# Deep RL Assignment1

Mufei Li mufei.li@nyu.edu

May 25, 2018

## 1 Warmup

We use the default setting in run_expert.py for collecting data. The setting for behavioral cloning is

- **random seed**: 0
- **batch size**: 64
- **learning rate**: 0.001
- **number of epochs**: 10
- **shuffle**: The dataset is reshuffled at every epoch.
- **optimizer**: Adam with its default setting in PyTorch 0.4.0
- **model**: a fully connected neural network with one hidden layer followed by a leaky ReLU using its default setting in PyTorch. Let an observation of the environment being $d$-dimensional. The number of parameters for the hidden layer is set to be $20d$.

The learning curve of the training plotted in TensorBoard is presented below:

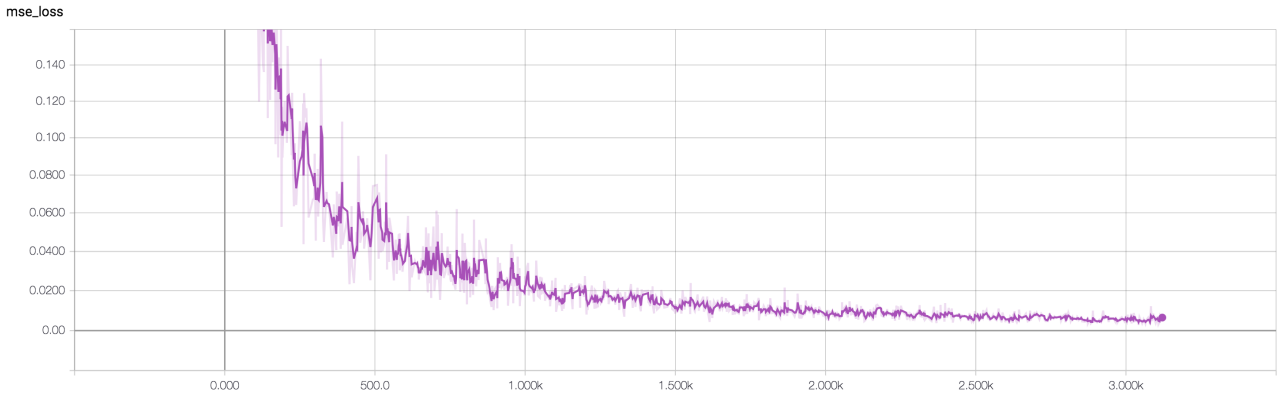


Figure 1: The learning curve for warmup.

## 2 Behavioral Cloning

### 2.1

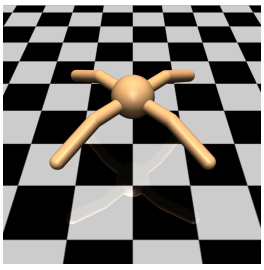

Figure 2: Ant-v1

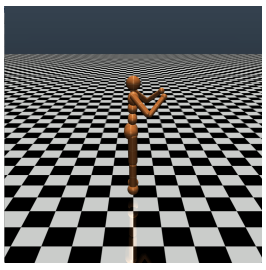

Figure 3: Humanoid-v1

The tasks we used for this question are **Ant-v1** and **Humanoid-v1**. The setting for the experiments is the same as in the warmup section. The statistics for both the expert and the trained (cloned) policy in the table[1] below are calculated based on 20 rollouts.

| Tasks | Observation Space | Action Space | Expert Mean | Expert Std | Cloned Mean | Cloned Std |
|---|---|---|---|---|---|---|
| Ant-v1 | 111 | 8 | 4836.24 | 123.77 | 4639.13 | 536.92 |
| Humanoid-v1 | 376 | 17 | 10417.81 | 57.65 | 278.59 | 65.72 |

The performance we obtained in Ant-v1 is comparable to that of the expert policy, but it is clearly not the case for Humanoid-v1.

### 2.2

We experiment with different **random seeds**. Random seed, being one of the most influential hyperparameters, affects both the weight initialization for PyTorch as well as environment initialization in OpenAI's Gym. To my knowledge, deep reinforcement learning (DRL) approaches, particularly policy gradient methods, are very sensitive to random seeds. On the contrary, imitation learning is intuitively more of the style of supervised learning despite that the tasks in hand may be the same. It is therefore interesting to explore how random seeds affect the performance of behavioral cloning.

