CS 780 Project

Brendan

December 9th, 2020

1 Task: What task are you solving

I will be solving classical control problems such as mountain car, cart-pole and acrobat. The problem more specifically will be to learn a *good* policy given expert demonstrations. A stretch goal for this project will be to create a method that can autonomously detect adversarial* demonstrations. Imitation learning which is the task that I am going to work with is trying to learn a policy for some task given expert demonstrations. There are two branches of imitation learning Inverse Reinforcement Learning (IRL) which attempts to recover the reward function from the demonstrations and then learning a policy form that reward function. In Behavior Cloning (BC) we attempt to learn the policy directly, without learning the reward function. A reward function is actually some function R_a that an algorithm attempts to maximize when doing Reinforcement Learning (RL), this results in a policy π.

1.1 Definition

The inputs are quite well defined in the domain of IL we have states (inputs) and actions (outputs), and a reward function. States are defined as the current state that a process or entity is in $s \in \mathcal{S}$ were $\mathcal{S}$ is the set of all possible states. Actions are defined as the action a process or entity is taking in that state $a \in \mathcal{A}$ where $\mathcal{A}$ is the set of all actions. More specifically to the Imitation Learning domain our input will be $\mathcal{D}$ a set of demonstrations. Each demonstrations will have a set of observed state-action pairs. We will attempt to learn a policy using these states-action pairs. The goal hear is to learn a policy $\widetilde{\pi}$ that imitates the expert policy π^*. An expert policy π^* maps states to actions such that it maximized the the expected value of the rewards function R_a for any given episode. An episode is one full expression of a policy trying to complete the task. In our case the episode either ends when the task is complete, the task in failed, or a maximum number of the time steps has been reached. For this task our demonstration set $\mathcal{D}$ will consist of episodes of the task completed using the expert policy.

1.2 Inputs

Our states can be more specifically defined, as a particular setting of a number of features the properly define the states:

$$\forall s \in \mathcal{S}\, \exists F_s : \{f_{s,1}, f_{s,2}, \cdots, f_{s,m}\} st. s \equiv F_s$$

Where $n = 1, 2, \cdots, N$ N being the number states, infinite, later we will talk about N as being the number of observed states. and $m = 1, 2, \cdots, M$ M being the number of features.

1.3 Re implementation

I will be doing a custom implementation of a CRF and and LSTM to solve this problem and then I will be re implementing Generative Adversarial Imitation Learning (GAIL) as my second Neural Network. The GAIL architecture is based upon Generative Adversarial Networks (GANs) and have been proven to work quite well in simulations.

2 Motivation: How would this make the world a better place?

The realm of Imitation Learning is very hot right now. As can easily be imagined what If we could teach robot or control system a new skill just by showing it what to do, without specifically programming it or gathering training data to run traditional machine learning algorithms on it to teach it.

2.1 Domain

The domain is Imitation Learning (IL). Where the goal is to learn a policy π from expert demonstrations such that the learn policy performs just as well in the environment as the expert policy pi^*. I will be using classical control tasks from Open-AI gym to attempt this task. Most IL paper show results from working with classical control tasks as a baseline to see if their method works. This is because the tasks are relatively simple, and many of them have close form solutions. This means that evaluation of the method is meaningful, simple, and easy to interpret.

2.2 Importance

These problems are already solved as they are quite easy. My view of importance is to try to create a method that is faster that some of the IL methods currently in use, as well as detecting adversarial demonstrations as stretch goal, which is a very import task in this domain right now.

2.3 Helpful

If a simple, theoretically sound, fast algorithm can be created that can detect adversarial demonstrations that would be of great interest and use for the IL community.

3 Data set: Which benchmark are you using for training and evaluation?

3.1 Download

As I said earlier there are close form solutions to many of these environments. So I was able to generate however much data I need to solve the problem. [**1606.01540**]

3.2 Confirmation

I promise that I have the ability to generate this data.

3.3 Train/Test Split

Due to the nature of this domain, I will be training on the entire set of expert data, and will be testing by seeing how my learned policy performers on a various number of starting states.

3.4 Suitability

The data is generated directly by a simulator or the task, so it fits the task perfectly.

