CS 181 LECTURE 4/11/24

Scribe Notes

Contents

1 Admin

HW 5 is due Friday and HW 6 will be out. Next lecture will be the last one where the content is covered on the midterm.
Midterm 2 is April 23rd.

2 Reward Design: Story in the World

Reward design is a hard real world problem. When we did Embedded EthiCS lecture, there was a choice you made as a project manager. There was an emphasis on how making decisions is tricky.

Another real world example is social media. Companies want you to stay on their platform so that you can click ads and generate revenue for them. So, one metric that companies track is time on platform. Facebook started as chronological feed of your friend's posts but now there are different priorities on which posts are first in your feed.

Facebook later made the move to promote engaging content. Engaging content can measured by likes and comments. "Dwell time" - which is how much time people stay on a post - is another way of gauging how engaging a post is. But people often stay on posts that are traumatic or extreme. Similarly, comments increase when a post is controversial. Thus, the posts that are more divisive are being promoted. People may go through darker periods in their life and would pay more attention to certain types of discourse. If they move on to a different period of their life, they may no longer want to see the content but the platforms keep feeding them the same type of content. This highlights how tricky these reinforcement learning problems are. There is a responsibility to build tools that do something beneficial in the world. You may need to pivot and adjust from decisions you make the first time.

3 Review From Last Lecture

In a Markov Decision Process, we have $\{S, A, R, T, \gamma\}$ and the goal is to find

$$\max_{\pi} \mathbb{E}_{\pi,T}[\sum \gamma^t r_t | s = s_0].$$

We found the Bellman Equation: $V^{\pi}(s) = r(s_0, \pi(s_0)) + \gamma \sum_{s'} T(s'|s_0, \pi(s_0)) V^{\pi}(s')$.

We ended with the following concept check: If $r'(s, a) \propto r(s, a) + \beta$, does the optimal policy π change?

The answer is: no! The expression for the new value would be as follows:

$$V^{\pi}_{r'} = \alpha V^{\pi}_{r} + \sum_{t=0}^{\infty} \gamma^t \cdot \beta$$

We notice that α is a multiplicative constant, so all the policies change by the same factor. Thus, whichever policy was the best before will still be the best/

4 Optimizing Policies

4.1 Policy Iteration

The first way of optimizing a policy is called policy iteration:

1. State with some π.
2. Evaluate π. We talked about solving a linear system of equations or evaluating iteratively (from last lecture).
3. Improve π.

Define the action value function, $Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} T(s'|s,a)V^\pi(s')$. This function says that if we are at state s and the policy π says for us to go to $\pi(s)$, but we are instead going to choose action a, then $T(s'|s,a)$ is the distribution of possible outcomes from action a. We are doing a one off excursion to not follow policy π. Afterwards, we follow policy π.

$$Q^\pi(s,a) = \underbrace{R(s,a) + \gamma \sum_{s'} T(s'|s,a)}_{\text{effect of "excursion"}} \underbrace{V^\pi(s')}_{\text{continue with } \pi}$$

So, we find $\pi(s)$ where:

$$\pi(s) = \text{argmax}_a Q^\pi(s,a)$$

In the discrete setting, our policy will converge to an optimal π^*.

In policy iteration, we go back and forth between improving π and evaluating π. We take big steps but each step is generally pretty expensive.

4.2 Value Iteration

Another way of optimizing is value iteration:

1. Start with some $Q_0(s,a)$. $Q_0(s,a)$ is a matrix so we sometimes call this the Q table. We can start with any $Q_0(s,a)$ and we will still converge to the same answer.
2. Set $Q_k(s,a)$ to be: $\underbrace{R(s,a) + \gamma \sum_{s'} T(s'|a)}_{\text{mini evaluation step}} \underbrace{\max_{a'} \underbrace{Q_{k-1}(s',a')}_{\text{old guess}}}_{\text{mini optimization step}}$.

Here, we have a tighter loop than policy iteration, so the value iterations are faster but we need to do a lot more iterations - hundreds of iterations - compared to tens of iterations in policy iteration.

4.3 Linear Programs

The third approach is applying linear programs for discrete spaces.

Solve:

$$\min_V \underbrace{c^T}_{\text{any c>0}} V$$
$$\text{s. t. } V(s) \geq R(s,a) + \gamma \sum_{s'} T(s'|s,a)V(s') \forall s, a$$

We are basically saying that $V(s)$ has to be greater than or equal to all possible $Q(s,a)$ since $Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s'|s,a)V(s')$.

4.4 Three Different Approaches

As technology changes, different approaches become easier to do. So, it is good to learn about the various approaches, knowing that one may be better than another depending on the scenario and time.

5 Concept Exercise: Optimizing a Policy

Suppose we are in a game where we desire to get to the end of the hallway.

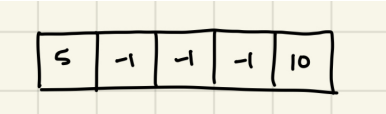


Figure 1: The game.

