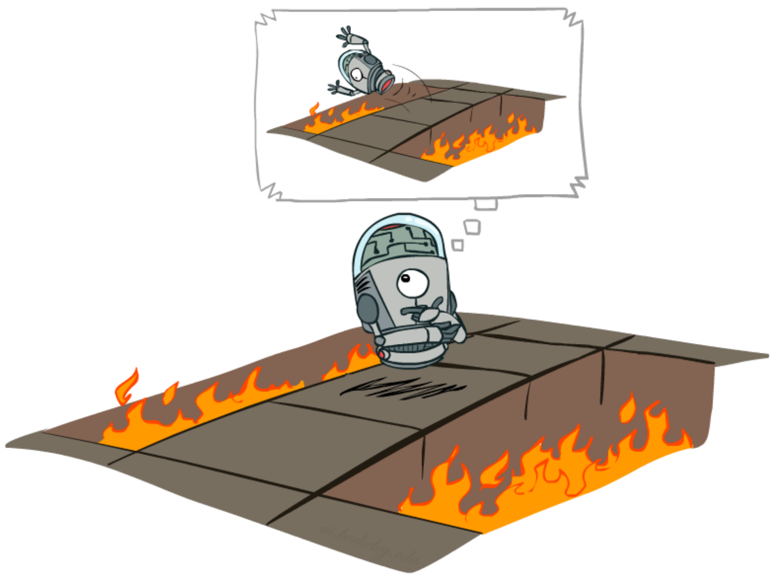
Outline

- Recap: Markov Decision Process (MDP)
- POMDP and Ranger Patrol Problem
- Reinforcement Learning (RL)
  - Q-Learning
  - Policy Gradient

Reinforcement Learning

- Learn an optimal policy for the environment without having a complete model

Don't know T or R (or just hard to enumerate)



Learn through Trial and Error

Don't have a **complete model of the environment**!

Have to actually learn what happens if take an action in a state

Reinforcement Learning

- Learn the optimal policy without knowing T or R

- Model-based RL
  - Build a model (estimate T and R) then find optimal policy

- Model-free RL
  - Directly evaluate without building a model

Model-Based RL with Random Actions

- Choose actions randomly
- Estimate $T(\cdot)$ and $R(\cdot)$ from sample trials (average counts)
- Use estimated $T(\cdot)$ and $R(\cdot)$ to compute estimate of optimal values and optimal policy (i.e., solve the MDP with estimated $T(\cdot)$ and $R(\cdot)$)

Model-Based RL with Random Actions

- Consider a trial

  Start at $(1,1)$

  $s = (1,1)$ action=tright (try going right) based on π

  Reward$= -0.01$; End up at $s' = (2,1)$

  $s = (2,1)$ action=tright (try going right) based on π

  Reward$= -0.01$; End up at $s' = (2,1)$

  $s = (2,1)$ action=tright (try going right) based on π

  Reward$= -0.01$; End up at $s' = (3,1)$

  $s = (3,1)$ action=tright (try going right) based on π

  Reward$= -0.01$; End up at $s' = (4,1)$

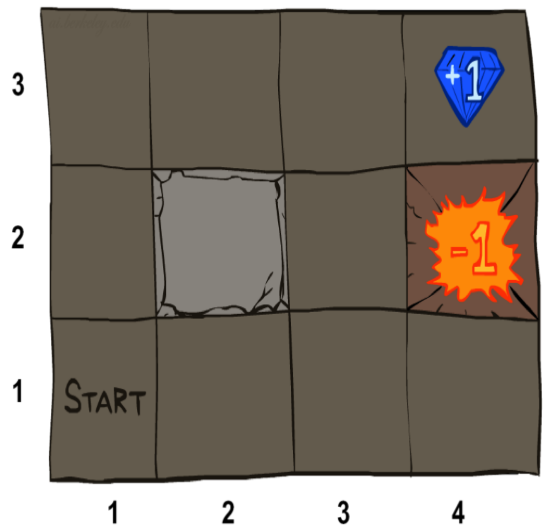
  $s = (4,1)$ action=tup (try going up) based on π

  Reward$= -0.01$; End up at $s' = (4,2)$

  $s = (4,2)$ No action available. Reward$= -1$; Terminate

  Estimate $T\big((2,1)|(2,1)\big)$? $R\big((2,1), tright\big)$?

Environment



Policy: Choose action randomly

Model-Free RL

- Can we find the optimal policy without explicitly estimating $T(\cdot)$ and $R(\cdot)$?

- Value-based method
  - Q-Learning

- Policy-based method + Actor-critic method
  - Policy-gradient
  - Advantage Actor-Critic (A2C)

Q-Learning

Recall $Q^*(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$

- Q-Learning
  - Start with some random guess of optimal Q values, $\hat{Q}^*(s,a)$
  - Agent interact with the environment following some policy π (no need to be optimal)
  - In one step of a trial: agent is in state s, take action $a = \pi(s)$, get reward r, end up in state s'
  - Update the estimated optimal Q value at (s,a) with

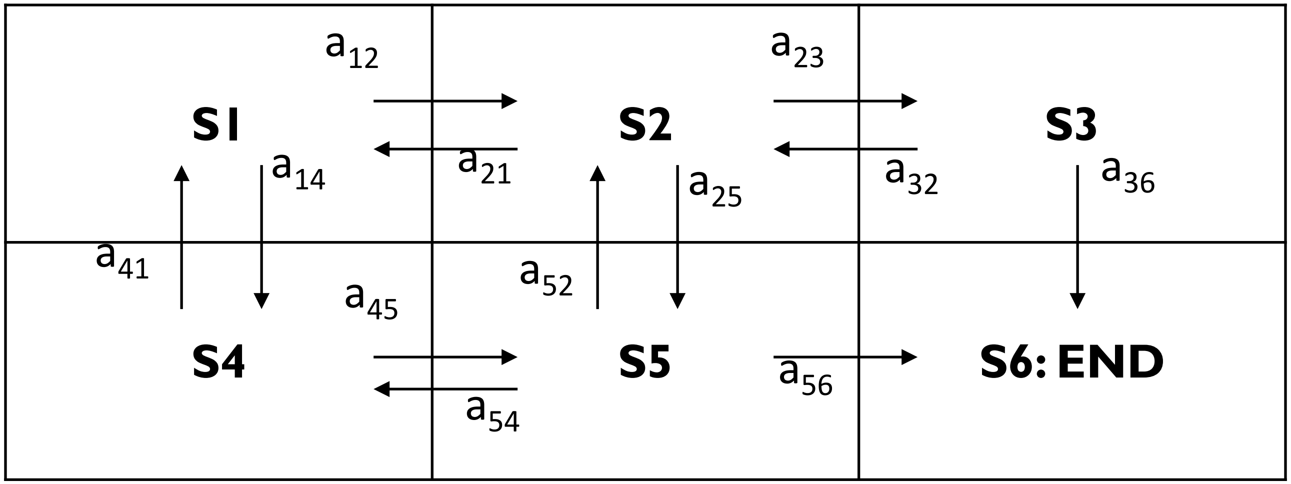
$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}^*(s',a'))$$

