CS 4649/7649
Robot Intelligence: Planning

RL

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

*Slides based in part on Dr. Mike Stilman and Dr. Pieter Abbeel's slides

Administrative– Final Project

• CS7649
- project proposal: Due Oct. 30 (email a pdf file to me and Saul)
- project final report: Due Dec. 4, 23:59pm, conference-style paper
- project presentation: Dec. 11, 11:30am - 2:20pm

• CS4649
- project reviewer assignment: Oct. 28 (2 ~ 3 reviewers/project)
- proposal review report: Due Nov. 6
- project review report(for the assigned project): Due Dec. 11, 11:30am
- project presentation review*(for all presentation): Due Dec. 11, 2:20pm

*presentation review sheets will be provided

MDP with unknown models

● Reinforcement Learning

- Model-based Learning

: Learn the model first, then solve the (approx.) MDP with VI or PI

- Model-free Learning

: Direct Evaluation [performs policy evaluation]

: Temporal Difference Learning [performs policy evaluation]

: Q-Learning [learns optimal state-action value function Q*]

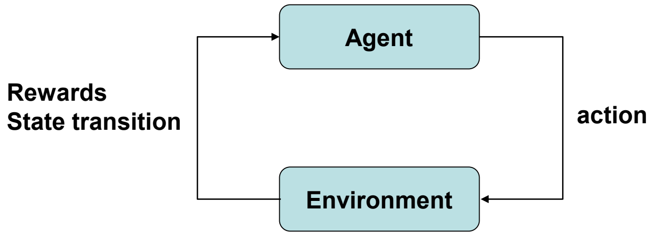
: Policy search [learns optimal policy from subset of all policies]

...

Reinforcement Learning

● Idea

- Receive feedback in the form of rewards
- Agent's is defined by the reward function utility(e.g. average/accumulated sum of the rewards)
- Must (learn to) act so as to maximize expected rewards
- Learning is based on observed samples of outcomes



Machine Learning

- Supervised Learning
 - The most common machine learning category
 - Trying to map some data points to some function(or function approximation) that best approximates the data.

- Unsupervised Learning
 - Analyzing data without any sort of function to map to. Figuring out what the data is w/o any feedback
 - Unsupervised in the sense that the algorithm doesn't know what the output should be. Instead, the algorithm has to come up with it itself.

- Reinforcement Learning
 - Figuring out how to play a multistage game with rewards and payoffs to optimize the life of the agent
 - Similar to supervised learning, but with reward.

RL examples: Inverted Pendulum

http://www.youtube.com/watch?v=b1c0N_Fs9wc&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e&index=9

RL examples: Helicopter Flying

http://www.youtube.com/watch?v=M-QUkgk3HyE&index=4&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e

Reinforcement Learning

● Markov Decision Process
- A set of states $s \in S$
- A set of actions (per state) A
- A transition model T(s'|s,a)
- A reward function R(s,a,s')

● Looking for a policy for MDP, but don't know T and/or R
- Don't know what the actions do and/or which states are good

● Reinforcement Learning – MDP with T and/or R unknown
- Model-based learning
- Model-free learning
  : Direct evaluation (performs policy evaluation)
  : Temporal difference learning (performs policy evaluation)
  : Q-Learning (learns optimal state-action value function Q)
  : …

Model-based Learning

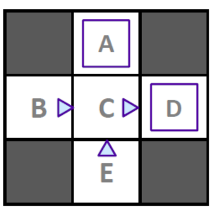
- Idea:
  - -Step 1: Learn the model empirically through experience
  - -Step 2: Solve for policy/values as if the learned model were correct

- Step 1: Empirical model learning
  - -Count outcomes s' for each s,a
  - -Normalize to give an estimate of T(s'|s, a)
  - -Discover an estimate of R(s,a,s') when we experience (s,a,s')

- Step 2: Solving the MDP with the learned model
  - -Value iteration, or policy iteration, as before

Model Learning Example

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3
E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$\hat{T}(s'|s,a)$
T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
...

