Int J Soc Robot (2012) 4:331–342
DOI 10.1007/s12369-012-0161-z

Robot Learning from Failed Demonstrations

Daniel H. Grollman · Aude G. Billard

Accepted: 13 June 2012 / Published online: 30 June 2012

Abstract Robot Learning from Demonstration (RLfD) seeks to enable lay users to encode desired robot behaviors as autonomous controllers. Current work uses a human's demonstration of the target task to initialize the robot's policy, and then improves its performance either through practice (with a known reward function), or additional human interaction. In this article, we focus on the initialization step and consider what can be learned when the humans do not provide successful examples. We develop probabilistic approaches that avoid reproducing observed failures while leveraging the variance across multiple attempts to drive exploration. Our experiments indicate that failure data do contain information that can be used to discover successful means to accomplish tasks. However, in higher dimensions, additional information from the user will most likely be necessary to enable efficient failure-based learning.

Keywords Robot Learning from Demonstration · Learning from Failure

1 Motivation

The standard Robot Learning from Demonstration (RLfD) scenario has an end-user who wants to adapt a robot to perform a new task, perform an old task in a new way, or operate in a new environment. Rather than hiring a roboticist to perform multiple rounds of analysis, modeling, programming, debugging and testing, RLfD aims to let one simply *demonstrate* the task (perhaps several times) in order to teach it to the robot. It can be similarly adjusted later if the user's need or situation change [2, 3].

This approach is well suited for tasks that humans can easily perform, but would rather not. Ideally, data collection is trivial: The robot watches a human as he or she does the task normally. Eventually, the robot learns the task and takes over, and the human then attends to other matters. Research has developed many methods for deriving autonomous controllers from observations of human performance, particularly focused on determining acceptable variance in execution and generalization over initial conditions and perturbations [7, 9].

However, the real world falls short of the ideal, and often passive observations are not enough for robot education. Instead, additional information is needed from the teacher, such as further demonstrations focused on correcting robot errors or specific modifications to the learned model. RLfD can then become a more interactive paradigm, sometimes called *tutelage*, where the robot observes, learns, performs, and gets feedback from the human to improve itself. Research has focused on making this process as intuitive for the human as possible, likening teaching the robot to the way one would teach a child [4, 11, 24].

Autonomous practice is another way in which robots can improve their performance. Using a known reward function, a robot can score itself and modify its behavior accordingly [5]. A benefit to this approach is that the human need not observe all of the robot's attempts. Downsides include the fact that the human must first explicitly write down the reward function (which may be non-trivial), and that the robot's repeated attempts may take more time and cause damage to the robot or the environment (if not performed in simulation). Recent work in Inverse Reinforcement Learning [22]

D.H. Grollman (✉) · A.G. Billard
Learning Algorithms and Systems Laboratory, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
e-mail: daniel.grollman@epfl.ch

A.G. Billard
e-mail: aude.billard@epfl.ch

addresses this first issue by attempting to estimate the reward function from observed task performance.

In all approaches where the robot improves its performance, an added advantage is that the robot can eventually learn to perform the task *better* than in the human's demonstrations. Often, techniques are compared based on the quality of the final controller, the amount of time spent learning, and possibly the amount of time spent teaching. However, there is a hidden cost not generally reported: the amount of time it takes the human to master the task themselves.

Almost all current RLfD approaches start with a successful (perhaps suboptimal) human demonstration of the task. For relatively simple tasks, such as pick-and-place [17], point to point motion [12], or washing a surface [8], such demonstrations are easily obtainable from nearly any human teacher with minimal overhead and can immediately be used for training. However, for more complicated tasks such as acrobatic helicopter flight [1], ball-in-cup [15], or unicycle riding [6], successful demonstrations are harder to come by. Indeed, often experimenters must either compensate trained experts, or learn the skills themselves, discarding data from failed attempts. In both cases, the often unreported expense (in time or money) to collect the demonstrations should be taken into account when evaluating the entire system.

From an end-user perspective, the requirement of successful demonstrations means that a user who wishes a robot to perform a task they themselves cannot do must either pay someone who can do it to teach their robot (similar to paying a programmer), or first learn the task themselves. However, if a robot could learn from unsuccessful (but non-catastrophic) *attempts* at performing the task, the user may still be able to teach it, using whatever limited skills they already possess.

Previously, failure information has been used mainly as a means to adjust a robot policy after it has been learnt [19, 21]. However, it is known that humans are capable of learning to perform tasks after only observing failed demonstrations [18, 25]. In this article we develop and examine RLfD approaches in an attempt to replicate that ability in a robot, based on the idea that failed demonstrations have educational worth in three respects: Firstly, they are examples of *what not to do*, so replication should be avoided. Secondly, they are indicative of what the human thinks a successful performance should be, so new attempts should explore around them. And thirdly, that multiple attempts indicate an appropriate breadth of exploration. From this point of view we attempt to perform Learning from Failure (LfF).

In doing so, we expect to see tradeoffs between the quality of the final controller, the skill level of the demonstrator, the number of demonstrations and the time spent learning. Similar to doing-it-yourself, a user would have to make their own decision if such tradeoffs are acceptable, or if they would rather pay a professional. What we attempt in this article is to lay the groundwork for providing them the tools with which users can do it themselves, if they choose. Additionally, as the tasks that we teach our robots become more complex, failed demonstrations may become more common, and these approaches may be utilized to better leverage all of the available data, rather than letting it be discarded.

Portions of this research were previously presented in [10]. Here we provide additional details in Sect. 3, and compare with reward-based learning in Sect. 4.3. Sections 6 and 7 contain ideas and experiments in extending the work to higher dimensions, and Sect. 8 concludes with future directions for LfF.

2 Robot Controller

We follow our previous work in RLfD and model robot controllers as autonomous dynamical systems (ADS) [12]. In particular, we treat the relationship between the current D-dimensional real-valued robot state (joint angles), ξ, and their velocities, $\dot{\xi}$, as a nonlinear function, $\dot{\xi} = f_\theta(\xi)$. The function itself is represented with a Gaussian mixture model (GMM) [23] in joint state-velocity space, with the probability of a given state-velocity pair

$$P_{\text{GMM}}(\xi, \dot{\xi}|\theta) = \sum_{k=1}^{K} \rho^k \mathcal{N}\left(\xi, \dot{\xi}|\mu^k, \Sigma^k\right) \tag{1}$$

with $\mathcal{N}$ as the standard normal distribution and collected parameters $\theta = \{K, \{\rho^k, \mu^k, \Sigma^k\}_{k=1}^K\}$. These are the number of components (positive integer) and the priors (positive real, $\sum_{k=1}^K \rho^k = 1$), means ($2D$ real vector) and covariances ($2D \times 2D$ positive semi-definite matrix) of each component.

This system is autonomous in that it is independent of time. Instead, the velocity of the system depends only upon the current state of the system, and the dynamics are defined over the entire state space. Because of these features, ADS controllers are robust to temporal and spatial perturbations, making them well suited for noisy, dynamic tasks.