Q-Learning

- Given estimated value of $Q^*(s, a)$, derive estimate of optimal policy

$$\hat{\pi}^*(s) = \underset{a}{\operatorname{argmax}}\, \hat{Q}^*(s, a)$$

Q-Learning Example

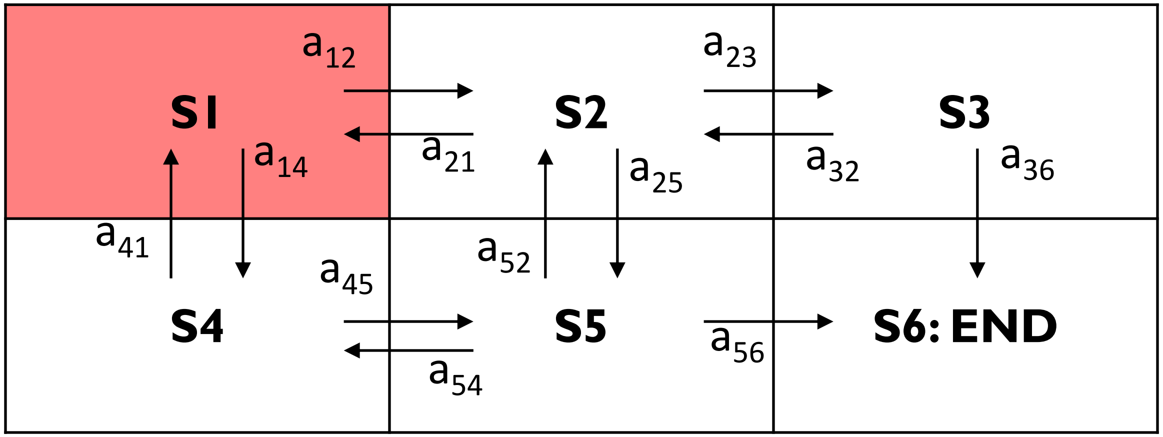


6 states, S1,..S6
12 actions a_{ij}
Deterministic state transitions (but you don't know this beforehand)
$R = 100$ in S6, $R = 0$ otherwise (again, you don't know this)
Use $\gamma = 0.5, \alpha = 1$
Random behavior policy

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



$\hat{Q}^*(S1, a_{12})$	0
$\hat{Q}^*(S1, a_{14})$	0
$\hat{Q}^*(S2, a_{21})$	0
$\hat{Q}^*(S2, a_{25})$	0
$\hat{Q}^*(S2, a_{23})$	0
$\hat{Q}^*(S3, a_{32})$	0
$\hat{Q}^*(S3, a_{36})$	0
$\hat{Q}^*(S4, a_{41})$	0
$\hat{Q}^*(S4, a_{45})$	0
$\hat{Q}^*(S5, a_{52})$	0
$\hat{Q}^*(S5, a_{54})$	0
$\hat{Q}^*(S5, a_{56})$	0
$\hat{Q}^*(S6, null)$	0

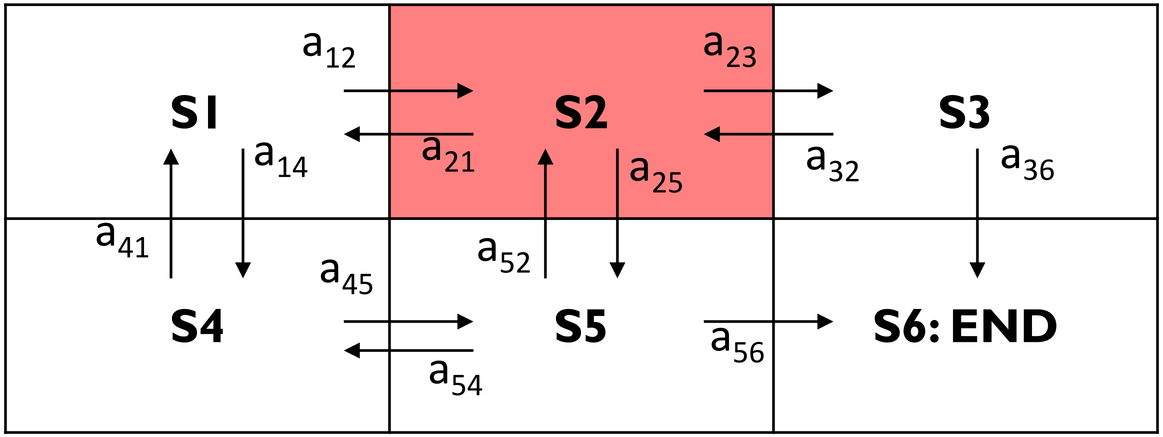
Start at S1, available actions: a_{12}, a_{14}
Probability of choosing each of them: 0.5
Choose a_{12}
Get reward 0, get to state S2
Update state-value function

$$\hat{Q}^*(S1, a_{12}) \leftarrow (1-\alpha)\hat{Q}^*(S1, a_{12}) + \alpha\left(r + \gamma \max_{a' \in \{a_{21}, a_{23}, a_{25}\}} \hat{Q}^*(S2, a')\right) = 0$$

