Exercise Set 3 - Reinforcement Learning
Chapter 5,6 - Prediction and control with 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 23th 2020 at 17:00** 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: Prediction methods with approximation

5.1 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?
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?

5.2 Neural Networks

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 standard (un-)supervised learning problems?
3. What does this mean for the weighting of the errors (such as in e.g. Eq. 9.1)?

5.3 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 difference error is not a great objective, we will take a closer look in the next chapter.*
3. Despite not being unbiased, 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)?

5.4 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 1. We model the problem as an MDP (Figure 1), with discount factor $\gamma = 1$. The goal location to the left is

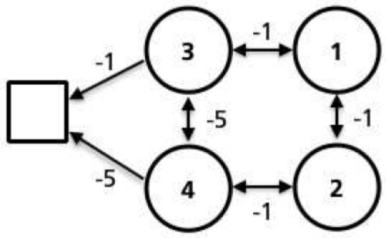

Figure 1: MDP showing possible actions per state and associated cost.

a terminal state. To use only 2 parameters to represent the value function, approximation can be used. 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. The TD fixed point is independent of the learning rate and solution strategies 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".

2. For the MDP as in Figure 1, you have access to the following set of trajectories:

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

   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?

5.5 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'.

5.6 *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 1. We model the problem as an MDP (Figure 1), 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 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.

Chapter 6: Off-policy and control with approximation

6.1 Geometry of linear value-function approximation (Theory)

1. Which error function is minimized by gradient Monte Carlo?
2. The Bellman error is zero only when the value error is zero (recall the Bellman equations). Why then does minimizing a TD objective not result in minimal value error ($\overline{VE}$) in the function approximation setting?
3. Is applying (full) gradient descent on the TD error a good approach to approximate the value function? Motivate your answer.

6.2 Homework: Geometry of linear value-function approximation (Application)

Consider the two-state MDP given in Figure 2. It consists of two states with one action, that transition into one another with reward 0. The features for both states are $\phi = 2$ for state s_0 and $\phi = 1$ for state s_1. We will now predict the value of the states using $v_w = w \cdot \phi$.

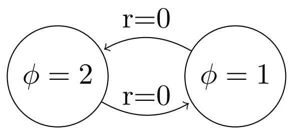

Figure 2: Two-state MDP

1. We can write the Bellman error vector as
$$\bar{\delta}_w = B^\pi v_w - v_w, \tag{1}$$
where B^π is the Bellman operator. What is the Bellman error vector after initialization with $w = 1$ and using $\gamma = 1$?
2. What is the Mean Squared Bellman Error?
3. What w results in the value functions that is closest (in least-squares sense) to the target values $B^\pi v_w$?
4. Plot v_w, B^π_w and $\Pi B^\pi v_w$. Explain what is happening. (hint: Refer to Figure 11.3 in the book).

6.3 Deep Q Networks

1. The DQN paper [1] relies, amongst others, on the use of an earlier idea called *experience replay* [2]. What does this trick do that is important for the algorithm?
2. An other important trick is the use of a separate target network that is frozen for periods of time. What does this trick do that is important for the algorithm?

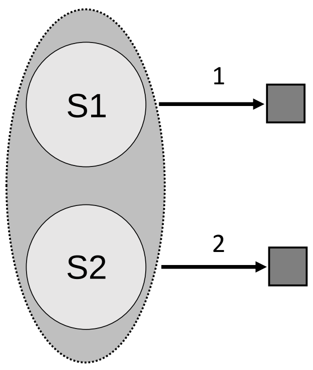

Figure 3: MDP exercise 6.5

6.4 Homework: Coding Assignment - Deep Q Networks

1. Download the notebook *RLLab4_DQN.zip* from canvas assignments and follow the instructions.
2. In the CartPole problem (in the notebook) our state is the current position of the cart, the current velocity of the cart, the current (angular) position of the pole and the (angular) speed of the pole. As these are continuous variables, we have an infinite number of states (ignoring the fact that a digital computer can only represent finitely many states in finite memory). Can you think of a way in which we can still use a tabular approach? Can you think of an example problem where this would not work? Explain why would this work for CartPole and not for the example you mentioned.

6.5 *Exam Question: Function approximation

1. Consider two types of function approximation for scalar s:
   (a) Using "Gaussian" radial basis features $\left(\phi_j(s) = exp(-\frac{(s-\mu_j)^2}{2\times width^2})\right)$.
   (b) Using polynomial features $\left(\phi_j(s) = s^j\right)$.

   Name one advantage of a) compared to b), and one advantage of b) compared to a). Assume the same number of features is used in both cases.
2. With linear function approximation, does gradient Monte Carlo (gradient MC) **always, sometimes, or never** converge to the same solution as semi-gradient TD(0)? Explain your answer.
3. Consider the simple MDP shown on the right. States S1 and S2 are indistinguishable (have the same features). Only a single action can be applied, that always ends the episode. The reward obtained is 1 or 2, respectively. Episodes start in S1 or S2 with equal probability.
   (a) For the shown MDP, what is the minimal mean squared Bellman error? Why?
   (b) For the shown MDP, what is the minimal mean square projected Bellman error? Why?

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[2] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.