To compute $\dot{\xi} = f_\theta(\xi)$ we first condition the GMM on the state to get a conditional distribution over velocities.

$$\begin{aligned}
P_{\text{GMM}}(\dot{\xi}|\xi, \theta) &= \sum_{k=1}^{K} \tilde{\rho}^k(\xi, \theta) \mathcal{N}\left(\dot{\xi}|\tilde{\mu}^k(\xi, \theta), \tilde{\Sigma}^k(\theta)\right) \\
\tilde{\mu}^k(\xi, \theta) &= \mu_{\dot{\xi}}^k + \Sigma_{\dot{\xi}\xi}^k \Sigma_{\xi\xi}^{k^{-1}} \left(\xi - \mu_\xi^k\right) \\
\tilde{\Sigma}^k(\theta) &= \Sigma_{\dot{\xi}\dot{\xi}}^k - \Sigma_{\dot{\xi}\xi}^k \Sigma_{\xi\xi}^{k^{-1}} \Sigma_{\xi\dot{\xi}}^k \\
\tilde{\rho}^k(\xi, \theta) &= \frac{\rho^k \mathcal{N}(\xi|\mu_\xi^k, \Sigma_{\xi\xi}^k)}{\sum_{k=1}^K \rho^k \mathcal{N}(\xi|\mu_\xi^k, \Sigma_{\xi\xi}^k)}
\end{aligned} \tag{2}$$

The result is itself a GMM (with derived parameters indicated by tildes) and can be used to generate ($\dot{\xi}$) either probabilistically (i.e., by sampling) or deterministically (i.e., by expectation). Note that $\tilde{\Sigma}^k$ does not depend on the current state. For clarity we drop the functional forms of the conditional parameters and write $\tilde{\mu}^k$ for $\tilde{\mu}^k(\xi, \theta)$, etc.

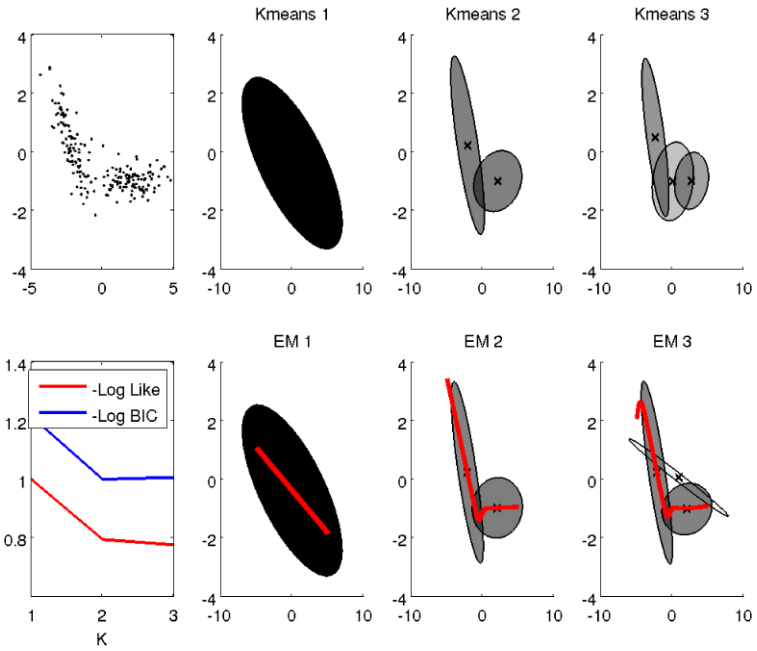


Fig. 1 Human demonstrations of state-velocity pairs are modeled as a GMM. Raw data (*top left*) is clustered via weighted K-means (*top*) and the parameters are tuned with Expectation Maximization (*bottom*). The appropriate value of K (2) is chosen by minimizing the BIC (*bottom left*). The resulting model can be used to generate smooth trajectories (*red lines*) by using the expectation of the conditional. When the training data is from successful demonstrations, this approach can reproduce the desired task

2.1 Parameter Fitting

To fit the parameters of the GMM to data, we use a weighted version of Expectation-Maximization (EM) [20]. Our data is collected state-velocity pairs from human demonstration attempts. Each of S attempts gives us a trajectory, $\tau_s = \{\xi_t, \dot{\xi}_t\}_{t=1}^{T_s}$, which we collect into a single dataset $\Xi = \{\tau_s\}_{s=1}^{S} = \{\xi_n, \dot{\xi}_n\}_{n=1}^{N}$ consisting of $N = \sum_{s=1}^{S} T_s$ points. With each point we associate a weight w_n.

For a given value of K, we initialize the μ^k randomly and use weighted K-means to seed the EM process. In weighted K-means, points are iteratively assigned to the nearest μ, and then the μs are updated to the weighted mean of all points assigned to them. These two steps alternate until no assignments need to be changed. If during the process a μ has no points assigned to it, it is re-initialized randomly.

From our K-means clusters, we initialize Σ^k as the covariance matrix of the datapoints in each cluster, and ρ^k as the number of points in each cluster, normalized by the total number of points. EM takes these initial values and iteratively adjusts them to maximize the likelihood $\mathscr{L}(\Xi|\theta)$.

In general, increasing K improves the fit. To avoid overfitting we use the Bayesian Information Criterion (BIC) [13]. We run EM for multiple K and compute

$$BIC(K) = -2\ln\big(\mathscr{L}(\Xi|\theta)\big) + K\left(1 + D + \frac{D(1+D)}{2}\right)\ln(N) \tag{3}$$

which penalizes the log-likelihood of the fit model based on the number of free parameters (K and D dependent) that must be fit. Over multiple random initializations we choose the K with the minimum value. The full parameter fitting process is illustrated in Fig. 1.

2.2 Learning from Success

When initialized with successful demonstrations, the learned GMM can deterministically generate $f_\theta(\xi)$ by taking the expectation of the conditional distribution in Eq. (2)

$$f_\theta^{\text{success}}(\xi) = \tilde{E}[\dot{\xi}|\xi,\theta] = \sum_{k=1}^{K} \tilde{\rho}^k \tilde{\mu}^k \tag{4}$$

Doing so makes the assumption that all of the observed data are initially correct, but corrupted by Gaussian noise. Alternatively, we could sample from the conditional distribution. However, when used to control a robot, the random samples may lead to large accelerations between timesteps and rapid oscillations in the robot's velocity. Using the expected value instead guarantees a smooth motion as shown in Fig. 1.

