Exercise Set 3 - Reinforcement Learning
Chapter 5,6 - Advanced TD methods and approximation

Instructions

This is the third exercise booklet for Reinforcement Learning. It covers both ungraded exercises to practice at home or during the tutorial sessions as well as graded homework exercises and graded coding assignments. The graded assignments are clearly marked as homework.

- Make sure you deliver answers in a clear and structured format. LaTeXhas our preference. Messy handwritten answers will not be graded.
- Pre-pend the name of your TA to the file name you hand in and remember to put your name and student ID on the submission;
- The deadline for this first assignment is **September 30th 2022 at 13:00** and will cover the material of chapter 5-6. All questions marked 'Homework' in this booklet need to be handed in on Canvas. The coding assignments need to be handed in separately through the Codegrade platform integrated on canvas.

Contents

Chapter 5: Advanced temporal difference learning

5.1 Advanced Temporal Difference Learning (Application)

~~Consider the same setting as in Exercise ??~~, i.e. we have an undiscounted Markov Decision Process (MDP) with two states A and B, each with two possible actions 1 and 2, and a terminal state T with $V(T) = 0$. The transition and reward functions are unknown, but you have observed the following episode using a random policy:

- $A \xrightarrow[r_3=-3]{a_3=1} B \xrightarrow[r_4=4]{a_4=1} A \xrightarrow[r_5=-4]{a_5=2} A \xrightarrow[r_6=-3]{a_6=1} T$

where the the arrow ($\rightarrow$) indicates a transition and a_t and r_t take the values of the observed actions and rewards respectively.

1. What are the state(-action) value estimates $V(s)$ (or $Q(s, a)$) after observing the sample episode when applying:
   (a) 3-step TD
   (b) Q-learning

   where we initialize state(-action) values to 0 and use a learning rate $\alpha = 0.1$
2. Choose a deterministic policy that you think is better than the random policy given the data. Refer to any of the state(-action) value estimates to explain your reasoning.
3. Let π_{random} denote the random policy used so far and $\pi_{student}$ denote the new policy you proposed. Suppose you can draw new sample episodes indefinitely until convergence of the value estimates.
   (a) Discuss how do you expect the final value estimates to differ if you ran Q-Learning with π_{random} as compared to $\pi_{student}$.
   (b) What problems may arise with π_{random} or $\pi_{student}$ respectively?
   (c) Do you think using an ϵ-greedy policy as behavior policy would be beneficial? Explain why/why not?

5.2 *Exam question - N-Step Temporal Difference Learning

This exercise has been taken from a previous exam (perhaps lightly edited) and can require a bit more insight. It should give you a good idea of the level of exam questions.

Consider the undiscounted and deterministic random walk environment in Figure 1 with 19 states and two terminal states. The state in the middle (J) is always the start state and we want to apply n-step TD using a random behavior policy choosing between going left or right with equal probability at each step. The rewards are indicated above each transition arrow. The initial value of each state is set to 0. We run n-step TD for different values of n and learning rate α. To the right

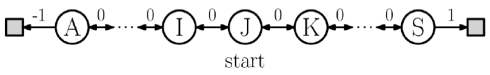

Figure 1: Random walk environment.

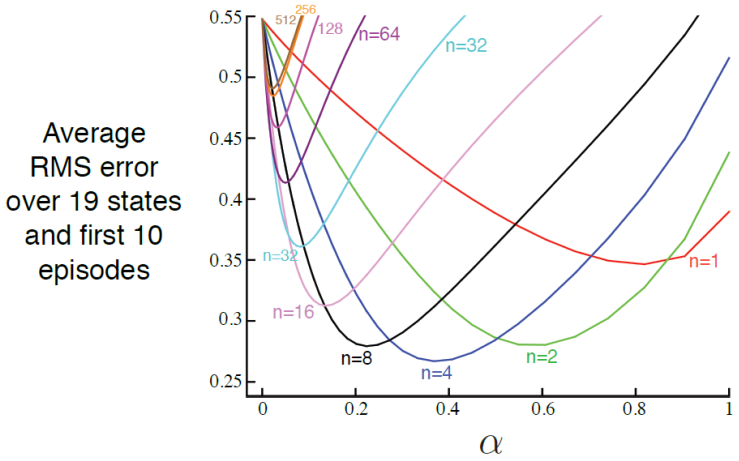

Figure 2: Performance of n-step TD methods as a function of α for various values of n on a 19-state random walk task.

you can see the average RMS error over all states compared to the true state values after 10 episodes.

1. We observe that we need a small learning rate when n is big in order to reduce the error. Why is this the case compared to when using a smaller n?

2. Why does an intermediate value for n work best in this scenario (assuming a good choice for α)? You can argue about the disadvantages of the corner cases.