$\hat{R}(s,a,s')$
R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
...

http://www.cs.berkeley.edu/~pabbeel/

Model-based vs Model-free

Goal: Compute expected age of **CS4649/7649 students**

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \qquad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

http://www.cs.berkeley.edu/~pabbeel/

Learning the Model in MBL

Estimate P(s) from samples

-Samples $x_i \sim P(x)$

-Estimate $\hat{P}(x) = num(x)/N, \text{ where } N = \text{total number of samples}$

Estimate P(s' | s,a) from samples

-Samples $s_0, a_0, s_1, a_1, \ldots$

-Estimate $\hat{P}(s'|s,a) = \dfrac{num(s_{t+1} = s', a_t = a, s_t = s)}{num(s_t = s, a_t = a)}$

Why does this work? B/C samples appear with the right frequencies!

MBL vs MFL

- Model-based RL

-First act in MDP and learn T/R

-Then value iteration or policy iteration with learned T/R

-Advantage: efficient use of data

-Disadvantage: need sufficient date/requires building a model for T/R

- Model-free RL

-Bypass the need to learn T/R

-Methods to evaluate V^{π}, the value function for a fixed policy π without knowing T, R:

(i) Direct Evaluation

(ii) Temporal Difference Learning

-Method to learn π^*, Q^*, V^* without knowing T, R

(iii) Q-Learning

RL examples: Table Tennis

http://www.youtube.com/watch?v=SH3bADiB7uQ&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e&index=2

MFL

- 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

$$x_i \sim P(x) \quad \hat{P}(x) = num(x)/N \quad E[f(x)] \approx \sum_x \hat{P}(x) f(x)$$

- Model-free: estimate expectation directly from samples

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

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

MFL: Direct Evaluation

- Goal: Compute values for each state under π

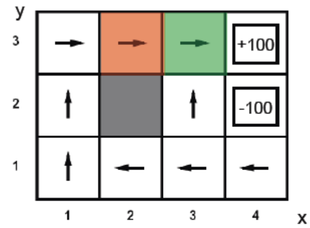
- Idea: Average together observed sample values
  - Act according to π
  - Every time you visit a state, write down what the sum of discounted rewards accumulate from state s onwards
  - Average those samples

Direct Evaluation Example

Observed Episodes (Training)

Episode 1	Episode 2
(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)	

Input Policy π



$\gamma = 1$, R = -1

Output Values

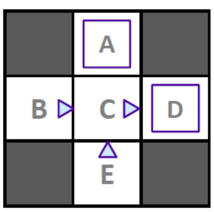
V(2,3) ~ (96 + -103) / 2 = -3.5

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

http://www.cs.berkeley.edu/~pabbeel/

Direct Evaluation Example

Input Policy π



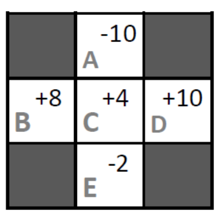
Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1	Episode 2
B, east, C, -1 C, east, D, -1 D, exit, x, +10	B, east, C, -1 C, east, D, -1 D, exit, x, +10

Episode 3	Episode 4
E, north, C, -1 C, east, D, -1 D, exit, x, +10	E, north, C, -1 C, east, A, -1 A, exit, x, -10

Output Values



http://www.cs.berkeley.edu/~pabbeel/

MFL: Direct Evaluation

- What is good about DE?
  - It's easy to understand
  - It doesn't require any knowledge of T, R
  - It eventually computes the correct average values, using just sample transitions
- What is bad about DE?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn

RL examples: Pancake Flipping

http://www.youtube.com/watch?v=W_gxLKSsSIE&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e&index=1

Why Not Use Policy Evaluation?

Simplified Bellman updates calculate V for a fixed policy:
Each round, replace V with a one-step-look-ahead layer over V

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

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

This approach fully exploited the connections between the states
Unfortunately, we need T and R to do it!

Key question: how can we do this update to V without knowing T and R?
In other words, how do we take a weighted average without knowing the weights?

Sample-based Policy Evaluation?