3 Learning from Failure

Using the expected value of the conditional only makes sense when observed data are evenly distributed around success, which is a very strong assumption in the case of failed demonstrations. We thus propose an approach based on a novel distribution, which we develop with three aims in mind, connected to the three ways in which failure data can be useful:

1. The probability of performing the same action as the human demonstration is reduced.

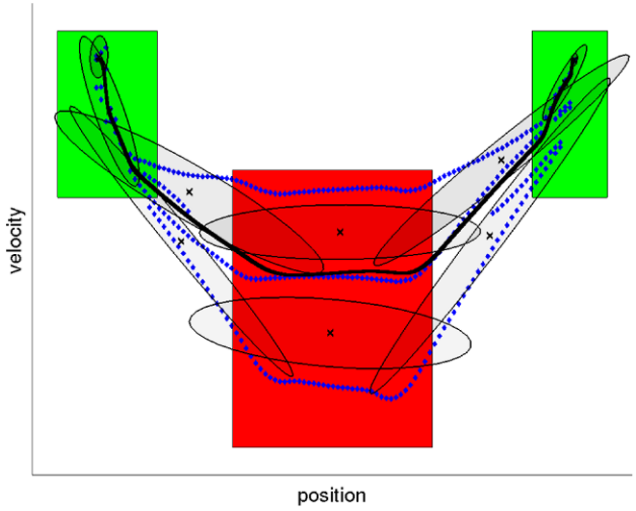


Fig. 2 When fit to failure data, the mean (*solid*) may no longer be an appropriate response. Instead, we aim to generate exploratory trajectories (*dotted*) that utilize variance in human demonstrations to mimic the human in areas of high confidence (*green*) and explore in areas of low confidence (*red*). *Dots* are values actually generated by our system

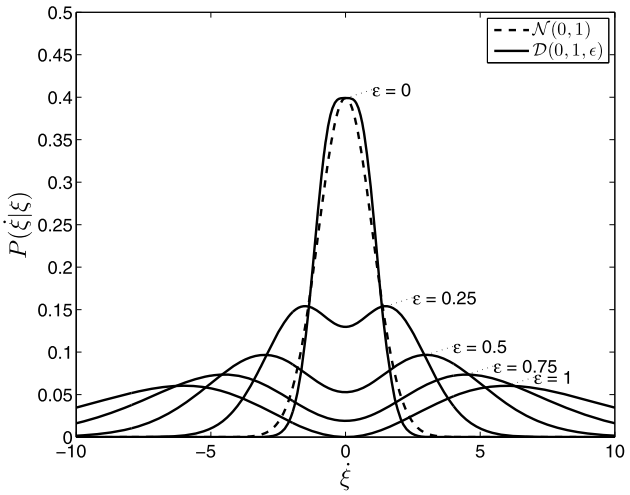


Fig. 3 The Donut distribution, a center-off distribution with variable width. Shown are the family of distributions generated by the simplified notation in Sect. 3.2.3

2. Areas around the human demonstration should have increased probability.
3. The span of exploration should be related to the variance in human demonstration.

Consider the failed demonstrations in Fig. 2. Rather than only producing the mean trajectory (solid line), we wish to also generate exploratory trajectories (dotted lines). Note that in areas of high demonstrated variance (red) we generate velocities that are further away from the observed data, and in areas of low variance (green) the velocities are closer to the human's demonstrations.

3.1 Donut Distribution

To generate our desired trajectories we introduce the Donut distribution, the center-off distribution with a variable width shown in Fig. 3. Our general approach will replace each component of the conditional distribution in Eq. (2) with a Donut, and use the most likely velocity for execution.

The Donut distribution is a difference of 2 Gaussians

$$\mathscr{D}(\mathbf{x}|\mu_\alpha, \mu_\beta, \Sigma_\alpha, \Sigma_\beta, \gamma) = \gamma\mathcal{N}(\mathbf{x}|\mu_\alpha, \Sigma_\alpha) - (\gamma-1)\mathcal{N}(\mathbf{x}|\mu_\beta, \Sigma_\beta) \quad (5)$$

where $\gamma > 1$ and we know that the priors must sum to 1.

Our aim is to have a distribution that smoothly moves from maximally to minimally likely at the center. In order to reach 0 at the center, while remaining positive everywhere else, we set $\mu_\alpha = \mu_\beta$. We further base this distribution on and compare to a standard Normal distribution $\mathcal{N}(\mu, \Sigma)$, so we take all of the μs to be the same. Likewise, we re-parameterize the donut distribution in terms of the scalar ratios of the variance of this base distribution to that of the donut's two components, r_α and r_β, such that $\Sigma_\alpha = \frac{1}{r_\alpha^2}\Sigma$, $\Sigma_\beta = \frac{1}{r_\beta^2}\Sigma$. This parameterization keeps the shape of the covariance constant, as desired.

$$\mathscr{D}(\mathbf{x}|\mu, \Sigma, r_\alpha, r_\beta, \gamma) = \gamma\mathcal{N}\left(\mathbf{x}|\mu, \Sigma/r_\alpha^2\right) - (\gamma-1)\mathcal{N}\left(\mathbf{x}|\mu, \Sigma/r_\beta^2\right) \quad (6)$$

Example donut distributions generated by the parameterization we will develop in comparison with a base distribution are shown in Fig. 3.

3.1.1 Height

Of interest is the height of this distribution at the mean, the likelihood of reproducing the demonstrations. Setting $\mathbf{x} = \mu$

$$\begin{aligned}
&\mathscr{D}(\mu|\mu, \Sigma, r_\alpha, r_\beta, \gamma) \\
&= \gamma\mathcal{N}\left(\mu|\mu, \frac{\Sigma}{r_\alpha^2}\right) - (\gamma-1)\mathcal{N}\left(\mu|\mu, \frac{\Sigma}{r_\beta^2}\right) \\
&= \frac{\gamma}{\sqrt{(2\pi)^D|\Sigma/r_\alpha^2|}} - \frac{\gamma-1}{\sqrt{(2\pi)^D|\Sigma/r_\beta^2|}} \\
&= \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}}\left[\gamma r_\alpha^D - (\gamma-1)r_\beta^D\right]
\end{aligned}$$

We recognize in the coefficient the height of the base distribution at $\mathbf{x} = \mu$ and so state the ratio of the height of the donut distribution to that of the base distribution as

$$\eta = \frac{\mathscr{D}(\mu|\mu, \Sigma, r_\alpha, r_\beta, \gamma)}{\mathcal{N}(\mu|\mu, \Sigma)} = \gamma r_\alpha^D - (\gamma-1)r_\beta^D \quad (7)$$

3.1.2 Width

We are also interested in the location of the maximum of the donut distribution with respect to that of the base distribution (the mean). This measurement is the radius of a hypersphere centered at the mean, which we relate to the standard deviation of the base distribution and call the width, corresponding to the area of exploration. To compute its value, we first need the gradient of the donut distribution:

$$\begin{aligned}&\nabla_x \mathscr{D}(\mathbf{x}|\mu, \Sigma_\alpha, \Sigma_\beta, \gamma)\\&\quad = -\gamma \mathscr{N}(\mathbf{x}|\mu, \Sigma_\alpha)\Sigma_\alpha^{-1}(\mathbf{x}-\mu)\\&\qquad + (\gamma-1)\mathscr{N}(\mathbf{x}|\mu, \Sigma_\beta)\Sigma_\beta^{-1}(\mathbf{x}-\mu)\end{aligned} \tag{8}$$

We solve for 0 to obtain:

$$-\frac{2\log[\frac{\gamma}{\gamma-1}(\frac{r_\alpha}{r_\beta})^{D+2}]}{(r_\beta^2 - r_\alpha^2)} = (\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu) \tag{9}$$

Without loss of generality, we can assume $\mu = 0$ and $\Sigma = \mathbf{I}$. The width λ is the absolute value of the offset from the mean proportional to norm of the variance, equal to the square root of the left-hand side:

$$\lambda^2 = \frac{2\log[\frac{\gamma}{\gamma-1}(\frac{r_\alpha}{r_\beta})^{D+2}]}{(r_\alpha^2 - r_\beta^2)} \tag{10}$$

3.2 Limits

Using the notions of height and width, we can more simply state our desired behavior. When imitating success, we want a distribution that is high and narrow, much like the standard Gaussian. However, when avoiding failure, we would want one that is low and wide. We can smoothly transition between these extremes to represent different levels of confidence in the fact that the mean is indeed a failure.

