Policy Optimization using Behavioral Cloning with Human Feedback

Sana Iqbal, Suchismita Padhy

Abstract

Reinforcement learning is based on the idea of training an agent to behave in a way that maximizes rewards from the environment, during a goal oriented task. However, it is not always possible to define a reward function for an environment because the task is complicated and the factors affecting a reward function cannot be clearly defined.

In this experiment we have proposed a method to train an agent in a similar complex environment with unknown reward function. We propose to ensemble Behavioral Cloning with the Human Preference Learning. We hypothesized that if we use the policy trained with Behavioral Cloning in the Human Preference Learning algorithm, that extracts a NN-based reward function from the environment based on the human feedback on the agents performance, this will fine tune the policy of the agent and improve its performance. Also, the extracted reward function will be a good approximation of the environment reward.

We have used the Hopper and the Humanoid environments from Mujoco in our experiment. We use multiple segments of fixed length for training a policy using the behavioral cloning with a multi-layered NN-based policy model. We use the trained policy to generate roll-outs in the environment. Segments of state- action of length 1.5 seconds are sampled from these roll-outs and are presented to the human observer. The human observer is shown two of these sampled segments at a time and the human chosen segment is assigned a label of 1 or 0 otherwise. These segment-label pairs are used to train a neural network based Reward function, which is then used to fine tune the policy. We have presented the comparison of performance of policies trained on Behavioral Cloning, human preferences learning and the proposed ensemble approach. We observed that the policies based on BC alone are not reliable whereas the policies trained using human preferences take a very long time to get trained to behave optimally. The policies that used our proposed method did fairly better than the above two approaches.

1 Introduction

The basic idea of reinforcement learning is to train an agent to learn behavior in dynamic environments as opposed to agents that execute preprogrammed behavior in a static world. In order to learn the behavior the agent needs to take decisions in the environment based on its observations. However we want the agent to take the best decisions that lead the agent towards its goal in the environment. For example a self driving car on the road can take any safe turn on road but we want it to make turns that take it safely to the final destination. The progress towards goal is described in terms of rewards from the environment. The agent trained using a goal-oriented learning is capable of deciding what action to take based on the observations to achieve a goal or maximize its rewards in the environment.

In computational reinforcement learning the goal is expressed in terms of reward functions, and for the agent to learn to behave in the environment it needs access to the reward function. In domains like computer games, reward functions are clear. Agent's goal is to maximize the expected score at the end of the game.This reward function acts as the feedback for the agent to improve its performance in the environment. However it is not always possible to design a reward function that captures the desirable behavior of the agent in the environment. Many tasks in the natural world involve goals that are complex and cannot be defined clearly, humans understand many things implicitly by intuition which is difficult to hard code. For example for tasks involving teaching a robot how to pour a glass of water, or have a dialogue with it, it is difficult to define the reward function for achieving the final goal.

2 Related Work

To work around such learning tasks where we cannot define the reward function of the environment, Imitation learning is used. Here the agent learns the behavior in an environment using the expert demonstrations in the environment, without defining a reward function. The process of imitation learning can either involve copying the expert without any consideration to why the expert is behaving the way it is or can be based on a method where the agent wants to understand the reasoning behind the expert behavior. The first method is called Behavior Cloning, it is the simplest approach of learning