3. To use off-policy data with n-step methods, we can use importance weights. Do you think off-policy n-step learning with a greedy target policy (like in Q-learning) would be effective? Explain your answer.

5.3 Homework: Coding assignment - Temporal Difference Learning

1. Download the notebook *RLLab3_TD.zip* from canvas assignments and follow the instructions.

2. In the lab notebook you plotted the average returns during training for Q-learning and SARSA algorithms. Which algorithm achieves higher average return? Do you observe the same phenomenon as in the Example 6.6 in the book? Explain why or why not.

5.4 Homework: Maximization Bias

Consider the undiscounted MDP in Figure 3 where we have a starting state A with two actions (L,R), one ending in the terminal state T (with $V(T) = 0$) and always yielding a reward of 0.7 and another action that transitions to state B with zero reward. From state B we can take four different actions each transitioning to the terminal state and yielding a reward of either 0 or 1 with equal probability. Suppose that from a behavior policy we sample two episodes where we try action L and eight episodes where we try action R followed by an action from state B (each action in B is used twice). The behavior policy chooses actions in B uniformly. The observed rewards for each of the four actions from state B are indicated in the Figure, e.g. the rightmost

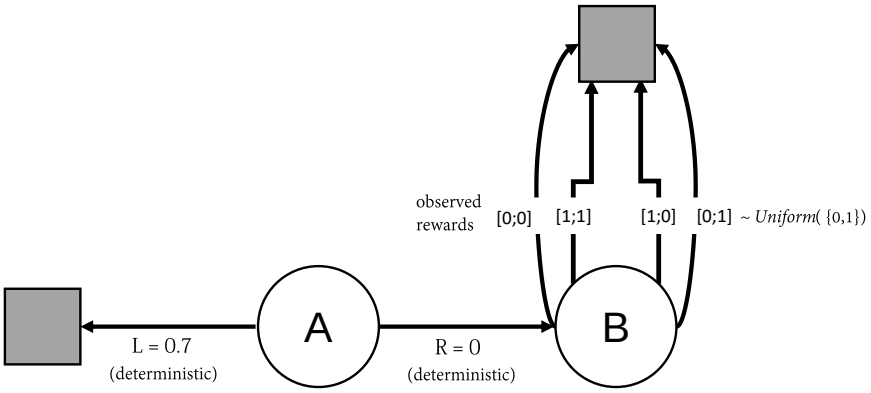

Figure 3: MDP: Maximization Bias

action received a reward of 0 and a reward of 1. *Note: The scenario in this exercise is a bit artificial, as it is designed highlight the effect of maximization bias.*

1. What are the Q-values of the four outgoing actions from state B after the updates corresponding to the described observations? Give your answers for both Q-learning and expected SARSA. Use an initial Q-value of 0.7 for all state-action pairs and a learning rate $alpha = 0.2$. For the purpose of this exercise, *the behavior policy is not updated.*

2. For the sake of this example, ignore any updates during the first ten transitions described above to the Q-values $Q(A, L)$ and $Q(A, R)$. So assume these are still at their initial value of 0.7. What happens to $Q(A, L)$ if now we execute action 'L' in state A and perform an update? Similarly, what happens to the value of $Q(A, R)$ if we execute 'R' in state A and perform an update? Give your answer again for both Q-learning and expected SARSA.

3. What are the true state-action values that we would expect to get (after convergence) if we continued sampling episodes.

4. This problem suffers from maximization bias. Explain where this can be observed. Do both Q-learning and SARSA suffer from this bias in this example? Why/why not?

5. To circumvent the issue of maximization bias, we can apply Double Q-learning. Use the given example to explain how Double Q-learning alleviates the problem of maximization bias.

Chapter 6: Prediction methods with approximation

6.1 Semi-gradient TD and the TD fixed point

We are considering how to travel to a goal location from various locations labeled 1, 2, 3, and 4. There are different travel costs between these locations. A "map" for this problem (showing the possible actions per state) and the associated costs are summarized in Figure 4. We model the problem as an MDP (Figure 4), with discount factor $\gamma = 1$. The goal location to the left is a terminal state. To use only 2 parameters to represent the value function, approximation can be used. We use a linear approximation $\hat{v}(s, \boldsymbol{w}) = \boldsymbol{w}^T \phi(s)$, For the four states and the terminal state, we use the following features respectively:

$$\phi(s1) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \phi(s2) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \phi(s3) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \phi(s4) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \phi(T) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

1. Assume the parameter for the value function approximator in the last time step is $w_t = [0.5, 0.5]^T$, the agent take a new step and receive a transition sample $(s_2, -1, s_4)$, please compute the one step updated value of parameters w_{t+1} with a learning rate α.