We must determine ways of setting r_α and r_β to achieve this behavior. Additionally, we must ensure that we generate a valid distribution. In other words, that it is everywhere positive and that the width is real. The fact that the distribution integrates to one is in the definition, in that $\gamma - (\gamma - 1) = 1$.

3.2.1 Positive

Without loss of generality we assume that $\mu = 0$, and to ensure that $\mathscr{D}$ is everywhere positive it must be that

$$\begin{aligned}&\gamma \mathscr{N}\big(\mathbf{x}|0, \Sigma/r_\alpha^2\big) \geq (\gamma-1)\mathscr{N}\big(\mathbf{x}|0, \Sigma/r_\beta^2\big)\\&\frac{\gamma}{\sqrt{|\Sigma_\alpha|}}\exp\big(-0.5\mathbf{x}^\top \Sigma_\alpha^{-1}\mathbf{x}\big) \geq \frac{\gamma-1}{\sqrt{|\Sigma_\beta|}}\exp\big(-0.5\mathbf{x}^\top \Sigma_\beta^{-1}\mathbf{x}\big)\\&\exp\big(-0.5\mathbf{x}^\top\big(\Sigma_\alpha^{-1} - \Sigma_\beta^{-1}\big)\mathbf{x}\big) \geq \frac{(\gamma-1)\sqrt{|\Sigma_\alpha|}}{\gamma\sqrt{|\Sigma_\beta|}}\\&-0.5\mathbf{x}^\top \Sigma^{-1}\mathbf{x}\big(r_\alpha^2 - r_\beta^2\big) \geq \log\left(\frac{\gamma-1}{\gamma}\right) + \log\left(\sqrt{\frac{|\Sigma_\alpha|}{|\Sigma_\beta|}}\right)\end{aligned}$$

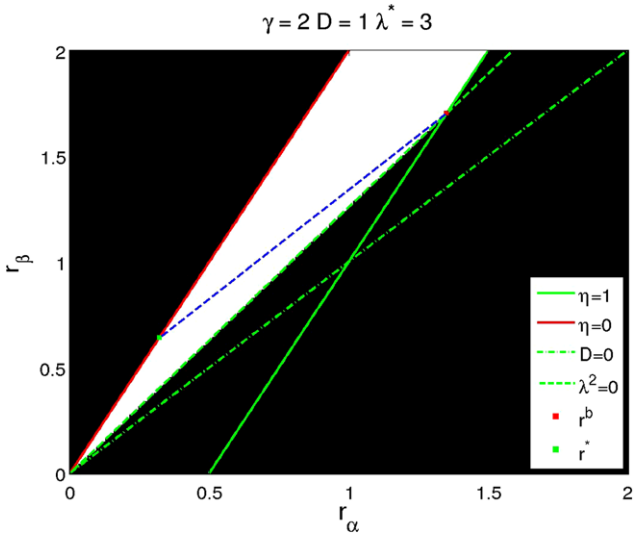


Fig. 4 A slice through $r_\alpha, r_\beta, \gamma$ space at $\gamma = 2$ showing which combinations of r_α and r_β produce valid donut distributions for dimensionality $D = 1$. Also shown are the location where we most closely approximate the base distribution r^b, and where we obtain the predetermined ($\lambda^* = 3$) maximum exploration (r^*), as well as the exploration line between them

Since $\mathbf{x}^\top \Sigma^{-1}\mathbf{x}$ is always positive, we require that $(r_\alpha^2 - r_\beta^2)$ is always negative, so that the left side is always positive and thus has a lower bound. Therefore we require that $r_\alpha < r_\beta$. This bound, indicated by the dash-dot green line in Fig. 4, is a necessary, but not generally sufficient condition, but sufficient for our needs when combined with the following.

3.2.2 Real

We do not allow λ to be imaginary, so we can constrain:

$$\begin{aligned}&\frac{2\log[(\frac{r_\alpha}{r_\beta})^{D+2}\frac{\gamma}{\gamma-1}]}{(r_\alpha^2 - r_\beta^2)} \geq 0\\&2\log\left[\left(\frac{r_\alpha}{r_\beta}\right)^{D+2}\frac{\gamma}{\gamma-1}\right] \leq 0\\&\left(\frac{r_\alpha}{r_\beta}\right)^{D+2}\frac{\gamma}{\gamma-1} \leq 1\\&\frac{r_\alpha}{r_\beta} \leq \sqrt[D+2]{\frac{\gamma-1}{\gamma}}\end{aligned}$$

Where we have used the fact that $(r_\alpha^2 - r_\beta^2) < 0$. This limit, indicated by the dashed green line in Fig. 4, supersedes the previous result.

3.2.3 Exploration

We further constrain $0 \leq \eta \leq 1$ (red and green solid lines) and show the space of valid settings of r_α and r_β by the white area in Fig. 4. To simplify our parameterization and aid in selecting these scalar values, we introduce an explo-

ration parameter, ε to control the behavior of the donut distribution, and make the covariance coefficients functions of it. Namely, when $\varepsilon = 0$, the distribution should most closely resemble the original normal distribution, corresponding to behaving in the standard learning-from-success fashion. This behavior is obtained by setting $\eta = 1, \lambda = 0$ and deriving

$$r_\beta(\varepsilon = 0) = \left[\gamma\left(\frac{\gamma - 1}{\gamma}\right)^{\frac{D}{D+2}} - (\gamma - 1)\right]^{-1/D}, \tag{11}$$

$$r_\alpha(\varepsilon = 0) = r_\beta(0)\sqrt[D+2]{\frac{\gamma - 1}{\gamma}} \tag{12}$$

Likewise, when $\varepsilon = 1$ we wish to obtain maximum exploration, where the likelihood of reproducing the observations is minimized ($\eta = 0$). We use a hyper-parameter λ^* to set the maximum width and derive

$$r_\beta(\varepsilon = 1) = \sqrt{\frac{4\log[\frac{\gamma-1}{\gamma}]}{\lambda^{*2}([\frac{\gamma-1}{\gamma}]^{\frac{2}{D}} - 1)D}}, \tag{13}$$

$$r_\alpha(\varepsilon = 1) = r_\beta(1)\sqrt[D]{\frac{\gamma - 1}{\gamma}} \tag{14}$$

We can smoothly transition from one extreme to the other by computing the coefficients as a function of exploration as:

$$r_\circ(\varepsilon) = (1 - \varepsilon)\big(r_\circ(0) - r_\circ(1)\big) + r_\circ(1) \tag{15}$$

Giving rise to the blue-dashed line of exploration in Fig. 4, and the family of distributions in Fig. 3.

