Imitation Learning

Report Author: Kevin (Yu-Teng Li)

[Project Spec] [Code Repo (private)]

Abstract

This project experiments with methods that study decision making as a supervised learning problem, namely Imitation Learning. In part 1, we implement a Behavioral Cloning model that achieves 40.1% of the performance of a human expert policy. In part 2, we implement DAgger, i.e. data aggregation, which is able to outperform a human expert within 8 iterations with only a 3-layer Multilayer Perceptron (MLP) backbone. Both experiments are performed in OpenAI Gym (MuJoCo) environments, including Ant, Half Cheetah, and Hooper.

Part 1 – Behavioral Cloning

We evaluate the Behavioral Cloning (BC) model in the Ants environment and have obtained the following results over 10 rollouts, with each trajectory having 1000 steps. The BC model is trained for 18,000 train steps with 3-layer MLP, MSE loss and Adam Optimizer with learning rate of 0.005.

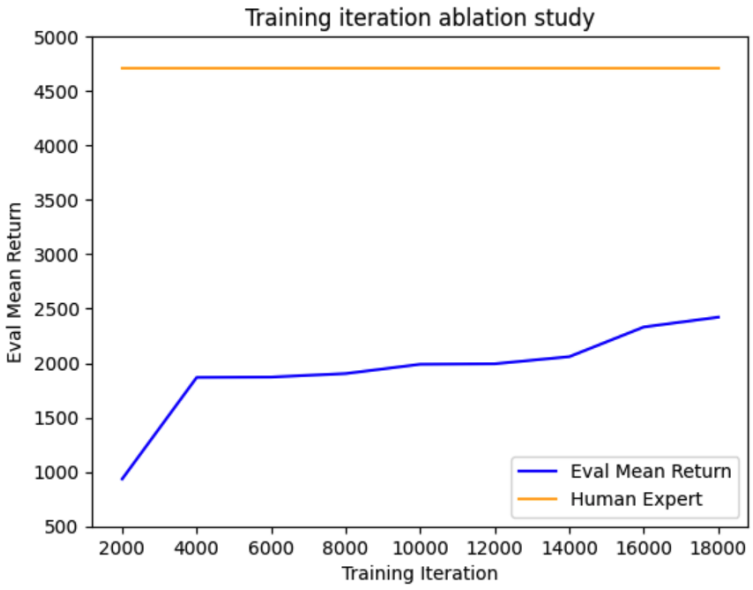
Environment	Eval Return Mean	Eval Return Std	Expert Return Mean	Policy/Expert Performance
Ants	2422.08447	238.83371	4713.65332	51.38%
Half Cheetah	2293.76807	225.23464	4205.77832	54.54%
Hopper	276.61502	7.99217	3772.67041	7.3%

With behavioral cloning, almost all runs are able to achieve human expert policy return in training time, regardless of the hyperparameters. Yet, it is reasonable to speculate that BC can easily overfit the data even with a small learning rate and suffers from the distribution shift of the policy in the evaluation stage. To further investigate the effect of some critical hyperparameters, we ran some ablation studies on a variety of *training iterations* and *learning rates*, using the Ants environment.

Ablation Study

Train Steps (num_agent_train_steps_per_iter)

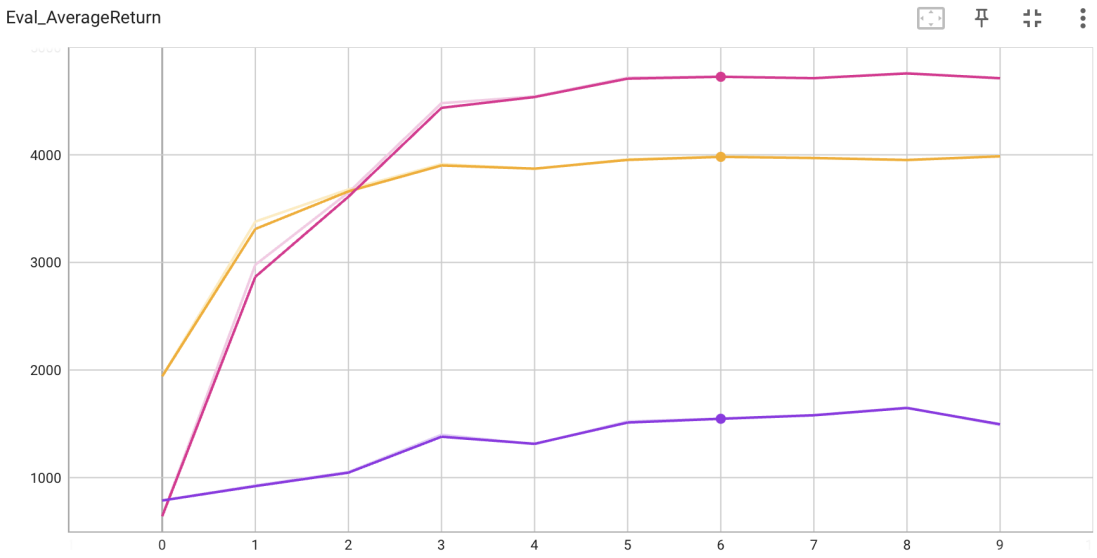
Since during behavioral cloning, the model only samples once from the expert data pool, perhaps the most intuitive way to optimize the model is to allow more training steps. Hypothetically, the more gradient descent steps the model makes, the better the model should perform. After 9 runs of different steps, we indeed observe an increase of mean eval return as the number of steps increases.