3.5 Feasibility

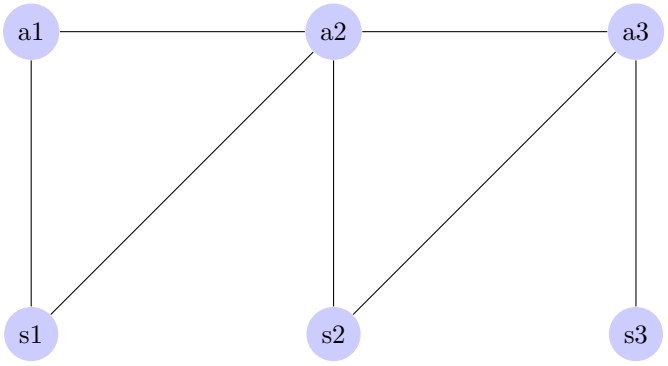
There is no need to label data for this task.

4 CRF Approach (How to solve the task with a CRF?)

4.1 Application

I will be using a linear chain CRF to solve this task. As there is only need to know the previous action the current action and current/previous state to understand all relevant information.

4.2 CRF factor graph



4.3 Features

As shown in the factor graph above I used Previous and current states as well as previous and current actions as features. At first I used all of these directly. However due to the infinite state space the CRF was not able to learn anything. So I decided on discritizing the states space. I rounded and fixed each observed states' features to 2 decimal places more than the range of values for that particular feature. Meaning if the range for a features was 1.4 I would round to 2 decimal places. If the range for a features was 0.56 I would round to 3 decimal places etc. Then I used a unique string representation of each of these discrete values. This resulted In a manageable features space, that I was able to achieve good results with. My CRFs had between 500 and 2500 weights. I used 100 demonstrations with around 100 to 200 state action pairs each for about 5000-10000 data points.

4.4 Customization

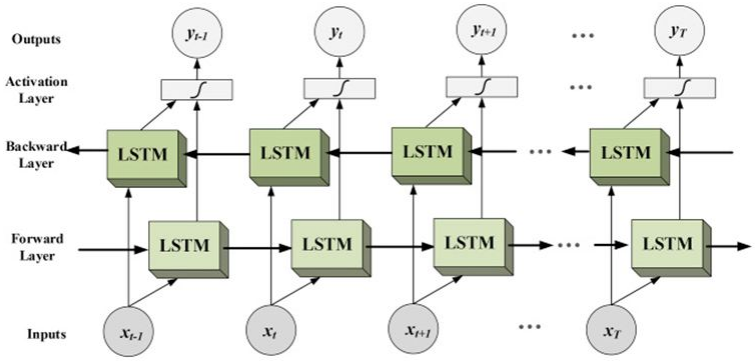
This implementation of the CRF worked better than I initially anticipated. For the stretch goal for this project I was working with a crf that does probabilistic reweighing demonstrations as it trains. I was not able to compile many results for this so nor did I have much to compare it too. InfoGAIL takes too long to train.

5 Neural Network Approach (Solve the task with a neural network?)

5.1 LSTM

5.1.1 Architecture

I used a bidirectional LSTM with a linear layer with softmax activation to output. I believe that this is a good architecture because it is able to learn the sequential nature of the task quite well. It did have one drawback though, It struggled to predict accurately at the beginning of the episode, when thee sequences were short.



5.1.2 Customization

The LSTM had difficulty deal with short sequences. My feeling that this was an issue with my data not the architecture itself. The sequences were all of complete trajectories, so when it encountered smaller sequences it made bad predicts. My solution to this was to augment my data. I took the training data and appended every sub sequence for each sequence that started at the initial position. Meaning if there was a sequence of length 10, I also included each sequence of length 1, 2, 3 ... 9. This improved performance drastically and made it competitive with my other implementations.

5.1.3 Size and Scalability

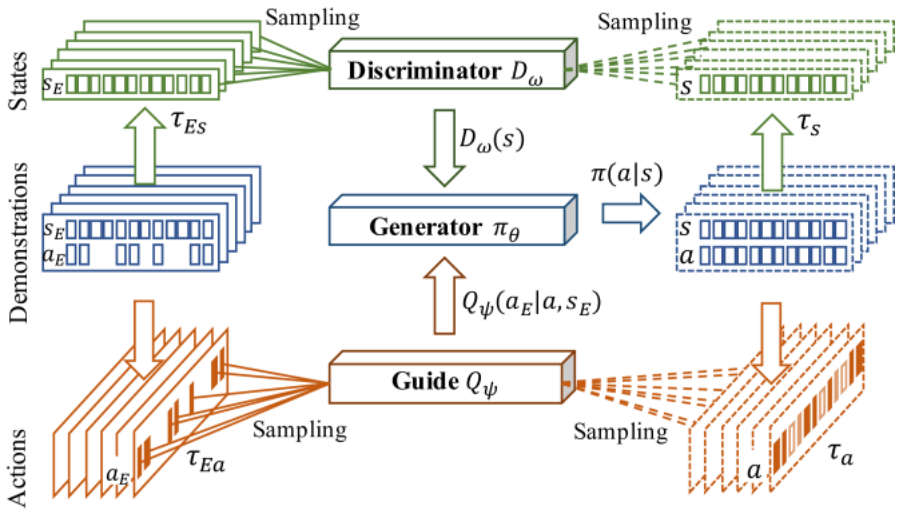
After permuting my sequences I had 500,000 sequences, and depending on the task I was solve I have between 1343 and 4923 parameters. These models took anywhere from 30 minutes to 2 hours to train. Which was less time than GAIL did but took much longer than the CRF which was nearly instantaneous.

