Conditional Cloning for Imitation Learning

Gautam Bhattacharya

***Abstract*—This report presents an approach to improve the performance of supervised imitation learning by conditioning action prediction on a demonstration of the task. The proposed method relies on a soft attention model, that determines which part of the demonstration to focus on in order to predict the best action. The proposed model is tested on the hopper and humanoid locomotion tasks in the Mujoco environment. While these tasks prove to be too challenging for the proposed model to match expert behavior, it is able to outperform the baseline supervised learning approach. Based on experimental findings, we propose techniques that could potentially improve performance.**

I. Introduction

The goal of imitation learning is to learn to predict the behavior and decisions an expert would choose, for example, the motions a person would take to grasp an object or the route a driver would take to get from home to work. Capturing purposeful, sequential decision-making behavior can be quite difficult for general-purpose statistical machine learning algorithms. A powerful idea for approaching problems of imitation learning is to structure the space of learned policies to be solutions to a Markov Decision Problems (MDP). The tasks then amounts inferring the reward structure (i,e, weights) of the MDP.

Another school of algorithms that have been shown to work well in practice are based on supervised learning [1]. That being said, the simplistic assumptions made by these algorithms leaves significant room for improvement. The goal of this project is to address some of these shortcoming.

II. Supervised Imitation Learning

Supervised learning represent perhaps the simplest approach to imitation learning. We are given a set of expert demonstrations $D = \{(s_1, a_1), (s_2, a_2), ..., (s_n, a_n)\}$, where s_i and a_i represent a *trajectories* of states and actions respectively. We can then train a classifier to predict actions, given individual states. In the work neural networks were used as the classifier. While this approach works well in certain constrained settings, there are aspects of the approach that are not favorable in many imitation learning settings.

Supervised imitation learning, or behavioral cloning assumes that the individual states are i.i.d, which contradicts the highly correlated nature of state trajectories. Another problem with behavioral cloning relates to the nature of the training data it is provided. Ideally we would like to provide human demonstrations of expert behavior, which the agent tries to imitate. The problem with expert data of this type is that it is often too good. Taking autonomous vehicles as an example; expert human data will typically contain perfect examples of driving. The learning algorithm will never see states that correspond to errors or bad decisions. Consequently, a simple supervised learning approach will have no notion of what actions to take in order to recover when such states are encountered. This leads to a compounding of errors.

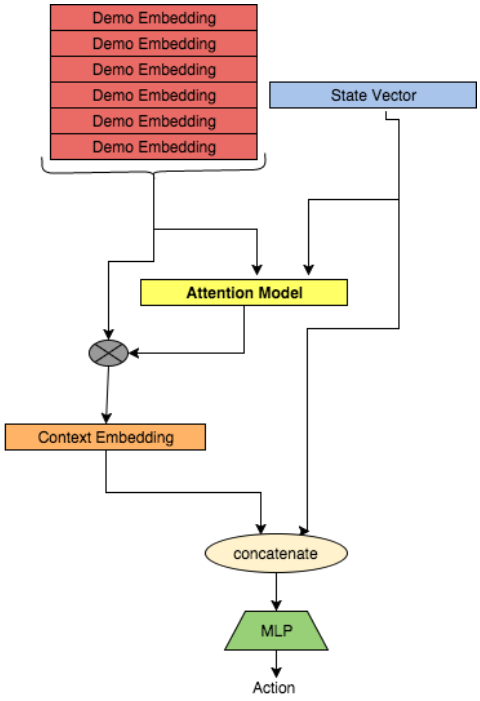

Fig. 1. Conditional Cloning Network Architecture

III. Conditional Cloning

Recently an approach was proposed for one-shot imitation learning, based on supervised learning [2]. The core idea is to condition the prediction of a classifier on a demonstration of the task. In this work a neural network is trained on multiple related tasks, i.e. tasks on the same environment. For example, particle reaching tasks where the position of the particle changes. At test time the network is conditioned on a *new* task demonstration (not seen during training) and asked to predict actions.

The goal in this work was to adapt this framework to a *single* task. The intuition behind conditioning action prediction on an expert demonstration is that the network may learn to better avoid reaching states that are difficult to recover from. That being said, by conditioning the network on expert behavior alone, it does not have information on how to recover. We are telling the network what it should do, without telling it what it should not.

