SDM366 Optimal Control and Estimation

Course Final Project
Spring 2024
Due June 15, 2024

Requirement and Deliverables:

- Students should work in a team of two or three students.
- Each team should finish both of the following two problems. You will be provided with a runnable simulation of a biped robot (RL policy) and a guiding of building a RL training framework for pendulums.
- Some of the questions are open-ended, and you may need to do some investigation to finish the project problems. You are expected to read papers and search and re-search online.
- PLAGIARISM IS NOT ALLOWED!

Reports: Each team should submit one report along with your solution codes. The report should provide all the background information, problem formulation (as control or optimization problem), existing methods, your proposed solution, your results, and discussions.

Problem I: State Estimation for a biped robot

State estimation is a crucial aspect of robotic control systems. It involves using sensor data and state space models to estimate the internal states of a system, which are often not directly measurable. One of the most widely used techniques in state estimation is the Kalman filter. Kalman filter is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, to produce estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

In this project, you will implement a state estimator for a bipedal robot. The state estimator's primary role is to provide observation inputs for the control policy of the bipedal robot, precisely, provide the linear velocity of the floating base. The control policy is already given, and you can read README.md for details. Without accurate observation, the provided bipedal robot will not function correctly and will fall down.

1. Figure out the State Space Model of a biped robot. You might need to refer to the GitHub repo of Cheetah-Software, which offers an example of how to build this model for a quadruped robot. In this paper, please read reference 1 and derive the state space model for biped state estimation problem.
2. Implement the state estimator based on the State Space Model derived in Part 1. Use proper state estimation algorithms to estimate the `base_lin_vel` and use it as the RL policy input. You are expected to submit runnable `.py` or `.ipynb` files, in which the biped robot

successfully walks with your estimator. Besides, you need to explain the state estimation algorithm in details, including detailed derivations and essential explanations.

3. Compare your state estimator results against the true state read directly from the simulator. Try different covariance matrices to see how they impact your estimation error. Discussion your results.

Tips:

(a) Run the given code and understand what kinds of states are provided by the simulator and what kinds of states need to be estimated. This will help you understand the objective of your estimator and how the biped robot works.

(b) Read the given resources and references, and you might need to search for some useful blogs or papers to help you understand them. This will help you learn how to build a State Space Model for a quadruped robot. The biped robot has similar properties to a quadruped robot, so you will need to build a model for the given biped robot.

(c) Use the model you build to implement your state estimation algorithms. Finish the code and test it on the given biped robot. If successful, the biped robot should be able to walk forward without falling. You can switch to simulation mode to compare your results by using USE_SIM.

References:

1. M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots - Consistent fusion of leg kinematics and IMU," Autonomous Systems Lab, ETH Zurich, Switzerland.

2. GitHub Repo: [mit-biomimetics/Cheetah-Software]

3. M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State Estimation for Legged Robots on Unstable and Slippery Terrain," in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, 2013, pp. 6058-6064.

Problem II: RL training of inverted singular and double pendulums

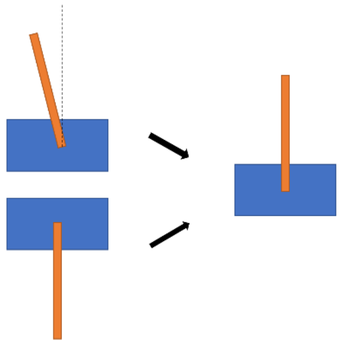
Reinforcement Learning (RL) has become a powerful tool in the field of robotics and control systems. It involves training an agent to make a sequence of decisions by rewarding desired actions and punishing undesired ones. One classic problem used to study and develop RL algorithms is the control of inverted pendulums, which are systems that are inherently unstable and require continuous balancing.

In this project, we focus on the RL training of two types of inverted pendulums: the single inverted pendulum and double inverted pendulum. The single inverted pendulum consists of a single rod pivoted at one end, requiring control inputs to keep it balanced around an upright position. The double inverted pendulum, on the other hand, consists of two rods connected end-to-end, creating a more complex and challenging system due to its increased degrees of freedom and non-linear dynamics.

