CSC 498: Assignment 3

Matthew Zhang, Claas Voelcker, Liquan Wang, Prof. Animesh Garg

Released: Wed 11/03/2021 – Due: Wed 11/17/2021

Name, First Name: _____

Student number: _____

Total points: 70 points

To complete the exercise, you can use the tex template provided in the materials github. Insert your answers into the solution space below each question. In case you are unfamiliar with Latex, you may also submit handwritten solutions, but make sure they are clean and legible.

Submit the exercise before 23:59 pm on the due date on quercus. To submit, please bundle your completed exercise sheet and your jupyter notebook into one zip file. Name the zip file `studentnumber_lastname_firstname.zip` and upload it on quercus.

Each student will have 3 grace days throughout the semester for late assignment submissions. Late submissions that exceed those grace days will lose 33% of their value for every late day beyond the allotted grace days. Late submissions that exceed three days of delay after the grace days have been used will unfortunately not be accepted. The official policy of the Registrar's Office at UTM regarding missed exams can be found here `https://www.utm.utoronto.ca/registrar/current-students/examinations`. If you have a compelling reason for missing the deadline, please contact the course staff as soon as possible to discuss hand in. For this assignment, you can hand in up to one week late with no penalty.

For assignment questions, please use Piazza and the office hours, but refrain from posting complete or partial solutions.

I Policy Gradient Theory

We proceed to define two policy classes: *Softmax Policy Class*

Let $\theta \in \mathbb{R}^d$, and $\psi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ be any function from state-action pairs to $\mathbb{R}^d$. Now consider the policy class defined by:

$$\pi_\theta(a|s) = \frac{\exp(\psi(s,a)^T\theta)}{\sum_{a'} \exp(\psi(s,a')^T\theta)} \tag{1}$$

Gaussian Policy Class

Let $\theta \in \mathbb{R}^d$ again, but now $\phi(s) : \mathcal{S} \to \mathbb{R}^d$ be any function from states to $\mathbb{R}$. Then, if $\mathcal{A} = \mathbb{R}$, we define the policy class as follows:

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(a - \phi(s)^T\theta)^2\right) \tag{2}$$

i.e. a Gaussian centered at $\phi(s)^T\theta$ with a constant variance σ^2.

1. What is the score function for the softmax policy class? What is the corresponding policy gradient update equation? (5)

   Solution:

2. What is the score function for the Gaussian policy class? What is the corresponding update equation? (5)

   Solution:

II Policy Gradient Implementation

We will be using the continuous MountainCar task from OpenAI Gym. For more details please see the documentation here: `https://gym.openai.com/envs/Pendulum-v0/`.

1. *Implement REINFORCE, Gaussian policy* (10)

   To start, download all necessary code from github for assignment 3 from the course webpage. Set up your Python environment and make sure you can run jupyter. If you have not done so already, you will need to set up the **Gym** package from OpenAI.

   Run the first section of the jupyter notebook assignment3.ipynb.

   Task 2 contains scaffolding code for a Gaussian policy agent. To be exact, the agent defines it's policy with the following distribution:

$$\pi(a|s,\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(a - \phi(s)^T\theta)^2\right) \tag{3}$$

where σ is a constant, and $\theta \in \mathbb{R}^3$ is a static constant. **Note**: the actions are bounded in [min-action, max-action]; when sampling actions that end up beyond this range, you can either resample the actions or set the actions to be clipped at the maximum/minimum.

In the code cell provided in the notebook, please replace the sections marked *TODO* with your own code. Your agent should be able to find a solution that is roughly optimal. You are free to chose parameters such as the learning rate, number of episodes, if you find this helps performance.

You may find your algorithm does not perform exceptionally well; this is fine, so long as it shows improvement as it trains. We added some plotting utilities to help you visualize this.

Please document the performance of this method, and perform some experimentation with multiple seeds/parameter values.

2. *REINFORCE with Value Function Learning* (10)

In the subsequent part of Task 2, you will need to implement a linear value function estimator parameterized by ξ:

$$\hat{V}_\xi(s) = \phi^T(s)\xi \tag{4}$$

you will use this quantity to rescale the rewards, such that:

$$A_t = r_t + \gamma\hat{V}_\xi(s_{t+1}) - \hat{V}_\xi(s_t) \tag{5}$$

and use A_t when computing the updates for your policy. This parameters ξ are updated with gradient descent, in a separate update immediately after the policy is updated. It will use the update equation from value function approximation, i.e.

$$\xi \leftarrow \xi - \frac{\alpha_V}{T}\sum_{t=1}^{T}\left(r_t + \gamma\phi^T(s_{t+1})\xi - \phi^T(s_t)\xi\right)\phi(s_t) \tag{6}$$

This uses a separate learning rate α_V that you can tune to increase performance.

Please document the performance of this method, and perform some experimentation with multiple seeds/parameter values.

Does this increase or decrease the performance of our model? What could we potentially do to improve the quality of this estimator?

Solution:

III Approximate Q Learning Theory

1. What is the main challenge when trying to solve an MDP using function approximation compared to tabular Q learning? (5)

Solution:

2. Name one advantage and one disadvantage DQN has over Policy Gradient methods like REINFORCE. (5)

Solution:

3. Many Q-learning methods opt to output a vector $Q(s, \cdot) \in \mathbb{R}^{|\mathcal{A}|}$ rather than each value $Q(s, a) \in \mathbb{R}$ individually. What is one advantage and one disadvantage of this approach? (5)

Solution:

4. In supervised learning, models are trained to minimize the difference between their output and data sampled from some fixed distribution. For example, regression with mean-squared error has the following error function: (5)

$$L(\theta) = \mathbb{E}_{(x,y)\sim D}\left[\|f_\theta(x) - y\|^2\right]$$

for some family of models f with parameter θ, and some fixed dataset D. We then optimize for θ to minimize this loss.

This looks similar to the Q-learning problem, but they are not the same. Outline and explain one difference between these two problems.

Solution:

IV DQN

As before, you will find the scaffolding code for a simple DQN agent in the Jupyter Notebook. Fill out all sections marked with ??? to complete the core logic of the DQN agent. Note that you will need to use PyTorch for this exercise, so you might need to refer to the PyTorch documentation (`https://pytorch.org/docs/stable/index.html`).

1. *Vanilla DQN Implementation* (10)

   For the first subtask, complete the vanilla DQN implementation. The train method returns all losses computed during the training. Visualize the losses using matplotlib and discuss the results. As always, only modify the code in the marked places and do not add or remove any libraries or dependencies. Your agent might not converge to a stable solution, if so, vary the parameters to investigate their impact on the performance.

   Note that the NN based training can take some time, so it is acceptable if you do not achieve very good performance with limited computational capacity. In addition, the agent is currently not using experience replay, which is sometimes necessary for the agent to converge properly. As a bonus (5 points) you can implement an additional agent using experience replay.

2. *Double DQN Implementation* (10)

   For the second subtask, complete the Double DQN implementation. Note that you have to change several parts of the previous algorithm, all relevant sections are marked. The code logic is mostly similar to before, but needs some additional variables, so several parts only have to be tweaked slightly.

   Run the algorithm, plot the loss function and discuss differences to the previous implementation.