Reinforcement Learning 1: Review of Course 1 and Projects

CMPUT 655

Fall 2020

Martha White

Comments: Sept. 16

- Discussion on Slack very nice
- Are there any other topics you would like to review?
- First 40-50 minutes: some review of discussions and exercise questions
- Next 30-40 minutes: discuss projects, and break into groups

Discussion Topics Summary

- Is the MDP formulation too restrictive? Why did the RL community decide we want to solve MDPs? Is there something better?

- Meta-parameters are usually a function of the data (environment). How do we determine the correct values of a meta-parameter for a particular environment? (Example: choosing “c” for UCB as a function of the magnitude of the reward)

- Is the reward hypothesis sufficient to describe all problems? What types of problems can we **not** solve?

- Can we use RL algorithms outside of control systems? If so, what types of problems. If not, why?

- How do we satisfy the Markov property in more complex real-world problems?

Discussion Topics Summary

- What is a “value function” and what is it trying to measure? Can we define other value functions besides the discounted returns, V(s) and Q(s, a)?

- In what types of real-world problems would we have known models of world dynamics?

- Can we define “optimal policy” with regards to other forms of optimality? For instance, lower variance of the return or better exploration?

Additional questions

- Are there other ways to discount (state-based, time-based, different from the exponential decrease approach)?
- Why use epsilon-greedy, since UCB is better? How do we use UCB in large state/action spaces? What about using gradient bandits? What other exploration strategies are there?
- Is UCB a Bayesian approach?
- Apart from being a simple problem formulation, what are the advantages of the K-armed bandit problem over complete RL? what are its use-cases in industry?
    - related: For machine learning projects (usually commercial ones) the online learning phase is finished before sending the item to be used, is it also the case for RL?
- In the bandit experiments, there seems to be an emphasis on means, but when drawing conclusions wouldn't knowing deviation provide additional insight?
- A single number for reward seems restrictive for multiple goals/sub goals. Is weighted sum approach appropriate or are there more sophisticated methods?

Additional theory questions

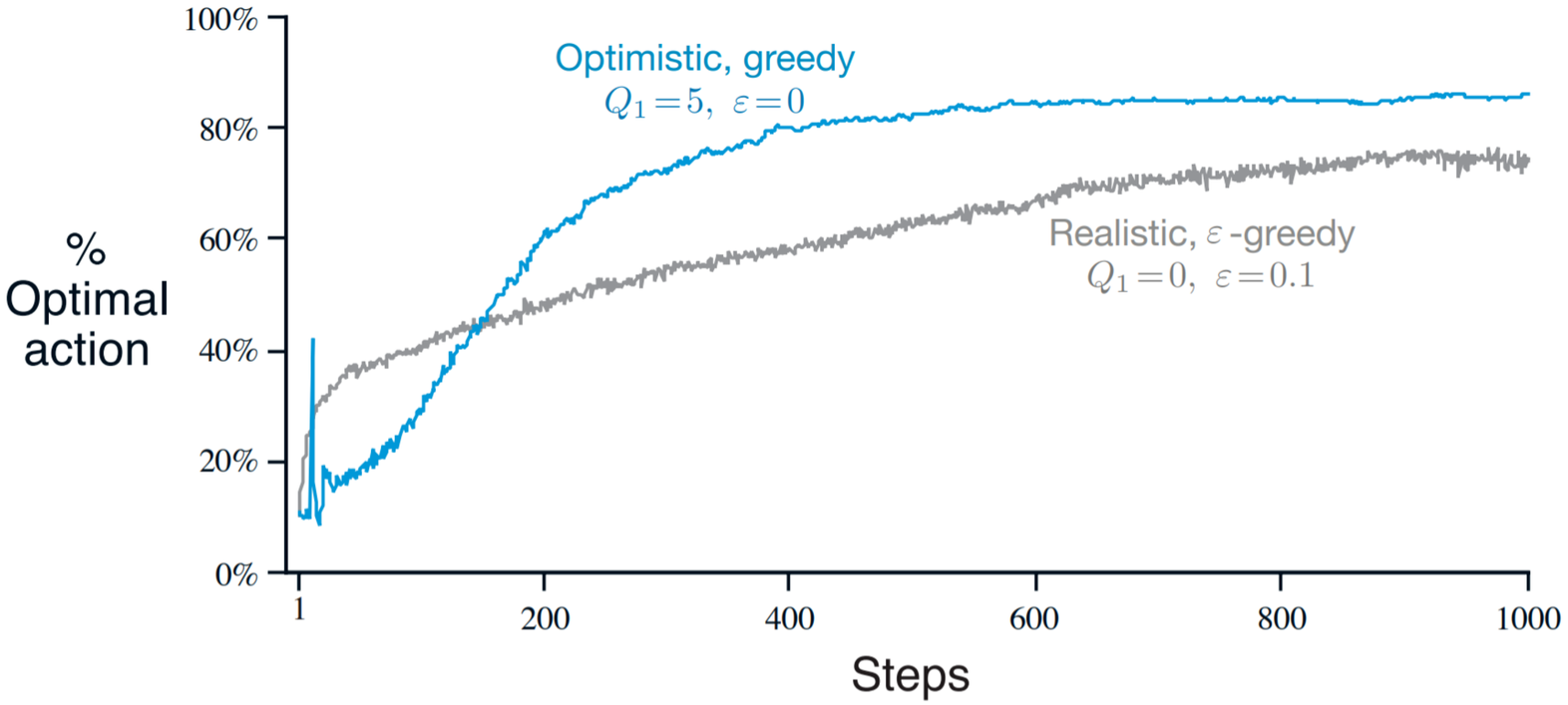
- How to prove that policy iteration is guaranteed to find the optimal policy in time polynomial in num_states and num_actions?
- For finite MDPs, the Bellman optimality eqn has a unique solution independent of policy. What is an example infinite MDP with many sol’ns
- How to prove convergence of iterative policy evaluation
- (I can show one or some of these in-class next time)