We want to improve our estimate of V by computing these averages

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

Take samples of outcomes s' (by doing the action!)
and compute the average:

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

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

Temporal-Difference Learning

- Idea: learn from every experience!
- Update V(s) each time we experience a transition (s, a, s', r)
- Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
- Policy still fixed, still doing evaluation!
- 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: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

Temporal-Difference Learning

- Idea: learn from every experience! Over time, updates will mimic Bellman's update!
- Update V(s) each time we experience a transition (s, a, s', r)
- Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
- Policy still fixed, still doing evaluation!
- 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: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

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

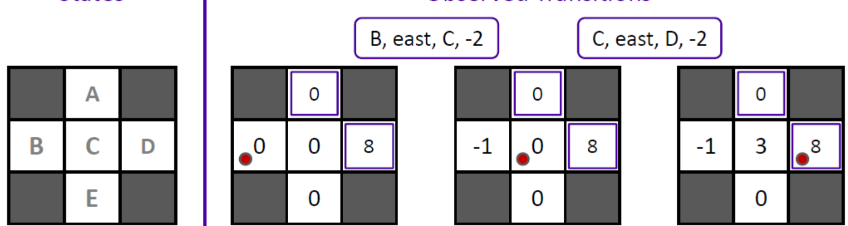
$$\bar{x}_n = (1-\alpha)\cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

Decreasing learning rate(α) can give converging averages

TD Learning Example

States

Observed Transitions



Assume: $\gamma = 1$, $\alpha = 1/2$

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

http://www.cs.berkeley.edu/~pabbeel/

Interim Summary

Model-based:
- Learn the model empirically through experience
- Solve for values as if the learned model were correct

Model-free:
- Direct evaluation:
  V(s) = sample estimate of sum of rewards accumulated from state s onwards

- Temporal difference value learning
  Move values toward value of whatever successor occurs: running average!

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

RL examples: Spider Walking

http://www.youtube.com/watch?v=RZf8fR1SmNY&index=6&list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e

Something Else than TD?

- TD value leaning is a model-free way to do policy evaluation, mimicking Bellman updates with running sample averages
- Idea: learn Q-values, not values
- Makes action selection model-free too!

Revisit Q-Learning

Value iteration:
- Start with $V_0(s) = 0$
- Given V_k, calculate V_{k+1} values for all states:

$$V_{k+1}(s) \leftarrow \max_{\pi(s)} \sum_{s'} P(s'|s,\pi(s))[r_{s'} + \lambda V_k(s')]$$

$$\downarrow$$

$$Q(s,a) = \text{Value of taking action a in state s}$$

$$\downarrow$$

Q iteration:
- Start with $Q_0(s,a) = 0$
- Given Q_k, calculate Q_{k+1} values for all states and actions:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,\pi(s))[r_{s'} + \lambda \max_{a'} Q_k(s',a')]$$

Revisit Q-Learning

- Since we don't know T and/or R, learn them(i.e. compute average) as we go
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$\text{Q } sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$$

Q-Learning, and Beyond

- Q-learning converges to optimal policy !!
- Caveats
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions.
  - Basic Q-learning keeps a table of all Q-values
  :Infeasible → Approximate Q-learning(feature-based)

- Policy Search
  - Problem: often the feature-based policies that work well (win games, maximize utilities) aren't the ones that approximate V / Q best
  - Solution
  : learn policies that maximize rewards, not the values that predict them
  - Start with an ok solution (e.g. Q-learning) then fine-tune by local optimization (e.g. hill climbing)

Summary

Things we know how to do:

- If we know the MDP
  - Compute V*, Q*, π* exactly
  - Evaluate a fixed policy π
- If we don't know the MDP
  - We can estimate the MDP then solve the MDP
  - We can estimate V for a fixed policy π
  - We can estimate Q*(s,a) for the optimal policy while executing an exploration policy

Techniques:

- Offline MDP Solution
  - Value/Policy Iteration
  - Policy evaluation
- Reinforcement Learning
  - Model-based RL
  - Model-free: Value learning
  - Model-free: Q-learning

- Idea: Compute averages over T using sample outcomes

*Online book: Sutton and Barto
http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html