Mini-Course 1, Module 4
Dynamic Programming

CMPUT 397
Fall 2020

Reminders: Sept 28, 2020

- Discussion Session during class on Wednesday
  - A question you might have: Do I have to go? Answer: Yes
- Any questions?

Review of C1M4
Dynamic Programming

Video 1: Policy Evaluation vs. Control

- Introduce the two classic problems of RL: prediction and control. Classic assumptions of DP
- Goals:
  - Understand the distinction between policy **evaluation** and **control**
  - Explain the **setting** in which dynamic programming can be **applied**, as well as its limitations

Video 2: Iterative Policy Evaluation

- How to turn Bellman equations into algorithms for **computing** value functions and policies
- Goals:
  - Outline the iterative policy evaluation algorithm for estimating state values for a given policy
  - Apply iterative policy evaluation to compute value functions, in an example MDP

Video 3: Policy Improvement

- **Key theoretical result** in RL and DP! How to make the policy better using the value function
- Goals:
  - Understand the **policy improvement theorem**; and how it can be used to construct improved policies
  - And use the value function for a policy to **produce a better policy**

Video 4: Policy Iteration

- Our first control algorithm. Why sequencing evaluation and improvement works!
- Goals:
  - Outline the **policy iteration algorithm** for finding the optimal policy;
  - Understand **“the dance of policy and value”**, how policy iteration reaches the optimal policy by alternating between evaluating a policy and improving it
  - Apply policy iteration to compute optimal policies and optimal value functions

Video 5: Flexibility of the Policy Iteration Framework

- Generalized Policy Iteration: a general framework for control
- **Goals**:
  - Understand the framework of generalized policy iteration
  - Outline value iteration, an important special case of generalized policy iteration
  - Differentiate synchronous and asynchronous dynamic programming methods

Video 6: Efficiency of Dynamic Programming

- DP is actually pretty good, compared to other approaches! What's the deal with **Bootstrapping**?

- **Goals**:

  - Describe Monte-Carlo sampling as an **alternative** method for learning a value function

  - Describe brute force search as an **alternative** method for finding an optimal policy; and

  - Understand the advantages of Dynamic programming and "**bootstrapping**" over these alternatives.

Key Terminology

- Policy evaluation
- Policy improvement
- Policy iteration
- Value iteration
- Generalized policy iteration

More Definitions and Terminology

- “I'm confused about what v_k is, my interpretation is its the state-value function for an arbitrary policy. I don't believe that is correct though. What is v_k?” —> It is our value estimate on the k-th step of Iterative Policy Evaluation

Difference between v and q

- “Why does the lower golf example (figure 3.3) which is supposed to be optimal have a -2 field over most of the green, where the above example with the putter has that area marked as only -1? Isn't q*() supposed to be optimal? There should be no areas where q*() has a worse result than v putt, right?”