from experts, it maps the desirable and undesirable actions to states directly. It is essentially a supervised learning method. For example using records of images from a vehicle camera and the commands executed by the human driver to route the vehicle, in a supervised learning algorithm to get a policy that predicts vehicle actions conditioned on the observation.The first problem with behavioral cloning approach is that for sequential tasks the policy prediction at any time affects future input observations/states , this violate the basic i.i.d. assumption made by most statistical learning approaches [1]. Any mistake by the learned policy leads the agent to encounter completely different observations than those under expert demonstration, leading to a compounding of errors. This problem of compounding of small errors called cascading errors in behavior cloning is a major drawback of the method [1]. The other method of imitation learning is the inverse reinforcement learning (IRL), which is based on the idea of learning the reason behind the expert's behavior. It assumes that the expert always behaves optimally with respect to some unknown cost function. The main principle of IRL is to learn a reward function under which the demonstration data is optimal.Since IRL trains an agent on the entire set of trajectories of the expert and not the individual state and action pairs, it prevents the problem of cascading errors. In IRL expert demonstrations of the desired task are used to extract a reward function of the environment and then agent is trained using that reward function. Since IRL assumes optimality in expert behavior as a prior on the space of policies, IRL algorithms can allow the learner to generalize expert behavior to unseen states much more effectively than if the learner had tried to produce a policy instead [2].Prior work has used maximum margin formulation [3] and probabilistic models that explain sub-optimal behavior as noise [4][5]. The probabilistic models still rely on defining a reward function using a linear combination of detailed features created using domain knowledge. This also includes indicators for successful completion of the task [6][7]. This approach though works in practice, increases the engineering burden significantly. Another set of methods have proposed learning non-linear costs using Gaussian processes [8] but operate on features instead of raw states. Neural network cost formulation can directly work on raw states and provide a rich and expressive representation. They have been used for small synthetic domains in previous works [9] and for real world robotic manipulation tasks that use torque control [10]. The problem with inverse reinforcement learning is that it is very expensive, we need to evaluate the fitness of a certain (that we are training) reward function to the expert policy by comparing the per-

formance of expert policy with all the possible policies . Yet another problem that arises in the inverse reinforcement learning is the requirement of having expert demonstrations. There are such tasks where it is not possible to have expert demonstrations because of the inability of humans to demonstrate the expert behavior. For example demonstrating a task to a robot with many degrees of freedom and very non-human morphology, although it is not possible for humans to perform that task but they can judge the actions of the robot as desirable or undesirable for the goal [11]. Christiano et. al. have proposed an approach for such tasks that allow a human to provide feedback to an agent on line when it is learning the policy in the environment. In their method called learning with human preferences a reward function for the task is extracted based on human feedback which is then used to train the agent for the task.

The other way to approach the imitation learning problem is proposed by Ho,Gupta, Ermon; 2016 in the apprenticeship learning formalism [12], in which they propose the learner must find a policy that performs at least as well as the expert policy on an unknown true reward function [3].

3 Preliminaries and Method

3.1 Motivation

It is important to communicate complex goals to our agent in order for it to choose optimal action. During the learning process the agent learns to distinguish between desirable and undesirable outcomes. The current behavior of the agent is to take actions that maximize a stated reward function. The biggest assumption in this approach is we already know the reward function and it remains fixed through the process. In real world setting however the reward function definition is not always possible, people understand many things implicitly by intuition or through common sense. Imagine teaching a robot how to pour a glass of water, or have a dialogue with it or train a autonomous car to drive on road. In such cases it is difficult to have a well defined reward function. If we instead set up proxies, for example we can design a detector for liquid and and measure the spill of the liquid but going through such engineering process may complicate the behavior of the system itself.

We can use expert data to make the agent learn the optimal behavior

for performing a task in a complex environment where we do not have a well defined reward function. Since most of the natural environments are complex and do not have a well defined reward function, we believe that its is important to understand how reinforcement learning can be used in such real word scenarios.

3.2 Our Method

We now present our method in which we proposed that we can use behavioral cloning and learning by human preferences to learn a reward function and train an optimal policy based on that reward function.

