Donut as I do: Learning from failed demonstrations

Daniel H Grollman and Aude Billard
Learning Algorithms and Systems Laboratory, EPFL
{daniel.grollman, aude.billard}@epfl.ch

***Abstract*— The canonical Robot Learning from Demonstration scenario has a robot observing human demonstrations of a task or behavior in a few situations, and then developing a generalized controller. Current work further refines the learned system, often to perform the task better than the human could. However, the underlying assumption is that the demonstrations are successful, and are appropriate to reproduce. We, instead, consider the possibility that the human has failed in their attempt, and their demonstration is an example of what not to do. Thus, instead of maximizing the similarity of generated behaviors to those of the demonstrators, we examine two methods that deliberately avoid repeating the human's mistakes.**

I. Introduction

One current problem in robotics is that while modern robots are physically capable of performing many useful tasks (e.g. heart surgery, space-station repair, bomb disposal), they are only able to do so under constant supervision and control of expert human operators or in highly engineered environments. That is, what are lacking are autonomous controllers for performing these skills in novel situations without human oversight. Robot Learning from Demonstration (RLfD) is an attempt to fill this void by enabling new robot controllers to be developed without requiring analytical deconstruction and explicit coding of the desired behavior.

Currently, RLfD typically first collects a set of examples, wherein a human user demonstrates acceptable task execution. These examples are then somehow put into the robot's frame of reference, and a generalized control policy is extracted. This policy can be improved via further interaction with the human, or by optimizing known or inferred criteria.

Many different approaches along these lines exist [1], [2], and as the field coalesces, overarching formalisms are being developed [3]. We note that early work was typically in discrete state and action spaces and took the human demonstrations as indicative of correct or *optimal* behavior. However, as recent research has shifted to continuous domains, there is more focus on optimizing the learned behavior beyond the demonstrator's capabilities.

In this paper, we propose to take the next logical step. That is, the initial assumption (which was valid in small, finite, discrete spaces) was that human demonstrations were optimal and the robot was told to "Do as I do." Currently, this assumption is being relaxed to allow improvement over the demonstrations, which still positively bias the robot: "Do *nearly* as I do." We here consider the situation where the humans are not only sub-optimal, but incapable of performing the task, and their demonstrations must be treated in part as negative biases: "Do *not* as I do."

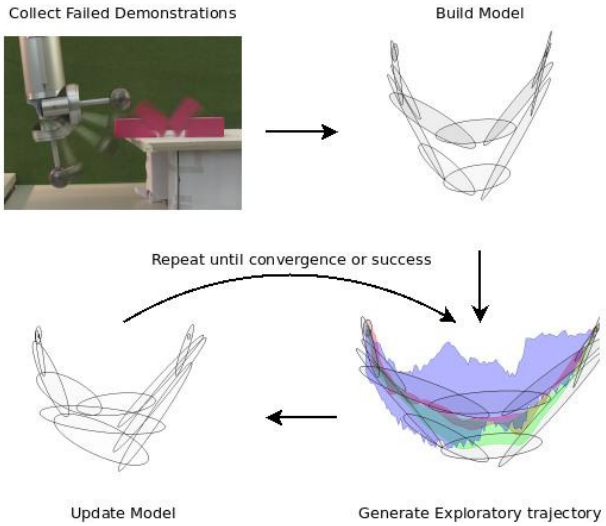


Fig. 1: An overview of our approach. After modeling collected failed demonstrations as in regular RLfD, we generate exploratory trajectories by assuming that when demonstrations disagree, they actually indicate what *not* to do. We update the model after exploration and repeat until successful task performance or trajectory convergence. In all figures distributions are shown $\pm$ 3 standard deviations.

II. Related Work

Of course, a human's failed demonstrations are not entirely devoid of useful information. We assume that the humans are attempting to perform the desired task, and not performing unrelated motions. Failure is then due perhaps to lack of skill or effort. We draw inspiration from work with humans showing that infants are able to successfully perform tasks that they have only seen failed examples of [4].

