10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Reinforcement Learning:
Value Iteration
+ Q-Learning

Matt Gormley
Lecture 23
Apr. 11, 2022

Reminders

- Homework 7: HMMs
  - Out: Fri, Apr. 1
  - Due: Tue, Apr. 12 at 11:59pm
  - (Re-released handout on Monday.)
- Homework 8: Reinforcement Learning
  - Out: Tue, Apr. 12
  - Due: Thu, Apr. 21 at 11:59pm

VALUE ITERATION

Definitions for Value Iteration

Whiteboard

- State trajectory
- Value function
- Bellman equations
- Optimal policy
- Optimal value function
- Computing the optimal policy
- Ex: Path Planning

RL: Optimal Value Function & Policy

- Optimal value function:

$$V^*(s) = \max_{a \in \mathcal{A}} R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) V^*(s')$$

  - System of $|\mathcal{S}|$ equations and $|\mathcal{S}|$ variables

- Optimal policy:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \underbrace{R(s,a)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) V^*(s')}_{\text{(Discounted) Future reward}}$$

RL Terminology

Question: Match each term (on the left) to the corresponding statement or definition (on the right)

Terms:

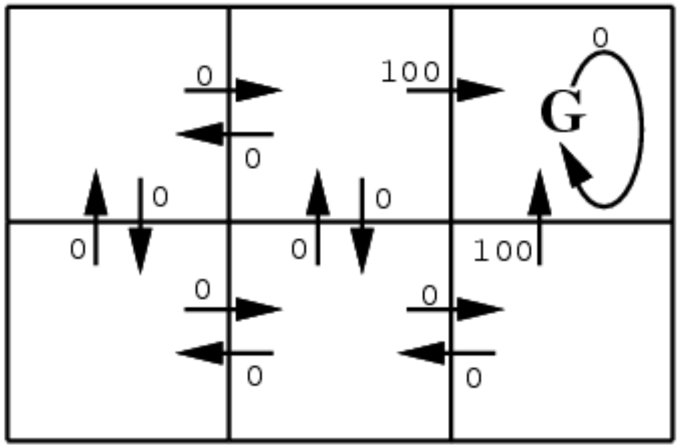
A. a reward function
B. a transition probability
C. a policy
D. state/action/reward triples
E. a value function
F. transition function
G. an optimal policy
H. Matt's favorite statement

Statements:

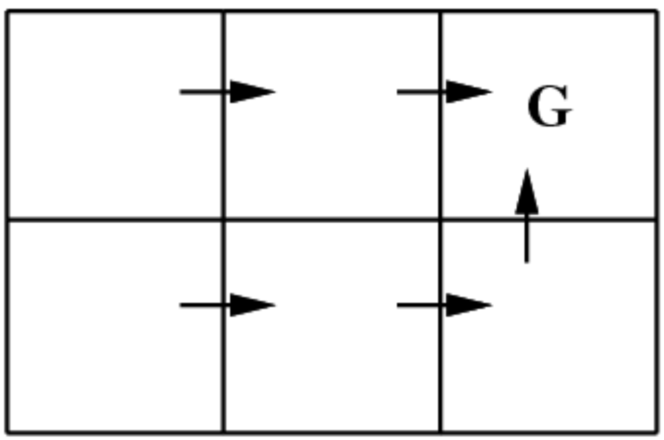
1. gives the expected future discounted reward of a state
2. maps from states to actions
3. quantifies immediate success of agent
4. is a deterministic map from state/action pairs to states
5. quantifies the likelihood of landing a new state, given a state/action pair
6. is the desired output of an RL algorithm
7. can be influenced by trading off between exploitation/exploration

Example: Path Planning

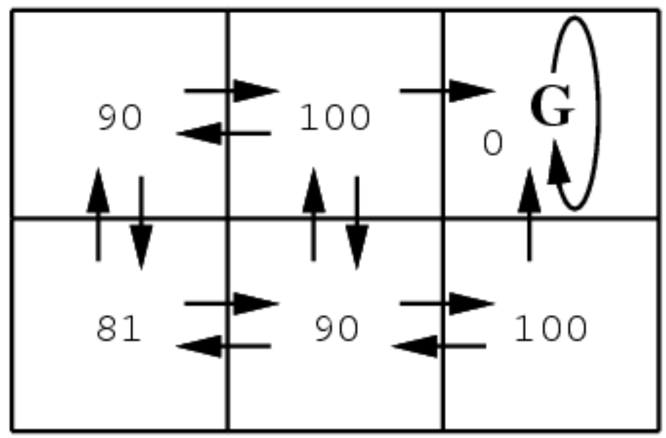
Example: Robot Localization



$r(s, a)$ (immediate reward) values



One optimal policy



$V^*(s)$ values

Figure from Tom Mitchell

Value Iteration

Whiteboard

– Value Iteration Algorithm

Value Iteration

Algorithm 1 Value Iteration

procedure ValueIteration($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
  Initialize value function $V(s) = 0$ or randomly
  while not converged **do**
    for $s \in \mathcal{S}$ **do**
      $V(s) = \max_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s')$
  Let $\pi(s) = \text{argmax}_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s'), \ \forall s$
  return π

