10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Reinforcement Learning

Matt Gormley
Lecture 23
Nov. 16, 2018

Reminders

- Homework 7: HMMs
  - Out: Wed, Nov 7
  - Due: Mon, Nov 19 at 11:59pm
- Schedule Changes
  - Recitation on Mon, Nov 26
  - Lecture on Fri, Nov 30
  - Recitation on Wed, Dec 5
- Peer Tutoring
  - Sign up by Mon, Nov 19

Q&A

MARKOV DECISION PROCESSES

Markov Decision Process

Whiteboard

- Components: states, actions, state transition probabilities, reward function
- Markovian assumption
- MDP Model
- MDP Goal: Infinite-horizon Discounted Reward
- deterministic vs. nondeterministic MDP
- deterministic vs. stochastic policy

Exploration vs. Exploitation

Whiteboard

- Explore vs. Exploit Tradeoff
- Ex: k-Armed Bandits
- Ex: Traversing a Maze

FIXED POINT ITERATION

Fixed Point Iteration for Optimization

- Fixed point iteration is a general tool for solving systems of equations
- It can also be applied to optimization.

$$J(\boldsymbol{\theta})$$

$$\frac{dJ(\boldsymbol{\theta})}{d\theta_i} = 0 = f(\boldsymbol{\theta})$$

$$0 = f(\boldsymbol{\theta}) \Rightarrow \theta_i = g(\boldsymbol{\theta})$$

$$\theta_i^{(t+1)} = g(\boldsymbol{\theta}^{(t)})$$

1. Given objective function:
2. Compute derivative, set to zero (call this function f).
3. Rearrange the equation s.t. one of parameters appears on the LHS.
4. Initialize the parameters.
5. For i in $\{1,\ldots,K\}$, update each parameter and increment t:
6. Repeat #5 until convergence

Fixed Point Iteration for Optimization

- Fixed point iteration is a general tool for solving systems of equations
- It can also be applied to optimization.

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

1. Given objective function:
2. Compute derivative, set to zero (call this function f).
3. Rearrange the equation s.t. one of parameters appears on the LHS.
4. Initialize the parameters.
5. For i in $\{1,\ldots,K\}$, update each parameter and increment t:
6. Repeat #5 until convergence

Fixed Point Iteration for Optimization

We can implement our example in a few lines of python.

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2+2}{3} = g(x)$$

$$x \leftarrow \frac{x^2+2}{3}$$

```python
def f1(x):
    '''f(x) = x^2 - 3x + 2'''
    return x**2 - 3.*x + 2.

def g1(x):
    '''g(x) = \frac{x^2 + 2}{3}'''
    return (x**2 + 2.) / 3.

def fpi(g, x0, n, f):
    '''Optimizes the 1D function g by fixed point iteration
    starting at x0 and stopping after n iterations. Also
    includes an auxiliary function f to test at each value.'''
    x = x0
    for i in range(n):
        print("i=%2d x=%.4f f(x)=%.4f" % (i, x, f(x)))
        x = g(x)
    i += 1
    print("i=%2d x=%.4f f(x)=%.4f" % (i, x, f(x)))
    return x

if __name__ == "__main__":
    x = fpi(g1, 0, 20, f1)
```

Fixed Point Iteration for Optimization

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

```
$ python fixed-point-iteration.py
i= 0 x=0.0000 f(x)=2.0000
i= 1 x=0.6667 f(x)=0.4444
i= 2 x=0.8148 f(x)=0.2195
i= 3 x=0.8880 f(x)=0.1246
i= 4 x=0.9295 f(x)=0.0755
i= 5 x=0.9547 f(x)=0.0474
i= 6 x=0.9705 f(x)=0.0304
i= 7 x=0.9806 f(x)=0.0198
i= 8 x=0.9872 f(x)=0.0130
i= 9 x=0.9915 f(x)=0.0086
i=10 x=0.9944 f(x)=0.0057
i=11 x=0.9963 f(x)=0.0038
i=12 x=0.9975 f(x)=0.0025
i=13 x=0.9983 f(x)=0.0017
i=14 x=0.9989 f(x)=0.0011
i=15 x=0.9993 f(x)=0.0007
i=16 x=0.9995 f(x)=0.0005
i=17 x=0.9997 f(x)=0.0003
i=18 x=0.9998 f(x)=0.0002
i=19 x=0.9999 f(x)=0.0001
i=20 x=0.9999 f(x)=0.0001
```