3.3 Donut Mixture Model

To perform learning from failure, we build a GMM from human demonstrations as usual, but instead of using the mean of the conditional as in Eq. (4), we find a maximum of the corresponding Donut Mixture Model (DMM):

$$P_{\text{DMM}}(\dot{\xi}|\xi) = \sum_{k=1}^{K} \tilde{\rho}^k \mathcal{D}\big(\dot{\xi}|\tilde{\mu}^k, \tilde{\Sigma}^k, \varepsilon\big) \tag{16}$$

We take $\gamma = 2$ as a constant, and the conditional means, covariances, and priors are computed as usual as in Eq. (2). For exploration, we set $\varepsilon = 1 - \frac{1}{1+\|\tilde{V}[\dot{\xi}|\xi,\theta]\|}$, where

$$\tilde{V}[\dot{\xi}|\xi,\theta] = -\tilde{E}[\dot{\xi}|\xi,\theta]\tilde{E}[\dot{\xi}|\xi,\theta]^\top + \sum_k \rho_k\big(\mu_k\mu_k^\top + \Sigma_k\big) \tag{17}$$

is the overall variance of the GMM. In doing so we connect the human demonstrator's own variance with the exploration of our system. ε then tends towards 1 in areas of high human variability, and 0 in areas of low variability, as desired.

We would like to generate the most likely velocity from this conditional distribution for use, but there is no analytical solution for the maxima of the DMM. Instead, we use gradient ascent, where the gradient of the entire DMM is equal to the weighted sum of the gradients of each individual component, as given by Eq. (8). As gradient ascent is only guaranteed to find a local maximum, there is some danger of being caught at a suboptimal value. In practice, we initialize the first gradient ascent step of every trajectory randomly from the overall distribution, and each successive step with the previously found maximum. To illustrate the approach, Fig. 2 shows all of the possible trajectories, determined via exhaustive search, that could be generated from the data.

3.3.1 *Parameter Update*

As we are learning from failure, there is no reason to expect that our first attempt will succeed. Thus, it becomes necessary to update the parameters of the DMM after each new trial. If the new trial is a failure like the human's demonstrations, the naive approach is to collect all of the data (demonstrations and trials) and re-estimate θ using EM. However, as the number of datapoints grows, the time necessary for EM does as well. We instead formulate a sparse update by making use of the weights in our weighted EM approach.

Given a θ derived from N datapoints, and a new trajectory τ' consisting of N' datapoints, we use a sample and merge approach to create a new θ'. First we sample N' points from our current model, and give them weight N/N'. We then add in the new data, with all points having weight 1. Rather than re-initializing with K-means, we start with θ and run EM from there to reach θ', holding K constant.

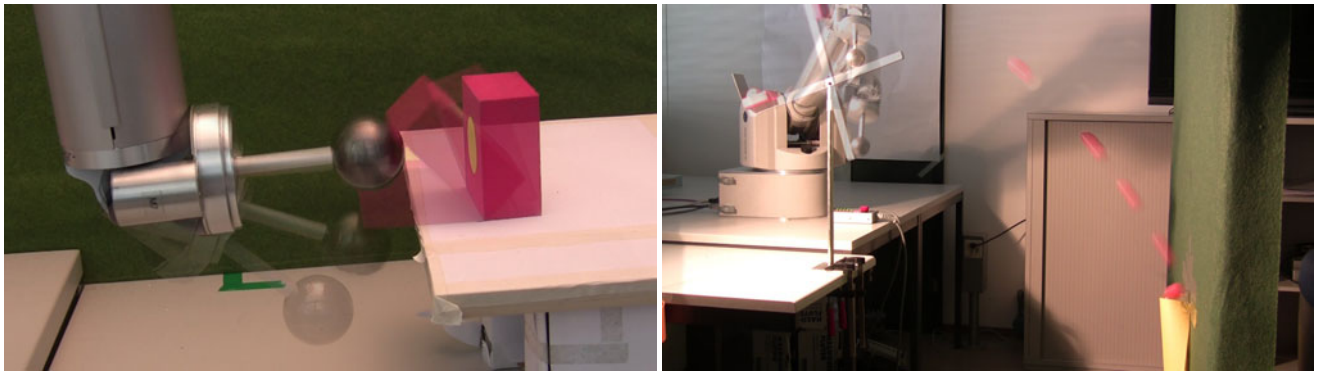
4 Experiments

To test if Learning from Failure is a viable approach, we here examine learning *solely* from failed demonstrations, future work may merge in techniques for learning from success as well. Our entire LfF framework is outlined in Algorithm 1, and all data in our experiments is collected informally, from members of our lab, using kinesthetic demonstration, where the robot is physically guided through to perform the task.

4.1 Tasks

We start with 2 one-DOF tasks. While success in 1D is by no means a guarantee that these techniques will scale up, failure to learn these tasks would be a strong indication that they certainly will not. Further, both of these tasks require accurate velocity control at certain points in time, so successful policies are unlikely to be discovered by random exploration. For each task we collect two initial failed attempts by a human for use as training data.

Our first task, illustrated in Fig. 5a, 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. The



(a) FlipUp (b) Basket

Fig. 5 Our robot tasks. (**a**) FlipUp: get the foam block to stand on end, (**b**) Basket: Launch the ball into the basket. Shown are successful trajectories learned with the DMM-ADS approach from 2 initial failed demonstrations

Algorithm 1 Our DMM LfF approach

```
Collect S human failed attempts
Build a GMM (θ) as described in Sect. 2.1.
while robot has not succeeded do {Robot trials}
  t = 0
  ξ_t = Current state of the system
  ξ̇_t ~ DMM(ξ̇|ξ, θ)
  while |ξ̇| ≠ 0 and not timeout do
    Maximize P(ξ̇_t|ξ_t) with gradient ascent
    apply ξ̇ to system
    t = t + 1
    ξ_t = Current state of the system {Nominally, ξ_{t−1} + ξ̇_{t−1}}
    ξ̇_t = ξ̇_{t−1} {Start gradient ascent here}
  end while
  update θ as in Sect. 3.3.1
end while
```

robot's end effector comes from below and makes contact with the exposed portion of the block, but the setup is such that the block cannot be lifted to a standing position while maintaining contact. Instead, there must be a 'flight' phase, so the robot must impart momentum to the block. However, too much momentum and the block will topple over.

The second task, in Fig. 5b, has the robot launching a small ball with a catapult. The goal is to get the ball to land in a basket attached to the wall opposite. The initial position has the robot's end effector already touching the catapult, so all necessary force must be built up relatively quickly.

4.2 Learning with no Reward

In the extreme, the robot has access only to the failed demonstrations, and no further information (such as comparisons between them). Additionally, after each unsuccessful robot trial, there is no scoring of the attempt. The robot then keeps making different attempts at the task, until it succeeds.

Using the DMM based ADS technique with parameter updating as described above, our system is able to learn successful policies for these two tasks. For the FlipUp task, we collect 10 different initial training sets, and our system averaged 4.2 trials to discover success. For Basket, over 3 different training sets it averaged 6.7 trials to success.