C1M1 Exercise Question 1

(Exercise 2.2 from S&B 2nd edition) Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1, R_1 = 1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = 2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

C1M1 Exercise Question 2

(Exercise 2.6 from S&B 2nd edition) The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?



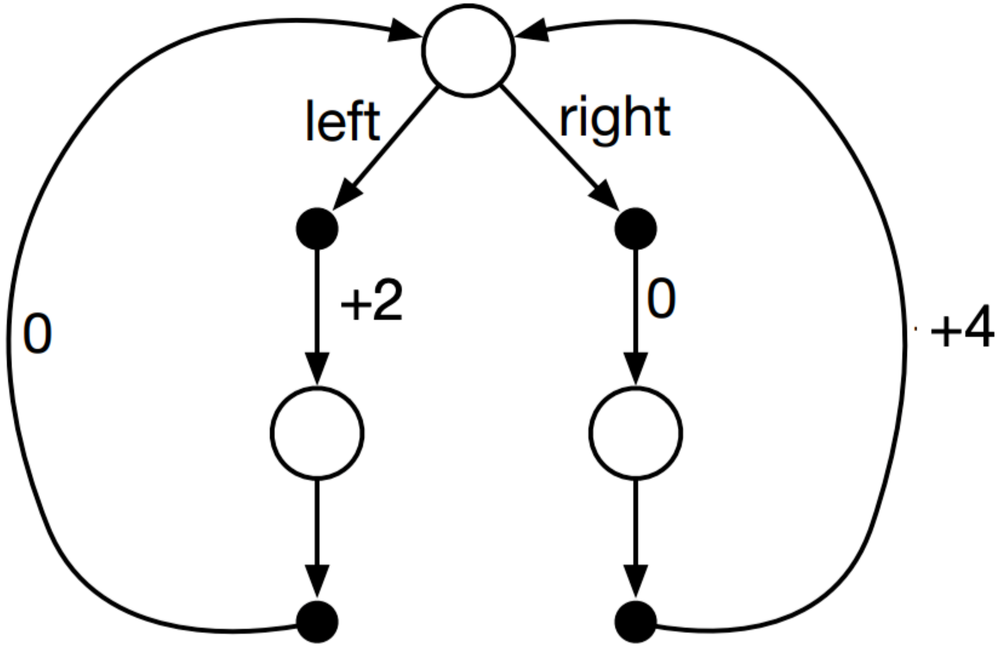
C1M2 Exercise Question 1

Assume you have a bandit problem with 4 actions, where the agent can see rewards from the set $\mathcal{R} = \{-3.0, -0.1, 0, 4.2\}$. Assume you have the probabilities for rewards for each action: $p(r|a)$ for $a \in \{1, 2, 3, 4\}$ and $r \in \{-3.0, -0.1, 0, 4.2\}$. How can you write this problem as an MDP? Remember that an MDP consists of $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma)$.

More abstractly, recall that a Bandit problem consists of a given action space $\mathcal{A} = \{1, ..., k\}$ (the k arms) and the distribution over rewards $p(r|a)$ for each action $a \in \mathcal{A}$. Specify an MDP that corresponds to this Bandit problem.

C1M3 Exercise Question 1

Consider the continuing MDP shown on the bottom. The only decision to be made is that in the top state, where two actions are available, left and right. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. What policy is optimal if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$?



C1M4 Exercise Question 1

1. In iterative policy evaluation, we seek to find the value function for a policy π by applying the Bellman equation many times to generate a sequence of value functions v_k that will eventually converge to the true value function v_π. How can we modify the update below to generate a sequence of action value functions q_k?

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right]$$

2. A deterministic policy $\pi(s)$ outputs an action $a \in \mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ directly. More generally, a policy $\pi(\cdot|s)$ outputs the probabilities for all actions: $\pi(\cdot|s) = [\pi(a_1|s), \pi(a_2|s), \ldots, \pi(a_k|s)$. How can you write a deterministic policy in this form? Let $\pi(s) = a_i$ and define $\pi(\cdot|s)$.