When we reach the left end of the hallway, we get 5 points. When we reach the right end of the hallway, we get 10 points. The game terminates when we get to either the left or right ends of the hallway. So, there are only three states where we have to make decisions.

$\gamma = 1$ and the actions are deterministic.

5.1 Using Policy Iteration

Let us start with the following policy:

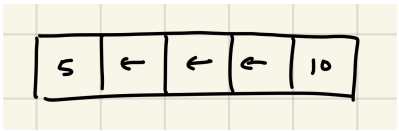
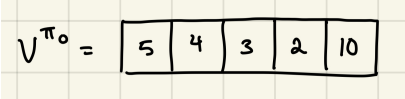


Figure 2: Initial policy π_0.

In this starting policy, we assign the action of left move for each state.

In policy iteration, we first evaluate the policy π_0.

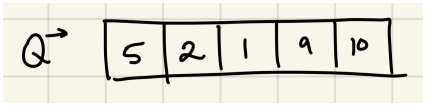


Let us label the states from left to right as 1, 2, 3, 4, and 5. We know that states 1 and 5 remain fixed at their point values 5 and 10.

Let us consider state 2. The value starts at -1 and moving to the left gets us to state 1 where the value is 5. So, the total value is -1+5 = 4.

Similarly, we can evaluate states 3 and 4. For state 3, we follow π_0 and move to the left two times, getting us -1+(-1)+5 = 3. For state 4, we have -1+(-1)+(-1)+5 = 2.

The next step in policy iteration is to improve π. We consider the resulting values if we do a right move one-off excursion.



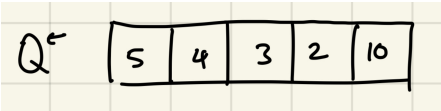
Again, states 1 and 5 remain the same.

For state 4, if we go right once, then we get to the end of the hallway. The points evaluate to -1+10 = 9.

For state 3, if we go right once, we end up at state 4. Then, we return to the original policy, and move left three times. Therefore, the points evaluate to -1+(-1)+(-1)+(-1)+5 = 1.

For state 2, after we move right once, we return to π_0 and move left two times to end up at the 5 point box. Therefore, the points evaluate to -1+(-1)+(-1)+5=2.

The other possible Q is moving to the left. Since our original policy was all moving to the left, this with evaluate to the same as the original policy values.



Now we compare the value outcomes between the left excursion and the right excursion.

For state 2, we see that $4 > 2$, so we choose left move.

For state 3, we see that $3 > 1$, so we choose left move.

For state 4, we see that $2 < 9$, so we choose right move. Thus, our new policy is Fig 3.

Then, we repeat this process of evaluating and improving the policy until no more changes are made.

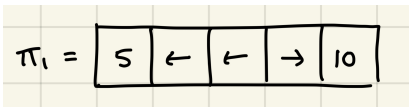


Figure 3: Next policy iteration, π_1.

5.2 Using Value Iteration

Let us start with the following Q table, Q_0:

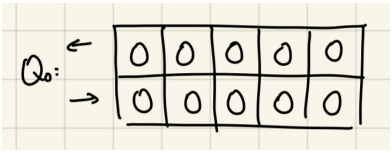


Figure 4: Q_0.

In value iteration, we update our Q table in the next step using the following equation:

$$R(s,a) + \gamma \sum_{s'} T(s'|a) \max_{a'} Q_{k-1}(s', a')$$

Therefore, Q_1 will be the following:

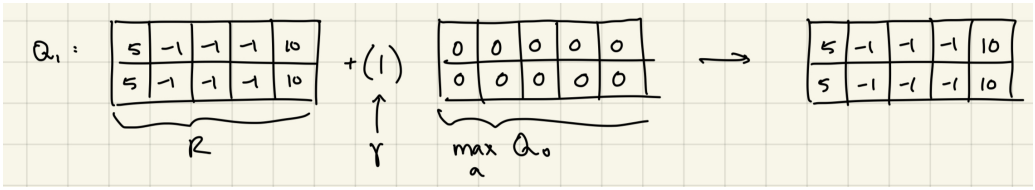


Figure 5: Q_1.

We follow the update step again.

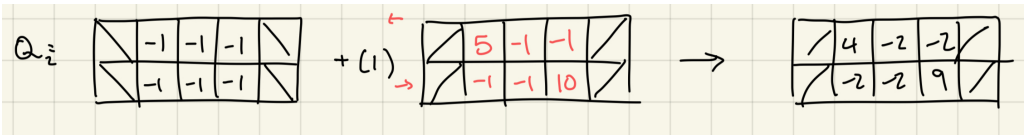


Figure 6: Q_2.

This time, we notice the numbers have changed for the $\max Q_1$ table. The first row considers the value at the next state if we took the action of going left. The second row considers the value at the next state if we took the action of going right.

Now, when we consider the value at each state, we take the larger of the two values in each column. For example, state 1 gives us a value of 4 instead of -2.

In value iteration, we continue iterating as we have, creating Q_3, Q_4, etc.