CS294 Deep Reinforcement Learning — UCB, Spring 2017 — William
Homework 1, Behavioral Cloning

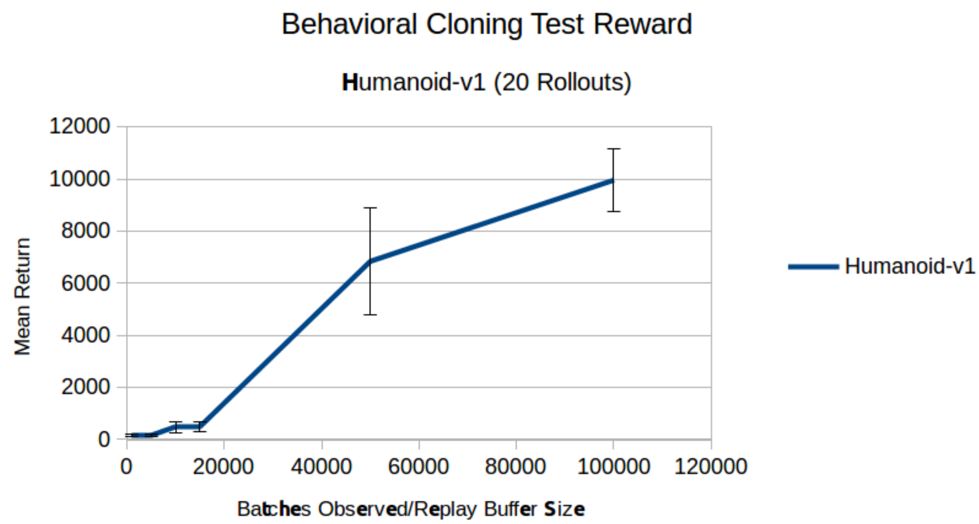
Task 2.1. Behavioral cloning on one successfull and one one unsuccessfull task.

Algorithm	Task	$E(t_{end})$	$\bar{r}$	$\sigma(r)$
Behavioral Cloning	Ant-v1	0.0003166649	4692.75305153	112.517654818
Expert	Ant-v1	–	4809.58391948	86.3039673066
Behavioral Cloning	Humanoid-v1	0.0588080287	474.056683013	200.906773452
Expert	Humanoid-v1	–	10401.0822864	52.0859149171

FIGURE 1. The results from behavioral cloning on two different tasks against an expert.

Details for Figure 2. The blcone algorithm was a neural network $\mathcal{N}$ with a 400 neuron ReLu layer followed by a 300 neuron ReLu layer followed by a m neuron linear layer where m denotes the size of the action space of the task. The network $\mathcal{N}$ was trained by minimizing $E : (s, y) \mapsto \|\mathcal{N}(s) - y\|^2$ for expert actions y drawn from a trajectory distribution generated by some expert policy $\pi : \mathcal{S} \to \mathcal{A}$ with $\mathcal{S}$ the state space and $\mathcal{A}$ the action space for the Humanoid-v1 task. Batches of size 64 were fed to bclone every time step $t \in \{1, \cdots, N\}$. The $E(t_{end})$ column yields the testing error of E accross the entire expert policy training set after the training the neural network on $N = 10000$ batches from a replay buffer of size N. In this comparison there were 10 rollouts to accumulate data. The $\hat{r}$ and $\sigma(r)$ columns denote the testing return (mean and standard devation respectively) over 20 testing rollouts of $\mathcal{N}$. In this experiment behavioral cloning on Ant-v1 but not Humanoid-v0.

Task 2.3. Behavorial cloning against a hyperparameter.



