Learning from Failed Demonstrations

Abdul Rafay Khalid

I. Introduction

MOST learning from demonstration techniques assume that the demonstrator is capable of perfectly performing a task and try to learn a model for the task based on them. However, certain tasks can only be performed by experts and thus the demonstrator may fail to generate any successful trials. These trials contain crucial information on what not to do. Motivated by the ability of infants to learn tasks from failed demonstrations [4], we attempt to find an approach that enables us to learn tasks from failures.

For this project we apply the approach to the classic problem of inverted pendulum on a cart and attempt to come up with methods to benefit from both the successful and unsuccessful portions of a demonstration.

April 4, 2015

II. Related Work

This project builds upon existing work in the Robot Learning from Demonstration field. For such an approach, human users are recorded multiple times while doing a task. These trajectories are then used by the robot to infer a model for the task. Schaal [5] describes some early work in which learning from demonstration is coupled with existing techniques in reinforcement learning for robotic pole balancing. More recent work in this area is described by Argall et al [1] which also analyses the various design choices. These approaches try to learn a mapping between the state and correct action to be taken by the robot, however they all assume that the examples used for training are correct.

More recently, there is an attempt to give up the assumption of perfectly correct demonstrations to in favor of suboptimal ones. The POWER algorithm proposed by [3] uses demonstrations to learn the mapping between states and actions which is then improved by using reinforcement learning. The problem with using reinforcement learning is that it requires an explicit reward function. Coming up with a good reward function requires domain specific knowledge. In addition such a system would fail if there were no successful demonstrations.

Our project builds upon the work of Grollman et al[2] which tries to address these challenges. It assumes that all the demonstrations are failures and tries to come up with strategies for exploration in order to efficiently find a successful trajectory. It does not require the use of a reward function and treats each failure equally. This project tries to apply their algorithm to a different kind of scenario. They demonstrate their system on two tasks; flipping a block and throwing a ball into a basket. For both of these tasks failure is a binary variable. We attempt to apply it to the case of a pendulum on a cart for which not all failures are equal and within each trajectory certain portions could be success while others could be failures. We need to come up with some heuristic to separate the trajectory as well as use their later work on giving weights to trajectories so that not all failures are treated equally.

III. Problem Definition

The problem is to generate exploratory trajectories which can result in success after looking at failed demonstrations.

Notation:

ξ represents the state of the robot

$\dot{\xi}$ represent the velocity of the robot

$\dot{\xi} = f_\theta(\xi)$ is a non linear function that captures the relationship between the state and velocity of a robot

Input:

$A_k = [(\xi, \dot{\xi})_0, (\xi, \dot{\xi})_1, ...(\xi, \dot{\xi})_N]$

$S = \{A_0, A_1, ..., A_{N_s}\}$ where A is a trajectory demonstrated to the robot that leads to failure and S is the set of N_s such trajectories

Output:

$\dot{\xi} = g_\theta(\xi)$ which is a mapping between the state and the velocities such that following this mapping leads the user to success.

IV. Methods

We intend to evaluate the Learning From Failed Demonstrations algorithm proposed by Grollman et al [2] to the case of an inverted pendulum on a cart. This is a standard problem in controls where the task is to keep the pendulum upright by applying force on the cart. Refer to figure 3. The algorithm for the method is reproduced below:

Collect S human failed attempts
Fit a GMM (θ) on S
while Robot has not succeeded **do**
  t = 0
  ξ_t = current state of the system
  $\dot{\xi}_t$ = DMM($\dot{\xi}$,ξ,θ)
  while $|\dot{\xi}| \neq 0$ and not timeout **do**
    Maximize P($\dot{\xi}|\xi$) with gradient ascent
    apply $\dot{\xi}$ to the system
    t = t + 1
    ξ_t = Current state of the system
    $\dot{\xi}_t = \dot{\xi}_{t-1}$
  end while
  update θ
end while

The algorithm takes the multiple failed trials as input and tries to fit a Gaussian Mixture model on the joint distribution of ξ and $\dot{\xi}$. In this paper Grollman et al have come up with a probability distribution called the Donut which is the difference of two Gaussians but has the interesting property that it is low where Gaussian is high. In the outer while

loop they construct a Donut Mixture Model based on the parameters of the GMM. The inner loop makes the robot follow a trajectory by choosing a velocity that corresponds to the maxima of the DMM given the current state. This allows us to explore trajectories which are different from what the user did.