In RLfD, the most closely related work assumes the demonstrations are correct, yet suboptimal, and incorporates Reinforcement Learning (RL) to optimize the learned system. For example, the PoWER algorithm uses demonstrations to initialize a policy, and then improves it with respect to a known reward function by performing exploratory rollouts, perturbing the the current policy by state-dependent noise [5]. One drawback is that if the initial demonstrations are very suboptimal (such as failures), this and similar techniques may be unable to locate a successful policy. By acknowledging the failure of demonstrations and deliberately avoiding them, we here provide an alternate means of generating exploratory trajectories.

Alternatively, if the reward function is not known, or the user does not wish to specify one, Inverse Reinforcement Learning (IRL) techniques can estimate one from the demonstrations themselves [6]. The current state-of-the-art requires that the reward function be linear in the feature space, so feature selection is very important. Further, they allow for the inclusion of prior information, indicating which features are the most important or what their values should be. By providing 'correct' values for some features, the system is told, in effect, to ignore certain aspects of the demonstrations. We note here that ignoring portions of the demonstrations is not the same as actively avoiding them.

The above techniques learn an initial model from demonstration, and improve it on their own. Alternate work continues to use the demonstrator during the learning process, to provide more information. For example, corrective demonstrations may be provided when the learned policy acts inappropriately [7], [8]. However, these techniques assume that the demonstrator is able to *perform* what should have been done, and that either the training was ambiguous, or the learning was incorrect. Instead users may simply *indicate* how the behavior should change using a set of operators, and therefore train a system that outperforms themselves, without needing to explicitly specify a reward function [9].

These approaches all assume that the initial demonstrations are basically correct, and that issues remaining after learning are due to stochastic humans or improper learning (poor generalization, simplified models, etc). Fully failed demonstrations are usually discarded, either explicitly by researchers, or implicitly in the algorithms themselves. We believe that these failures have instructive utility and can place constraints on what should and should not be explored.

III. Methodology

We compare approaches to RLfD of motion control based on Dynamical Systems (DS) [10] and Gaussian Mixture Models (GMM)[11]. Taking the state of the robot, ξ and its first derivative $\dot{\xi}$ to be D-dimensional vectors, a demonstration is a trajectory through this state-velocity space, $\mathbf{x}^n = \{\xi_t^n, \dot{\xi}_t^n\}_{t=1}^{T^n}$. From a set of N demonstrations $\mathbf{X} = \{\mathbf{x}^n\}_{n=1}^N$ of possibly different lengths ($T^i \neq T^j, i \neq j$), we approximate the distribution of observed state-velocity pairs with a GMM where the probability of a given pair is:

$$P(\xi, \dot{\xi}|\theta) = \sum_{k=1}^{K} \rho^k \mathcal{N}(\xi, \dot{\xi}; \mu^k, \Sigma^k) \tag{1}$$

$\mathcal{N}$ is the standard normal distribution and $\theta = \{K, \{\rho^k, \mu^k, \Sigma^k\}_{k=1}^K\}$ are the collected parameters, termed the number of components (positive integer) and the priors (positive real, sum to 1), means ($2D$ real vector) and covariances ($2D \times 2D$ psd matrix) of each component.

To deal with mismatches in the size of the state and velocity spaces, we first normalize our data such that all dimensions are mean zero and have unit variance. We then fit these parameters using a combination of the Expectation-Maximization algorithm [12] (initialized with Kmeans) to maximize the probability of observed data for a given value of K, and the Bayesian Information Criterion [13], a penalized likelihood method, to select K itself. As K is discovered in a data-driven fashion, our overall technique can be considered nonparametric – the total number of parameters used to model the data is not set apriori.

To compute $\dot{\xi}$ for a given ξ we require the conditional:

$$P(\dot{\xi}|\xi, \theta) = \sum_{k=1}^{K} \tilde{\rho}^k(\xi, \theta) \mathcal{N}(\dot{\xi}; \tilde{\mu}^k(\xi, \theta), \tilde{\Sigma}^k(\theta)) \tag{2}$$

$$\tilde{\mu}^k(\xi, \theta) = \mu_{\dot{\xi}}^k + \Sigma_{\dot{\xi}\xi}^k \Sigma_{\xi\xi}^{k^{-1}} (\xi - \mu_\xi^k) \tag{3}$$

$$\tilde{\Sigma}^k(\theta) = \Sigma_{\dot{\xi}\dot{\xi}}^k - \Sigma_{\dot{\xi}\xi}^k \Sigma_{\xi\xi}^{k^{-1}} \Sigma_{\xi\dot{\xi}}^k \tag{4}$$

$$\tilde{\rho}^k(\xi, \theta) = \frac{\rho^k \mathcal{N}(\xi; \mu_\xi^k, \Sigma_{\xi\xi}^k)}{\sum_{k=1}^K \rho^k \mathcal{N}(\xi; \mu_\xi^k, \Sigma_{\xi\xi}^k)} \tag{5}$$

Note that $\tilde{\Sigma}^k$ does not depend on the current state. For clarity we drop the functional forms of the conditional parameters.

