Reinforcement Learning
Exercise 6

November 19, 2019

In this exercise we will implement a feedback controller (iterative Linear Quadratic Regulator, iLQR) and a model (Kernel Ridge Regressor, KRR) for optimizing a trajectory for the Inverted Pendulum environment.

This assignment is optional — solving it will grant you extra course points, but you can get the full grade without submitting it.

1 Introduction

1.1 The model

In Model-Based Reinforcement Learning we learn an approximated model of the environment by gathering data from it. This approximated model is then used to leverage the cost of sampling from the environment for gathering new samples.

For this exercise, we provide a model based on a KRR, where the hyper-parameters are already given from prior knowledge. The model is optimized using Ridge Regression as:

$$\min C(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{Y}_i - \mathbf{w}^\top \mathbf{X}_i)^2 + \frac{1}{2}\lambda||\mathbf{w}||^2 \tag{1}$$

where $\mathbf{w}$ are the hyper-parameters, λ is the weight-decay, and $\mathbf{Y}_i$ are the targets for each of the inputs $\mathbf{X}_i$. For our model KRR, the inputs are mapped to a feature space. The mapping function is an RBF which you have seen in previous exercises.

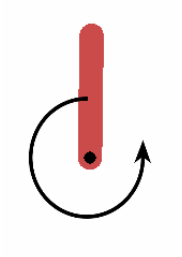


Figure 1: The Inverted Pendulum environment.

1.2 The problem to learn

For this exercise we will be using the Inverted Pendulum environment. The problem is stated as an optimal trajectory optimization by using a KRR as model, and the iterative-LQR as feedback controller.

The Inverted Pendulum control problem consists of an inverted pendulum starting from a random position where the goal is to swing the pole up and keep the balance in the upright position. The action space is defined as:

$$\mathbf{u} = (f)^\top, \tag{2}$$

where f is the force applied. The states of the environment are slightly modified from the standard `gym` environment:

$$\mathbf{x} = (\theta,\ \dot{\theta})^\top. \tag{3}$$

The inputs to the approximated dynamics model are the state-action pairs: $\mathbf{X} = (\mathbf{x},\ \mathbf{u})^\top$. The targets for our model are the difference between consecutive states $\mathbf{Y}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$.

2 Optimizing a trajectory using iLQR

You should have now knowledge from the lectures about the iterative-LQR. For reference on LQR check the PDF in MyCourses: *Platt, Introduction to Linear Quadratic Regulation*.

Task 1 – 20 points. Design a cost function for the inverted pendulum that can be used with the iLQR. Decide the state and action cost matrices depending on what you want to penalize. Derive the gradient and Hessian of your cost function **analytically** (we will implement this in Task 2) and show them in the report.

Hint: For this exercise you will need to install *autograd*, `pip install autograd`.

Question 1 – 10 points. Could iLQR handle a non-linear cost function? Argue **why/why-not?**

Task 2 – 20 points. Implement the cost function, gradient and Hessian that you have previously designed in `iLQR.py` and `main.py`.

Hint: Use the `normalize_angle()` provided by `utils.py` to normalize the θ of the state.

Hint: The dimensions of the cost gradient and Hessian matrices should be the same as defined in the `__init__` function (excluding the time dimension).

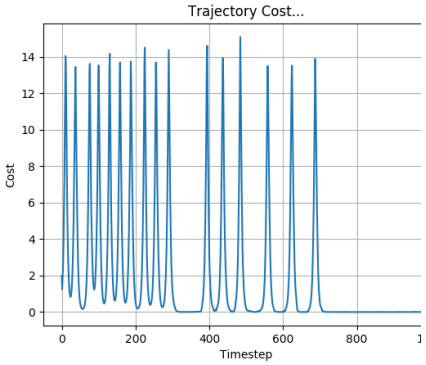
Question 2 – 10 points. Discuss the reason behind the terms included in your cost function.

Task 3 – 10 points. Implement the required steps for performing the iLQR update in `main.py`. Run your code and observe how the cost is reducing over time and the optimal actions are computed. Figure 2a shows an example of the expected behavior. Use $\alpha = 0.99$.

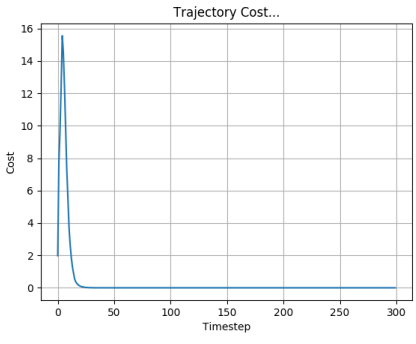
Task 4 – 10 points. Modify the code in `main.py` and `iLQR.py` to use the dynamics equation of the system instead of the approximated model. Figure 2b shows expected results.

Question 3 – 10 points. How could the uncertainty of the approximated model be taken into account for performing predictions? You can have a look at the KRR model used in `main.py` or literature on other possible models that could be used.[1]

Question 4 – 10 points. How does the horizon for performing predictions using the approximated model affect the learning?



(a) Example of cost using the approximated model after 1000 timesteps.



(b) Example of cost using the true dynamics model after 300 timesteps.

Figure 2: Cost to reach the optimal state from position at a given timestep.

[1] For reference check the documentation: `https://scikit-learn.org/stable/modules/kernel_ridge.html`

3 Submission

Your report should be submitted as a **PDF file** as RL_Exercise3_*LastName_FirstName*.pdf. The report must include:

1. **Answers to all questions** posed in the text.
2. The **cost trajectory** for Task 3 and Task 4.

In addition to the report, you must submit as separate files, in the same folder as the report:

1. Python code used to solve **all task exercises**.
2. NumPy file `list.npy` of the trajectory with the optimal actions.

Please, avoid uploading unnecessary files such as self-generated folders (e.g. __pycache__, __MACOSX).

Deadline to submit your answers is **2nd of December 2019, 11:55 am** (in the morning before the Monday exercise session).

If you need help solving the tasks, you are welcome to visit the exercise sessions on Monday, Tuesday and Wednesday.

Good luck!