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



$\hat{Q}^*(S1, a_{12})$	0
$\hat{Q}^*(S1, a_{14})$	0
$\hat{Q}^*(S2, a_{21})$	0
$\hat{Q}^*(S2, a_{25})$	0
$\hat{Q}^*(S2, a_{23})$	0
$\hat{Q}^*(S3, a_{32})$	0
$\hat{Q}^*(S3, a_{36})$	0
$\hat{Q}^*(S4, a_{41})$	0
$\hat{Q}^*(S4, a_{45})$	0
$\hat{Q}^*(S5, a_{52})$	0
$\hat{Q}^*(S5, a_{54})$	0
$\hat{Q}^*(S5, a_{56})$	0
$\hat{Q}^*(S6, null)$	0

At S2, available actions: a_{21}, a_{23}, a_{25}

Probability of choosing each of them: $\frac{1}{3}$

Choose a_{23}

Get reward 0, get to state S3

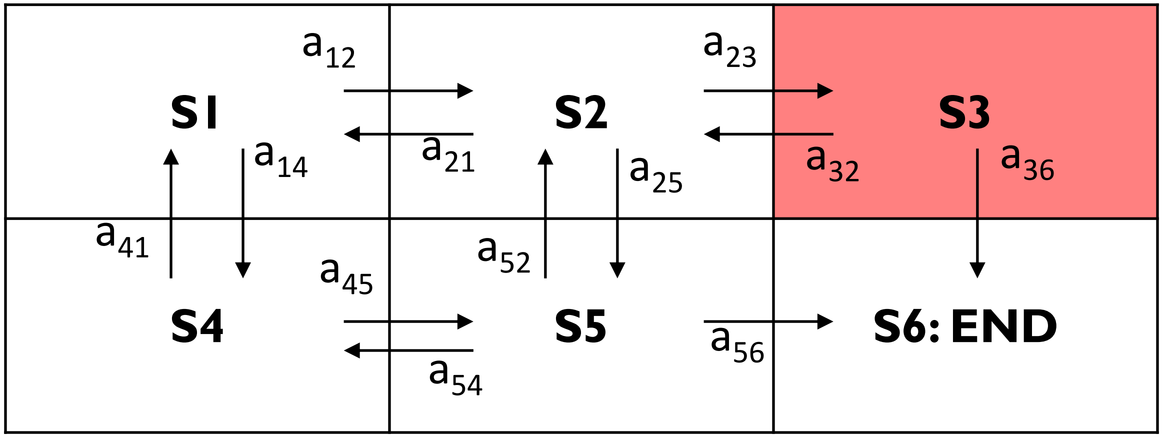
Update state-value function

$$\hat{Q}^*(S2, a_{23}) \leftarrow (1-\alpha)\hat{Q}^*(S2, a_{23}) + \alpha\left(r + \gamma \max_{a' \in \{a_{32}, a_{36}\}} \hat{Q}^*(S3, a')\right) = 0$$

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



$\hat{Q}^*(S1,a_{12})$	0
$\hat{Q}^*(S1,a_{14})$	0
$\hat{Q}^*(S2,a_{21})$	0
$\hat{Q}^*(S2,a_{25})$	0
$\hat{Q}^*(S2,a_{23})$	0
$\hat{Q}^*(S3,a_{32})$	0
$\hat{Q}^*(S3,a_{36})$	0
$\hat{Q}^*(S4,a_{41})$	0
$\hat{Q}^*(S4,a_{45})$	0
$\hat{Q}^*(S5,a_{52})$	0
$\hat{Q}^*(S5,a_{54})$	0
$\hat{Q}^*(S5,a_{56})$	0
$\hat{Q}^*(S6,null)$	0

At S3, available actions: a_{32}, a_{36}
Probability of choosing each of them: 0.5
Choose a_{36}

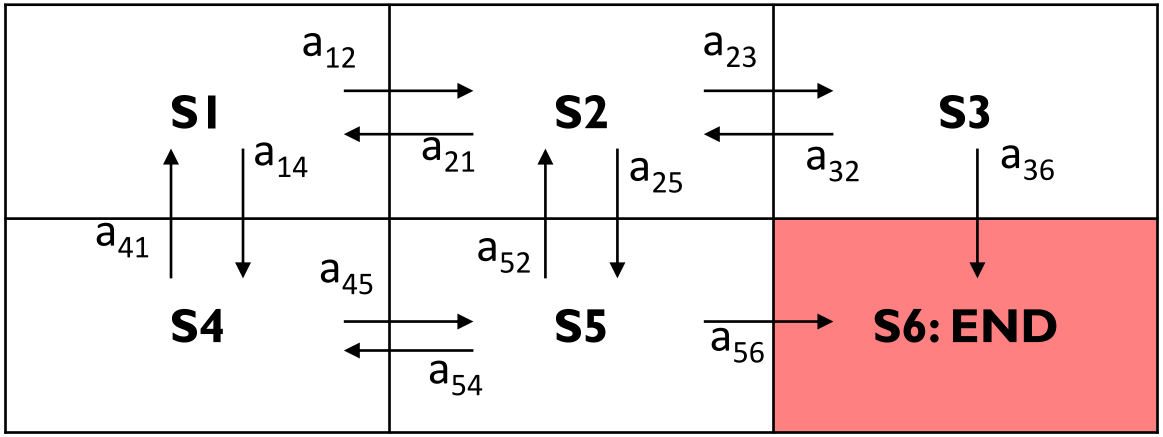
Get reward 0, get to state S6
Update state-value function

$$\hat{Q}^*(S3,a_{36}) \leftarrow (1-\alpha)\hat{Q}^*(S3,a_{36}) + \alpha\left(r + \gamma \max_{a' \in \{null\}} \hat{Q}^*(S6,a')\right) = 0$$