This conditional distribution over $\dot{\xi}$ is itself a GMM, with an overall mean and variance:

$$\tilde{E}[\dot{\xi}|\xi, \theta] = \sum_{k=1}^{K} \tilde{\rho}^k \tilde{\mu}^k \tag{6}$$

$$\tilde{V}[\dot{\xi}|\xi, \theta] = -\tilde{E}[\dot{\xi}|\xi, \theta]\tilde{E}[\dot{\xi}|\xi, \theta]^\top + \sum_{k=1}^{K} \tilde{\rho}^k (\tilde{\mu}^k \tilde{\mu}^{k\top} + \tilde{\Sigma}^k) \tag{7}$$

As illustrated in Figure 1, our general approach is to:

1) Collect a set of failed demonstrations ($\mathbf{X}$).
2) Build a model of what the demonstrators did (θ).
3) Use the model to generate a tentative trajectory that explores near the demonstrations ($\mathbf{x}^*$).
4) Run $\mathbf{x}^*$, update θ.
5) Repeat steps 3-4 until success or convergence.

Step 3 is the key, where we generate full trajectories from an initial state (ξ_1^*) by predicting a velocity, updating the state with that velocity, and repeating: $\mathbf{x}^* = \{\xi_t^*, \dot{\xi}_t^*\}_{t=1}^{T^*}, \xi_{t+1}^* = \xi_t^* + \dot{\xi}_t^*$. Prediction stops when $\dot{\xi} = 0$ or a predetermined time-length (1.5 times the longest demonstration) is passed. With faster computation we could generate velocities online, to react to perturbations and noise in the actuators. We use a low-level high-gain PID controller to avoid these issues.

For step 4, there are multiple ways to update the GMM incrementally [12]. A naive approach is to retrain the GMM on all of the available data. However, computation grows with the number of datapoints. We instead sample a fixed number of points from the current GMM, weight them to represent the total number of data points, and then combine them with the new trajectory for re-estimation. Note that K is unchanged in this approach.

Below we describe our different techniques for predicting a velocity for a given state from learned models. They are compared graphically in Figure 2. Our main intuition is that while the demonstrators are not succeeding, they are at least *attempting* to (roughly) perform the task. Thus, exploring in the vicinity of the demonstrations, while avoiding them exactly, may lead us to discover a way to succeed.

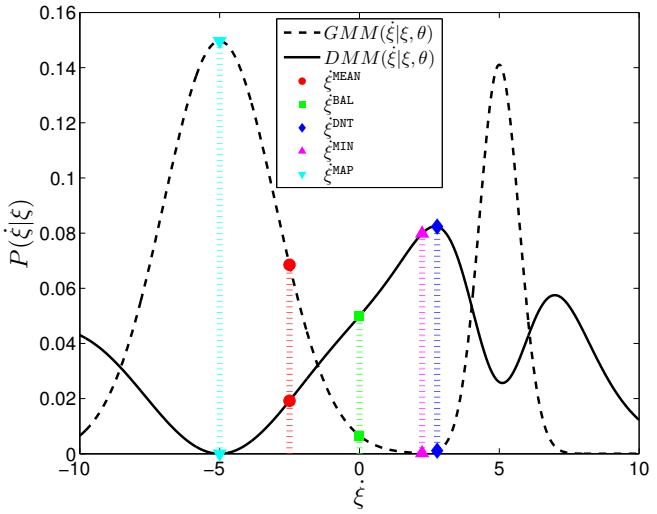


Fig. 2: Illustration of velocities generated by the different techniques introduced for a particular state. The dashed distribution is the GMM, and the solid distribution is what arises after replacing all Gaussians with Donuts.

