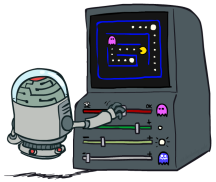
CS 188: Artificial Intelligence

Reinforcement Learning II



Instructor: Nathan Lambert, University of California, Berkeley

1

Reinforcement Learning

- We still assume an MDP:
  - A set of states $s \in S$
  - 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 $\pi(s)$

- New twist: don't know T or R, so must try out actions
- Big idea: Compute all averages over T using sample outcomes

2

The Story So Far: MDPs and RL

Known MDP: Offline Solution

Goal	Technique
Compute V*, Q*, π*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

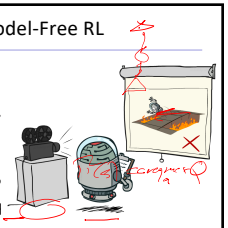
Unknown MDP: Model-Based		Unknown MDP: Model-Free	
Goal	Technique	Goal	Technique
Compute V*, Q*, π*	VI/PI on approx. MDP	Compute V*, Q*, π*	Q-learning
Evaluate a fixed policy π	PE on approx. MDP	Evaluate a fixed policy π	Value Learning

3

The Story So Far: Model-Free RL

- Passive Reinforcement Learning
  - Goal: Learn state-values
  - Direct Evaluation
    - Learn state-values in a batch from samples
  - Temporal Difference Models
    - Iteratively learning state-values after each episo
    - Incremental update leans towards convergence

$$V^{\pi}(s) \leftarrow V^{\pi}(s) +$$



- Q-Learning
  - Iterating on policy-conditioned values is challenging
  - What is a better way to get a policy from utility estimate?

4

Sample-Based Policy Evaluation?

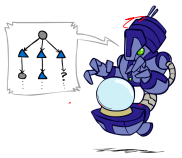
- We want to improve our estimate of V by computing these averages:

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

- Idea: Take samples of outcomes s′ (by doing the action!) and average

$$sample_1 = R(s,\pi(s),s'_1) + \gamma V^{\pi}_{k}(s'_1)$$
$$sample_2 = R(s,\pi(s),s'_2) + \gamma V^{\pi}_{k}(s'_2)$$
$$\cdots$$
$$sample_n = R(s,\pi(s),s'_n) + \gamma V^{\pi}_{k}(s'_n)$$

$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n}\sum_{i} sample_i$$



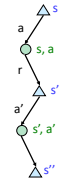
5

Model-Free Learning

- Model-free (temporal difference) learning
  - Experience world through episodes

$$(s,a,r,s',a',r',s'',a'',r'',s'''' \ldots)$$

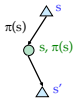
  - Update estimates each transition (s,a,r,s')
  - Over time, updates will mimic Bellman updates



6

Temporal Difference Learning

- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average



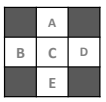
Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

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

7

Example: Temporal Difference Learning

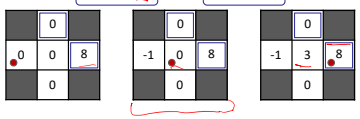
States



Observed Transitions

B, east, C, -2

C, east, D, -2



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

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

8

Approximating Values through Samples

- Policy Evaluation:

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

- Value Iteration:

$$V_{k+1}(s) \leftarrow \max_{a} \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s')\right]$$

- Q-Value Iteration:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a')\right]$$

9

Q-Learning

- Q-Learning: sample-based Q-value iteration

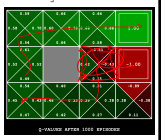
$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a')\right]$$

- Learn Q(s,a) values as you go
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s, a)$
  - Consider your new sample estimate:

$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$ no longer policy evaluation!

  - Incorporate the new estimate into a running average:

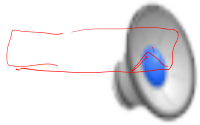
$Q(s, a) \leftarrow (1-\alpha)Q(s, a) + (\alpha)[sample]$



[Demo: Q-learning – gridworld (L10D2)]
[Demo: Q-learning – crawler (L10D3)]

10

Video of Demo Q-Learning -- Gridworld



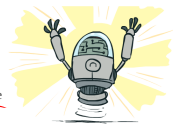
11

Video of Demo Q-Learning -- Crawler

12

Q-Learning Properties

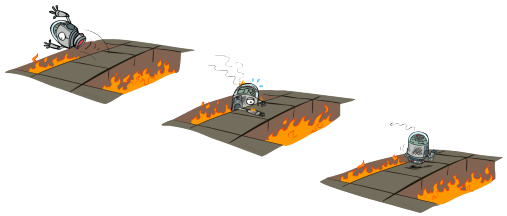
- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!
- This is called off-policy learning
- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions (!)



