report

September 12, 2022

0.1 Problem 1

For BC Ant, we use the following choice of parameters, which is the default parameters except where noted.

```
#@markdown expert data
expert_policy_file = 'cs285/policies/experts/Ant.pkl' #@param
expert_data = 'cs285/expert_data/expert_data_Ant-v4.pkl' #@param
env_name = 'Ant-v4' #@param ['Ant-v4', 'Walker2d-v4', 'HalfCheetah-v4', 'Hopper-v4']
exp_name = 'bc_Ant' #@param
do_dagger = False #@param {type: "boolean"}
ep_len = 10000 #@param {type: "integer"}
save_params = False #@param {type: "boolean"}

num_agent_train_steps_per_iter = 10000 #@param {type: "integer"})
n_iter = 1 #@param {type: "integer"})

#@markdown batches & buffers
batch_size = 10000 #@param {type: "integer"})
eval_batch_size = 10000 #@param {type: "integer"}
train_batch_size = 100 #@param {type: "integer"}
max_replay_buffer_size = 1000000 #@param {type: "integer"}

#@markdown network
n_layers = 2 #@param {type: "integer"}
size = 64 #@param {type: "integer"}
learning_rate = 5e-3 #@param {type: "number"}

#@markdown logging
video_log_freq = -1 #@param {type: "integer"}
scalar_log_freq = 1 #@param {type: "integer"}

#@markdown gpu & run-time settings
no_gpu = False #@param {type: "boolean"}
which_gpu = 0 #@param {type: "integer"}
seed = 1 #@param {type: "integer"}
```

The data is recorded in `q1_bc_Ant_10evals_Ant-v4_12-09-2022_09-09-39`.

Same parameters, just with the task changed to Hopper. We tried all four available tasks. The

Hopper task is the only one that managed to reach below 30% with the above parameters. The agent is just too good.

The data is recorded in `q1_bc_Hopper_10evals_Hopper-v4_12-09-2022_09-12-03`

```
(array(4730.07714844), array(97.9336853))
4713.6533203125
(array(984.69366455), array(156.83201599))
3772.67041015625
```

	mean of return	std of return	expert performance
BC Ant	4730	98	4713
BC Hopper	984	156	3773

Table 1. The mean and standard deviation of the policy returns over 10 evaluation rollouts, in Ant and Hopper environments. The hyperparameters are described above. Note that the mean of return for Ant is larger than the expert performance.

For experiment on Hopper, we changed the episodes per training iteration, because seeing the exponential convergence (in DAgger Hopper), we think that it lacks both expert data and training steps. We first tried varying only the number of episodes per training iteration from 1 gradually to 100, but obtained the same agent (up to floating point error).