Variant 1: without Q(s,a) table

Value Iteration

Algorithm 1 Value Iteration

procedure VALUEITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
    Initialize value function $V(s) = 0$ or randomly
    while not converged **do**
        for $s \in \mathcal{S}$ **do**
            for $a \in \mathcal{A}$ **do**
                $Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s')$
            $V(s) = \max_a Q(s,a)$
    Let $\pi(s) = \mathrm{argmax}_a\, Q(s,a),\ \forall s$
    return π

Variant 2: with Q(s,a) table

Synchronous vs. Asynchronous Value Iteration

Algorithm 1 Asynchronous Value Iteration

procedure ASYNCHRONOUSVALUEITERATION($R(s,a)$, $p(\cdot|s,a)$)
  Initialize value function $V(s) = 0$ or randomly
  while not converged **do**
    for $s \in \mathcal{S}$ **do**
      $V(s) = \max_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s')$
  Let $\pi(s) = \text{argmax}_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s'), \ \forall s$
  return π

asynchronous updates: compute and update V(s) for each state one at a time

Algorithm 1 Synchronous Value Iteration

procedure SYNCHRONOUSVALUEITERATION($R(s,a)$, $p(\cdot|s,a)$)
  Initialize value function $V(s)^{(0)} = 0$ or randomly
  $t = 0$
  while not converged **do**
    for $s \in \mathcal{S}$ **do**
      $V(s)^{(t+1)} = \max_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s')^{(t)}$
    $t = t + 1$
  Let $\pi(s) = \text{argmax}_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s'), \ \forall s$
  return π

synchronous updates: compute all the fresh values of V(s) from all the stale values of V(s), then update V(s) with fresh values

Value Iteration Convergence

very abridged

Theorem 1 (Bertsekas (1989))
V converges to V^*, if each state is visited infinitely often

Theorem 2 (Williams & Baird (1993))
if $max_s|V^{t+1}(s) - V^t(s)| < \epsilon$
then $max_s|V^{t+1}(s) - V^*(s)| < \frac{2\epsilon\gamma}{1-\gamma}, \forall s$

Theorem 3 (Bertsekas (1987))
greedy policy will be optimal in a finite number of steps (even if not converged to optimal value function!)

Holds for both asynchronous and sychronous updates

Provides reasonable stopping criterion for value iteration

Often greedy policy converges well before the value function

Value Iteration Variants

Question:
True or False: The value iteration algorithm shown below is an example of **synchronous** updates

Algorithm 1 Value Iteration

procedure VALUEITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
    Initialize value function $V(s) = 0$ or randomly
    while not converged **do**
        for $s \in \mathcal{S}$ **do**
            for $a \in \mathcal{A}$ **do**
                $Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s')$
            $V(s) = \max_a Q(s,a)$
    Let $\pi(s) = \text{argmax}_a Q(s,a), \ \forall s$
    return π

POLICY ITERATION

Policy Iteration

Algorithm 1 Policy Iteration

procedure POLICYITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)

Initialize policy π randomly

while not converged **do**

Solve Bellman equations for fixed policy π

$$V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,\pi(s)) V^\pi(s'),\ \forall s$$

Improve policy π using new value function

$$\pi(s) = \operatorname*{argmax}_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^\pi(s')$$

return π

Policy Iteration

Algorithm 1 Policy Iteration

procedure POLICYITERATION($R(s, a$ … n, $p(\cdot|s, a)$ transition probabilities)

Initialize policy π randomly

while not converged **do**

Solve Bellman equations for fixed policy π

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^{\pi}(s'), \ \forall s$$

Improve policy π using new value function

$$\pi(s) = \operatorname*{argmax}_a R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi}(s')$$

return π

Compute value function for fixed policy is easy

System of |S| equations and |S| variables

Greedy policy w.r.t. current value function

Greedy policy might **remain the same** for a particular state if there is no better action

Policy Iteration Convergence

In-Class Exercise:

How many policies are there for a finite sized state and action space?

