Reinforcement Learning
Exercise 4

November 4, 2018

In this exercise we will implement a policy gradient estimation solution for the cartpole environment and compare it to the actor-critic approach.

1 Policy Gradient

In this part of the exercise you will be implementing policy gradient for the cartpole environment with a **continuous action space**.

1.1 Environment with continuous action space

The environment used for this exercise is a modified version of CartPole. The standard CartPole uses a discrete action (applying force of either -10 or 10), while this version can use an arbitrary value between -25 and 25.

Task 1 Implement policy gradient for the Cartpole environment with continuous action space. Use `agent.py` for implementing the reinforcement learning itself (for example, the agent and policy classes). Instantiate those classes in `cartpole.py`, similarly to how it was done in Exercise 1. Use a small neural network as your policy (one hidden layer with 20 neurons makes a good starting point).

Hint: The `agent.py` file from the first exercise session contains some useful code. Can you recognize the algorithm used there? If you want to, feel free to use it as a starting point for your implementation.

Hint: Your policy should output a probility distribution over actions. A good (and easy) choice would be to use a normal distribution (`from torch.distributions import Normal`).

Hint: Using a continuous action space will take more time to train than it did previously (for the basic discrete case).

Question 1 (and also a hint): Normalizing your discounted rewards to zero mean and unit variance should improve the stability of your training. Why is it so?

Question 2 Compare using a constant variance (use $\sigma = 0.5$) to using decaying exponentially decaying variance and to making variance an output of your neural network. Which of these approaches works best? Which takes longer to train? Explain your answers.

2 Actor critic

Task 2 Revisit the policy gradient solution for the continuous cartpole from task 1 and implement the actor-critic algorithm. In the initial setup, perform updates at the end of each episode. Put your implementation in a separate file called `ac_cartpole.py`.

Hint: Computing both the policy and the value function may require a larger neural network. If your model doesn't learn, you may have to increase the number of neurons in the hidden layer.

Question 3 Compare the result to the policy gradient approach. What are the benefits of actor-critic? Which algorithm learns faster? **Remember to run both programs multiple times and average the results.**

Task 3 Plot a heatmap of critic's value function estimation for different x and θ (position and angle) values. Explain the plot shortly.

Task 4 Update the actor-critic code to perform TD updates every 10 timesteps, instead of updating your network at the end of each episode. Make sure to handle the end of each episode correctly (as in the value iteration exercise, value of the terminal state is 0).

Question 4 Does performing more frequent updates improve the total time needed to learn to balance the pole? **Remember to run both programs multiple times and average the results.**

Question 5 What are the benefits of actor-critic methods compared to conventional policy gradient methods.

Question 6 What are the advantages of policy gradient methods compared to action-value methods such as Q-learning? When would you use Q-learning? When would you use a policy policy gradient method?

3 Submission

Your report should contain answers to questions posed in the text and an analysis of the results. Furthermore enclose the parameters you used in your code to achieve the outcomes you describe in your answers. Include trained models (the files should be saved when training is finished) and the code used for the exercises.

Deadline to submit your answers to part 1 and part 2 of this exercise is 18.11, 23:55. Please remember to submit both parts at the same time.

If you need help solving the tasks, you are welcome to visit the exercise sessions on Monday (12:15-14:00) and Wednesday (10:15-12:00).

Good luck!