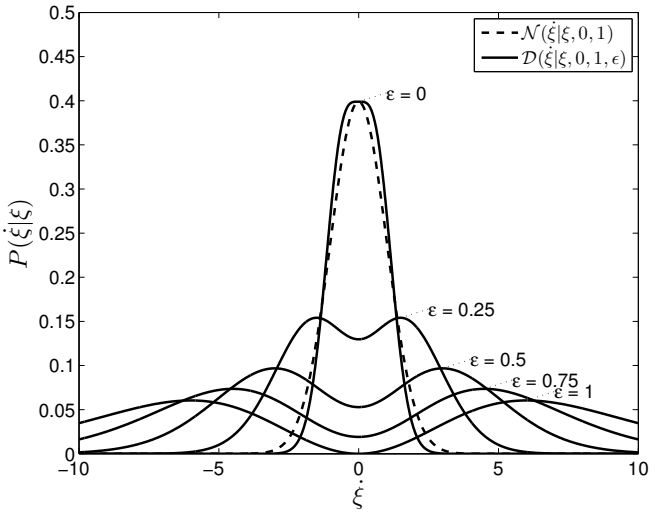


Fig. 3: The donut pseudo-inverse has an additional exploration parameter that determines how far away the peaks are from the base distribution.

A. Approach 1: Balanced Mean

A standard way to use GMMs in RLfD is to assume that the demonstrations are optimal, but corrupted by mean-zero Gaussian noise. Thus the expected value of the conditional distribution (Equation 6) estimates the noise-free function:

$$\dot{\xi}^{\texttt{MEAN}} = \tilde{E}[\dot{\xi}|\xi,\theta] \tag{8}$$

However, in our scenario this assumption does not hold. Particularly, as the demonstrations are failures, we do not assume they are Gaussianly distributed around success, and thus do not expect their mean to succeed. Further, incorporating the mean of a GMM back into the model will not lead to improvement, as the mean itself becomes more likely.

As an alternative we employ a balanced mean approach, which allows for improvement over iterations. We divide $\mathbf{X}$ into two classes: $\mathbf{X}^+$ and $\mathbf{X}^-$, reasoning that demonstrators that fail do not blindly repeat themselves. Rather, they try to correct themselves, and may end up failing in a different fashion. A binary division is the simplest case, but the approach may scale to multiple classes and dimensions.

From these two classes, we derive two GMMs parameterized by θ^+ and θ^-. Our overall estimated velocity is a weighted average of the means from each class:

$$\dot{\xi}^{\texttt{BAL}} = \alpha\tilde{E}[\dot{\xi}|\xi,\theta^+] + (1-\alpha)\tilde{E}[\dot{\xi}|\xi,\theta^-] \tag{9}$$

where $\alpha \in [0,1]$ is a mixing ratio. We set $\alpha = 0.5$, but in the future may make α state-dependent, perhaps measuring the relative variance in the predictions from the two classes.

The generated trajectory is necessarily in between the means of the two classes, so this approach requires that the demonstrations 'span' the area where correct behavior lies. If the classes are balanced to begin with (same number of demonstrated points), then Equations 9 and 6 are the same. However, as the generated trajectories are incorporated, only one class (and its mean) is updated at each iteration, shifting the overall mean. By assumption correct behavior lies somewhere between the two classes, and this technique approaches it in a fashion similar to that of binary search.

A further advantage of this technique is that when the classes 'agree' (produce nearly the same mean) then $\dot{\xi}^{\texttt{BAL}} \approx \dot{\xi}^{\texttt{MEAN}}$. This situation corresponds directly with assuming that the demonstrations are noisily correct[1]. Thus, overall, this approach will follow the demonstrations when the two classes agree and explore in between the data when the two classes disagree, which follows from our intuition that the demonstrations, while failures, are not all wrong.