In-Class Exercise:

Suppose policy iteration is shown to improve the policy at every iteration. Can you bound the number of iterations it will take to converge? If yes, what is the bound? If no, why not?

Value Iteration vs. Policy Iteration

- Value iteration requires O(|A| |S|^2) computation per iteration

- Policy iteration requires O(|A| |S|^2 + |S|^3) computation per iteration
- In practice, policy iteration converges in fewer iterations

Algorithm 1 Value Iteration

procedure VALUEITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
    Initialize value function $V(s) = 0$ or randomly
    while not converged **do**
        for $s \in \mathcal{S}$ **do**
            $V(s) = \max_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s')$
    Let $\pi(s) = \text{argmax}_a R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s'),\ \forall s$
    return π

Algorithm 1 Policy Iteration

procedure POLICYITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
    Initialize policy π randomly
    while not converged **do**
        Solve Bellman equations for fixed policy π

$$V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,\pi(s)) V^\pi(s'),\ \forall s$$

        Improve policy π using new value function

$$\pi(s) = \underset{a}{\text{argmax}}\, R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^\pi(s')$$

    return π

Learning Objectives

Reinforcement Learning: Value and Policy Iteration

You should be able to…

1. Compare the reinforcement learning paradigm to other learning paradigms
2. Cast a real-world problem as a Markov Decision Process
3. Depict the exploration vs. exploitation tradeoff via MDP examples
4. Explain how to solve a system of equations using fixed point iteration
5. Define the Bellman Equations
6. Show how to compute the optimal policy in terms of the optimal value function
7. Explain the relationship between a value function mapping states to expected rewards and a value function mapping state-action pairs to expected rewards
8. Implement value iteration
9. Implement policy iteration
10. Contrast the computational complexity and empirical convergence of value iteration vs. policy iteration
11. Identify the conditions under which the value iteration algorithm will converge to the true value function
12. Describe properties of the policy iteration algorithm

Q-LEARNING

Q:

What can we do if we don't know the reward function / transition probabilities?

Today’s lecture is brought to you by the letter Q

Source: https://en.wikipedia.org/wiki/Avenue_Q#/media/File:Image-AvenueQlogo.png

Today’s lecture is brought to you by the letter Q

Source: https://vignette1.wikia.nocookie.net/jamesbond/images/9/9a/The_Four_Qs_-_Profile_(2).png/revision/latest?cb=20121102195112

Today's lecture is brought to you by the letter Q

Source: https://www.npr.org/2017/06/03/531044118/there-may-not-be-flying-but-quidditch-still-creates-magic

Value Iteration

Algorithm 1 Value Iteration

procedure VALUEITERATION($R(s,a)$ reward function, $p(\cdot|s,a)$ transition probabilities)
  Initialize value function $V(s) = 0$ or randomly
  while not converged **do**
    for $s \in \mathcal{S}$ **do**
      for $a \in \mathcal{A}$ **do**
        $Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s')$
      $V(s) = \max_a Q(s,a)$
  Let $\pi(s) = \text{argmax}_a Q(s,a), \ \forall s$
  return π

Variant 1: with Q(s,a) table

$Q^*(s,a)$

- $Q^*(s,a) = \mathbb{E}[$total discounted reward of taking action a in state s, assuming all future actions are optimal$]$

$$= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) V^*(s')$$

- $V^*(s') = \max_{a' \in \mathcal{A}} Q^*(s', a')$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \left[\max_{a' \in \mathcal{A}} Q^*(s', a') \right]$$

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

- Insight: if we know Q^*, we can compute an optimal policy π^*!

$Q^*(s,a)$ w/deterministic transitions

- $Q^*(s,a) = \mathbb{E}[$total discounted reward of taking action a in state s, assuming all future actions are optimal$]$

$$= R(s,a) + \gamma V^*\big(\delta(s,a)\big)$$

- $V^*\big(\delta(s,a)\big) = \max_{a' \in \mathcal{A}} Q^*(\delta(s,a), a')$

$$Q^*(s,a) = R(s,a) + \gamma \max_{a' \in \mathcal{A}} Q^*(\delta(s,a), a')$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a)$$

- Insight: if we know Q^*, we can compute an optimal policy π^*!

Q-Learning

Whiteboard

- Q-Learning Algorithm
  - Case 1: Deterministic Environment
  - Case 2: Nondeterministic Environment
- ϵ-greedy Strategy