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



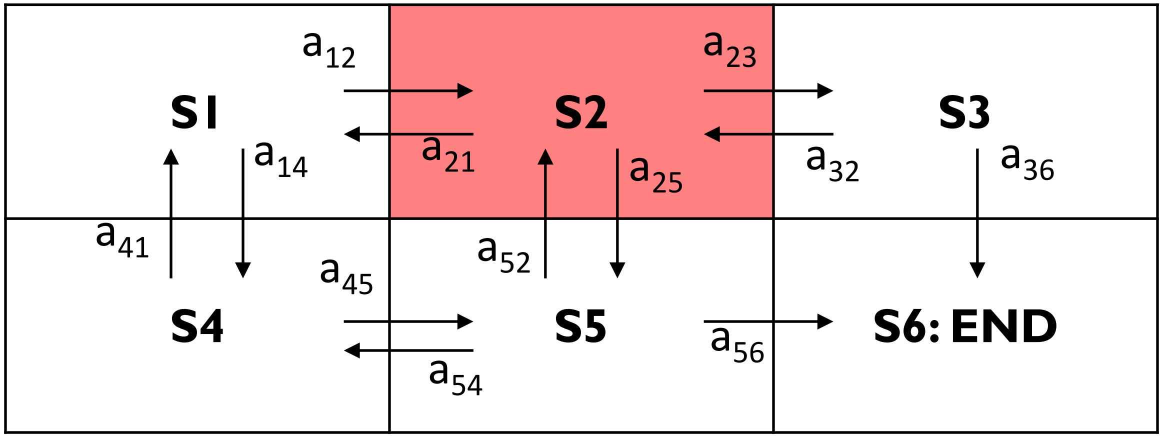
$\hat{Q}^*(S1, a_{12})$	0
$\hat{Q}^*(S1, a_{14})$	0
$\hat{Q}^*(S2, a_{21})$	0
$\hat{Q}^*(S2, a_{25})$	0
$\hat{Q}^*(S2, a_{23})$	0
$\hat{Q}^*(S3, a_{32})$	0
$\hat{Q}^*(S3, a_{36})$	0
$\hat{Q}^*(S4, a_{41})$	0
$\hat{Q}^*(S4, a_{45})$	0
$\hat{Q}^*(S5, a_{52})$	0
$\hat{Q}^*(S5, a_{54})$	0
$\hat{Q}^*(S5, a_{56})$	0
$\hat{Q}^*(S6, null)$	0 $\rightarrow 100$

Terminal state, get reward 100, $\hat{Q}^*(S6, null) \leftarrow 100$

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



$\hat{Q}^*(S1, a_{12})$	0
$\hat{Q}^*(S1, a_{14})$	0
$\hat{Q}^*(S2, a_{21})$	0
$\hat{Q}^*(S2, a_{25})$	0
$\hat{Q}^*(S2, a_{23})$	0
$\hat{Q}^*(S3, a_{32})$	0
$\hat{Q}^*(S3, a_{36})$	0
$\hat{Q}^*(S4, a_{41})$	0
$\hat{Q}^*(S4, a_{45})$	0
$\hat{Q}^*(S5, a_{52})$	0
$\hat{Q}^*(S5, a_{54})$	0
$\hat{Q}^*(S5, a_{56})$	0
$\hat{Q}^*(S6, null)$	100

Start a new episode!

Start at S2, available actions: a_{21}, a_{23}, a_{25}

Probability of choosing each of them: $\frac{1}{3}$

Choose a_{23}

Get reward 0, get to state S3

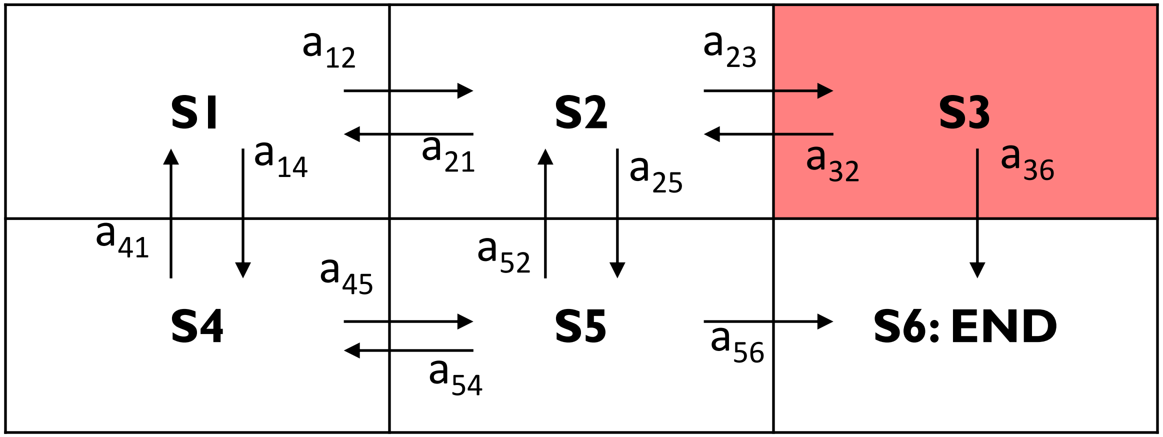
Update state-value function

$$\hat{Q}^*(S2,a_{23}) \leftarrow (1-\alpha)\hat{Q}^*(S2,a_{23}) + \alpha\left(r + \gamma \max_{a' \in \{a_{32},a_{36}\}} \hat{Q}^*(S3,a')\right) = 0$$

Q-Learning Example

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^*(s',a')\right)$$

$R = 100$ in S6, $\gamma = 0.5, \alpha = 1$



$\hat{Q}^*(S1, a_{12})$	0
$\hat{Q}^*(S1, a_{14})$	0
$\hat{Q}^*(S2, a_{21})$	0
$\hat{Q}^*(S2, a_{25})$	0
$\hat{Q}^*(S2, a_{23})$	0
$\hat{Q}^*(S3, a_{32})$	0
$\hat{Q}^*(S3, a_{36})$	0 $\rightarrow 50$
$\hat{Q}^*(S4, a_{41})$	0
$\hat{Q}^*(S4, a_{45})$	0
$\hat{Q}^*(S5, a_{52})$	0
$\hat{Q}^*(S5, a_{54})$	0
$\hat{Q}^*(S5, a_{56})$	0
$\hat{Q}^*(S6, null)$	100