B. Approach 2: Donut MAP

An alternate technique for generating velocities from a GMM is to use the Maximum a posteriori (MAP) value:

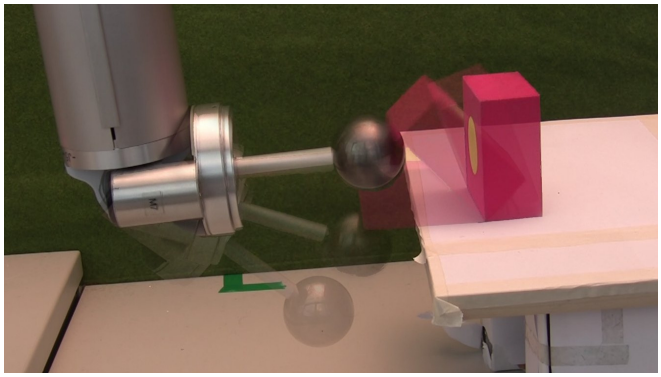
$$\dot{\xi}^{\texttt{MAP}} = \text{argmax}_{\dot{\xi}} P(\dot{\xi}|\xi,\theta) \tag{10}$$

Doing so makes sense when operating in non-convex spaces, where the combination of two successful demonstrations may not be appropriate. For example, turning left and turning right at a cliff's edge are both viable, but their convex mean (walk forward) is not. Similar to the standard mean, we observe that using the MAP incrementally will lead to rapid convergence and minimal improvement. However, with our assumption that demonstrations are indicative of what not to do, an alternate possibility is to generate those trajectories that are least likely under the model of the demonstrations, guaranteeing that we perform not-like the users:

$$\dot{\xi}^{\texttt{MIN}} = \text{argmin}_{\dot{\xi}} P(\dot{\xi}|\xi,\theta) \tag{11}$$

Two issues arise when using the minimum likelihood technique: 1) local minima only exist *between* the demonstrations, beyond the data the conditional probability continues

[1]Another possibility is that there is a bias in the demonstrations.



(a) FlipUp

(b) Basket

Fig. 4: Our robot tasks. FlipUp: get the foam block to stand on end, Basket: Launch the ball into the basket. Shown are successful trajectories learned with the Donut MAP approach from 2 initial failed demonstrations.

to decline. 2) local minima are always off of the data, so even when demonstrations are in agreement we avoid them.

We address both issues by instead finding a maxima of a mixture of pseudo-inverses of the Gaussian distribution. Each individual component of the GMM ($\mathcal{N}(\dot{\xi}; \tilde{\mu}^k, \tilde{\Sigma}^k)$) is replaced by its pseudo-inverse, the donut distribution ($\mathcal{D}(\dot{\xi}; \tilde{\mu}^k, \tilde{\Sigma}^k, \epsilon)$), resulting in a Donut Mixture Model (DMM). The additional exploration parameter ($\epsilon \in [0, 1]$) allows us to generate a spectrum of distributions whose peaks smoothly move from that of the underlying base distribution to a configurable maximum distance away, as seen in Figure 3. Further details about $\mathcal{D}$ are in the appendix.

We use the overall variance of the conditional (Equation 7) to set exploration: $\epsilon = 1 - \frac{1}{1+||V[\dot{\xi}|\theta,\xi]||}$. Our reasoning is that if the variance of the conditional is low, then the multiple demonstrations are in agreement as to what velocity should be associated with the current state. In this situation, it makes sense to do what the model predicts. However, if the variance is high, the demonstrations do *not* agree, and it would therefore be sensible to try something new.

Our actual desired velocity is the most likely velocity:

$$\dot{\xi}^{\texttt{DNT}} = \text{argmax}_{\dot{\xi}} \sum_{k=1}^{K} \tilde{\rho}^k \mathcal{D}(\dot{\xi}; \tilde{\mu}^k, \tilde{\Sigma}^k, \epsilon) \tag{12}$$

where each of the conditional Gaussian components has been replaced by its pseudo-inverse. However, as there is no closed form for the optima of a GMM we use gradient ascent to find a *local* maximum in the area around an initial guess, $\dot{\xi}'$. For a new trajectory we initialize $\dot{\xi}'_1 = \dot{\xi}_1^{\texttt{MEAN}}$ and take $\dot{\xi}'_{t+1} = \dot{\xi}_t^{\texttt{DNT}}$ for the other timesteps.