[Demo: Q-learning – auto – cliff grid (L11D1)]

13

Active Reinforcement Learning



14

RL: Helicopter Flight

[Andrew Ng] [Video: HELICOPTER]

15

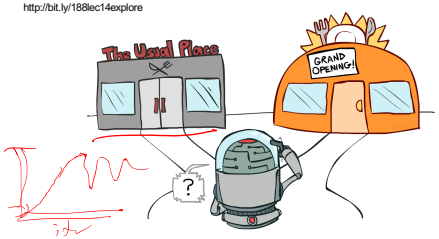
RL: Learning Locomotion

[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016] [Video: GAE]

16

Exploration vs. Exploitation

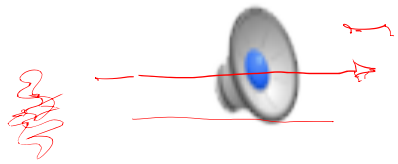
http://bit.ly/188lec14explore



17

Video of Demo Q-learning – Manual Exploration – Bridge Grid

http://bit.ly/188lec14explore



18

How to Explore?

http://bit.ly/188lec14explore

- Several schemes for forcing exploration
  - Simplest: random actions (ε-greedy)
    - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability 1-ε, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

[Demo: Q-learning – manual exploration – bridge grid (L11D2)]
[Demo: Q-learning – epsilon-greedy -- crawler (L11D3)]

19

Video of Demo Q-learning – Epsilon-Greedy – Crawler

20

Exploration Functions

- When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring
- Exploration function
  - Takes a value estimate u and a visit count n, and returns an optimistic utility, e.g. $f(u, n) = u + k/n$

Regular Q-Update: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q(s',a')$

Modified Q-Update: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

  - Note: this propagates the "bonus" back to states that lead to unknown states as well!

[Demo: exploration – Q-learning – crawler – exploration function (L11D4)]

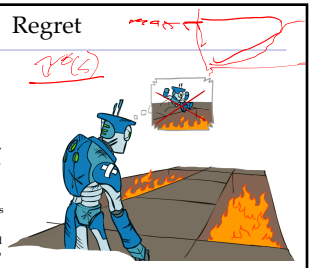
21

Video of Demo Q-learning – Exploration Function – Crawler

22

Regret

- Even if you learn the optimal policy, you still make mistakes along the way!
- Regret is a measure of your total mistake cost: the difference between your (expected) rewards, including youthful suboptimality, and optimal (expected) rewards
- Minimizing regret goes beyond learning to be optimal – it requires optimally learning to be optimal
- Example: random exploration and exploration functions both end up optimal, but random exploration has higher regret



23

Discuss: Model-Free Learning

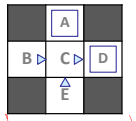
- act according to current optimal (based on Q-Values)
- but also explore…

Q-learning

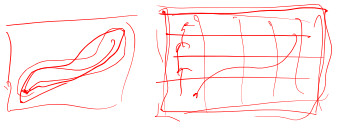
24

Discuss: Model-Based Learning

~~Input Policy π~~

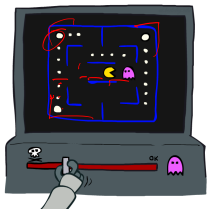


- act according to current optimal
- also explore!



25

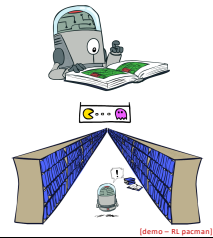
Approximate Q-Learning



26

Generalizing Across States

- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
  - This is a fundamental idea in machine learning, and we'll see it over and over again



[demo – RL pacman]

27

Example: Pacman

Let's say we discover through experience that this state is bad:

In naïve q-learning, we know nothing about this state:

Or even this one!

[Demo: Q-learning – pacman – tiny – watch all (L11D5)]
[Demo: Q-learning – pacman – tiny – silent train (L11D6)]
[Demo: Q-learning – pacman – tricky – watch all (L11D7)]

28

Video of Demo Q-Learning Pacman – Tiny – Watch All

29

Video of Demo Q-Learning Pacman – Tiny – Silent Train

30

Video of Demo Q-Learning Pacman – Tricky – Watch All

31