CS294-112 HW1

Huanjie Sheng, 25928718

September 3, 2018

2 Behavioral Cloning

2.2 Comparison between behavioral cloning and expert policy

	exp(mean)	exp(std)	bc(mean)	bc(std)
Ant-v2	4823.29695049	100.901786378	4825.32305121	113.912392909
Humanoid-v2	10401.0887849	41.1015403998	2219.26290071	1302.88074545

exp: expert; bc: behavioral cloning

We can see that behavioral cloning performs well in ***Ant-v2*** but very badly in ***Humanoid-v2***.

Here are the parameters I used for this simulation:
max_timesteps: 1000, num_rollouts: 20, epochs: 500

Here are the policy architecture used:
hidden_layers=2, layer_size=64, activation_func=tf.nn.relu

2.3 Hyperparameter tuning in behavioral cloning

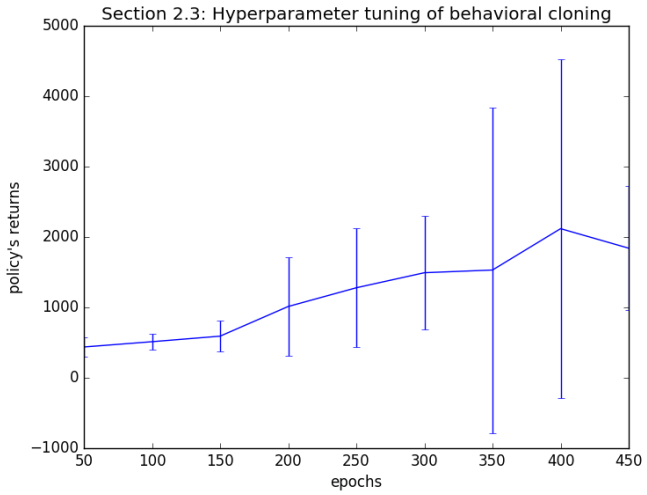

Figure 1: Hyperparameter tuning of epochs with the same model as in 2.2. Here are the parameters I used:
expert_policy_file: ./experts/Humanoid-v2.pkl,
envname: Humanoid-v2,
max_timesteps: 1000,
num_rollouts: 20,
I chose epochs because it directly affects whether the training process converges. Besides, it's easy to tune.

3 DAgger

3.2 Comparison between behavior cloning and expert policy

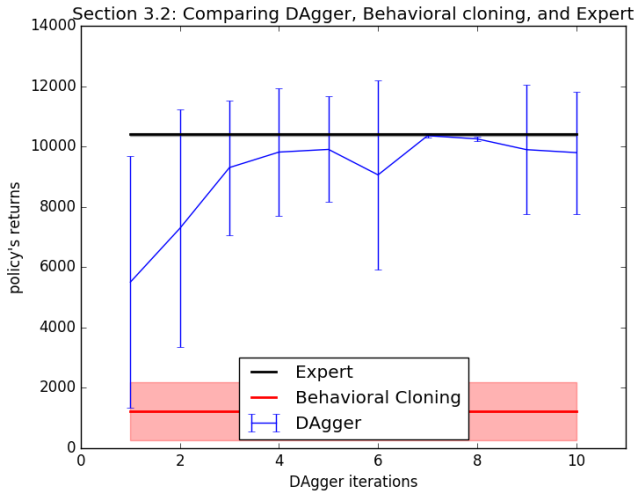

Figure 2: I used the same neural network mode las in behavioral cloning with two hidden layers of 64 hidden nodes per layer, which use ReLU as activation functions.
Here are the parameters I used:
expert_policy_file: ./experts/Humanoid-v2.pkl,
envname: Humanoid-v2,
max_timesteps: 1000,
num_rollouts: 20,
epochs: 500,
dagger_runs: 10,
It should be noted that "num_rollouts" applies to both the expert demonstration and the correction for each dagger iteration.

4 Policy architecture

4.1 Policy architecture of behavioral cloning

	mean	std
expert	10410.3113272	34.5014719634
original	1627.43551917	1271.28443921
hidden_layers=6	627.643695252	173.365926011
layer_size=16	529.67947482	202.227319372
activation_func=tt.nn.tanh	597.21387704	144.390361908

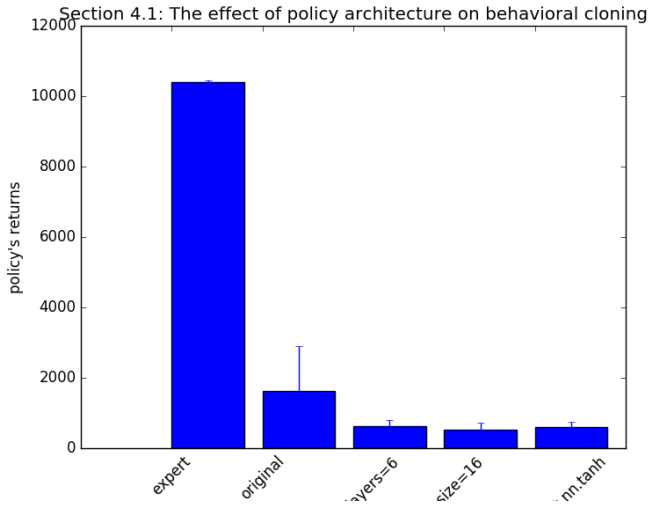

Figure 3: From left to right: expert, original model, model with six hidden layers, model with 16 hidden nodes per layer, model using tanh as activation function.
Here are the parameters I used:
expert_policy_file: ./experts/Humanoid-v2.pkl,
envname: Humanoid-v2,
max_timesteps: 1000,
num_rollouts: 20,
epochs: 300,
All modification performs worse than my original model which is copied from the TensorFlow's tutorial. More layers is worse probably because it overfits. Less hidden nodes is worse probably because it underfits. tanh is worse might be because I did not normalize the input data.