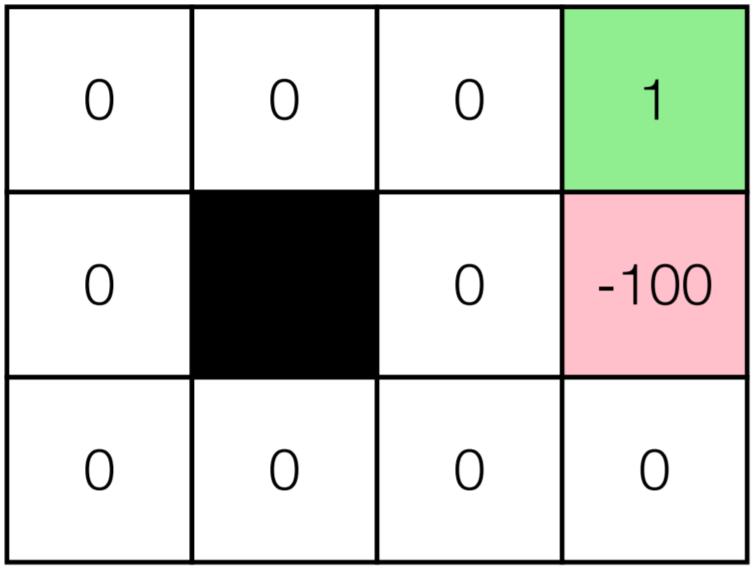
At S3, available actions: a_{32}, a_{36}
Probability of choosing each of them: 0.5
Choose a_{36}

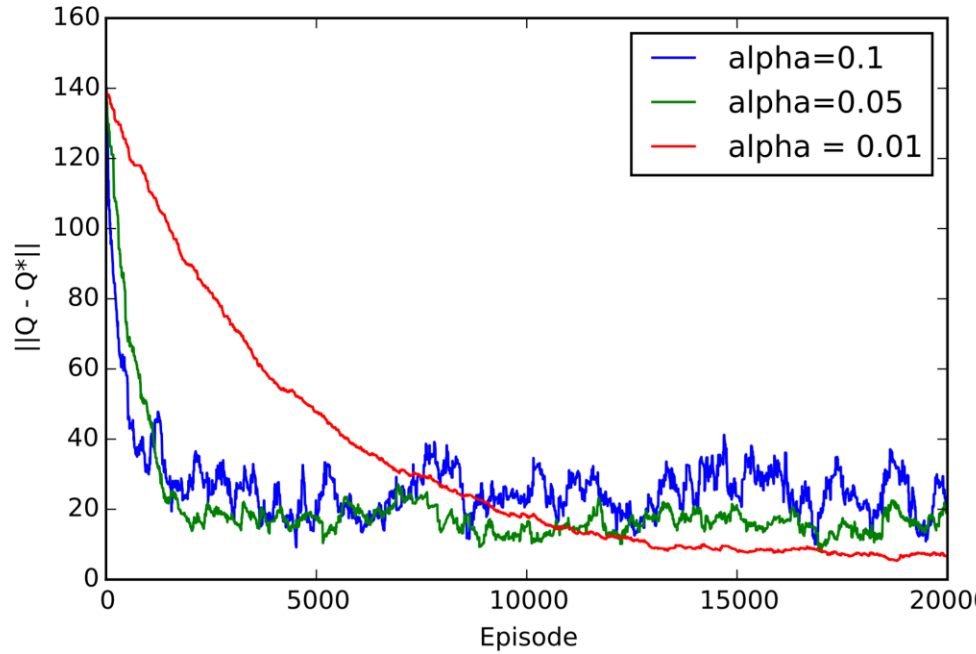
Get reward 0, get to state S6
Update state-value function

$$\hat{Q}^*(S3, a_{36}) \leftarrow (1-\alpha)\hat{Q}^*(S3, a_{36}) + \alpha\left(r + \gamma \max_{a' \in \{null\}} \hat{Q}^*(S6, a')\right) = 50$$