IV. Experimental Setup

We test the above approaches on two tasks that are difficult for human demonstrators to perform. The accompanying video shows the tasks being demonstrated and the results of learning with Donut MAP. Our robot is the Barrett WAM, and we collect demonstrations kinesthetically, by placing the used joint in gravity-compensation mode and physically guiding it in attempts to perform the task while recording joint angles (ξ) at 500Hz. Velocities $\dot{\xi}$ are computed as the single-step difference between samples ($\dot{\xi}_t = \xi_{t+1} - \xi_t$).

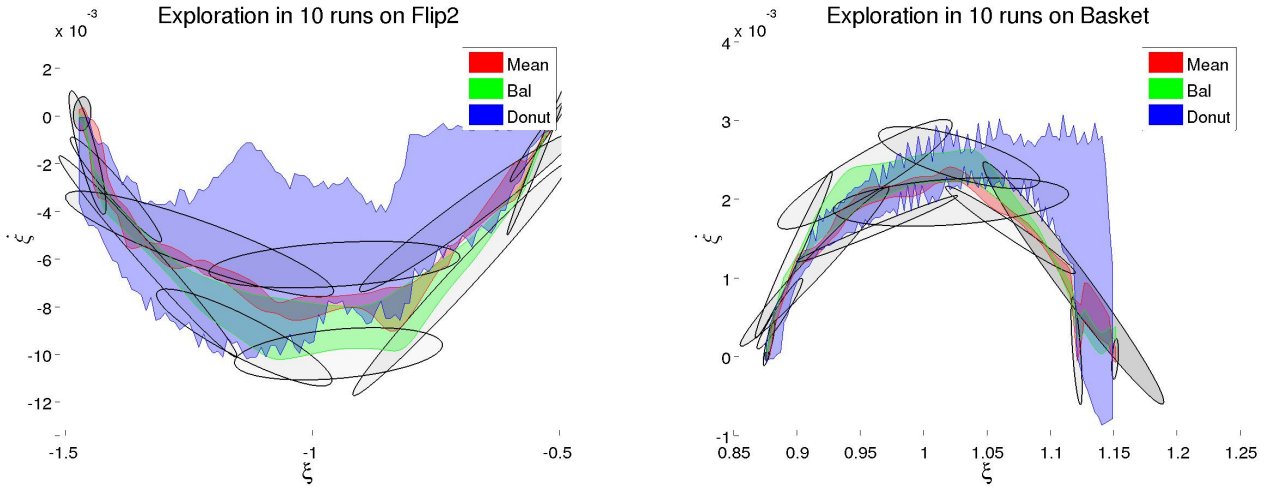
While the tasks themselves are fairly simple (1DOF) and performable by human demonstrators, it often takes them a few tries to get it right. In standard RLfD, these failures would be ignored, and only the successful performances used. We instead learn only from these failures. We note that all of our demonstrations are *complete*, in that task failure does not lead to early termination of the attempt.

To evaluate our techniques we are concerned not only with whether or not the task is eventually performed successfully (which it is), but also with the breadth of possibilities that are generated. That is, as continued failure is observed, we want to generate trajectories that diverge more from the demonstrations, exploring where no human had gone before, while at the same time reproducing the parts of the task that the different demonstrators agree on.

A. Task 1: Flip Up

Our first task, illustrated in Figure 4a, is to get a square foam block to stand on end. The block is set at the edge of a table, with a protruding side, but not fixed to the table in any way. Using the robot's wrist, the end effector comes from below and makes contact with the exposed portion. The setup is such that the block cannot be lifted to a standing position while in contact with the robot. Instead, there must be a 'flight' phase, where the block continues to move beyond the point in time when the robot ceases contact. Thus, the robot must impart momentum to the block. However, too much momentum and the block will topple over.

We collect 2 demonstrations of this task. In the first, too little momentum is transferred, and the block falls back to the initial position. In the second, too much momentum is imparted, and the block topples the other way. The resulting initial GMM is shown in state-velocity space in Figure 5a. We see that both demonstrations have the same basic shape, but differ in their maximum velocity and their timing. Further, they agree on starting and ending positions of the task. These agreements should be reproduced in the trial trajectories, while areas of disagreement are more explored.



(a) FlipUp

(b) Basket

Fig. 5: Illustrations of the initial GMMs and the space of trajectories explored for each task by the techniques.