The primary objective of this problem is to apply the proper RL algorithm to train control policies that can successfully balance both types of inverted pendulums. This involves defining appropriate state and action spaces, designing a reward function that encourages stable balancing, and employing RL algorithms such as the vanilla policy gradient or the Proximal Policy Optimization (PPO) algorithm to learn the control policies.

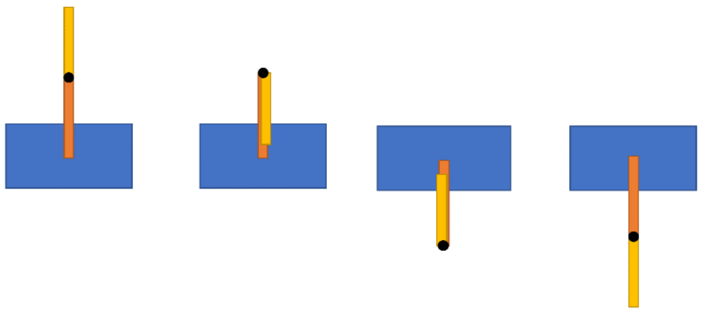
For each of the following two problems, please (1) You are expected to explain the RL algorithm you used in the report, with clear problem statement. (2) Provide necessary discussion about the results, like comparing different algorithms or analyzing the rewards graphs. (3) Provide runnable codes should also be provided in your submission (save your trained model).

1. For the inverted pendulum, you need to finish the following two sub-problems.
    a) The pendulum starts with a slight inclination. The goal is to stabilize the pendulum in the upright position by controlling the movement of the cart.
    b) The pendulum begins in the downward position. The objective is to swing the pendulum up and stabilize it in the upright position.



Objective 1: has two sub-problems

2. (Bonus) For the inverted double pendulum, you need swing and stabilize the double inverted pendulum from a downward initial phase, to any other phase (double inverted pendulum has more than phases). <u>The submission requirement is the same as question 1.</u>

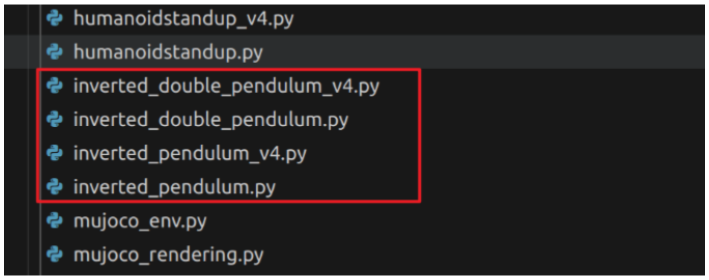


Objective 2: double pendulum has multiple phase

Tips:

(a) Read the inverted_double_pendulum and inverted_pendulum provided by OpenAI gym (gymnasium), understand the action space and observation space of the problems.

(b) Read the given simulation codes and the codes in `gymnasium`, understand how to run the simulation of the two pendulums and how `gymnasium` run an agent for RL training. During this process, you might also need to watch some online courses on the Internet about the basic terms of RL (like agent, policy, value function, etc.) and how to train a policy for control.



```
humanoidstandup_v4.py
humanoidstandup.py
inverted_double_pendulum_v4.py
inverted_double_pendulum.py
inverted_pendulum_v4.py
inverted_pendulum.py
mujoco_env.py
mujoco_rendering.py
```

Model codes of pendulums in `gymnasium`

(c) Implement an RL algorithm by yourself (*using package like Stable Baselines3 will save your time quite a bit*), and train the required pendulum models. Save the training results, including models, learning rates, rewards, and other training records.

References:

1. Simulation reference: Gymnasium Documentaion

2. Popular RL package: stable-baselines3

3. OpenAI RL Tutorial: Spinning Up

4. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.