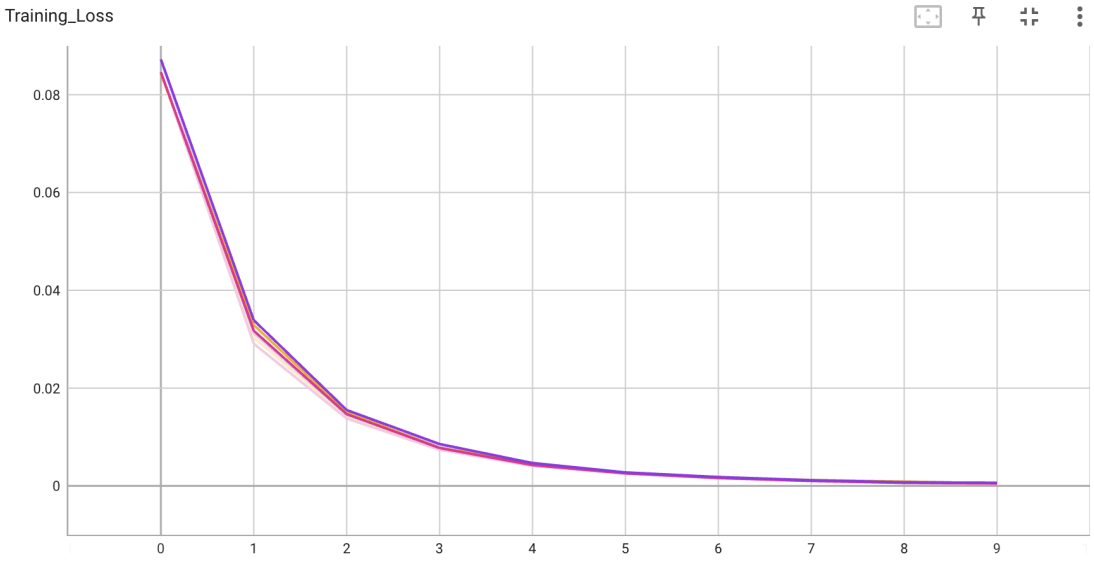


Part 2 – DAgger

DAgger, i.e. data aggregation, method mitigates the distributional shift problem by introducing policy-generated data during the training process. Below we run the same 3-layer MLP model in the same set of environments – Ants, Half Cheetah, and Hopper. All results are averaged over 10 runs, with each trajectory of 1000 steps.

Environment	Eval Return Mean	Eval Return Std	Expert Return Mean	Policy/Expert Performance
Ants	4710.125	81.675	4713.65332	99.9%
Half Cheetah	3987.8959	84.04	4205.77832	94.8%
Hopper	1487.9367	778.649353	3772.67041	39.4%

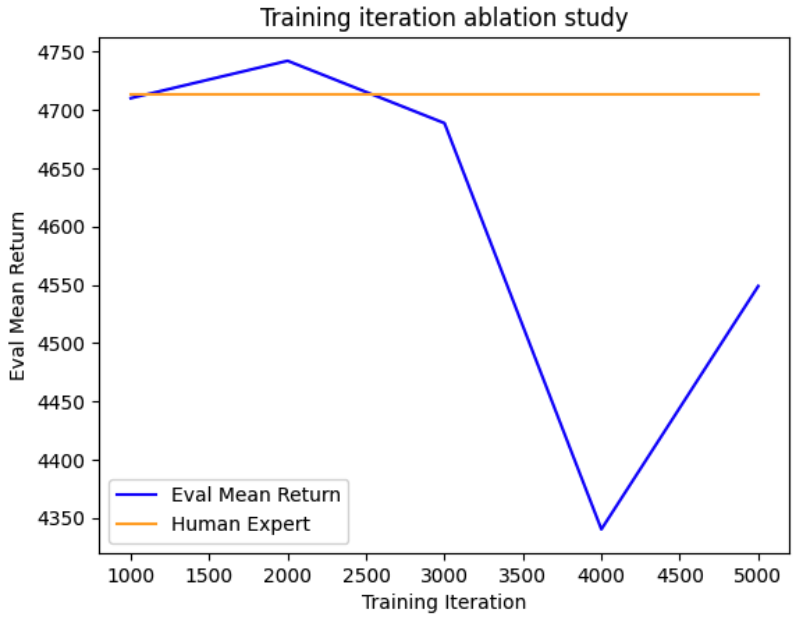




*Eval Average Run and Training loss (pink: **Ants**, yellow: **Half Cheetah**, purple: **Hooper**)*

Ablation Study

To compare the result directly with Behavioral cloning, we also chose Training Steps as a hyperparameter for the ablation study and observed that in DAgger, the more training steps does not necessarily correlate with higher eval mean return, which is vastly different from the phenomenon in behavioral cloning.



Conclusion

In this report we studied two methods of imitation learning – Behavioral cloning (BC) and DAgger. After multiple runs of experiments and some (rough) ablation studies, we can reasonably speculate that BC does not work for sequential decision making problems, which aligns with common belief, and that DAgger can work reasonably well given the right amount of exposure to policy-generated data during its training. In the two ablation studies on "train steps", the results indicated that higher train steps in BC lead to better reward returns, whereas in DAgger, higher steps do not necessarily increase the reward. One interpretation would be that, the more gradient descent steps the model makes, the likelier it might overfit, and the mistakes made by an overfitting model will compound as the policy produces data for the coming iterations, deteriorating the performance substantially. Yet it is also worth noting that even with the same set of hyperparameters, the performance can vary by a nontrivial amount every time the model is run, which also conveys that treating sequential decision making as a supervised learning problem might not be ideal in most cases.