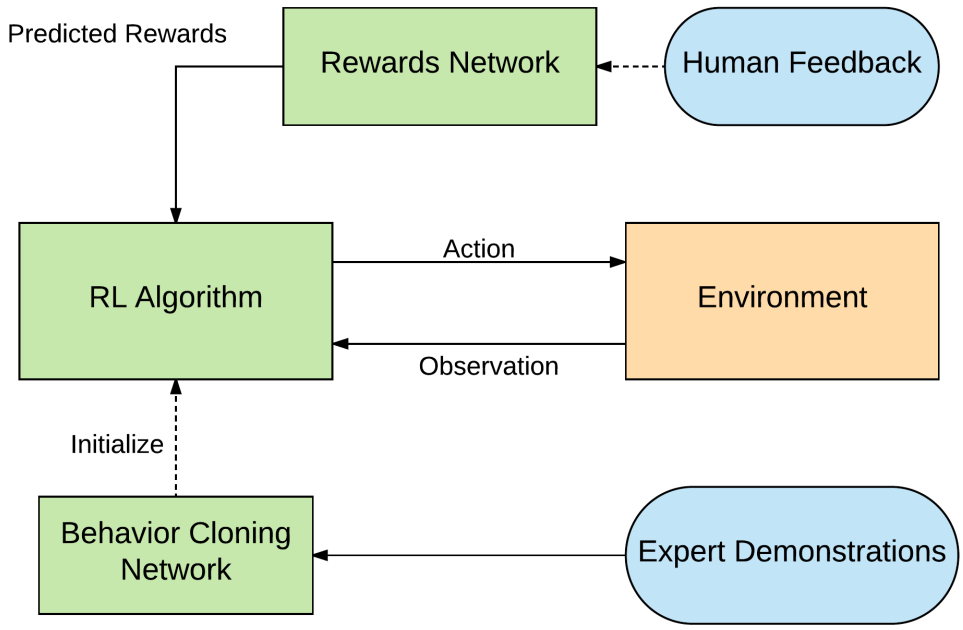

Figure 1: Block Diagram showing different components used in our method.

In our project we want to address this problem of amount of exploration required to learn an optimal policy especially in the continuous control tasks where expert demonstrations are not enough for the agent to learn the optimal behavior as in the above example where we want to train a multi degree freedom robot of non-human morphology. We propose that combining the

supervised policy learning and human feedback based RL paradigm into a single framework will allow us to train agents more efficiently based on their strengths. So, the main idea behind the experiment is that we want to improve the learning of the agent from the expert demonstrations by ensemble of behavioral cloning and human preference learning. Behavioral cloning alone does not allow the agent to learn the motivations of the expert behavior, it blindly copies the actions of the expert at a given state of the environment. Copying the expert may not always be the optimal way of doing a task for example a human is left handed and always lifts a glass of water by his left hand in his demonstrations, this may not always be the most optimal way of doing the task by a robot who has same strength in both hands. We propose that in order for the agent to learn the motivation behind certain actions during a task, we can ask the agent to perform actions based on its current policy and get feedback from the humans on its performance. This is similar to the concept of human learning, where a human learns to act optimally in the environment based on the feedback on its actions or the performance.

The primary components used by our method are shown in Figure 1. In the beginning, we collect demonstrations of the task being performed by an expert. We then use a multi-layered Neural Network which is trained to imitate the behavior of the expert. This network forms our initial policy. Using this policy, we generate simulated trajectories. We sample segments of fixed lengths from these trajectories and render these segments as small video clips. These video clips are presented to the human user in pairs through a web interface to record the userś preference. We expect the human to provide us with 1 bit of information indicating which segment in the pair seems to do the task better. This preference is used to train another multi-layered Neural Network which takes the observations and actions as inputs and predicts the reward. In the next step, we use policy gradients to improve our initial policy and for this training we use the rewards from the rewards network we trained in the previous step. The whole algorithm is described as follows.

1. Collect demonstrations from expert and clone experts policy to get an initial policy.

2. Generate roll outs from this policy and sample pairs of segments.

3. Ask human to choose one of the pairs of segments.

4. Train a Reward Function (neural network) to predict rewards given

observations and corresponding actions so that the segments chosen by human in (3) receive high rewards and vice versa.

5. Improve the policy learnt in step 1 by training it further using the reward function created in step 4 using Policy Gradient. So we learn a new policy that maximizes the rewards predicted by the network in step 4.