Q-Learning

- Impact of α

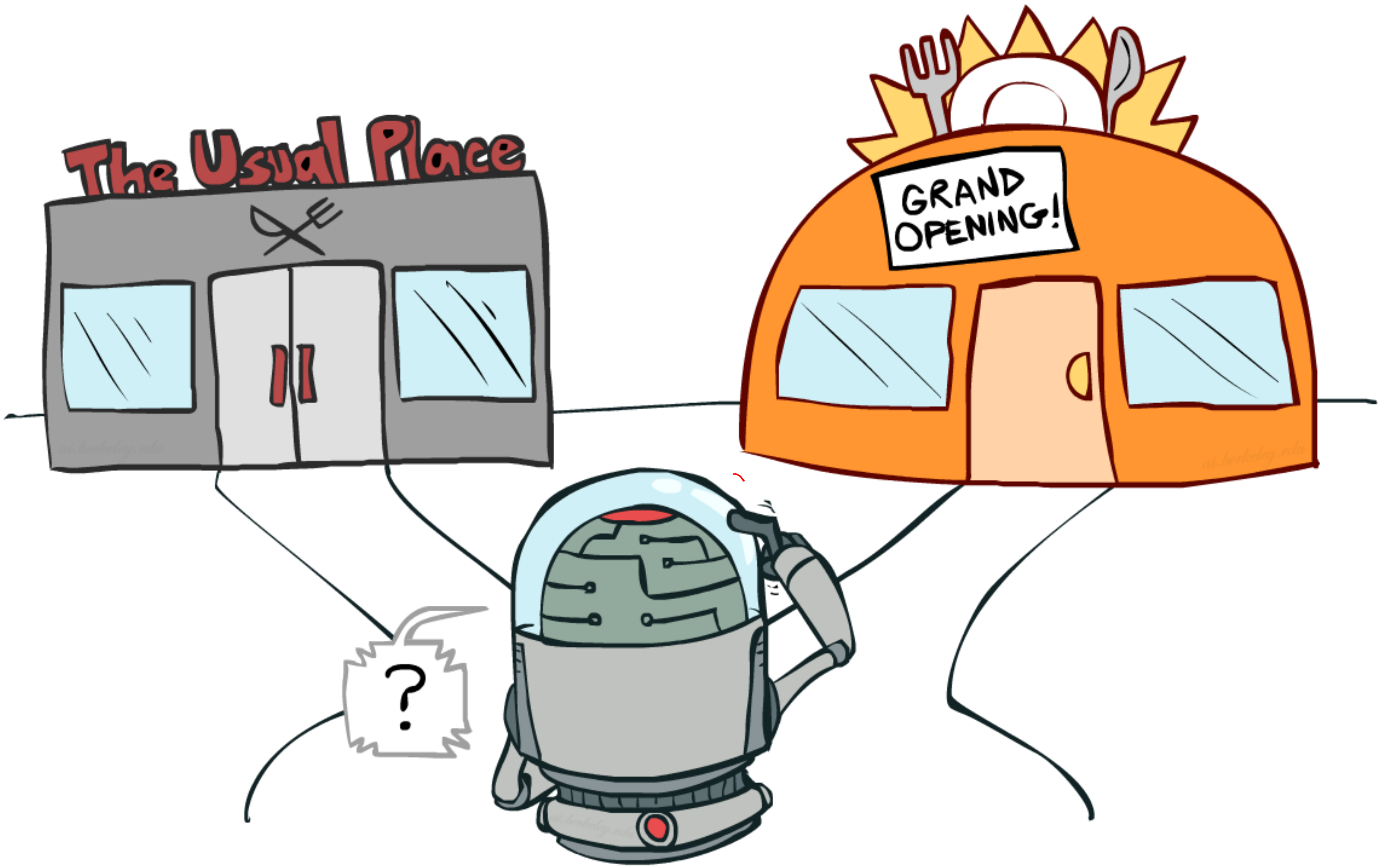




Progress of Q-learning methods for different values of α

- Implication: Let α decrease over time

Exploration vs Exploitation



Simple Approach: ϵ-Greedy

- With probability $1 - \epsilon$
  - Choose action $a = \underset{a'}{\operatorname{argmax}} \hat{Q}^*(s, a')$
- With probability ϵ
  - Select a random action

- For Q-learning
  - Guaranteed to compute optimal policy π^* based on $\hat{Q}^*(s, a)$ given enough samples with $\epsilon > 0$
  - However, the policy the agent is following is never the same as π^* (because it select a random action with probability ϵ)

Policy Gradient

- Key ideas
  - Parameterize the policy
  - Update the parameters towards the direction that increase the objective function (e.g., expected reward)

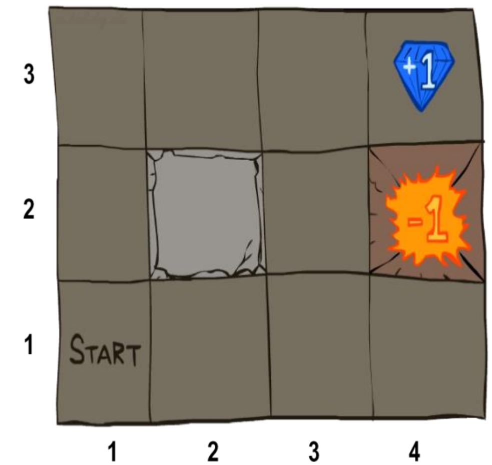
Similar to gradient ascent algorithm

Example

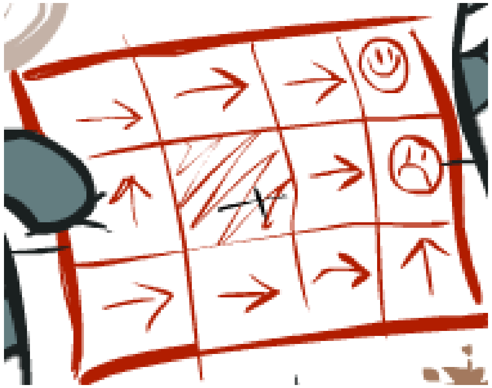
- $\pi = \pi_\theta$ where $\theta \in \mathbb{R}^4$
- $q_k = \dfrac{e^{\theta_k}}{\sum_j e^{\theta_j}}$
- Goal: Find θ that maximizes J_θ (now a standard optimization problem!) where J_θ is often chosen to be $J_\theta = \mathbb{E}_{s \sim \rho_{\pi_\theta}}[V^\pi(s)]$ where ρ_{π_θ} is the stationary distribution
- Policy gradient update (gradient ascent):

$$\theta^{i+1} \leftarrow \theta^i + \alpha \nabla_\theta J_\theta$$

Environment



Policy



Policy Gradient

- Challenge: hard to compute the gradient w.r.t. policy parameters due to uncertainties in MDPs
  - Finite difference methods
  - Policy gradient theorem

Policy Gradient – Policy Gradient Theorem

$$J(\theta) = \mathbb{E}[V^{\pi}(s)]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)]$$

- Estimate gradient through sampling
  - Sample possible histories
  - Compute gradient as average value of $Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)$
  - How to compute $Q^{\pi_\theta}(s,a)$?
    - REINFORCE: Directly use discounted reward from sampled history

REINFORCE

REINFORCE

Initialize θ arbitrarily
For each episode $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$
For $t = 1 \dots T-1$
$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$
(v_t is the discounted reward starting from time t)
Return θ

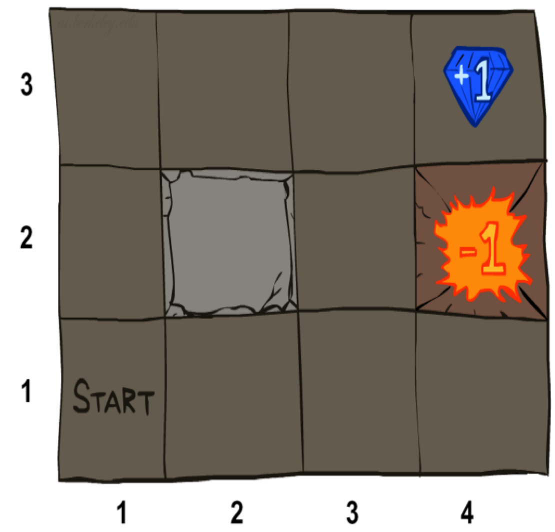
- Similar to stochastic gradient descent
- Use one sample to compute gradient and update parameters

Example

$$q_k = \frac{e^{\theta_k}}{\sum_j e^{\theta_j}}$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