2. For the MDP as in Figure 4, you have access to the following set of trajectories (with actions not shown):

   $$\{(s_1, -1, s_3, -1, T), (s_2, -1, s_4, -5, T)\}$$

   where the end of an episode means you reached a terminal state. What solution do TD algorithms converge to when repeatedly trained on this dataset with the given feature function ?

3. Comment on the solution you found under 2. Where is the solution good or bad? Where the solution seems to be bad, can you understand why that is the case?

4. The TD fixed point is independent of the learning rate and certain algorithms based on finding it are said to "never forget". Elaborate what is meant by this and provide one advantage and one disadvantage of "never forgetting".

5. (Deep) neural networks are popularly used as function approximators (instead of linear function approximators). In this case $\hat{v}(s, \boldsymbol{w}) = \mathrm{NN}_w(s)$, where NN_w is a neural network with parameters (weights and biases) w. Assuming you have access to a 'autograd()' function that stores $\partial \mathrm{NN}_w(s)/\partial w$ to w.grad, how would you implement an update of the q-function for a transition (s, a, r, s', a')?

6.2 Basis functions

1. Tabular methods can be seen as a special case of linear function approximation. Show that this is the case and give the corresponding feature vectors.

2. You want to design the feature vectors for a state space with $s = [x, y]$. You expect that x and y interact in some unknown way. How would you design a polynomial feature vector for s?

3. What happens to the size of the polynomial feature vector if the number of variables in your state space increases?

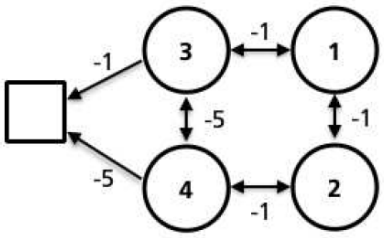

Figure 4: MDP showing possible actions per state and associated cost.

4. You are working on a problem with a state space consisting of two dimensions. You expect that one of them will have a larger impact on the performance than the other. How might you encode this prior knowledge in your basis function?

5. You can view coarse coding as a special case of Radial Basis Functions. Why?

6.3 Function approximation and state distribution

1. Consider the state distribution, $\mu(s)$. How does it depend on the parameters of the value function approximator?

2. How does this differ from the data distribution in standard (un-)supervised learning problems?

3. What does this mean for the weighting of the errors (such as in e.g. Eq. 9.1)?

6.4 Preparatory question: Off-policy approximation

Off-policy learning with approximation is a tricky topic. Before we'll dive into it in the next chapter, we'll investigate what happens if we apply the methods you know so far in this setting. On Canvas, you'll find a notebook prepared with an exercise on a problem called 'Baird's Counterexample'.

6.5 *Exam question: Value function approximation

We are considering how to travel to a goal location from various locations labeled 1, 2, 3, and 4. There are different travel costs between these locations. A "map" for this problem (showing the possible actions per state) and the associated costs are summarized in Figure 4. We model the problem as an MDP (Figure 4), with discount factor $\gamma = 1$. The goal location to the left is a terminal state. For the four states, we use the same features as in exercise 5.4, respectively:

$$\phi(s1) = \begin{bmatrix}2\\0\end{bmatrix}, \phi(s2) = \begin{bmatrix}0\\2\end{bmatrix}, \phi(s3) = \begin{bmatrix}1\\0\end{bmatrix}, \phi(s4) = \begin{bmatrix}0\\1\end{bmatrix}, \phi(T) = \begin{bmatrix}0\\0\end{bmatrix}$$

1. What is the optimal value for each of the states for time horizons of 1, 2, 3 and 4? *Hint: use value iteration!*

2. Consider the features used to represent the states. With these features, what is the value assigned to each of the states under optimal policy such that the mean squared error ($\overline{VE}$) is minimized? Assume that the state distribution $\mu(s)$ is uniform.

6.6 Homework: Gradient Descent Methods

1. Why is the Monte Carlo target, G_t, an unbiased estimate of $v_\pi(S_t)$?

2. Gradient Monte-Carlo approximates the gradient of the mean squared value error (see e.g. equations 9.1, 9.4 and 9.5 in the book). Similarly, derive a weight update that minimizes the mean squared temporal difference error. Compare the result to Semi-Gradient TD, and comment on the origin of the name Semi-Gradient method. *Note: It turns out the mean squared temporal difference error is not a great objective, we will take a closer look in the next chapter.*

3. Despite not corresponding to the gradient of any loss function, Semi-Gradient methods that use bootstrapping have certain advantages w.r.t. Monte Carlo approaches. Why, for example, would you prefer bootstrapping in the Mountain Car problem (book p.244 example 10.1)? Note that an episode in the Mountain Car problem only terminates once the goal is reached.