4 Experiment

4.1 Setup

In our algorithm we ensemble the behavioral cloning method and the human preference learning method. We generate demonstrations for the Hopper and Humanoid environments using expert policies in MujoCo. We use these expert demonstrations for training a policy using behavioral cloning algorithm, we then use the trained policy in the learning by human preferences. We allow the agent to perform actions based on the policy learnt from the behavioral cloning and then give it feedback on some of its randomly chosen actions. We cannot give feedback on the entire sequence of actions because human feedback loop is computationally costly. Based on the feedback on the performance of the agent, a reward function is trained and this reward function is used to update the policy in each iteration of human feedback.

For our expert demonstrations dataset we use several segments of length 60 for training a policy using the behavioral cloning with a NN-based policy model with 3 hidden layers. We use the trained policy to generate roll-outs in the environment. Segments of state- action of length 1.5 seconds are sampled from these roll-outs and are presented to the human observer. Figure 2 shows a screen shot of the comparison as it is shown to the user. The human observer is shown two of these sampled segments at a time and the human chosen segment is assigned a label of 1 or 0 otherwise. These segment-label pairs are used to train a neural network based Reward function, which is then used to fine tune the policy.

4.2 Results

We used a batch size of 10000 to train the behavioral cloning network. The figure 3a shows the loss vs number of iterations of the behavioral cloning

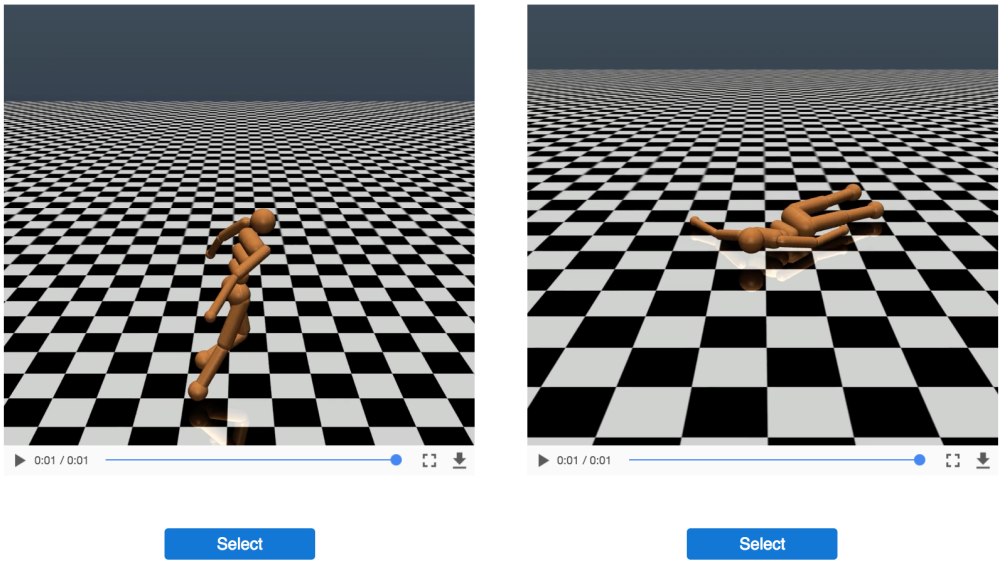

Figure 2: Screen shot of the comparison window.

network for the humanoid task. We see the loss decreasing rapidly during the initial few iterations and then decreases by small amounts before plateauing. The results for the Hopper task is similar to this.