4.3 Learning with Reward

To compare our approach with current state-of-the-art techniques, we introduce a continuous reward function. Note that in practice, reward functions for user-desired task may be non-trivial to write down. Thus, being able to learn in the absence of one would be a useful skill for an RLfD system.

To use DMM-ADS with continuous reward, we leverage the weighted datapoint capabilities of EM. During initial parameter fitting, each datapoint is weighted by the reward associated with the entire trajectory it is in. For parameter update, the total reward accrued replaces N.

We compare here against PoWER (Policy learning by Weighing Exploration with the Rewards), a policy iteration technique for robot motions [14]. It is generally initialized with one successful demonstration and used to improve robot performance beyond that of the human, but we here apply it to learning from failure.

PoWER operates by weighing the parameters, rather than the datapoints. From each failed demonstration we extract a different set of parameters, θ_s, with an associated weight ω_s. A new trial's parameters is then computed as the weighted average of all previous trials, plus some Gaussian noise Σ_p

$$\theta' \sim \mathcal{N}\left(\frac{1}{\sum_{s=1}^{S}\omega_s}\sum_{s=1}^{S}\omega_s\theta_s, \Sigma_p\right) \tag{18}$$

Note that K must be constant across all θs.

Table 1 Summary of results (# of attempts to achieve success) for learning with reward from failed demonstrations

	FLIPUP	BASKET
Donut	4.30 ± 0.48	7.67 ± 0.58
PoWER	4.60 ± 2.17	11.00 ± 5.29
Human	5.2 ± 3.11	3.50 ± 1.73

In the FlipUp task (Fig. 5a) there are two failure modes: If the block falls back to the starting position, reward is measured as $\omega = \exp(-\text{argmin}_t|\phi_t|)$, with ϕ being the angle of the block with respect to the normal of the table. If the block instead passes to the other side, $\omega = \exp(-6|\dot{\phi}_{t^*}|)$, where t^* is the time at which the block passes the upright position, and 6 is a scaling constant to account for the magnitude difference between the ϕ and $\dot{\phi}$. For the Basket case (Fig. 5b), reward is computed as $\omega = \exp(-|y|)$, where y is the vertical offset of the ball from the lip of the basket when it makes contact with the wall. For both tasks the necessary information is extracted from a fast stereo vision pair. We ran both algorithms on the same data sets (of two failed demonstrations each), and show results in Table 1. For comparative purposes we also show the number of trials the humans took to successfully complete the task. We note that the FlipUp task was learnt quicker by the robot than the human, and vice-versa for the Basket task, indicating that what seems easier for one does not carry to the other.

We see that Donut slightly outperforms PoWER on the FlipUp task, and that this difference is more noticeable on the more complicated Basket task. Further, while the means may not be significantly different, there is an order of magnitude improvement in the variances. We believe this is due to the more targeted way in which Donut explores.

5 Issues

We have thus successfully shown that failure data does contain useful information for task learning, and have demonstrated an approach that can use only that data to discover an appropriate robot controller. However, there are several issues with the DMM-ADS that will make scaling up to higher dimensions and more complicated tasks difficult.

A first issue is how the system extrapolates beyond the demonstrations. The GMM-ADS on which our system is based is designed to represent observed data, capturing well the nonlinearities of the distribution near the human's demonstrations. However, further away, the model breaks down, and generated velocities may not accurately predict what a human would have done, as shown in Fig. 6.

In 1D, we are never far from the observed data - no matter what velocity we apply, the system remains in the space explored by the human, so this issue is moot. However, even just in 2D, the system quickly enters regions of the state space that were unexplored during human attempts. Without a reasonable model of human behavior, using the donut distribution does not make sense. To address this issue, we will require an alternate representation of the robot's motion.

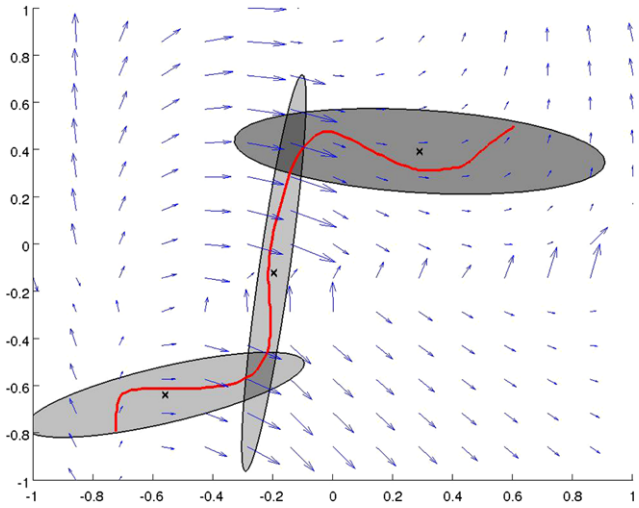


Fig. 6 An illustration of the extrapolation problem. Far from observed data, a GMM-ADS may behave other than the human would have (*lower right*)

A second issue concerns the use of gradient ascent. While the locations of the maxima of a single donut do have an analytical solution, those of the entire DMM do not. Thus, we are forced to use the slow and only locally optimal gradient ascent to generate velocities. While in 1D this process can occur relatively quickly, in higher dimensions it will be more difficult to ensure real-time computability.

Further, gradient ascent only guarantees finding a local maximum. Due to the multi-modal nature of the DMM, where each component can generate up to two peaks, there are many suboptimal maxima that can be discovered. In our experiments, we initialize our search at the last known maximum, which alleviates this concern. However, in further studies we have seen the system get stuck in suboptima.

Lastly, we are worried about the possible interference between donuts. While one donut is guaranteed to decrease the likelihood of its mean as exploration increases, two donuts in close proximity may accidentally increase each other's means, as in Fig. 7. Because we use the overall variance of the GMM to set the exploration, this issue rarely arises, as GMMs with components that are close to each other tend to have small variances, leading to small exploration, and minimal overlap in the resulting donuts. However, as our models increase in complexity, this situation may arise more often and may lead to known failures being replicated.