B. Task 2: Basket Ball

The second task we consider also depends on accurate velocity control. Our basketball setup, shown in Figure 4b, has the robot launching a small ball with a catapult, with the goal of having the ball land in a basket attached to a wall opposite. Our initial position has the robot's end effector already touching the catapult, so all necessary force must be built up relatively quickly.

We again collect two demonstrations, one from each class in the Balance Mean approach. The first causes the ball to rebound off of the wall above the basket, and the second below. The initial model is in Figure 5b.

V. Discussion

As expected, the standard mean and MAP techniques rapidly converge to unsuccessful policies. While some exploration takes place (due to changes in the underlying models), the generated trajectories are very limited and cover a small portion of the available state-velocity space. Likewise, the minimum technique leads to issues, generating velocities that are not physically safe for the robot.

In terms of finding a successful policy, both the Donut MAP (DNT) and Balanced Mean (BAL) techniques converge within 10 iterations. However, the exploration exhibited by each algorithm is distinctly different. To illustrate the breadth of the search from each technique, we show the spread of 10 generated trajectories from each algorithm for each task in Figure 5. For this illustration, we have assigned all generated trajectories to the same class (+). For comparison, we show the area explored by the standard mean.

We immediately see that of the three, the donut technique covers the widest area, by an order of magnitude. While both mean-based approaches are limited to generating trajectories inside the span of the demonstrations, the donut is not, which will allow it to succeed if all demonstrations are in one class (e.g, too low). However, this advantage has a downside. Because there are more possibilities to explore, in our experiments the donut method took more iterations to succeed than the balanced mean.

Further, as expected, exploration with both techniques increased in the middle portions of both tasks, where the demonstrations disagreed the most. At the beginning (and to a lesser extent the end), the generated trajectories more closely resemble the humans'. This behavior is more visible in Figure 5a, which has more variance. We believe the decreased agreement at the end of the movement comes from accumulated drift during trajectory generation.

What these plots do not show is the order in which trajectories are generated. For the BAL technique, the initial trajectory is at the midpoint between the two demonstrations. Successive trajectories then approach the negative class incrementally. Donut, on the other hand, is much more erratic in its exploration: The technique will generate a few trajectories on one side of the data (slower than all demonstrations), and then jump to exploring in between the demonstrations, and then jump again to being faster.

We believe this behavior (and some of the visual jagginess) arises from our use of gradient ascent in the velocity generation and our initialization. Since we are only finding a local maximum, it may be that the generated velocity is actually relatively unlikely. However, it will keep being selected until the model has shifted enough to remove the local optimality. Further, as we initialize with the mean at $t = 1$, we will always start at the local maxima nearest to it, which may unnecessarily curtail our exploration.

VI. Future Work

We are examining ways to alleviate these issues and improve Donut's performance. One approach is to use sampling in an attempt to find the global maxima instead of a local one. However, each additional sample would require its own gradient ascent, which is computationally costly. Further, the global maxima may shift greatly from one timestep to the next, generating potentially unsafe velocities and torques.

We have also considered introducing a forgetting factor into our GMM update. Currently, we resample only to speed up the estimation of the updated parameters, and weigh our samples to represent the total number of datapoints. We could instead force the old data to have the same weight as the newly generated trajectory (or some percentage of the old weight), which may speed exploration. However, there is then the worry that the original demonstrations will be lost.

In terms of the BAL approach, we currently hand-assign trajectories to one of the two classes. For our tasks, this is an acceptable method. However, as the behaviors become more complex and high-dimensional, it may no longer be. One possibility would be to first use unsupervised clustering to automatically divide the data into (possibly more than two) classes. Additionally, making the mixing parameter (α) dependent on the current state may enable us to explore more heavily when the demonstrations disagree more, in much the same way the donut does now.

Looking at our work through the lens of reinforcement learning, you can think of us as currently using a binary reward signal: Success or failure, and we assume all of our demonstrations fail. Rather than exploring randomly in the space of possible trajectories, we generate guesses based on the intuition that failure is likely due to the parts of the demonstrations that differ. However, it is still the case that some failures are worse than others. Using a more continuous reward may allow us to better leverage the available information and converge faster, while learning from both failed and successful examples. One approach we are investigating is to weigh the trajectories by the reward when building our model. Alternatively, we could embed the donut exploration directly into a standard RL technique.