C1M4 Exercise Question 1

The policy iteration algorithm on page 80 has a subtle bug in that it may never terminate if the policy continually switches between two or more policies that are equally good. This is ok for pedagogy, but not for actual use. Modify the pseudocode so that convergence is guaranteed. Note that there is more than one approach to solve this problem.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $policy\text{-}stable \leftarrow true$
   For each $s \in \mathcal{S}$:
   $old\text{-}action \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
   If $old\text{-}action \neq \pi(s)$, then $policy\text{-}stable \leftarrow false$
   If $policy\text{-}stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

C1M4 Exercise Question 3

5. **(Challenge Question)** A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, she wins as many dollars as she has staked on that flip; if it is tails, she loses her stake. The game ends when the gambler wins by reaching her goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of her capital to stake, in integer numbers of dollars. This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler's capital, $s \in \{1, 2, ..., 99\}$ and the actions are stakes, $a \in \{0, 1, ..., \min(s, 100 - s)\}$. The reward is +1 when reaching the goal of \$100 and zero on all other transitions. The probability of seeing heads is $p_h = 0.4$.

(a) What does the value of a state mean in this problem? For example, in a gridworld where the value of 1 per step, the value represents the expected number of steps to goal. What does the value of state mean in the gambler's problem? Think about the minimum and maximum possible values, and think about the values of state 50 (which is 0.4) and state 99 (which is near 0.95).

(b) Modify the pseudocode for value iteration to more efficiently solve this specific problem, by exploiting your knowledge of the dynamics. *Hint: Not all states transition to every other state. For example, can you transition from state* 1 *to state* 99*?*

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$

Identifying Projects

- If you already have a group and project idea, post it to Slack!
- If you are comfortable with the project, you can go ahead and submit a proposal on eClass
  - We will go through these and give comments when needed about scope
- If you still want more validation, then send a draft on slack, to Andy, Shivam and I as a Direct Message (including your teammates). We can give minor comments about it there
- Today in class we will spend some time discussing projects, so you can find teammates

Advice on Identifying a Project

- Execution matters as much as (or more than) ideas
  - There is a tendency to over-value coming up with a cute algorithm/idea
  - Research is about understanding, not just about solving problems
- It is likely too early to come up with ideas on your own; it is perfectly fine to take a question specified by someone else
  - Your job as a (new) grad student is to execute thoroughly and precisely
- Specify a clear question and identify steps towards answering it

Some project types

- Empirical Study — requires a thorough evaluation
- Algorithmic Project — a new idea, where sometimes it is enough to do a demonstration rather than a thorough study
- Theory Project — provide an answer to an open theoretical question
- Applied Project — this one is not listed in the google doc, because an Applied project might take too much time in this course, unless you are bringing a partially completed

Empirical Study

- An empirical study is not easy!
  - Ignore most of the papers in RL; they are not really doing empirical work, typically they are really just showing that their algorithm runs
- It requires 1) designing an experiment to highlight what you would like to see and 2) using a sound, and typically exhaustive methodology
- You need sufficient runs, fair choices for algorithms, etc.
- e.g., https://arxiv.org/abs/1602.08771

Algorithmic Project

- You need to clearly motivate what problem you are solving and why your proposed idea solves this problem
- You need results substantiating these claims
  - Theory, without experiments, is enough
  - Else, your experiments should provide evidence for your claims
- Ask yourself: do I have evidence supporting my claims about this algorithm? Did I design my experiment to highlight what I want to show about this algorithm?

Some topic areas

- Policy evaluation and prediction, especially in the off-policy setting (data gathered under one policy, want to evaluate another)
- Improving (stable) online control, with policy gradient methods and/or action-value methods
- Optimization strategies for RL, and (hyper)parameter selection
- Exploration
- Using learned models in RL
- Batch RL, including learning from a batch for pre-training

An example of a concrete project

- Title: Understanding the role of partial greedification in API
- Motivation: Policy gradient methods can be seen as an instance of API, where the greedification step is not fully greedy. Instead, the policy parameters are only updated a part of the way to the greedy policy. This contrasts methods like Q-learning. There is some reason to believe, however, that avoiding fully greedy steps is beneficial for stability in learning and convergence to better solutions, particularly as the action-values are incorrect during learning. We investigate the benefits, if any, of using partially greedy steps in a simple toy environment. We vary the accuracy of the action-values during learning (both due to estimation and approximation), as well as the amount of greedification per step, within Soft Actor-Critic for discrete action problems.

Identifying a question and steps

- Question: do n-step methods for learning values in Actor-Critic help?
- Step 1: Hypothesize why they could help (e.g., more accurate value estimates and so lower bias in the gradient estimate)
- Step 2: Pick an environment on which to test this
- Step 3: Implement the standard Actor-critic algorithm, with n = 1, including specifying the function approximator (e.g., start with tile-coding)
- Step 4: Run the experiment with just that implementation, and ensure you have a clear set-up

Finding topics and groups

- Goals: to meet your fellow classmates, start finding potential collaborators and start finding a project
- In your group, answer the following questions:
- **Do you already have a project in mind? If yes, what is it?**
- **If no, what topics are you interested in?**
- After discussing with the first group for 10 minutes, we can do one more random shuffle