5.1.4 Rational

An LSTM by nature is good at modeling sequential data. I chose this over a CNN, RNN of GRU because I thought that the Gate parameters within the model would be able to learn and discriminate between adversarial data. This did not end up working out.

5.2 GAIL

GAIL is a GAN, so it trains both a discriminator and a generator. The generator attempts to learn a policy from the state-action pairs. The discriminator tries to sub samples the training states, and the states generated from the generator, and tires to discriminate between them. The model converges when the discriminator can no longer learn a difference between the distribution of the training states and the generated states. The generator Learns the sequential nature of the task, because that is an essential part of learning to generate data that comes from the same distribution as the training data.



5.2.1 Size and Scalability

I trained GAIL on 100 expert demonstrations and there are about 20000 - 30000 learn-able weights. This would seem like it would be difficult for GAIL to learn, which is true. But for the sake of equal comparison I wanted to use the same number of expert demonstrations to train. GAIL re-samples the demonstrations however so it was still able to learn and good policy.

5.2.2 Rational

GAIL is one of the most know methods of Imitation Learning right now. It ability to solve both sequential and non sequential tasks is impressive, and it scales quite well. [**ho2016generative**] One drawback in training time. The extension of GAIL InfoGAIL is a very promising for the stretch goal for this project, detecting adversarial demonstrations [**li2017infogail**]. However I did not have time to implement or train InfoGAIL.

6 Evaluating: Success

6.1 Measures

The are really only two metrics I can use to score the models, accuracy and reward. Accuracy was evaluated on a test set of 20 expert demonstration, this is not a very good metric because it does not show how the model actually performs as a policy. Rewards are calculated by summing the output of the reward function at each time-step for an episode. This is the best way to judge a models success by.

6.2 Comparison

In this section we explore the results of my models and how they compare to one another. I use two metrics accuracy, the mean accuracy of the model on a test set of 20 Episodes. This is not necessarily a good metric to use as it does not necessarily coincide with good performance in the environment. The second metric is mean rewards over 10 episodes. Here we use the model to predict what the next action should be given the previous state-action pairs in that episode. This is a very good metric because it shoes how your model actually is as a policy, and can be compared directly to the expert policy. In the following tables we compare the performances of each of our three models with both the expert policy and a random policy. Entries in bold font have a significantly different mean from the Expert at a 0.05 level of significance. This means that significant entries are actually bad in our case, since our goal is to imitate the expert policy. This brings us to another problem with using accuracy, the expert has 100% accuracy, which makes paired t tests conducted with the expert on accuracy hard to interpret.

Table 1: Mountain Car

Metric	Accuracy	Reward
Expert	1.00 ± 0.00	-99 ± 5.92
CRF	**0.97 ± 0.013**	**-108.5 ± 6.47**
LSTM	0.978 ± 0.003	-106.7 ± 7.78
GAIL	**0.995 ± 0.005**	-105.25 ± 7.73
Random	NA	**-200 ± 0**

Table 2: Cart Pole

Metric	Accuracy	Reward
Expert	1.00 ± 0.00	200 ± 0.00
CRF	0.99 ± 0.024	200 ± 0.00
LSTM	0.999 ± 0.0008	200 ± 0.00
GAIL	**0.99 ± 0.0013**	200 ± 0.00
Random	NA	**26.45 ± 5.25**

Table 3: Acrobot

Metric	Accuracy	Reward
Expert	1.00 ± 0.00	-103.1 ± 27.53
CRF	**0.976 ± 0.013**	-91.95 ± 16.69
LSTM	**0.99 ± 0.0036**	-85.5 ± 10.34
GAIL	**0.9957 ± 0.0031**	-96 ± 23.65
Random	NA	**-500 ± 0**