VII. Conclusion

Current work in Robot Learning from Demonstration uses ideas from Reinforcement Learning to deal with suboptimal, noisy demonstrations and to improve the robot's performance beyond that of the human. However, an underlying assumption is that the human has successfully completed the desired task. We instead assume the negation, that the humans have failed, and use their demonstrations as a negative constraint on exploration. We have proposed two techniques for generating tentative trajectories and shown that they converge to successful performance for two robot tasks.

Acknowledgements

This work was supported in part by the European Commission under contract numbers FP7-248258 (First-MM) and FP7-ICT-248311 (Amarsi). The authors thank Florent D'Halluin and Christian Daniel for their assistance.

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, May 2009.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot Programming by Demonstration," in *Handbook of Robotics*. MIT Press, 2008, vol. chapter 59.

[3] E. A. Billing and T. Hellström, "A formalism for learning from demonstration," *Paladyn*, vol. 1, no. 1, pp. 1–13, 2010.

[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, pp. 838–850, 1995.

[5] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Neural Information Processing Systems*, Vancouver, Dec. 2008.

[6] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *International Conference on Intelligent Robots and Systems*, Nice, France, Sept. 2008, pp. 1083–1090.

[7] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artifical Intelligence Research*, vol. 34, no. 1, pp. 1–25, Jan. 2009.

[8] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 2483 – 2488.

[9] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *International Conference on Human-Robot Interaction*, Arlington, VA, Mar. 2007, pp. 57–64.

[10] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kineshetic demonstrations," *IEEE Transactions on Robotics*, pp. 1463–1467, 2008.

[11] H. G. Sung, "Gaussian mixture regression and classification," Ph.D. dissertation, Rice, 2004.

[12] R. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*. Kluwer Academic Publishers, 1998, pp. 355–368.

[13] X. Hu and L. Xu, "Investigation on several model selection criteria for determining the number of cluster," *Neural Information Processing. - Letters and Reviews*, vol. 4, no. 1, pp. 1–10, July 2004.

Appendix

The Donut distribution is a pseudo-inverse of the base normal distribution $\mathcal{N}(\dot{\xi};\tilde{\mu},\tilde{\Sigma})$. It is defined as:

$$\mathcal{D}(\dot{\xi};\tilde{\mu},\tilde{\Sigma},\epsilon) = 2\mathcal{N}(\dot{\xi};\tilde{\mu},\frac{1}{{r_\alpha}^{\frac{2}{D}}}\tilde{\Sigma}) - \mathcal{N}(\dot{\xi};\tilde{\mu},\frac{1}{{r_\beta}^{\frac{2}{D}}}\tilde{\Sigma}) \quad (13)$$

where the component distributions' means are the same as the base distribution's. Their covariances are defined by scalar ratios r_α and r_β which are themselves determined from the desired exploration, $\epsilon \in [0,1]$ and a maximum width (peak-to-mean) λ^*. Equation 13 integrates to one, and if $r_\alpha < r_\beta$ is everywhere positive.

When talking about the Donut distribution, we define the height (η) as the ratio between the Donut's height at the mean and that of the base distribution and the width (λ) as the ratio between the peak-to-mean distance and the standard deviation of the base distribution.

The Donut distribution approximates the base distribution ($\eta = 1, \lambda = 0$) when

$${r_\alpha}^b = \frac{\sqrt[3]{0.5}}{2(\sqrt[3]{0.5}-1)+1} \quad (14)$$

$${r_\beta}^b = 2{r_\alpha}^b - 1 \quad (15)$$

and achieves maximum width ($\lambda = \lambda^*, \eta = 0$) at

$${r_\beta}^* = \frac{2}{\lambda^*}\sqrt{\frac{\log[0.5]}{0.5^2-1}} \quad (16)$$

$${r_\alpha}^* = {r_\beta}^*/2 \quad (17)$$

We interpolate between these two points based on ϵ:

$$r_\alpha = (1-\epsilon)({r_\alpha}^b - {r_\alpha}^*) + {r_\alpha}^* \quad (18)$$

$$r_\beta = (1-\epsilon)({r_\beta}^b - {r_\beta}^*) + {r_\beta}^* \quad (19)$$

and set $\lambda^* = 6$, giving the distributions seen in Fig. 3.