While in principle a generic neural network architecture can be used to do this type of conditional prediction, one of key insights from the work in [2] is that designing a network architecture that suits the task is crucial for good performance.

IV. Models

The task of conditioning action prediction on a demonstration can be broken into two distinct parts. The classifier stage of the process represents a standard feed-forward neural network trained in a supervised fashion.

The other key component of the approach involves producing conditioning information from the demonstration sequence. Typically this is achieved by producing a single vector to represent the demonstration. However, Depending on the complexity of the task and the length of the sequence, it is not trivial to extract meaningful information from the demonstration. The network architecture proposed for producing the conditioning information attempts to take these factors into account.

Demonstration Network

The demonstration network represents the crux of the model. If this network is unable to learn something useful from the demonstration sequence, the agent network (classifier) can learn to simply ignore the conditioning information. Indeed, this appears to be the case for some of the models experimented with in this project.

1) Summarizing Sequences with RNNs: Recurrent neural networks (RNNs) are a natural choice to represent sequential data. After propagating a sequence through a recurrent network, the RNN hidden activation corresponding to the last time-step of the sequence can be treated as a summary of the sequence.

$$h_t = G(x_t, h_{t-1}) \tag{1}$$

Where h_t is the RNN hidden activation at time t, x_t is the input and G is the RNN non-linearity. We see that there is a recursive relationship between h_t and h_{t-1}. Thus the last hidden activation h_T contains information from all preceeding time-steps.
Crucially, it allows us to represent a sequence of potentially arbitrary length with a single vector. However it becomes more challenging to capture all the relevant information from the sequence as it gets longer. In this work sequences of length 250 and 500 were used, which proved to be too long for this basic summary model to capture. Initial experiments showed performance worse that behavioral cloning and I did not explore this model further.

2) State Dependent Conditioning: In the one-shot imitation learning approach, the authors make extensive use of soft attention models [3]. There are two main advantages of using an attention network. Firstly, the model only focuses specific parts of a demonstration sequence, thus the RNN does not need to memorize the entire demonstration. The second benefit of augmenting the demonstration network with an attention model is that this makes the conditioning vector state dependent, which intuitively seems to be a desirable feature.

In this model a RNN is also used to embed the demonstration, producing a sequence of embeddings d_j for $j = 1, 2, ..., T$.
The attention network is a single layer neural network that takes the state vector s_i and a single embedding of the input sequence as input. The network processes each of the input embeddings sequentially, producing a sequence of scores of the same length (as the demonstration sequence).

$$g_{ij} = [s_i; d_j]; e_{ij} = a(g_{ij}) \tag{2}$$

Where [] represents the concatenation function, and a is given by :

$$a(g_{ij}) = tanh(W_a g_{ij} + b_a) \tag{3}$$

The attention network outputs a set of unnormalized scores, which are subsequently normalized using the softmax function.

$$\alpha_{ij} = \frac{exp(e_{i,j})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{4}$$

Finally, the context embedding (conditioning vector) for state $s_i$is given by:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} d_j \tag{5}$$

This context embedding is concatenated to the state vector s_i and fed as input to the agent network.

Conditioning Information

In the imitation learning setting we are provided with a set of expert trajectories in order for our agent to learn from. To makes this concrete, an expert trajectory consists of a sequence of state-action pair (s_i, a_i) for $i = 1, 2, ..., T$. Thus we can condition the agent network on the combined state-action sequence, or the individual state or action sequences.

Initially only state sequences were used as conditioning information, however the models was similar to behavioral cloning. Somewhat surprisingly, the best performance was achieved by a model conditioned on the action sequence alone.

Agent Network

The agent network used for all the experiments in this work is also a neural network. The input to this network is the concatenated state and context vectors $[s_i; c_i]$, and the target is the expert action corresponding to the state. In this work only continuous action spaces were considered, an the network is hence trained by minimizing the mean squared error between the agents prediction and the target action.

A. Architecture and Training

- A bidirectional RNN with 200 hidden units in each layer was used for producing the demonstration embeddings. The network used Gated recurrent units (GRU) activations.
- The attention network is single-layer fully connected network. The number of hidden units depends on length of the demonstration sequence. Here demonstrations of either 250 or 500 time-steps were considered. Tanh nonlinearity was used as activation function.
- The agent network is a two-layer network with 100 hidden units in each layer. Once again, tanh activation are used for non-linear transformation.