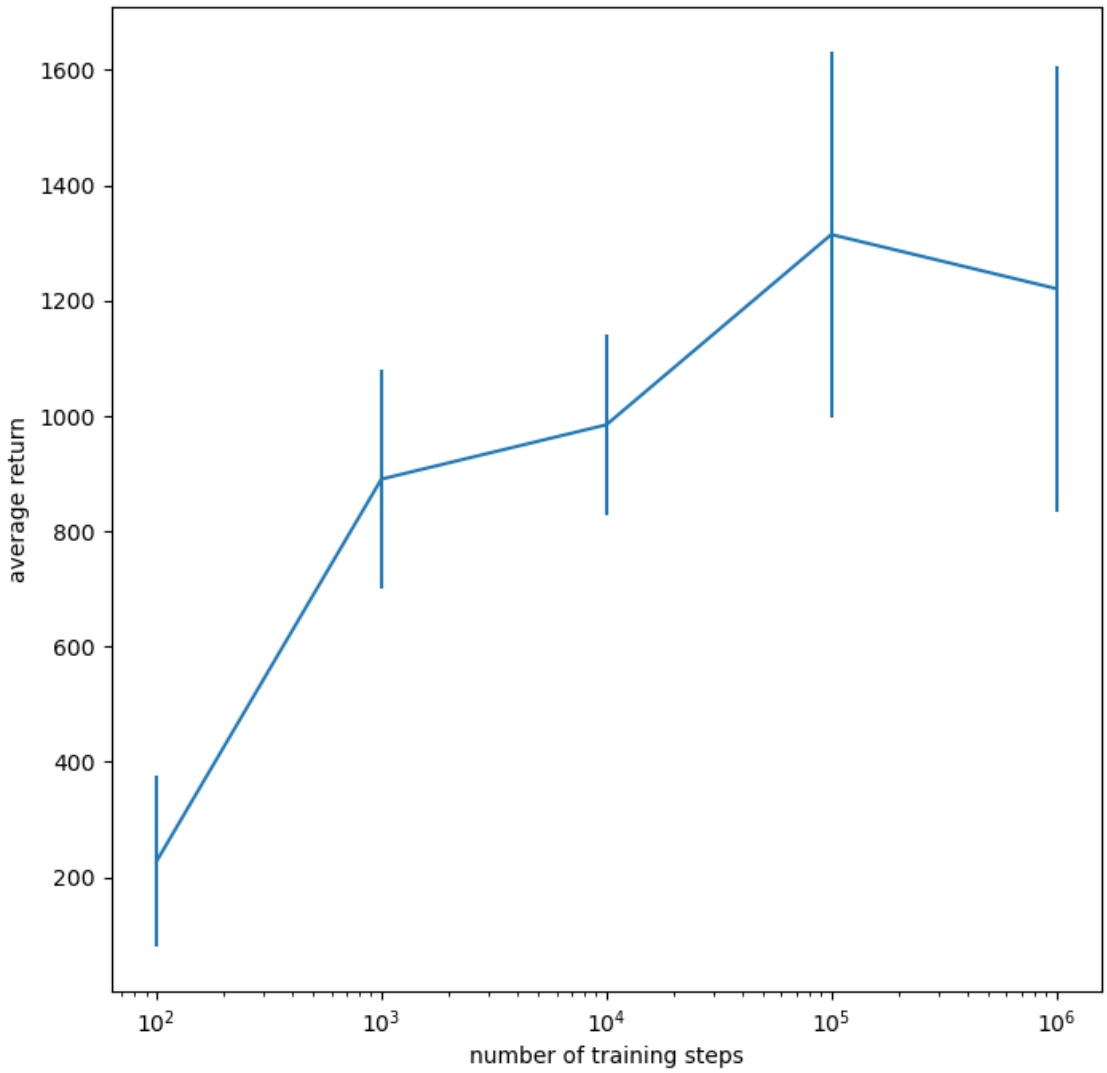
At first we thought there was a bug, but then I realized that the Behavior Cloning agent *NEVER* interacted with the environment! That's why it never improved – it obtained exactly the same expert-behavior dataset, loaded from the pickled file! There's no bug in the code, only a bug in my mind.

So we varied number of training steps from 10^2 to 10^6 and got the desired variation in performance. It appears that the training has converged at 10^5 steps, and just in time. To train it for 10^7 steps would have costed 200 minutes of computing time.

The data is recorded in

- `q1_bc_Hopper_100trainsteps_Hopper-v4_12-09-2022_09-30-13`
- `q1_bc_Hopper_1000trainsteps_Hopper-v4_12-09-2022_09-31-01`
- `q1_bc_Hopper_10000trainsteps_Hopper-v4_12-09-2022_09-32-00`
- `q1_bc_Hopper_100000trainsteps_Hopper-v4_12-09-2022_09-32-41`
- `q1_bc_Hopper_1000000trainsteps_Hopper-v4_12-09-2022_09-35-42`

Figure 1. Average return as a function of number of training steps.