A. Generate Trajectories and Build Model

For the inverted pendulum system , we use $\xi = \theta$ and $\dot{\xi} = F$. I modeled the inverted pendulum system using the Gazebo simulation environment.This is interfaced with a ROS node to record trajectories either based on keyboard input or using some control law. I initially tried generating demonstrations with the keyboard but found it particularly difficult. Instead I generated my demonstrations by simulating a noisy proportional controller. A bunch of demonstrations are collected using the simulation control node and Gazebo. The first step in the algorithm is to build a Gaussian mixture model from the acquired demonstrations. This has been implemented as a matlab script that takes the trajectories generated and fits the mixture model using Expectation Maximization with the minimum Bayes Information Criterion.

B. Perform Exploration and Update Model

Once the Gaussian mixture model have been trained we need to find exploratory trajectories on the basis of that model. To perform the exploration we need to compute a Donut Mixture Model from A GMM. This is constructed from a difference of two Gaussian distribution ans is parametrized by an exploration parameter ϵ. By varying the parameter from 0 to 1 we can go from a distribution that replicates the Gaussian to one which is something like the inverse of a Gaussian. Figure 2 shows the variation of the distribution as we modify ϵ. A ROS node is written in python using PyPr which allows us to compute the most likely Force according the DMM for the current state. This node acts as a service which is accessed by the simulation controller to apply the appropriate force on the basis of the state received from the simulation. This interaction is shown in figure 1. Inside the exploration node we first compute the conditional distribution $P(\dot{\xi}|\xi)$ from the joint distribution $P(\dot{\xi}, \xi)$ which is represented by the GMM. Depending on the current state we compute the exploration parameter as follows.

$$\epsilon = \frac{1}{1 + Var[\dot{\xi}|\xi]}$$

Var represents the variance in the values of force applied for that particular state across the trajectories.In cases when the trajectories agree this is close to 0 while in cases when there is disagreement this will approach 1. Depending on the current value of exploration we convert the GMM to DMM. As we do not have a closed form solution for the arg max of the DMM, we use gradient ascent to find the local maxima and return that as the appropriate force. The simulation controller then applies this force in the simulation and the process continues until the trajectory reaches a predetermined length.
To update the model, we simply retrain our model with the additional data from the new exploration. The process of generating trajectories continues until we get a trajectory of our choice.

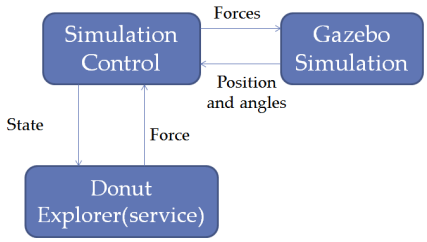


Fig. 1. Data exchange between the nodes

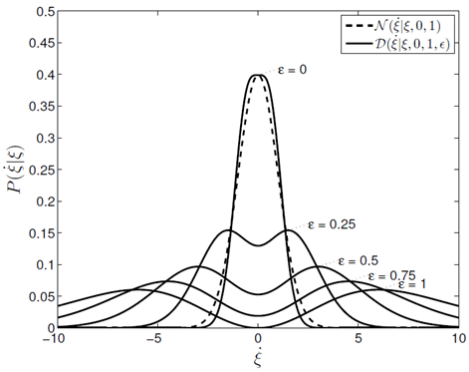


Fig. 2. Variation in Donut with the exploration parameter

C. Building model with cost

For this project, we also try assigning a cost to each trajectory and using that cost to improve the model. Failure in the inverted pendulum system is not discrete and to model how good or bad a trajectory is we use the following cost.

$$C = \frac{1}{T} \sum_{t=0}^{T} |\theta|$$

As this algorithm tries to build a model of failed demonstrations, bas demonstrations should contribute more to the model while the better ones should contribute less. In order to achieve this, we train a separate GMM on each of the trajectories. We then draw samples from each of these GMMs. The number of samples that is chosen from each model is proportional to the cost of the trajectory. These samples are used to generate a GMM which is then used for exploration. A similar approach is taken when updating the model based on explored trajectories.