We experimented with two training strategies. First we insured that the mini-batch of state vectors being input to the agent network were conditioned on the specific sequence containing those states. At test time an arbitrary demonstration was used as conditioning for all time steps.
The second strategy involved selecting random demonstrations to to provide conditioning information. This approach is more harmonious is terms of training at test conditions. Intuitively, any expert demonstration should provide useful information regarding what action to take in a given state. This training approach was more successful in our experiments. Apart from a mini-batch strategy the remained of network training is standard. Backpropagation and the ADAM optimizer are used to learn the networks weights. Training is stopped when the loss on a held out validation set stops reducing.

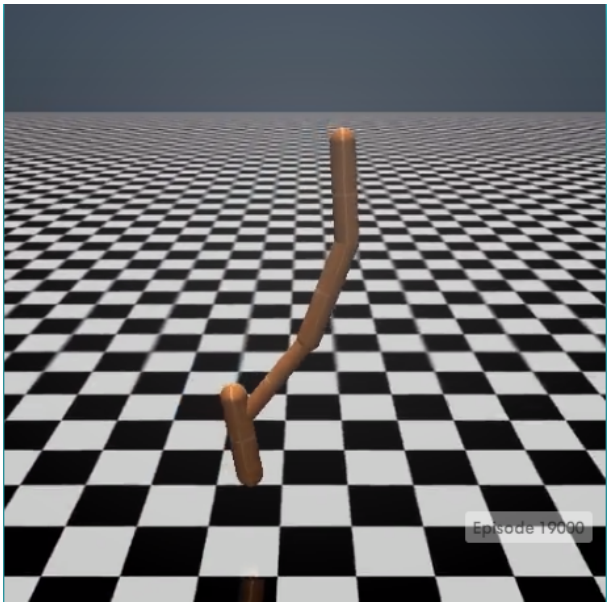

Fig. 2. Hopper Environment : Make hopper go forward without falling over

V. Experiments and Results

The proposed conditional cloning model was tested on two locomotive tasks in the Mujoco environment. The two tasks considered, hopper and humanoid, represent challenging continuous control tasks. The goal in these tasks is to make the agent move forwards as fast as possible with falling over.

As the results will show, these tasks appear to be too challenging for the model to perform close to expert behavior. That being said, the proposed model is able to outperform behavioral cloning, indicating that the network is capturing some relevant information from the conditioning data.

Expert Data

A policy trained using trust region policy optimization (TRPO) is used to produce training data. 50000 training points and 10000 validation points were collected for each of the experiments. The amounted to 120 rollouts in the 500 time-step case, and 240 in the 250 time-step case. Noise was added to the actions in order to explore a larger part of the space [2].

TABLE I
Hopper 250

Algorithm	Avg. Reward	Std.
Expert policy	802.7	4.2
Behavioral cloning	845.3	114.1
Conditional cloning	845.3	7.7

Table 1. compares the performance of the different algorithms on the hopper environment over 250 time-steps. The results are averaged over 100 rollouts.We see that behavioral cloning is able to match expert behavior over this duration. Encouragingly, the proposed method outperforms both behavioral cloning and the expert policy. However as we will see, this result is counter-intuitive. On viewing videos of the learned policy, it is clear that the policy learned by the proposed method learns to accelerate quickly, only to fall over right after 250 time-steps. This policy performed significantly worse that behavioral cloning when the environment was allowed to run for 500 time-steps.

TABLE II
Hopper 500

Algorithm	Avg. Reward	Std.
Expert policy	1731.1	9.7
Behavioral cloning	942.4	193.1
Conditional cloning	907.3	94.3

Tables 2 compares results when a 500 time-step demonstrations is used as conditioning information. Consequently, the environment is also allowed to run for 500 time-steps at test time. Here we see that neither behavioral cloning or the proposed method are able to match the expert policy. The proposed method does not do as well as behavioral cloning, however, the result is better than the policy conditioned on a 250 time-step demonstration.

The experiments on the humanoid environment were far less successful. The is perhaps to the significantly higher dimensionality of the observation and action spaces compared to the hopper environment. One surprising finding from these experiments was that the model conditioned on 500 time-steps performed significantly better than the model conditioned on 250 steps, even though neither policy could progress more than 150 steps.

