HW3 — 6.884: Computational Sensorimotor Learning

Cameron Hickert

May 2020

1 Problem 1: Learning a policy with Behavioral Cloning

The source code is included (for this problem and the next) in Python files in the zipped folder. To execute the code for this problem – which learns a policy with behavioral cloning, generates training loss plots, calculates the average L2 distance between the object location and the goal location, and creates videos showing evaluation of the learned policy – run the p1.py file. For this problem and the next, I used parts of the PPO code my group has been using in our project, since it is a cleaner version of the PPO algorithm I implemented in HW1.

For the policy I used an actor-critic model, where the actor is a multilayer perceptron feed-forward neural network with two hidden layers of size 128 and 64, respectively, and the critic is a multilayer perceptron feed-forward neural network with two hidden layers of size 64 and 32, respectively. Both used tanh activation functions.

For behavioral cloning the agent learns the policy produced by maximizing the expected (over the expert dataset) log probability of selecting the action a chosen by the expert given state s. We can write this as:

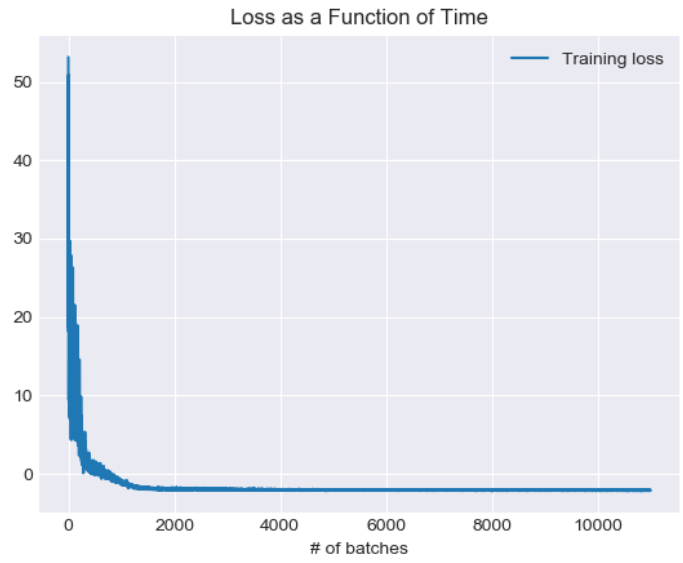
$$E_{(s,a)\sim D}[log\pi_\theta(a|s)] \tag{1}$$

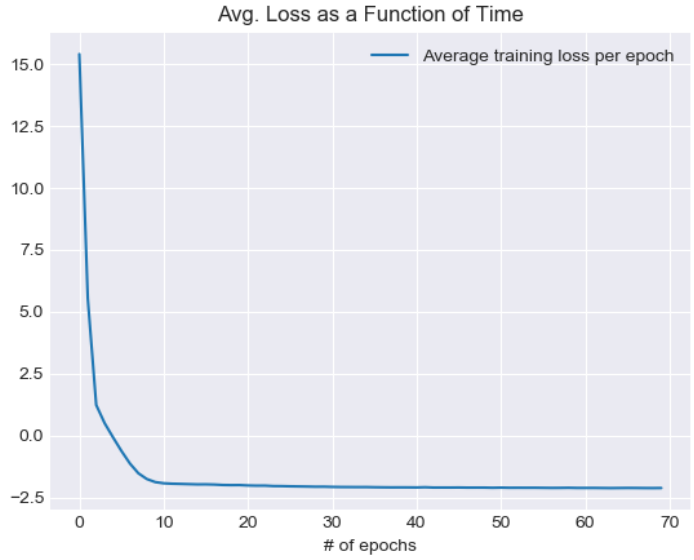
where θ represents the parameters of the policy π, and D is the expert dataset. Thus, the loss we design is the negative of this, where we sample and take the mean to approximate the expectation:

$$L_{BC} = -\frac{1}{n}\sum_{i=1}^{n}[log\pi_\theta(a|s)] \tag{2}$$

which represents an average (as a means of approximating the expectation) where, as before, the $[log\pi_\theta(a|s)]$ is the log probabilities over the actions given the state. This average is calculated using n state-action pairs drawn from the expert dataset.

Two training plots showing loss as a function of time are below.





From these charts, we can see that the model successfully learns to roughly "clone" the behavior of the expert. After evaluating the learned policy on 100 episodes, I found the average L2 distance between the object location and the

goal location to be 0.205, which is well within the 0.22 bound suggested in the problem statement.

The video showing evaluation of the resulting policy on 10 episodes is included in the zipped file. These videos show the model is able to execute fairly successful pushes (though not entirely optimal, since the pushes in the expert dataset are sub-optimal).

2 Problem 2: Fine-tuning with Reinforcement Learning

The source code for this problem is included in Python files in the zipped folder. To execute the code for this problem run the p2.py file. This code learns a policy from scratch and also fine-tunes the policy learned in Problem 1, generates and saves a chart with the associated learning curves, calculates the average L2 distances between the object location and the goal location over 100 episodes, and creates videos showing the evaluation of each of the three policies on 10 episodes.

For the joint loss fine-tuning case, I modified the objective as such:

$$L_{total} = L_{actor} + L_{critic} + L_{entropy} + \lambda L_{BC} \tag{3}$$

where L_{BC} is the loss defined in Problem 1 above. I found that $\lambda = 0.1$ worked best for my experiments.

The learning curves for each of the policies can be seen below. It's clear that the agents using the policy learned through behavioral cloning greatly outperform the agent using a policy learned from scratch. We also see that the joint-loss fine-tuned learner slightly outperforms the vanilla fine-tuned learner. For both, the learning is on a slight upward trajectory even at termination time; we can imagine that these trends would continue, given more compute or time (the experiment was very slow to run on my machine). With a relatively small λ coefficient, it makes sense that the joint-loss and vanilla cases would not vastly differ.

The mean L2 distances between the object location and the goal location averaged over 100 episodes for each of the three policies are shown here:

Policy	Avg. L2 Dist.
From Scratch	0.500
Vanilla Fine-tuned	0.188
Joint-loss Fine-tuned	0.176

From this we can see that the policies using behavioral cloning do indeed outperform the from-scratch policy. They also outperform the policy learned in Problem 1, which indicates the fine-tuning was successful. Furthermore, we see that the joint-loss fine-tuned policy slightly outperformed the vanilla fine-tuned policy, consistent with the learning curves above.

The videos showing evaluation of the three policies on 10 episodes each are included in the zipped file. These videos support the above conclusion by showing that the models learned via behavioral cloning and fine-tuning greatly outperform the model learned from scratch (which appears to be failing entirely).

Acknowledgements: In working on this problem set, I worked with or spoke to Anurag, Josh, and Jordan.