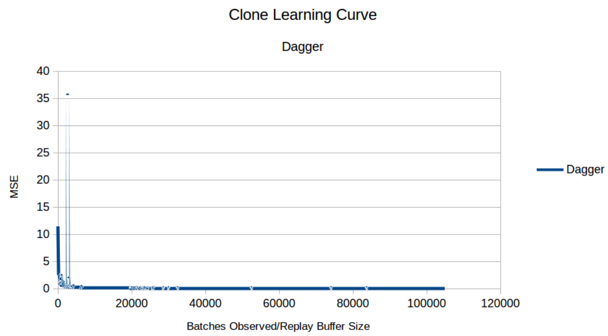
Algorithm	R	N	ϵ	$E(t_{end})$	$\bar{r}$	$\sigma(r)$
bclone	1	1000	1.00E-03	6.456817627	179.452871551	44.3650606799
bclone	5	5000	1.00E-03	0.0836024135	178.129472845	53.3310386856
bclone	10	10000	1.00E-03	0.0588080287	474.056683013	200.906773452
bclone	15	15000	1.00E-03	0.0215217602	495.870674664	180.387783861
bclone	50	50000	1.00E-03	0.010508501	6848.28015071	2073.80294628
bclone	100	100000	1.00E-03	0.00993759	9964.98563503	1199.9020977
bclone	100	100000	4.00E-04	0.0129574696	6246.07468031	1535.16340293
expert	–	–	–	–	10401.0822864	52.0859149171

FIGURE 2. The results from varying the the number of observed rollouts, N, that the behavioral cloning algorithm observes.

Details for Figure 2. The blcone algorithm was a neural network $\mathcal{N}$ with the same network architecture as that described in Task 2.2. The network $\mathcal{N}$ was trained by minimizing $E : (s, y) \mapsto \|\mathcal{N}(s) - y\|^2$ for expert actions y drawn from a trajectory distribution generated by some expert policy $\pi : \mathcal{S} \to \mathcal{A}$ with $\mathcal{S}$ the state space and $\mathcal{A}$ the action space for the Humanoid-v1 task. Batches of size 64 were fed to bclone every time step $t \in \{1, \cdots, N\}$. The $E(t_{end})$ column yields the testing error of E accross the entire expert policy training set after the coressponding number of batches in the N coilumn. The ϵ column denotes the learning rate of $\mathcal{N}$. The $\hat{r}$ and $\sigma(r)$ columns denote the testing return (mean and standard devation respectively) over 20 testing rollouts of $\mathcal{N}$.

This particular hyperparaemter was chosen for experimentation as it stands to reason that more time learning directly on the expert policy (with more data) will lead a more general cloned policy. Although varying this hyperaparameter in general lead to more successful rollouts, the variance of these rollouts was significantly higher as cloned policy $\mathcal{N}$ often diverged from the state-action trajectory distribution of the expert.

Task 3.2. DAgger as an improvement to behavioral cloning.



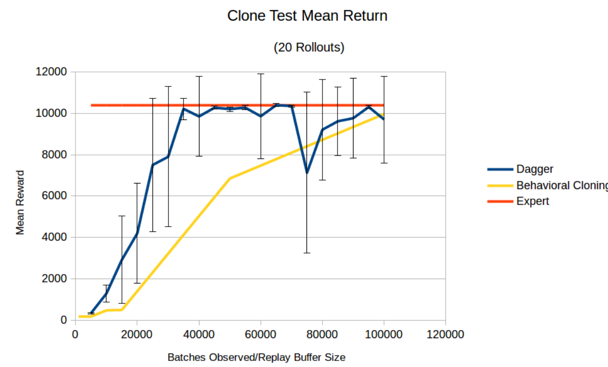


FIGURE 3. (Left) A plot of the clone mean squared error curve using dagger on Humanoid-v1. (Right) A comparison of the clone test mean reward as the number of learning iterations (and datapoints) increase on the same task.

Details for Figure 3. The blcone algorithm was a neural network $\mathcal{N}$ with the same network architecture as that described in Task 2.2. The network $\mathcal{N}$ was trained by minimizing $E : (s, y) \mapsto \|\mathcal{N}(s) - y\|^2$ for expert actions y generated by some expert policy $\pi : \mathcal{S} \to \mathcal{A}$ with $s \in \mathcal{S}$ the state space and $a \in \mathcal{A}$ the action space for the Humanoid-v1 task. **However**, at every iteration the action taken was $\mathcal{N}(s)$ and **not** $\pi(s)$, the action of the expert policy[1]. Batches size, learning rate, and evaluation metrics are the exact same as Figure 2.

In the above Figure, it is clear that providing expert labels for states generated by $\mathcal{N}$ avoids the disribution mismatch problem, and at $N \simeq 45,000$ we have that $\mathcal{N}$ achieves the not only the same average return as π but also the same standard deviation $\sigma(r)$. Beyond this point we suspect $\mathcal{N}$ began to overfit to the replay buffer, but further investigation is required.

[1]This is the essential setp of the DAgger algorithm.