V. Results

Figure 4 shows the simulation developed for the project. We show results for three cases. We first describe and analyse the results on synthetic data to evaluate correctness of our implementation. We then show the performance of this algorithm on the inverted pendulum with and without using reward.

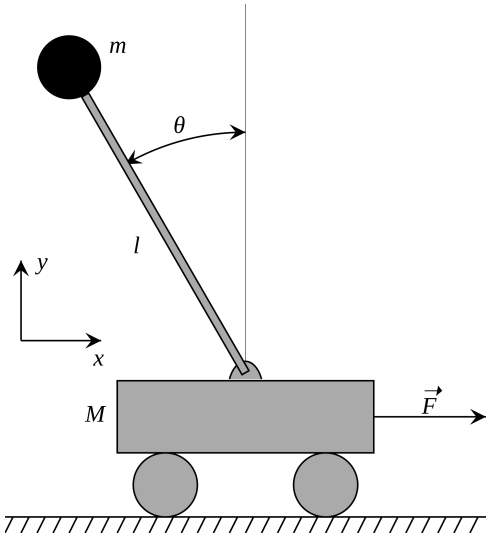


Fig. 3. Diagram of the inverted pendulum system

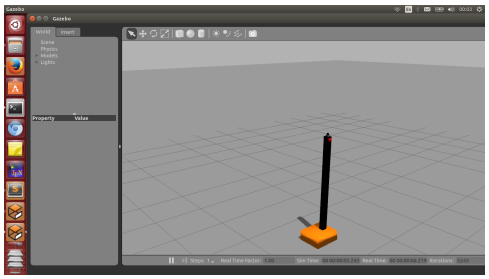


Fig. 4. Simulation of the inverted pendulum system

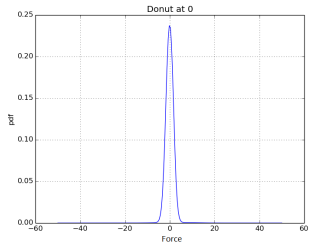


Fig. 5. Donut at 0

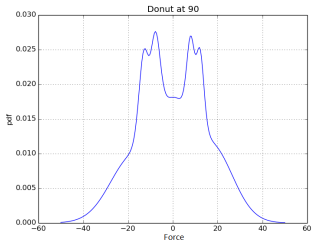
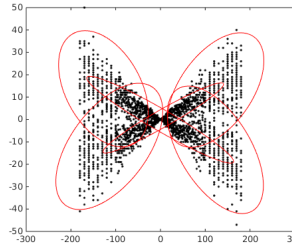


Fig. 6. Donut at 90



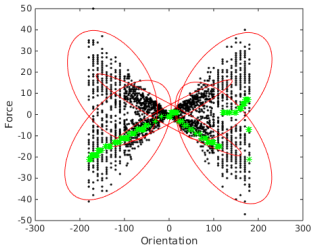
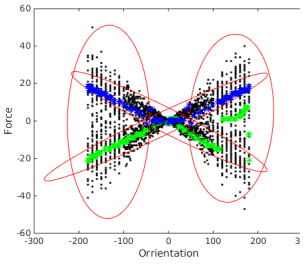


Fig. 7. Synthetic Data with GMM Fig. 8. Synthetic Data Explo 1



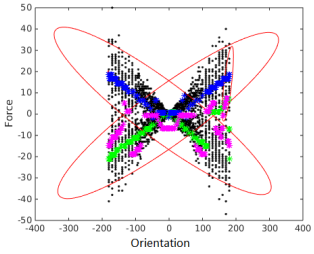


Fig. 9. Synthetic Data Explo 2 Fig. 10. Synthetic Data Explo 3

A. *Synthetic Data*

We generate synthetic data characterizing the relationship between theta and Force to test the implementation. For each value of theta, the value of force is taken to be proportional to it with a Gaussian noise added which is proportional to theta. In one demo the constant of proportionality is negative while in the other one it is positive. Figure 7 shows the data with the fitted model. Notice that as designed there is less variance for smaller theta and more variance for larger theta. The pdf of the DMMs generated at theta = 0 and theta = 90 is shown in figures 5 and 6. It can be seen that as expected in low variance are the donut resembles a Gaussian. On the other hand for a state for which the variance is high the pdf is higher away from the center. We notice that the pdf is mutimodal. This means that when we use a technique such as gradient ascent to find the mode, we may find any of the two modes depending on the initialization thus introducing unwanted randomness into the algorithm.