(a) (b)

Figure 3: Training of Behavioral cloning and rewards network for the Humanoid task

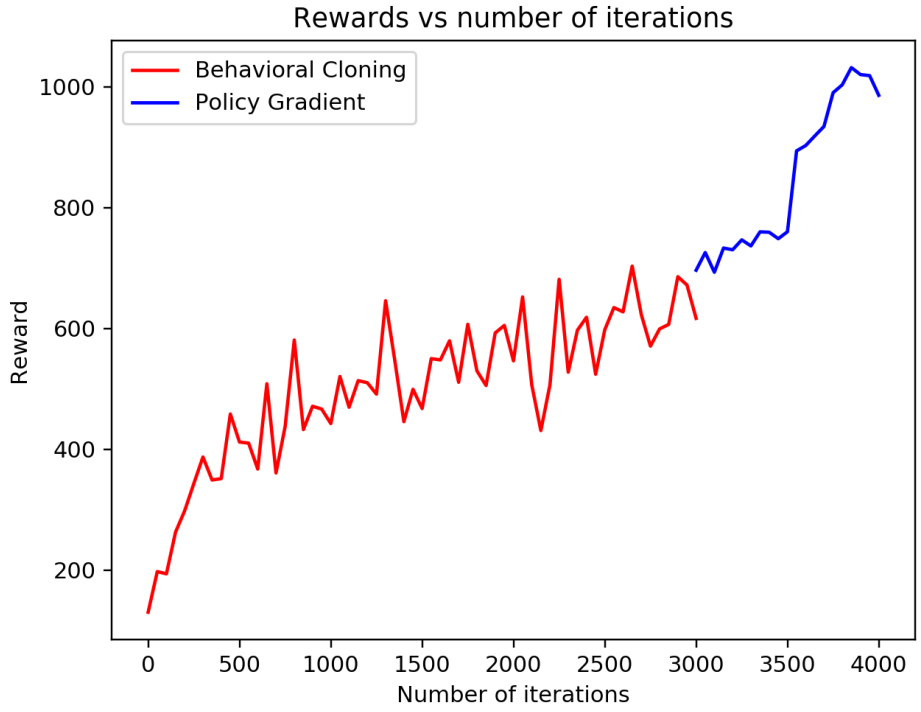

Figure 4: Rewards for the humanoid task.

We used a smaller batch size of 64 while training the rewards network. The figure 3b shows a plot of the loss versus the number of iterations for the rewards network.

Figure 4 shows the reward our policy receives for the Humanoid task. We can see that the behavioral cloning step helps us to reach a decent reward. The policy gradient network starts from there and the reward keeps getting better with number of iterations of this network.

We made another important observation during our experiment. In addition to helping the policy gradient with better initialization, the incorporation of behavioral cloning also generates more relevant roll outs in the first step of the algorithm. With a randomly initialized policy, the initial roll outs some time are not related to the task under consideration, making the collection of preference significantly more complex. Since there is no clear way for the user to prefer one of the two unrelated segments, it requires many

more queries to train the rewards network reliably. This makes the approach difficult to scale to more complex tasks and larger environments.

5 Conclusion and Future works

For the project, we collect segments using a randomly chosen start point and show to the user to collect the feedback. One of the key demerits of this approach is that when the randomly chosen segments are similar, it becomes difficult for the user to indicate a preference. This happens more frequently in the beginning and towards the end of the training. A possible solution to this is to sample segments in a more intelligent manner. Picking the pairs of segments that have maximum variance may help alleviating this issue. Another possible way to improve our policy is to exploit expert demonstrations more effectively.

References

[1] Stphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudk, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011.

[2] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[3] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

[4] Deepak Ramachandran and Eyal Amir. Bayesian Inverse Reinforcement Learning. *Learning*, 51:2586–2591, 2007.

[5] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings*

of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI'08, pages 1433–1438. AAAI Press, 2008.

[6] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative Entropy Inverse Reinforcement Learning. In *Proceedings of the 14th International Con- ference on Artificial Intelligence and Statistics*, volume 15, pages 182–189, 2011.

[7] Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf, and Jan Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biol. Cybern.*, 108(5):603–619, October 2014.

[8] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. pages 19–27, 2011.

[9] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

[10] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016.

[11] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. 2017.

[12] Jonathan Ho, Jayesh K. Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. *CoRR*, abs/1605.08478, 2016.