

Recall from that in a previous exercise we implemented DQN, where we used function approximation to learn on continuous state spaces. To extend the action space to continuous action spaces, one typically uses policy gradient methods. In this exercise, we solve continuous action versions of Cart Pole using Reinforce.

## Exercise 1: Reinforce

- 1.1 First, we need a parameterized policy  $\pi_\theta$  that gets as an input a state  $x$  and returns a distribution of possible actions  $\pi_\theta(\cdot \mid x)$ . Similar to the previous exercise, we also use a neural network for this. The implementation uses a neural network to predict the mean  $\mu$  of a gaussian, whereas the standard deviation  $\sigma$  is fixed. Depending on  $\sigma$ , the noisiness of the policy increases. Similar to the  $\epsilon$ -greedy policy, this is needed to explore new actions. Read the `Policy` class in `policy.py` to understand it, and run the file to generate a plot of the action distribution.
- 1.2 Now, we want to implement Reinforce. Complete the function `reinforce` provided in `reinforce.py`. Note that to save time and because Reinforce is not a very stable algorithm, we set the training of the episodes to 5000 (on the computer this is tested, it took around 5 minutes). Feel free to adjust the value.
- 1.3 (Bonus) Additionally learn a value function approximation and use it as a baseline in `reinforce`.