VALUE ITERATION

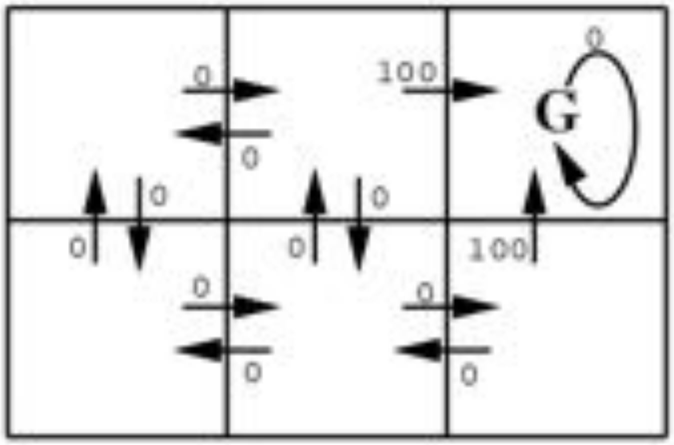
Definitions for Value Iteration

Whiteboard

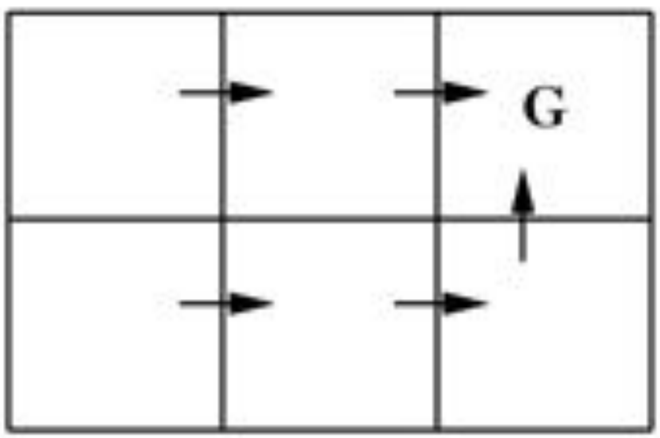
- State trajectory
- Value function
- Bellman equations
- Optimal policy
- Optimal value function
- Computing the optimal policy
- Ex: Path Planning

Example: Path Planning

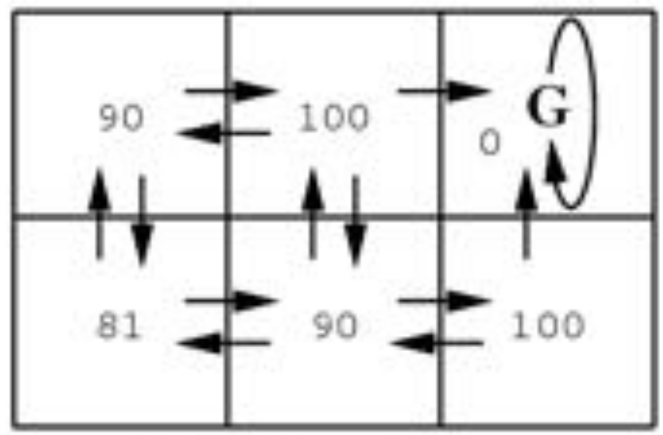
Example: Robot Localization



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



One optimal policy



$V^*(s)$ values

Figure from Tom Mitchell

Value Iteration

Whiteboard

- Value Iteration Algorithm
- Synchronous vs. Asychronous Updates
- Convergence Properties

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 $\pi(s) = \operatorname{argmax}_a Q(s,a), \ \forall s$
  return π

Policy Iteration

Whiteboard

– Policy Iteration Algorithm
– Solving the Bellman Equations for Fixed Policy
– Convergence Properties
– Value Iteration vs. Policy Iteration

Policy Iteration

Algorithm 1 Policy Iteration

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

2: Initialize policy π randomly

3: **while** not converged **do**

4: Solve Bellman equations for fixed policy π

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

5: Improve policy π using new value function

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

6: **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?

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**
      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

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