Figure 8 to 10 show the exploration paths taken in the space. In Figure 9 note that the blue trajectory aligns with the demonstrations when orientation is close to zero and moves towards edges of the Gaussian for larger values of orientation. After the model updates to that of figure 10 the region near zero also becomes a high variance region and erratic exploration is witnessed in the pink trajectory.

B. *Inverted Pendulum without reward*

We use the simulation control node to save demonstration trajectories using simple proportional control with noise. Unlike the synthetically generated data this data is recorded from actual gazebo simulation and thus there will be more datapoints in areas where the pendulum spends more time. For this experiment, following the paper exactly we treat all failures as equal. In order to assess our performance quantitatively we will evaluate cost for the first 7.5 secs, 15 secs and the complete trajectory length of 30 seconds.

Figures 11 to 16 show the 3 std dev ellipses for the models that are fitted on the data. The most recent exploration is shown in green. In most cases, the data is primarily described by two broad gaussians capturing the uncertain region with large theta and a third narrow gaussian that captures the behavior near zero. We can also observe the randomness introduced by using Gradient Ascent in these results. If we look at fig 14 and and 16 which show trajectories 7 and 11 respectively we can observe that even though the model looks very similar the trajectories taken are different. In 7 half the trajectory lies below the data while the other half lies above, while for 11 all of the trajectory lies below the data.

As for the performance, looking at the 18 and table V-B clearly shows that the exploration performed by the Donut method is

very erratic. Looking at the cost for the whole trajectory and comparing it to the cost of actual demonstrations in table V-B we note that the explorations perform very poorly. In order to do well throughout the demonstration we need to have a good mapping from angle to force for all values of the angle. The results indicate that the space spanned by the successful trajectories is very small and it may be very difficult for such a method to reach there just using exploration. However, not all hope is lost.

If we observe the cost for 15 seconds and 7.5 seconds, we note that trajectories 3 and 9 performed to a decent extent. Comparing them to the actual demonstrations, we see that trajectory 3 did as well as the best actual demonstration for the first 15 seconds. In addition trajectory 9 outperformed the best demonstration by a factor of 3. These results mean that with sufficient the algorithm will partially find the correct mapping. This indicates that we might be able to use this method in an incremental fashion. That is learning the correct force for some theta, then using that to bootstrap further exploration which will only explore for the remaining angles.

Tr No (Actual)	1	2	3	4
Cost (30s)	93.0757	104.7473	91.6079	76.1565
Cost (15s)	61.4735	88.4744	84.9328	66.5753
Cost (7.5s)	29.5136	43.7652	46.4241	37.9770

TABLE I
COSTS FROM ACTUAL TRAJECTORIES

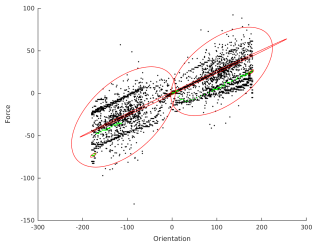


Fig. 11. Inverted Pendulum Exp 1

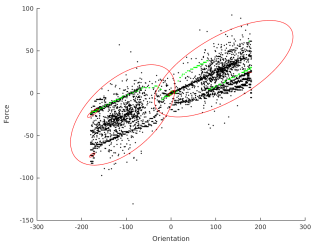


Fig. 12. Inverted Pendulum Exp 3

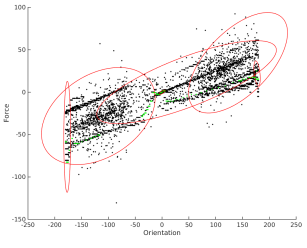


Fig. 13. Inverted Pendulum Exp 5

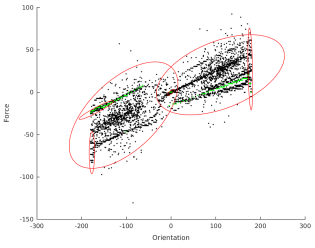


Fig. 14. Inverted Pendulum Exp 7

C. Inverted Pendulum with reward

As described in the previous section, we can use the cost of the trajectories to drive the model building process. We use the same 4 actual demonstrations to build the model and carry on exploration. However for this experiment, we build and update the models using the cost for the 30s trajectory.