Environment



Start at $(1{,}1)$

$s = (1{,}1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (2{,}1)$

$$\pi_\theta(s_1, a_1) = \frac{e^{\theta_R}}{e^{\theta_R} + e^{\theta_L} + e^{\theta_U} + e^{\theta_D}}$$

$$\frac{\partial \log \pi_\theta(s_1, a_1)}{\partial \theta_U} = \frac{e^{\theta_R} + e^{\theta_L} + e^{\theta_U} + e^{\theta_D}}{e^{\theta_R}} \times e^{\theta_R} \times (-1)$$

$$\frac{1}{(e^{\theta_R} + e^{\theta_L} + e^{\theta_U} + e^{\theta_D})^2} \times e^{\theta_U} = -\frac{1}{4}$$

$$v_t = -1.05$$

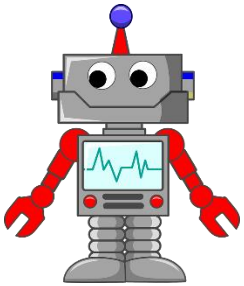
$$\theta_U = \theta_U + (0.5) \times \left(-\frac{1}{4}\right) \times (-1.05)$$

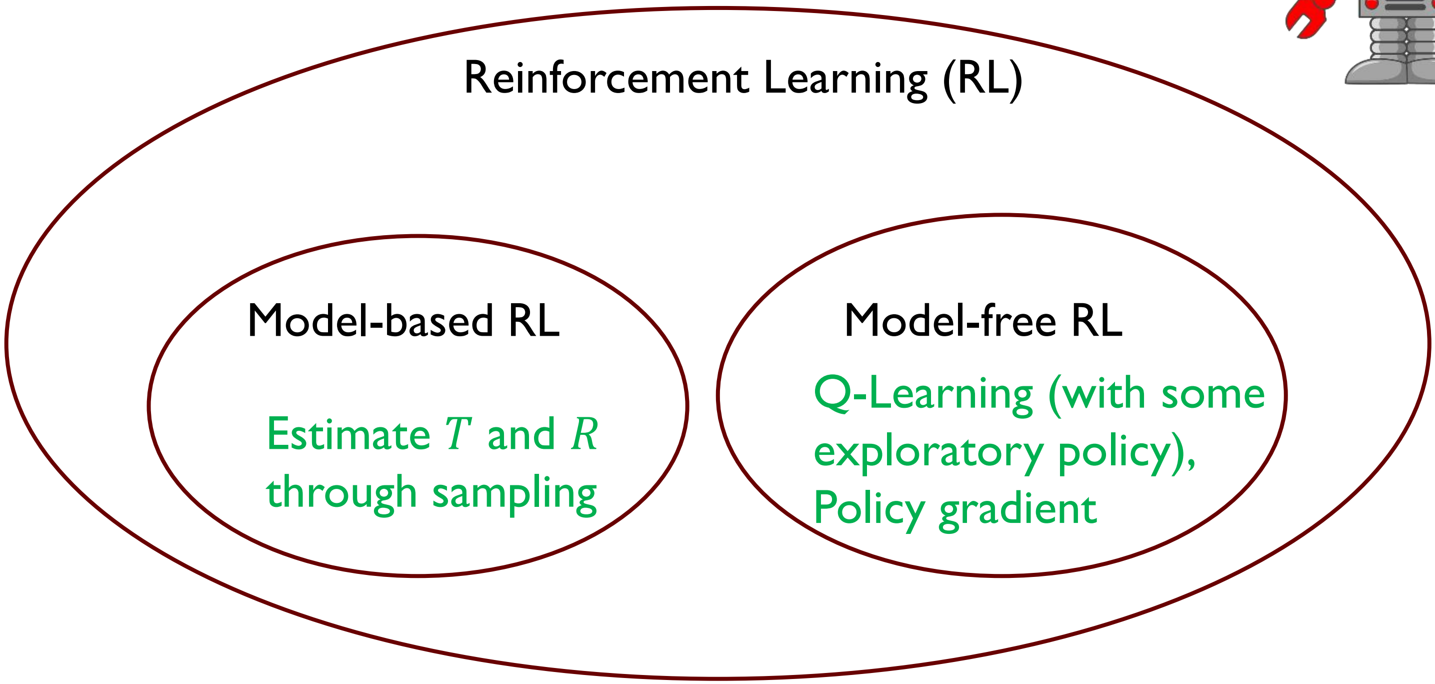
Reinforcement Learning in Practice

- Q-Learning and Proximal Policy Optimization (or PPO, a policy gradient-based approach) are commonly used

- Existing code packages:
  - OpenAI Gym: https://github.com/openai/gym
  - Rllib: https://docs.ray.io/en/latest/rllib/index.html

Summary





Reinforcement Learning (RL)
Model-based RL
Estimate T and R through sampling
Model-free RL
Q-Learning (with some exploratory policy), Policy gradient

References and Additional Resources

Other Resources

- http://courses.csail.mit.edu/6.825/fall05/rl_lecture/rl_examples.pdf
- http://www.cs.cmu.edu/afs/cs/academic/class/15780-s16/www/slides/rl.pdf
- http://incompleteideas.net/book/bookdraft2017nov5.pdf
- https://towardsdatascience.com/a-review-of-recent-reinforcment-learning-applications-to-healthcare-1f8357600407
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- https://arxiv.org/pdf/1312.5602.pdf

Acknowledgment

- The slides are prepared based on slides made by Patrick Virtue, Dave Touretzky, Chun Kai Ling, Tai Sing Lee and Zico Kolter, and some examples are borrowed from Meg Aycinena and Emma Brunskill

Backup Slides

Bellman Equation Explained

- Let $V_t^*(s)$ be the maximal expected total reward assuming
  - We begin in s
  - We have t time steps remaining

- Necessary condition

$$V_t^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V_{t-1}^*(s')]$$

$$V_0^* = 0$$

- Pick the action which maximizes *current* + *future reward* (assuming continued optimal behavior)

Bellman Equation Explained

$$V_t^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V_{t-1}^*(s')]$$

- Value function $V^*(s)$ can be viewed as $V_t^*(s)$ as $t \to \infty$

- Bellman Equation
  - $V^*(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^*(s')]$ Fixed point

Value Iteration

Value Iteration

Initialize $V_0^*(s) \leftarrow 0$

Typical termination condition: difference$< \epsilon$

Iterate $V_{i+1}^*(s) \leftarrow \max_{a\in A} [R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V_i^*(s')]$

Based on state-value Bellman Equations

$$V^*(s) = \max_{a\in A} [R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^*(s')]$$

- Consider the finite horizon view: Pretend we have a really long horizon
- aka 'Value update', 'Bellman backups/updates'
- Guaranteed to converge to $V^*(s)$

What is the optimal policy π^*?

