CS 188: Artificial Intelligence
Spring 2010

Lecture 11: Reinforcement Learning
2/23/2010

Pieter Abbeel – UC Berkeley

Many slides over the course adapted from either Dan Klein, Stuart Russell or Andrew Moore

Announcements

- P0 / P1 / W1 / W2 in glookup
  - If you have no entry, etc, email staff list!
  - If you have questions, see one of us or email list.
  - W1, W2: can be picked up from 188 return box in 283 Soda
- W3: Utilities --- Due Thursday.
- Recall: readings for current material
  - Online book: Sutton and Barto

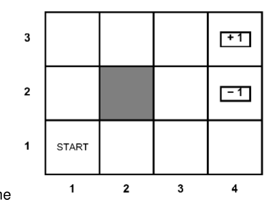
    http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html

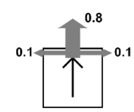
MDPs recap

- Markov decision processes:
  - States S
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s') (and discount γ)
  - Start state s_0
- Solution methods:
  - Value iteration (VI)
  - Policy iteration (PI)
  - Asynchronous value iteration
- Current limitations:
  - Relatively small state spaces
  - Assumes T and R are known

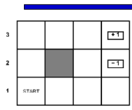
MDP Example: Grid World

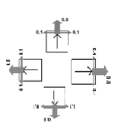
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Rewards come at the end
- Goal: maximize sum of rewards





MDP Example: Grid World





MDP = (S, A, T, R, s_0, γ)

Set of states S

Set of actions A

Transition model T

Initial state s_0

Discount factor γ

Value Iteration

- Idea:
  - $V_i(s)$: the expected discounted sum of rewards accumulated when starting from state s and acting optimally for a horizon of i time steps.
  - Start with $V_0(s) = 0$, which we know is right (why?)
  - Given V_i, calculate the values for all states for horizon i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

  - This is called a value update or Bellman update
  - Repeat until convergence
- Theorem: will converge to unique optimal values
  - Basic idea: approximations get refined towards optimal values
  - Policy may converge long before values do

Complete procedure

1. Run value iteration (off-line)

   Returns V, which (assuming sufficiently many iterations is a good approximation of V*)

2. Agent acts.

   At time t the agent is in state s_t and takes the action a_t:

$$\arg\max_a \sum_{s'} T(s_t, a, s')[R(s_t, a, s') + \gamma V(s')]$$

8

MDPs recap

- Markov decision processes:
  - States S
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s') (and discount γ)
  - Start state s_0
- Solution methods:
  - Value iteration (VI)
  - Policy iteration (PI)
  - Asynchronous value iteration
- Current limitations:
  - Assumes T and R are known
  - Relatively small state spaces

9

Reinforcement Learning

- Reinforcement learning:
  - Still assume an MDP:
    - A set of states s ∈ S
    - A set of actions (per state) A
    - A model T(s,a,s')
    - A reward function R(s,a,s')
  - Still looking for a policy π(s)
  - New twist: don't know T or R
    - I.e. don't know which states are good or what the actions do
    - Must actually try actions and states out to learn

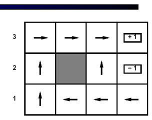
10

Example: learning to walk

Before learning (hand-tuned) | One of many learning runs | After learning [After 1000 field traversals]

[Kohl and Stone, ICRA 2004]

Passive Learning

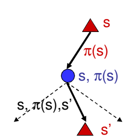


- Simplified task
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You are given a policy π(s)
  - Goal: learn the state values
  - … what policy evaluation did
- In this case:
  - Learner "along for the ride"
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - We'll get to the active case soon
  - This is NOT offline planning! You actually take actions in the world and see what happens…

12

Recap: Model-Based Policy Evaluation

- Simplified Bellman updates to calculate V for a fixed policy:
  - New V is expected one-step-look-ahead using current V
  - Unfortunately, need T and R



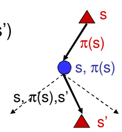
$$V_0^\pi(s) = 0$$

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

13

Model-Based Learning

- Idea:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct
- Simple empirical model learning
  - Count outcomes for each s,a
  - Normalize to give estimate of **T(s,a,s')**
  - Discover **R(s,a,s')** when we experience (s,a,s')
- Solving the MDP with the learned model
  - Iterative policy evaluation, for example



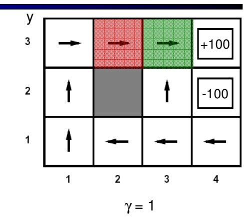
$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^{\pi}(s')]$$

14

Example: Model-Based Learning

- Episodes:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



$\gamma = 1$

T(<3,3>, right, <4,3>) = 1 / 3

T(<2,3>, right, <3,3>) = 2 / 2

15

Model-Free Learning

- Want to compute an expectation weighted by P(x):

$$E[f(x)] = \sum_x P(x)f(x)$$

- Model-based: estimate P(x) from samples, compute expectation

$$x_i \sim P(x)$$

$$\hat{P}(x) = \text{count}(x)/k$$

$$E[f(x)] \approx \sum_x \hat{P}(x)f(x)$$

- Model-free: estimate expectation directly from samples

$$x_i \sim P(x) \qquad E[f(x)] \approx \frac{1}{k}\sum_i f(x_i)$$

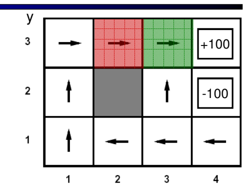
- Why does this work? Because samples appear with the right frequencies!

16

Example: Direct Estimation

- Episodes:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



$\gamma = 1$, R = -1

V(2,3) ~ (96 + -103) / 2 = -3.5

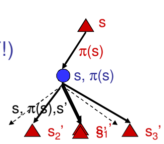
V(3,3) ~ (99 + 97 + -102) / 3 = 31.3

18

Sample-Based Policy Evaluation?

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

- Who needs T and R? Approximate the expectation with samples (drawn from T!)



$$sample_1 = R(s,\pi(s),s_1') + \gamma V_i^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_i^{\pi}(s_2')$$
$$\cdots$$
$$sample_k = R(s,\pi(s),s_k') + \gamma V_i^{\pi}(s_k')$$

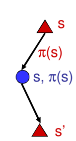
$$V_{i+1}^{\pi}(s) \leftarrow \frac{1}{k}\sum_i sample_i$$

Almost! But we only actually make progress when we move to i+1.

19

Temporal-Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience (s,a,s',r)
  - Likely s' will contribute updates more often
- Temporal difference learning
  - Policy still fixed!
  - Move values toward value of whatever successor occurs: running average!



Sample of V(s): $sample = R(s,\pi(s),s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

20

Exponential Moving Average

- Exponential moving average
  - Makes recent samples more important

$$\bar{x}_n = \frac{x_n + (1-\alpha)\cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

  - Forgets about the past (distant past values were wrong anyway)
  - Easy to compute from the running average

$$\bar{x}_n = (1-\alpha)\cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

- Decreasing learning rate can give converging averages

21

Policy evaluation when T (and R) unknown --- recap

- Model-based:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct
- Model-free:
  - Direct estimation:
    - V(s) = sample estimate of sum of rewards accumulated from state s onwards
  - Temporal difference (TD) value learning:
    - Move values toward value of whatever successor occurs: running average!

$$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$
$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha) sample$$

23