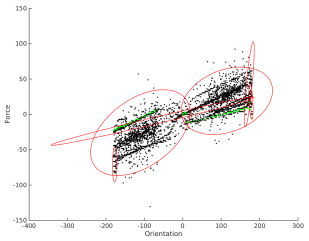


Fig. 15. Inverted Pendulum Exp 9

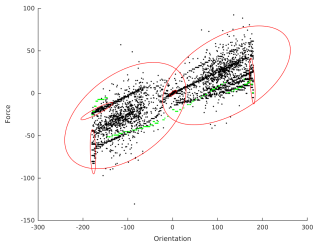


Fig. 16. Inverted Pendulum Exp 11

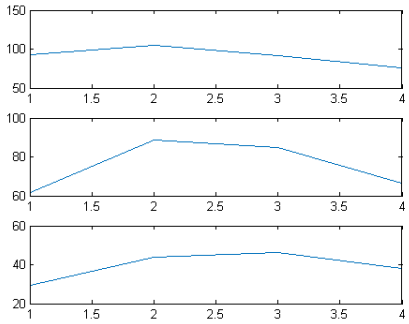


Fig. 17. Cost for the 4 actual Demonstration. Top to Bottom (30s, 15s,7.5s)

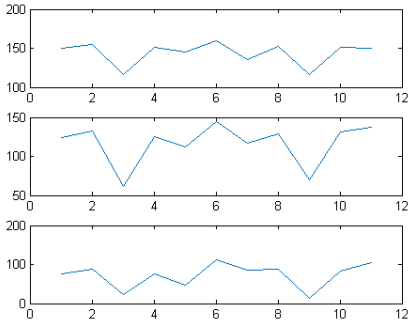


Fig. 18. Cost for the 11 explorations. Top to Bottom (30s, 15s,7.5s)

The fitted models and the exploration is shown in figures 20 to 25. If we compare the models in the case between reward and no reward, we notice that the initial model is very similar however it evolves very differently. From two broad gaussians describing the data it turns into a single variance of very large variance.

As is evident from Figure 19 which shows the cost for different trajectories this method performs very poorly on the data. After the first trajectory the performance goes down and continues like that. Even with exploration there is no improvement even after 10 trials. Also, the performance remains consistent throughout the trajectory as the cost follows the same trend for 30s,15s and 7.5s.

We can get some idea from the formed models why this approach is failing. As we keep on giving higher weights to the poorer trajectories, the model loses information about the relatively successful trials. As we can fail in very different

Tr No (Explored)	1	2	3	4	5	6	7	8	9	10	11
Cost (30s)	150.3399	154.4538	116.2481	151.1992	145.2563	159.6868	135.1678	152.8853	116.3262	151.6808	150.2374
Cost (15s)	123.8537	131.9417	61.9360	125.5217	111.6739	144.7901	116.9440	128.9745	70.4682	131.7492	137.8092
Cost (7.5s)	76.0133	87.1472	22.5199	75.6113	46.4094	112.0619	85.4763	86.8828	12.5943	84.1958	103.7207

TABLE II
COSTS FROM EXPLORED TRAJECTORIES WITHOUT REWARD

ways, the whole model now has a very large variance. Now the Donut explorer sees that variance for all states is high and thus it maximizes exploration. However as we discussed before the space of successful demonstrations is very small and thus simply doing extreme exploration is not very likely to help us get there. We keep exploring poorer trajectories, the variance keeps increasing and our chances of finding a successful trajectory keep going down.