0.2 Problem 2

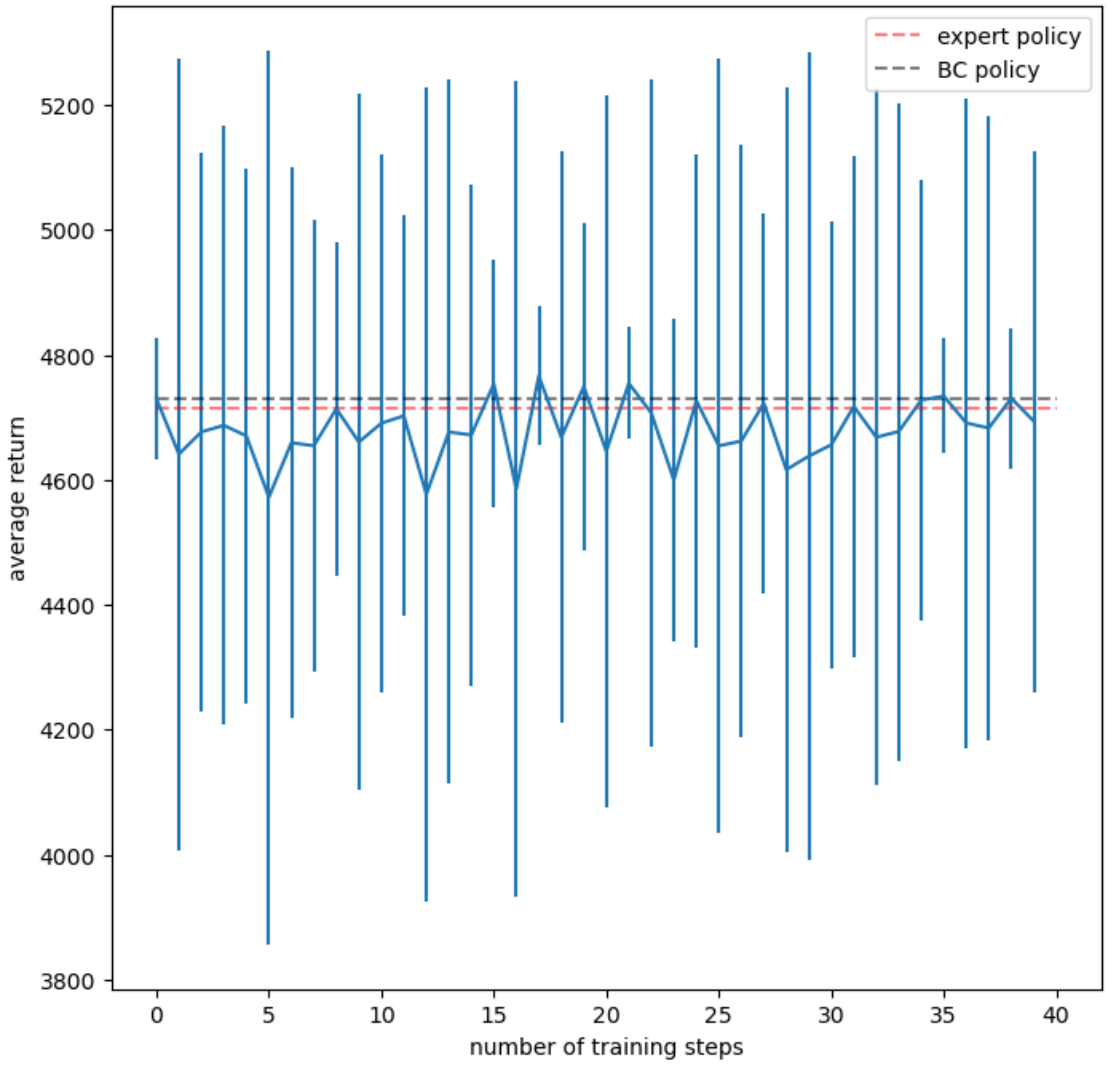
The DAgger Ant experiment uses the same parameters as the BC Ant, but with 40 iterations.

```
#@markdown expert data
expert_policy_file = 'cs285/policies/experts/Ant.pkl' #@param
expert_data = 'cs285/expert_data/expert_data_Ant-v4.pkl' #@param
env_name = 'Ant-v4' #@param ['Ant-v4', 'Walker2d-v4', 'HalfCheetah-v4', 'Hopper-v4']
exp_name = 'dagger_ant_10eval' #@param
do_dagger = True #@param {type: "boolean"}
ep_len = 10000 #@param {type: "integer"}
save_params = False #@param {type: "boolean"}

num_agent_train_steps_per_iter = 10000 #@param {type: "integer"})
```

```
n_iter = 40 #@param {type: "integer"})

#@markdown batches & buffers
batch_size = 10000 #@param {type: "integer"})
eval_batch_size = 100000 #@param {type: "integer"}
train_batch_size = 100 #@param {type: "integer"}
max_replay_buffer_size = 1000000 #@param {type: "integer"}

#@markdown network
n_layers = 2 #@param {type: "integer"}
size = 64 #@param {type: "integer"}
learning_rate = 5e-3 #@param {type: "number"}

#@markdown logging
video_log_freq = -1 #@param {type: "integer"}
scalar_log_freq = 1 #@param {type: "integer"}

#@markdown gpu & run-time settings
no_gpu = False #@param {type: "boolean"}
which_gpu = 0 #@param {type: "integer"}
seed = 1 #@param {type: "integer"}
```

The data is recorded in `q2_dagger_ant_10eval_Ant-v4_11-09-2022_22-38-54`

Figure 2.1. DAgger Ant performance.



The Hopper DAgger experiment uses the same parameters as the DAgger Ant, just with a different environment.

The data is recorded in `q2_dagger_hopper_10eval_Hopper-v4_11-09-2022_21-55-24`.

Figure 2.2. DAgger Hopper performance.