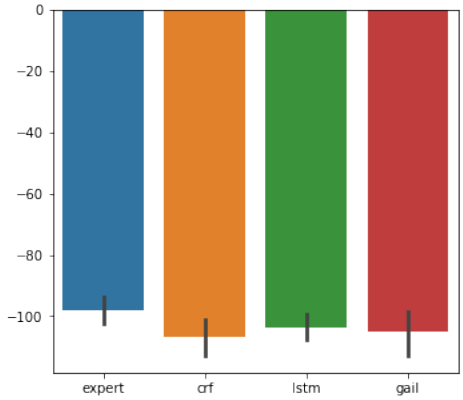
6.3 Significance

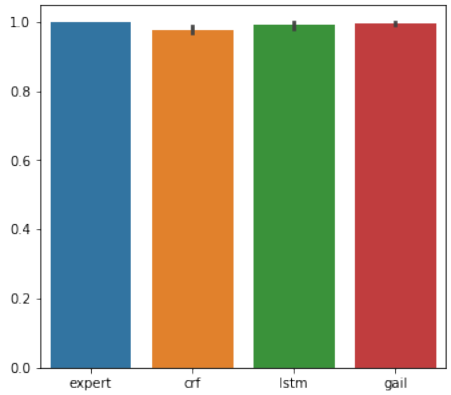
As I stated in the last section the bold entries in bold had significantly different means from the expert. However in other analysis we found that all of the models were significantly different from a random policy and non of the models were significantly different from one another based upon reward.

6.4 Plot /Graphs

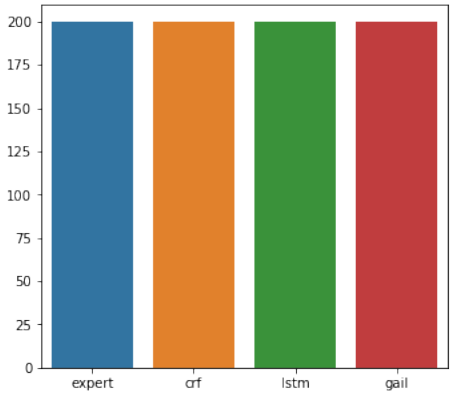
Following are Plots of how the differnt policies performed.

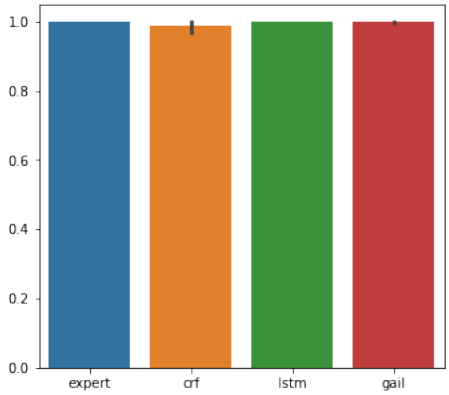
6.4.1 Mountain Car



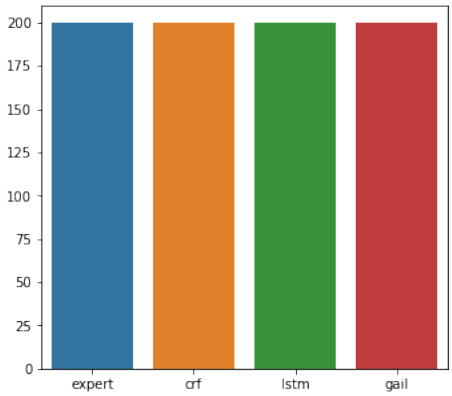


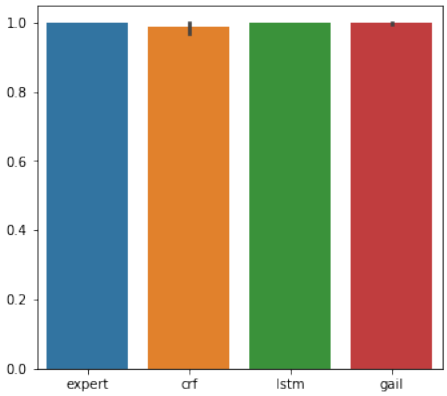
6.4.2 Cart-Pole





6.4.3 Acrobot


200
175
150
125
100
75
50
25
0
expert
crf
lstm
gail


1.0
0.8
0.6
0.4
0.2
0.0
expert
crf
lstm
gail