As the performance of behavioral cloning on Humanoid-v1 is significantly worse than that of the expert policy in the previous part, we choose this task for our experiment. During the experiment, we fix a collection of demonstrations from the expert policy and use exactly the same setting of the experiment as before except that we try 10 random seeds: $0, 1, \cdots, 9$. For each choice of random seed, we set the seed for both PyTorch and Gym environment. The results of the experiment are presented below:

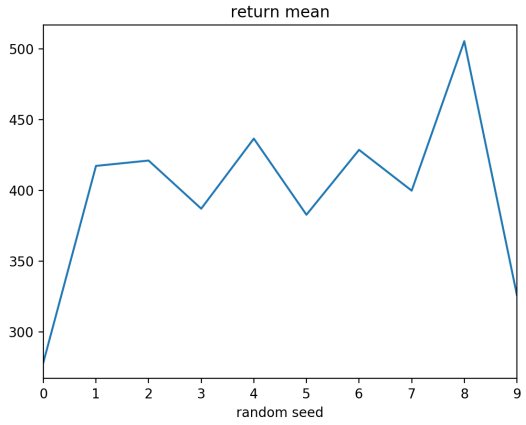


Figure 4

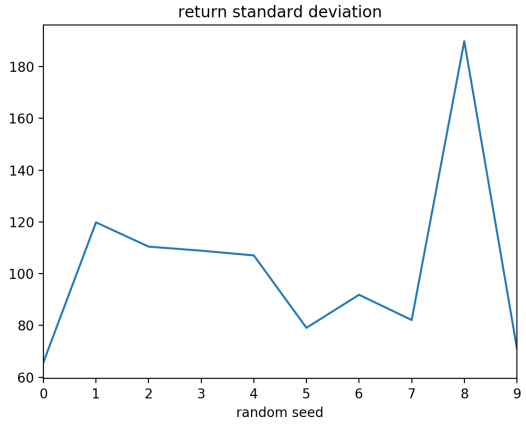


Figure 5

From the figure, we can clearly see a fluctuation for both the mean and the standard deviation of the returns as we vary the choice of random seed. However, the model does appear to work to some extent for all choices of random seed, suggesting that this approach is more robust than some existing DRL algorithms. A more thorough examination should require experiments for a bigger range of random seeds.

## 3 DAgger

We examine the performance of the expert policy, behavioral cloning and DAgger on the task **Walker2d-v1**. The hyperparameters are chosen as before with the random seed fixed to be 0. We start with a collection of data from 20 rollouts of the expert policy and add additional data from 20 rollouts of the expert policy in each iteration of the DAgger algorithm.

[1]The numbers for the observation space and the action space simply indicate the dimension of the spaces.

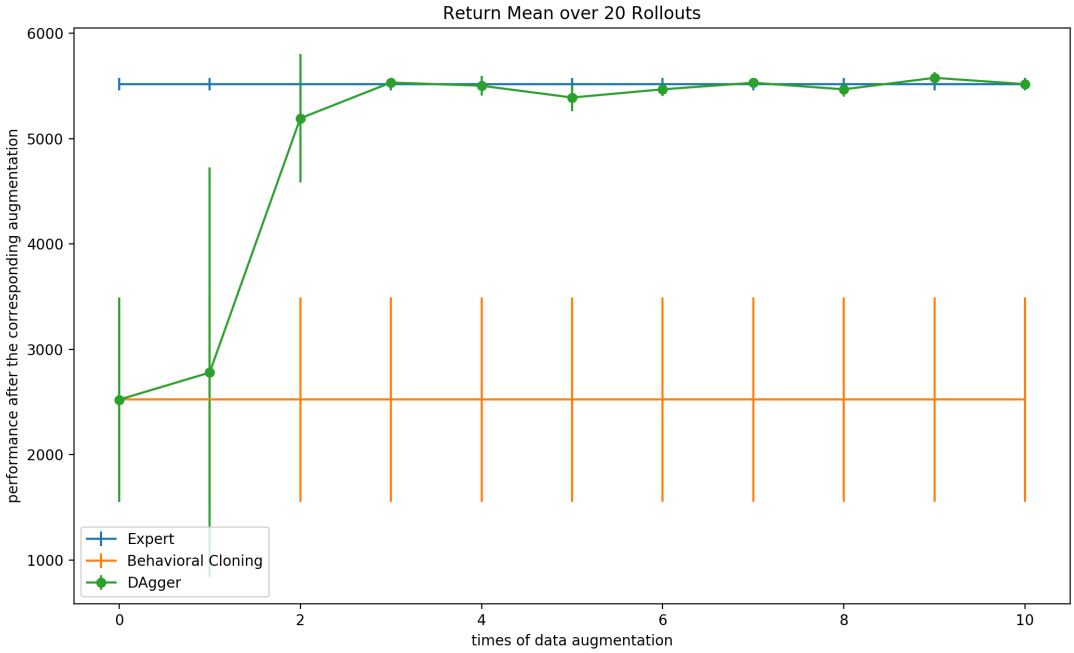


Figure 6: Comparison of the expert policy, naive cloned policy and policy trained using DAgger, with the vertical bars indicating the standard deviations. The performance of the expert policy and the clone policy are unrelated to the number of data augmentation performed.