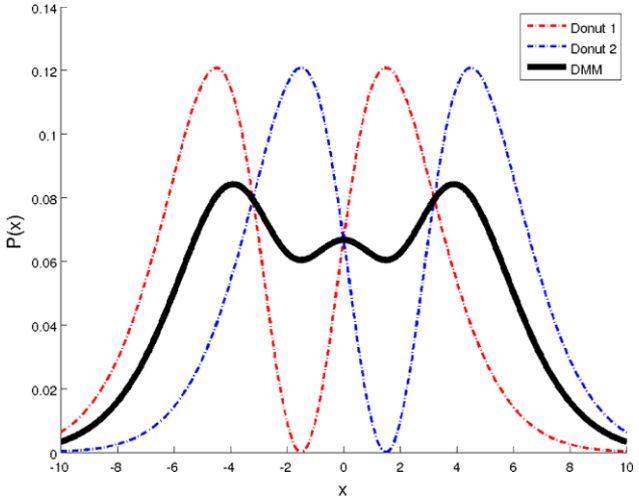


Fig. 7 As exploration increases, two proximal donut distributions (*red* and *blue*) may interfere, increasing the likelihood of each other's means in the overall distribution (*black*), contrary to design

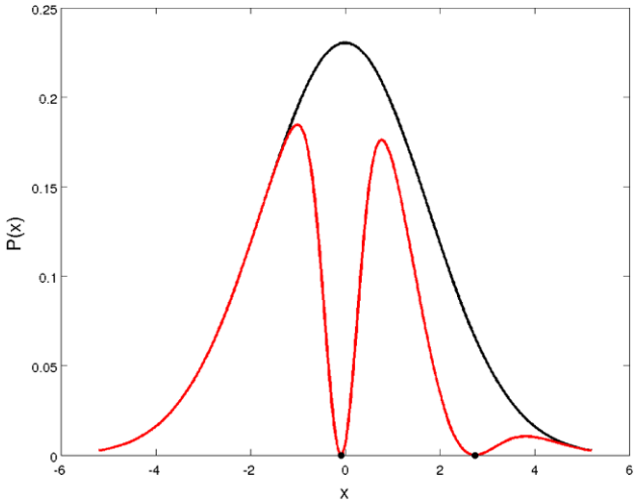


Fig. 8 A multidonut distribution, which avoids interference between 'holes.' Exploration around observed human failures is now controlled by the width (variance) at each hole, as well as the overarching Gaussian distribution

6 Higher Dimensions

To address these issues, when moving to higher dimensions we switch from gradient-based computation with a DMM in state-velocity space to a sampling-based approach using one distribution with multiple areas of low density ("holes") in parameter space. In this section we describe the approach, and in the next some experiments to test its feasibility.

6.1 Parameter Representation

In 1D we used the donut distribution to directly modify the velocities generated by an ADS. However, in higher dimensions the system will inevitably move beyond the limits of the observed data, rendering the ADS model insufficient. We therefore 'lift' the donut into a parameter space which will allow us to change the underlying representation as needed.

Given a set of parameters, $\{\theta^s\}_{s=1}^S$ from S human failed demonstrations, we now build a distribution over parameters. We will want the specific parameters that are known to be bad to be unlikely, while the area around them is more or less likely dependent upon the human's own exploration. As we now operate in parameter space, the underlying controller can be changed. For example, a GMM-ADS can be used, where θ is as before. Or, a spline controller could be used, where θ would be the spline points and coefficients.

In operating over parameters, our approach is similar to that of PoWER. Recall that PoWER draws θ' from a Gaussian centered on the weighted mean of previous trials as in Eq. (18). Comparing this with Eq. (4) we see that the mean of PoWER's distribution has the same form as the expectation of a GMM, with $K = S$, $\rho_k = \frac{\omega_s}{\sum_{s=1}^S \omega_s}$, $\mu_k = \theta_s$. We can likewise use Eq. (17) to derive a (non-unique) setting for Σ_k in terms of PoWER's variance Σ_p. PoWER can then be viewed as drawing from a Gaussian approximation of a GMM, $\theta' \sim \mathcal{N}(\tilde{E}, \tilde{V})$. An alternative would be to draw from the full GMM instead, perhaps replacing each component with a donut. However, when we did so we encountered the interference problem discussed above.

6.2 MultiDonut

To avoid interference, we change from a mixture of Donuts to a single distribution with K "holes," which we call the multidonut distribution. The probability of a point is

$$P(\mathbf{x}) = \frac{1}{Z}\mathcal{N}\left(\mathbf{x}|\mu^0, \Sigma^0\right) \times \prod_{k=1}^{K}\left(1 - \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu^k\right)^\top \times \Sigma^{k^{-1}}\left(\mathbf{x}-\mu^k\right)\right)\right) \tag{19}$$

and a 1D illustration is shown in Fig. 8.

The naught distribution parametrized by μ^0 and Σ^0 constrains the data to be near the observed human demonstrations. The 'holes' take the place of individual donuts and are centered on the attempts and thus reduce the probability of exactly replicating the known failures. The covariance of each hole now plays the role of the exploration parameter, and we will explore several different methods for setting them. Because the holes are multiplied in, the probability of a point can never be above the minimum probability of any component, removing interference. Z is a normalization constant to ensure that the distribution integrates to 1.

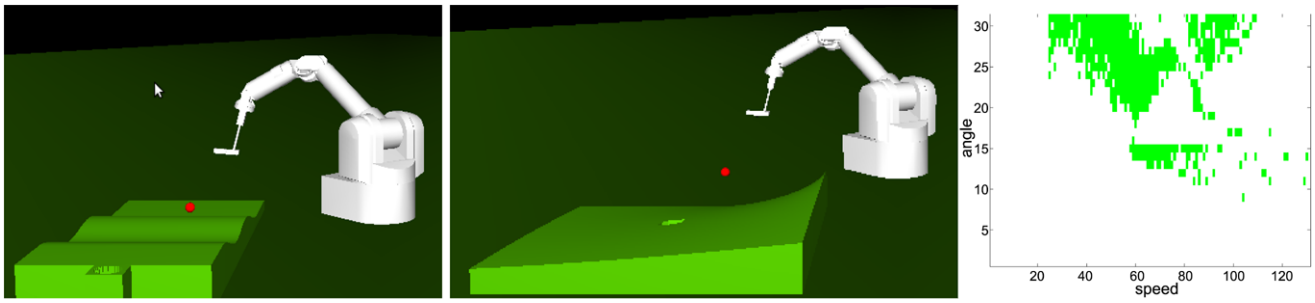


Fig. 9 The two fields used in our minigolf simulator. Users set the 4 (ball x/y, hitting speed and angle) parameters by drawing lines in 2D, the endpoints of which determined values. The simulator execution determined success or failure. *Right, green areas* are successful settings in a 2D version of the golf simulator (position held constant). The large set of successful parameters makes this task easy for humans

6.3 Sampling

As a single θ suffices for an entire trial, consistency between values is not an issue, so we can use sampling instead of gradient ascent. To draw samples from the multidonut distribution, we use rejection sampling where we first draw a possible sample from a proposal distribution $\mathbf{x} \sim h(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu^0, \Sigma^0)$. Since the product of the 'holey' part of the multidonut distribution is no more than one, we see that $P(\mathbf{x}) < \frac{1}{Z}h(\mathbf{x})$. We then accept $\mathbf{x}$ as a sample with probability proportional to the ratio between P and $\frac{1}{Z}h(\mathbf{x})$:

$$\frac{\frac{1}{Z}\mathcal{N}(\mathbf{x}|\mu^0, \Sigma^0)\prod_{k=1}^{K}(1-\exp(-\frac{1}{2}(\mathbf{x}-\mu^k)^\top \Sigma^{k^{-1}}(\mathbf{x}-\mu^k)))}{\frac{1}{Z}\mathcal{N}(\mathbf{x}|\mu^0, \Sigma^0)}$$

The normalization constants cancel, as do the naught distributions, leaving us with the probability of acceptance as

$$\prod_{k=1}^{K}\left(1-\exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu^k\right)^\top \Sigma^{k^{-1}}\left(\mathbf{x}-\mu^k\right)\right)\right) \tag{20}$$

Drawing samples thus scales with the number of holes and the widths (Σ^k). As we only need one sample to run an entire trial, this issue is somewhat negligible.

