Reinforcement learning project

Elie Kadoche

April 24, 2024

This is a reinforcement learning project on policy gradient methods. In this document you will find the instructions. More explanations will be given in the board during the practical sessions. Do not hesitate to ask for help if you need to. The overall project will be noted on 20. The notation is given for information and may slightly change. Your solutions will be gathered in a **document** and in the **code**.

- In the **document**, you will write your answers, thoughts and approaches for each Section. Even if you do some mistakes, you can describe them and explain how you corrected them. It will be taken into account in the notation.
- In the **code**, each time there is a comment "⟶ TODO", there is some code missing that you need to write. Be mindful of the code quality. Each line of code should be associated to a comment where you describe what you do.

Setting up

You need to install Python 3.8+ on your machine. You need to install the required packages with the following commands.

```
pip install gymnasium
pip install numpy
pip install scipy
pip install torch
```

1 Cartpole environment (/2)

Your objective is to understand how the cartpole environment works and what is the problem we want to solve. The environment is implemented in the file `./src/envs/cartpole_v0.py`.

You do not need to write any code for this Section. You need to write in a document whare are the states, the rewards and the actions. Write the Markov Decision Process associated to the problem.

You can read the Section 5 of the `./documents/cartpole.pdf` document for help. And you can use script 0 to test a random policy on the environment.

```
python ./0_test_env.py
```

2 Policy deep neural network (/2)

In the file `./documents/neural_network.py`, a simple PyTorch implementation of neural network trained with the gradient descent method is given. You can execute it with the following command, to see how training is performed.

```
python ./documents/neural_network.py
```

We want to build a policy $\pi_\theta(a|s) = P(a|s, \theta)$ that gives the probability of choosing an action a in state s. The policy is a deep neural network parameterized by some weights θ. The policy is also referred to as "actor" and is defined the file `./src/models/actor_v0.py`.

Depending on what you understood from the cartpole environment, you need to change in the actor code the `input_size` and `nb_actions` variables. In script 1, you need to find how to select an action from the output of the policy. Then, you can use the script 1 to test the policy. How does the policy performs on cartpole? Why?

```
python ./1_test_model.py
```

3 REINFORCE algorithm (/6)

We want to find the parameters θ that maximize the performance measure $J(\theta) = \mathbb{E}_{\pi_\theta}[G_0]$ with $G_t = \sum_{k=0}^{\infty} \beta^k r_{t+k+1}$ and $\beta \in [0,1]$ being a discount factor. To do so, we use the gradient ascent method: $\theta_{k+1} = \theta_k + \alpha \nabla_{\theta_k} J(\theta_k)$ with α being the learning rate.

The performance measure depends on both the action selection and the distribution of states. Both are affected by the policy parameters, which make the computation of the gradient challenging.

The policy gradient theorem gives an expression for $\nabla_\theta J(\theta)$ that does not involve the derivative of the state distribution. The expectation is over all possible state-action trajectories over the policy π_θ:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)].$$

In the REINFORCE algorithm, we use a Monte-Carlo estimate over one episode, i.e., one trajectory:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty} G_t \nabla_\theta \ln \pi_\theta(a_t|s_t).$$

Your objective is to complete script 2 to train the policy according to the REINFORCE method. To solve the problem, you need to achieve a cumulative reward of at least 500 when training the policy.

1. Write the code to compute the discounted sum of rewards (/1).
2. Write the REINFORCE loss for the policy (/3).
3. Change the α and the β parameters to solve the problem (/1).
4. Find a good metric to decide when the training should be stopped (/1).

```
python ./2_reinforce.py
```

After each training, you can use script 3 to test your policy. The cumulative reward should be equal to 500.

```
python ./3_test_model.py
```

4 Markov property (/5)

For this Section, uncomment the corresponding lines of scripts 1, 2 and 3 at the top of the files. In this Section, we work on a slightly different version of the environment.

```
# -----------------------------------------
# ---> TODO: UNCOMMENT FOR SECTION 4 ONLY
env = CartpoleEnvV1()
policy = ActorModelV1()
actor_path = "./saved_models/actor_1.pt"
# -----------------------------------------
```

In the `./src/envs/cartpole_v1.py` file, the accelerations are removed from the states. Because the observation size is then of size 2, the actor is updated accordingly in the file `./src/models/actor_v1.py`. Test and train back a model with scripts 1 and 2. Does it converge? Why?

Find a way to modify the states in `./src/envs/cartpole_v1.py` file, without using the accelerations, so that the environment becomes Markovian again. Update the `./src/models/actor_v1.py` network accordingly and train the model back again with the REINFORCE algorithm. Does it converge? Why?

```
python ./1_test_model.py
python ./2_reinforce.py
python ./3_test_model.py
```

5 Open questions (/5)

You can choose to work on one or several of the following subjects. You are also encouraged to work on other subjects, if you have any ideas.

5.1 Different environments (moderately easy)

You can find other reinforcement learning problems with their corresponding environments and adapt your REINFORCE training script and your actor model to solve it. You could also create your own simple reinforcement learning problem, determine the sates, the actions, the rewards and code your own environment to solve it.

5.2 REINFORCE with baseline (moderately difficult)

To reduce the variance of the REINFORCE method, we can subtract to G_t a baseline $v(s_t)$. With $v(s)$ being the estimate of the state value given by another neural-network, called critic. Based on the actor network, build the network for the critic in the `./src/models/critic.py` file. And based on the REINFORCE script 2, build the `./4_reinforce_baseline.py` training script. Use script 5 to test your model.

```
python ./4_reinforce_baseline.py
python ./5_test_model.py
```

5.3 Actor-critic algorithm (moderately difficult)

Explain and code the one step actor-critic algorithm.

```
python ./6_actor_critic.py
python ./7_test_model.py
```

5.4 Continuous actions (difficult)

In this project, we considered a discrete action space. How could we use policy gradient methods for continuous action spaces? You can give your ideas, you can code them, and ultimately, you can even find an environment with a continuous action space and try to solve it.