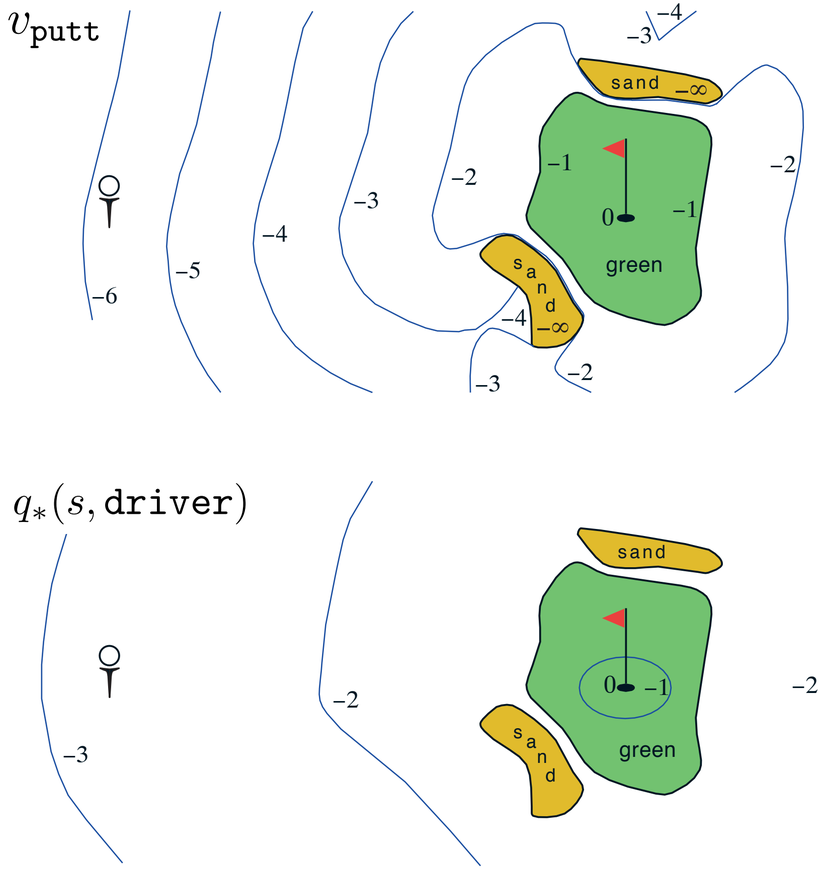


Figure 3.3: A golf example: the state-value function for putting (upper) and the optimal action-value function for using the driver (lower). ■

Value Iteration vs Policy Iteration

- “When would we ever not want to use Value Iteration?”
- “Since value iteration only need one "sweep", does it means value iteration is less precise and need take more iteration to find the optimal policy than the policy iteration?”

Exercise Question

- “What is the time complexity in policy/value iteration when doing an entire sweep?”
- Think about it in the worst-case, assuming deterministic reward outcomes that depend on s, a, s’
- Think about more typical cases, and so what could be an average case performance

Convergence Criteria

- “Iterative Policy Evaluation algorithm will loop until the value function of the previous state and current state hit a small enough value, is the small enough value a predefined threshold, and what is the commonly used threshold in this case?”

- “For asynchronous updates, when do we know when we have v* ? How do we determine if the value of v is unchanging since it has converged to v* or because we haven't updated some states, and are we forcing the agent to eventually update all states to check ?”

- “For policy evaluation part, under the same policy, why is our sequence of value functions monotonously improving for every state?”

Policy Improvement

- “Why does the policy improvement theorem guarantee the new policy is strictly better than the old policy? Why can't the new policy be just as good as the old policy if the optimal policy still has not been found?”

- “In policy improvement video, you said that “policy pi prime is strictly better if the value is strictly greater at least one state”, but what about other states? if one of the state following pi prime have lower value than pi will pi prime still be strictly better than pi?”

Policy Improvement Result

$$
\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \;\big|\; S_t = s\big] \\
&\leq \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \;\big|\; S_t = s\big] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \;\big|\; S_t = s\big] \\
&= v_{\pi'}(s).
\end{aligned}
$$

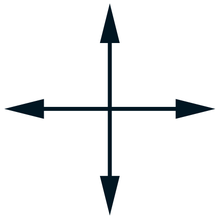
Asynchronous DP

- “When we talked about dynamic programming we said that there was a way to do the updates in place with a single array. How would that method look when doing the updates asynchronously?”
- “What kind of DP algorithms are usually used to perform asynchronous updates? How do we make sure such algorithms will update all states to guarantee convergence?”
- “When is it more beneficial to perform a synchronous sweep rather than asynchronous?”
- “In the case that there are two separate arrays, the order that we update the values does not matter. In the one array case, can it be done with random sampling?”

Additional Clarifications

- “is it possible to use DP in non-episodic models?” —> Yes
- “The Monte Carlo method which is quite famous is described to be a optimization of averages of the policy taken over a lot of instances. This seems to me a very unsophisticated method? So, why is such a method so widely used in RL?” —> Its actually not very widely used
- “How do you make sure the optimal solution found by value iteration is global maximum instead of local maximum?” —> we have not talked about having a (smooth) optimization surface that could have local maxima. Value iteration is guaranteed to converge to the optimal solution (the global max)

Practice Questions



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

$p(6, -1 | 5, \texttt{right}) =$

$p(10, r | 5, \texttt{right}) =$

$p(7, -1 | 7, \texttt{right}) =$

Worksheet Q1

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?

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