The results indicate that for such as complex problem, there is a need to come up with a better way to use the reward. It may be helpful to use shorter portions of the trajectory, which could then lead to partial improvements.

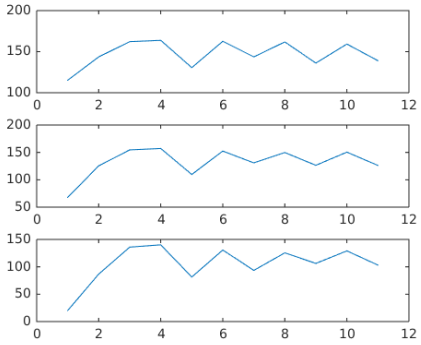


Fig. 19. Cost for the 11 explorations with reward. Top to Bottom (30s, 15s,7.5s)

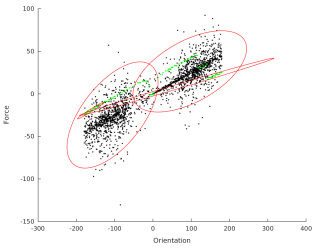


Fig. 20. Inverted Pendulum with reward Exp 1

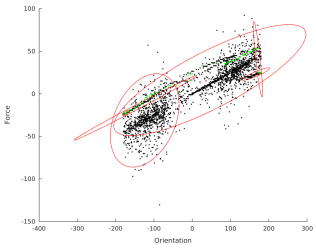


Fig. 21. Inverted Pendulum with reward Exp 3

D. Higher Dimensions

I also tried using both the velocity of the cart and the pendulum orientation as the state, however that did not have a qualitative improvement on the results. Therefore, I abandoned it in favor of simplicity.

VI. Conclusion and Future Work

This project showed that the inverted pendulum problem is much harder than the sort of experiments used by the authors.

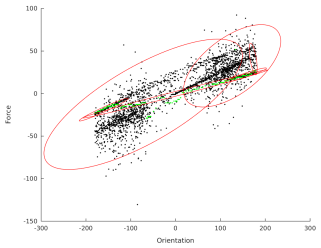


Fig. 22. Inverted Pendulum with reward Exp 5

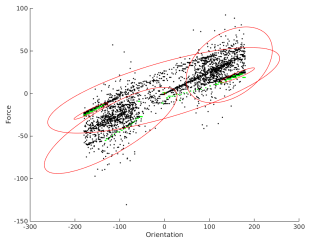


Fig. 23. Inverted Pendulum with reward Exp 7

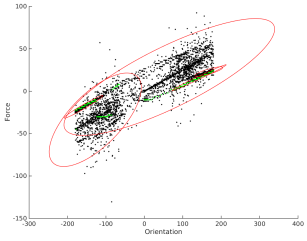


Fig. 24. Inverted Pendulum with reward Exp 9

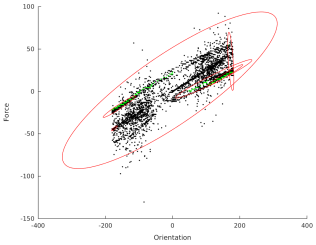


Fig. 25. Inverted Pendulum with reward Exp 11

Although the reward methodology failed miserably, there is some hope in using the approach without reward. The results for short durations during exploration look promising for the case with no reward. The best future direction to pursue is to use the exploration in an iterative fashion. That is we explore until we improve a part the response for certain angles in the state and use that mapping to bootstrap subsequent demonstrations. However, even with that it appears that this algorithm might not prove effective for this class of problems. In addition, incorporating reward into this and using this as a part of a learning from successful demonstrations approach is a difficult but interesting future direction of work.

VII. Acknowledgments

I am grateful to Jan Petur Petersen for developing the PyPr package. I have used that extensively in the exploration node. In addition i have used the statistics toolbox in Matlab for fitting mixture models.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3804–3809.

[3] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.

[4] A. N. Meltzoff, "Understanding the intentions of others: re-enactment of intended acts by 18-month-old children." *Developmental psychology*, vol. 31, no. 5, p. 838, 1995.
[5] S. Schaal *et al.*, "Learning from demonstration," *Advances in neural information processing systems*, pp. 1040–1046, 1997.