Reinforcement Learning — Exercises Lectures 6-9

Instructions

This is the second 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. The graded homework assignments are clearly marked. Please note the following:

- 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 second assignment is *October 4th 2019* and will cover the material of lectures 5-8 (all questions marked 'homework' in this booklet).

Contents

Lecture 5: Value function approximations

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)?
4. 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?
5. 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?

5.3 Preparatory question off-policy approximation

Off-policy learning with approximation is a tricky topic. Before we'll dive into it in the next lecture, 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'.

5.4 Homework: Gradient Descent Methods

gradient mc?

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

$$\sum_{a,s',r} \pi(a|S_t)p(s', r|S_t, a)[r + \gamma\hat{v}(s', \mathbf{w}_t)] \tag{1}$$

result in a semi-gradient method?

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?

Lecture 6: Approximations in off-policy learning

6.1 Homework: Geometry of linear value-function approximation

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

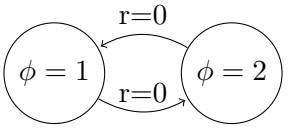

Figure 1: Two-state MDP

1. We can write the Bellman error vector as

$$\bar{\delta}_w = B^\pi v_w - v_w, \tag{2}$$

   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).

Lecture 7: Policy-gradient methods

7.1 REINFORCE

In the lecture, we have seen that introducing a constant baseline b does not introduce a bias to our policy gradient.

$$\nabla J = \mathbb{E}_\tau \Big[\Big(G(\tau) - b \Big) \nabla \log p(\tau) \Big] \tag{3}$$

We now want to consider the variance when introducing a baseline.

1. Derive the optimal constant baseline that minimizes the variance of the policy gradient. Interpret your result.

2. Consider the simple example from the lecture in a bandit setting (i.e. no states):

$$r = a + 2 \tag{4}$$
$$a \sim \mathcal{N}(\theta, 1) \tag{5}$$
$$\nabla_\theta \log \pi(a) = a - \theta \tag{6}$$

   Can you argue what should be the optimal constant baseline in this case? You can use your result from 1.

3. Now, consider a baseline that is not constant, but dependent on the state $b(s_t)$. We want to establish that in this case, the policy gradient remains unbiased. Show that

$$\mathbb{E}_\tau \Big[\sum_{t=1}^{T} \nabla \log \pi(a_t|s_t) b(s_t) \Big] = 0. \tag{7}$$

   Hint: you can use the linearity of expectation or the law of iterated expectation to "decouple" parts of the full trajectory τ.

Lecture 8: Policy gradient theorem

8.1 Limits of policy gradients

To explore the behavior of PGT gradients with different parametrizations we will use a stateless continuous bandit environment. In this environment, the agent performs a single action and receives a single reward before the episode is terminated. Furthermore, we assume that the reward function is known and the policy is represented by a normal distribution parametrized by θ_μ, θ_σ (we will ignore that standard deviation can technically become negative in this case) and a hyperparameter k (we treat k as a design choice and thus its value cannot be changed during optimization):

$$r = a - a^2 \tag{8}$$

$$\pi(a|\theta) = \frac{1}{\sigma(\theta_\sigma)\sqrt{2\pi}} \exp\left(- \frac{(a - \mu(\theta_\mu))^2}{2\sigma(\theta_\sigma)^2} \right) \tag{9}$$

$$\mu(\theta_\mu) = \theta_\mu \tag{10}$$

$$\sigma(\theta_\sigma) = k\theta_\sigma \tag{11}$$

1. Compute gradients of expected reward $\mathbb{E}_a[r]$ with respect to parameters θ_σ and θ_μ
2. How does different choice of hyperparameter k influence the gradient? Do you think that this choice can affect convergence speed? Why (not)?
3. Consider three different parameterizations that represent the same policy $\mathcal{N}(0, 0.1)$. :
   (a) $\theta_\mu = 0$, $\theta_\sigma = 1$, $k = 0.1$
   (b) $\theta_\mu = 0$, $\theta_\sigma = 0.1$, $k = 1$
   (c) $\theta_\mu = 0$, $\theta_\sigma = 0.01$, $k = 10$

   Calculate gradients ∇_{θ_μ} and ∇_{θ_σ} for all of them and perform a single gradient update with learning rate $\alpha = 0.001$. What can you observe when you look at new policies?
4. Plot the gradient directions with different $k = \{0.1, 1, 10\}$ using the notebook we provided. How does choice of k affect the gradient? Which parametrization should be the fastest to converge?
5. If you consider the results you obtained in this exercise, what is the drawback of simple policy gradient?

8.2 Homework: Compatible Function Approximation Theorem

Consider the actor-critic method that optimizes the expected return J using the gradient

$$\nabla J = \mathbb{E}_\tau \left[\sum_{t=1}^{T} \nabla \log \pi(a_t|s_t) q_\pi(s_t, a_t) \right] \tag{12}$$

where π is our parameterized policy (actor) and q_π is the true state-action value under policy π. In the lecture we have seen that we can approximate q_π with a parameterized value function $\hat{q}_w$ which is unbiased if it fulfills the condition

$$\nabla_w \hat{q}_w = \nabla_\theta \log \pi_\theta(s, a) \tag{13}$$

$$\text{e.g. } \hat{q}_w = w^T \nabla_\theta \log \pi_\theta(s, a) \tag{14}$$

1. Show that $\mathbb{E}_a\Big[\hat{q}_w(s,a)\Big] = 0,\ \forall s \in S$. Briefly interpret this result.

2. Evaluate the expectation $\mathbb{E}_a\Big[q_\pi(s,a) - v_\pi(s)\Big]\ \forall s \in S$. The term $A(s,a) = q_\pi(s,a) - v_\pi(s)$ is also known as the advantage function.

3. What do you conclude from the results in 1. and 2.

4. Consider the following policy

$$\pi_\theta(s,a) = \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}} \tag{15}$$

corresponding to a softmax with linear parametrization with respect to the state-action features ϕ_{sa}. Give an expression for the parametrization of $\hat{q}_w$ that satisfies the compatible function approximation theorem.

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.