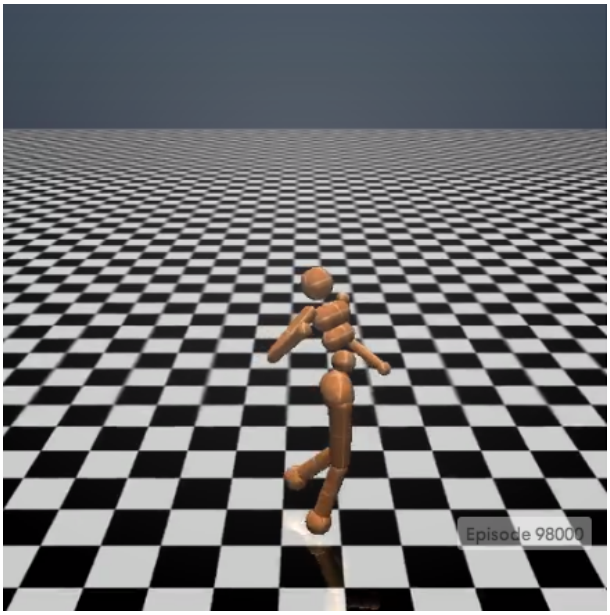

Fig. 3. Humanoid Environment : Make a humanoid walk

TABLE III
HOPPER 500

Algorithm	Avg. Reward	Std.
Expert policy	2167.4	32.5
Behavioral cloning	802.4	177.2
Conditional cloning	799.6	114.3

From table 3 we see than neither of the supervised learning strategies comes close to the expert policy. Given their large standard deviations, behavioral cloning and the proposed method perform roughly the same on this environment.

Effect of Ensembles

A common strategy to improve the performance of neural network models is to collect an ensemble of predictions and take the average. In the setting of this work, we can consider an ensemble of demonstrations. That is, we can make action predictions considering multiple different demonstrations, and then take the average action.

TABLE IV
HOPPER 500

Algorithm	Avg. Reward	Std.
Behavioral cloning	947.4	197.1
Conditional cloning	1033.2	119.3

TABLE V
HUMANOID 500

Algorithm	Avg. Reward	Std.
Behavioral cloning	812.4	212.3
Conditional cloning	923.6	143.4

The ensemble models considered here are of size 10. For behavioral cloning, this amounts to averaging ten predictions for the same state vector (not training 10 different models, and averaging their individual predictions). While this was not expected to help performance, it was done to keep the comparison as fair as possible. For the conditional cloning approach 10 demonstrations are randomly selected from the training set and used as conditioning information for each state vector. The final action is the average of these 10 predictions.

From the results in table 4 and 5 we see that considering an ensemble of demonstrations clearly helps. In the case of the humanoid environment, the ensemble approach was able to complete 150-200 steps (somewhat consistently), which was significantly better than the other supervised learning algorithms.

VI. DISCUSSION

From our experiments we find that while the proposed method is able to do better than the baseline, the improvement is marginal. That being said, the policy learned by the conditional network appears to be different from that of behavioral cloning. This suggests that useful information is can be obtained from a demonstration sequence. One potential reason for the poor performance is the difficulty of the tasks considered.

The proposed method would appear to be better suited to tasks well defined goal states. Multi-task and transfer learning tasks like particle reaching would also appear to be a good fit.

Another potentially interesting future direction for this research would be to try to get the agent to learn how to recover.This would involve training the model in a DAgger like fashion [4]. First the trained model can be used to generate trajectories. Then rather than use the actions produced by the model, we can use the actions taken by the expert as the corresponding conditioning information. Intuitively, the model would now be learning to predict actions an expert would take to get out of bad states and ideally learn to recover.

REFERENCES

[1] Pomerleau, Dean A. ALVINN, an autonomous land vehicle in a neural network. No. AIP-77. Carnegie Mellon University, Computer Science Department, 1989.

[2] Duan, Yan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. "One-Shot Imitation Learning." arXiv preprint arXiv:1703.07326 (2017).

[3] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[4] Ross, Stphane, Geoffrey J. Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning." In AISTATS, vol. 1, no. 2, p. 6. 2011.