Policy Iteration

- Not necessary to get an accurate estimate of V^* to induce π^*

- Policy iteration
  - Compute optimal policy π^* directly
  - Iterate between 2 steps
    - Policy Evaluation (check how good current policy is)
    - Policy Improvement (get a 'better' policy)

Policy Iteration

- Policy Evaluation
  - Method 1: Iterative approach
    - $V_{i+1}^{\pi}(s) \leftarrow R(s,\pi(s)) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V_i^{\pi}(s')\,, V_0^{\pi}(s) \leftarrow 0$
  - Method 2: Solve system of linear equations
    - $V^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s')$
- Policy Improvement
  - $\pi^{new}(s) \leftarrow \underset{a\in A}{\operatorname{argmax}} \left[R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s')\right]$
  - Note that when π is optimal, π^{new} is the same as π
- Policy iteration converges to the optimal policy in a finite number of steps
- Often converge faster than value iteration

Policy Gradient – Policy Gradient Theorem

- Basis: Given function $f(\cdot)$ and discrete-valued random variable $\boldsymbol{X} \sim p(\boldsymbol{x}|\boldsymbol{\theta})$

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X})] = \mathbb{E}_{\boldsymbol{X}}[\overbrace{\boxed{f(\boldsymbol{X}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{X}|\boldsymbol{\theta})}}^{g(X)}]$$

- Can be approximated by sampling X and compute average $g(X)$!

Policy Gradient – Policy Gradient Theorem

- Basis: Given function $f(\cdot)$ and discrete-valued random variable $\boldsymbol{X} \sim p(\boldsymbol{x}|\boldsymbol{\theta})$

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X})] = \mathbb{E}_{\boldsymbol{X}}[\overset{g(X)}{\boxed{f(\boldsymbol{X}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{X}|\boldsymbol{\theta})}}]$$

Derivation: $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X})] = \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{\theta}) f(\boldsymbol{x}) = \sum_{\boldsymbol{x}} f(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} p(\boldsymbol{x}|\boldsymbol{\theta})$

$$= \sum_{\boldsymbol{x}} f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{x}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{\theta})}$$

$$= \sum_{\boldsymbol{x}} f(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}|\boldsymbol{\theta})$$

$$= \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{X}|\boldsymbol{\theta})]$$

- Can be approximated by sampling X and compute average $g(X)$!

Policy Gradient – Policy Gradient Theorem

- Given $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X})] = \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{X}|\boldsymbol{\theta})]$, rewrite the gradient of the objective function $J(\theta)$ with respect to policy parameters

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{s\sim\pi_\theta}[V^\pi(s)] = \nabla_\theta \mathbb{E}_{s\sim\pi_\theta}\left[\sum_a \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(s,a)]$$

- Estimate gradient through sampling
  - Sample possible histories
  - Compute gradient as average value of $Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(s,a)$
  - How to compute $Q^{\pi_\theta}(s,a)$?
    - REINFORCE: Directly use discounted reward from sampled history

Example

$$q_k = \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} \qquad \theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

- Consider a trial

Start at $(1,1)$

$s = (1,1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (2,1)$

$s = (2,1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (2,1)$

$s = (2,1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (3,1)$

$s = (3,1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (4,1)$

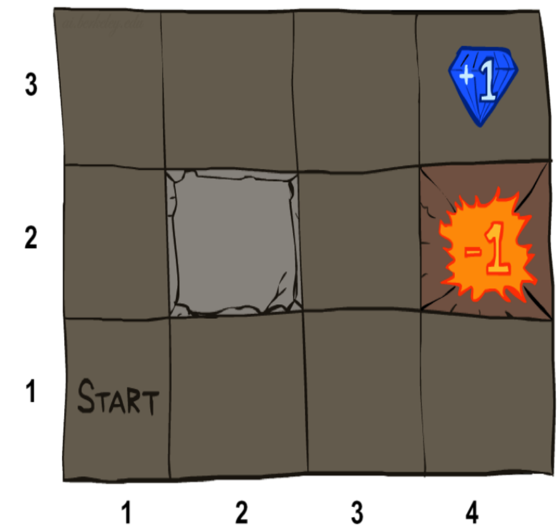
$s = (4,1)$ action=tup (try going up)

Reward= -0.01; End up at $s' = (4,2)$

$s = (4,2)$ No action available. Reward= -1; Terminate

$\gamma = 1, \alpha = 0.5, \theta$ is initialized to 0, how to update θ?

Environment



Example

$$q_k = \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} \qquad \theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

Start at $(1,1)$

$s = (1,1)$ action=tright (try going right)

Reward= -0.01; End up at $s' = (2,1)$

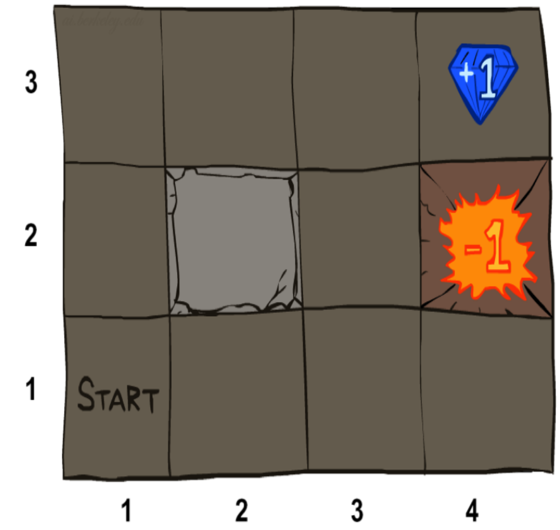
$\pi_\theta(s_1, a_1) =$

$\frac{\partial \log \pi_\theta(s_1, a_1)}{\partial \theta_U} =$

$v_t =$

$\theta_U = \theta_U +$

Environment



Discussion: How is it different for the update of θ_R?