7 High Dimensional Experiments

We now present some exploratory experiments to judge the suitability of our sampling-based multidonut approach for finding success when initialized with failure. These tests used a simulated robot arm that played mini-golf, as shown in Fig. 9. Previously, this setup was used to learn appropriate hitting parameters, but only from successful demonstrations [16]. The system takes 4 parameters, the x and y position of the ball, and the desired hitting speed and angle. In some experiments we held the ball's position constant, and only varied speed and angle. A 2D GUI allowed human users to select the parameters for execution and view the resulting shot. Due to the nature of the field, multiple parameter settings can lead to success.

We used two fields, the 'wavy' field and the 'arctan' field, shown in Fig. 9 left and center, respectively. We also used an alternate abstract platform with only one 'correct' goal point, where users received color-based feedback as they tested parameters instead of viewing full golf swings.

Over all platforms, we collected data from 6 humans, denoted S, G, M, J, K, and D, who selected parameters until they succeeded at the task. Some tasks, such as golf with only 2 parameters, were noticeably easy, requiring only a few (<10) human attempts to succeed, perhaps due to the multiple possible successful settings as seen in Fig. 9 right. Others, such as the abstract 4D space, required more (>50).

From varying amounts of human failure data (neglecting the last, successful attempt) we built models that generated new exploratory parameters. Each model was run until it discovered success, and we compare them based on the number of trials (averaged over multiple random restarts).

The models we tested were:

1. Positive only: Model all of the human's demonstrations with a single Gaussian (no holes).
2. Fixed-width Multidonut: The above Gaussian, with holes at each of the demonstrations. All Σ^k are equal.
3. Incremental Multidonut: As above, but all newly generated robot trials are also used as holes.
4. Growing Multidonut: As 2, but each new failure widens the holes proportional to how close they are to it.

Additionally, we explored the assumption that the human improves over time, and so the later trials might be taken as reverse tick 'less bad' than the earlier ones. Doing so led to the models:

5. Weighted-Positive: Linear or Exponential weights are applied to the data before fitting the Gaussian.
6. Last: The above Gaussian has its mean set to be the last human trial (a failure).
7. Variable-width Multidonut: The covariances of the holes are scaled to match the weights on the data.

These models are illustrated in Fig. 10.

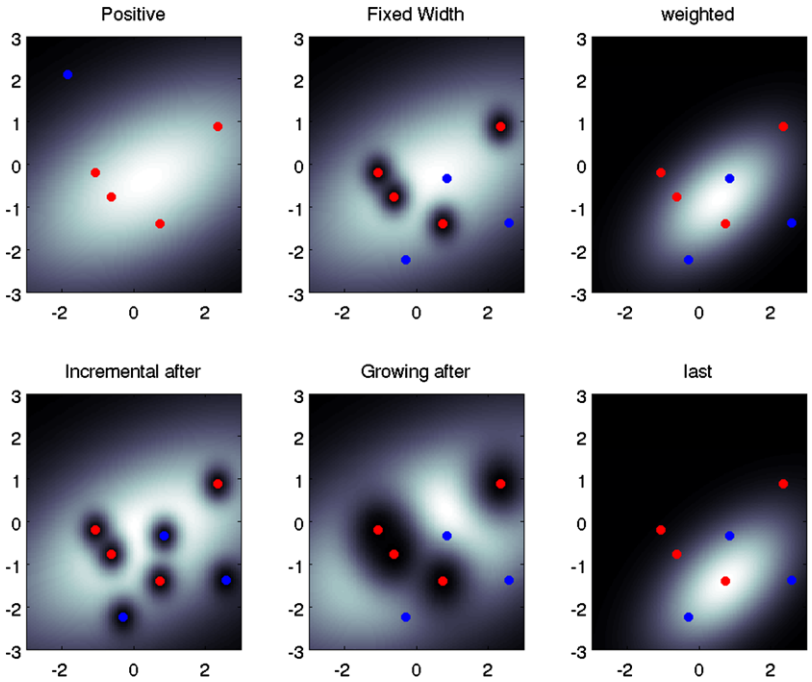


Fig. 10 An overview of the distributions used in our higher-dimensional experiments. Shown in 2D, lightness corresponds to the likelihood of generating arbitrary 2D parameters. *Red dots* are human attempts, *blue* are system-generated trials. Incremental and growing distributions initially start identical to the fixed width distribution

Table 2 Sample of results using various multidonut models

Human (*S*)	Positive [1]	Fixed [2]	Last [6]	Variable [7]
S (7)	3356	6547	**335**	4331
G (8)	**13000**	274418	64756	176828
M (18)	1061	759	**420**	597
K (33)	446	135	**51**	187
J (29)	60	59	198	**56**
D (9)	147	80	**45**	83

Over multiple humans, trials, and hyperparameter settings, the best overall performer was model 6. Illustrative (bad) results are presented in Table 2 for one of the more difficult, single-point success cases.[1] Note that for some datasets (J, M), the negative models (2, 7) perform very well. However, this behavior is not consistent over datasets. From these results we infer that the "improving over time" assumption for human attempts is valid, but that our model of how to use the other failed attempts needs improvement.

8 Discussion

In this article, we demonstrate that failed demonstration data is not without merit, to be discarded in favor of a single successful one. Instead, it has information that can lead a robot to learn to perform a task it has never observed. Our proposed method does this by explicitly avoiding the reproduction of known bad values while exploring based on some basic assumptions as to the nature of multiple human attempts.

In scaling our approach to higher dimensions, we addressed several issues such as interference between negative models, extrapolation beyond observations, and dealing with local optima. We further introduced the assumption that humans themselves improve over time, and used it to further guide our exploration. Unfortunately, while we were able in a few cases to lead to rapid convergence, our approach was generally not competitive with a baseline "search near the human's last attempt" technique. From this we conclude that the human, who has access to much richer feedback and a better sense of the system's dynamics, is a good guide.

Thus, we believe the main issue is a lack of feedback from the human during the system's exploratory trials. Without knowledge as to whether or not the behavior is improving, it is impossible to determine how the distribution over parameters should change. The local density of human attempts is not enough—a high concentration could indicate either that the human believes success to be near, or that an area has been 'explored out' and the system should try elsewhere. Incorporating temporal information by weighing the data was aimed at alleviating this, but it was insufficient.

If the system is able to monitor its own success, such as with a reward function, then this dilemma can be resolved. However, we believe that the requirement of an explicit reward function may be too strong. Writing one down may take extensive domain expertise or analytical skills that an end user does not have. For many tasks, there are multiple

[1] In situations where success is more common, such as in Fig. 9 right, all approaches faired generally equally.

ways to fail, with some better than others. While people may intrinsically be able to compare them, formulating an exact mathematical statement may be beyond their ability.

Instead, one solution may be to incorporate aspects of tutelage into the LfF framework. For example, the robot's exploratory trials could graded by a human observer. Abstract grades such as "better," "worse," or "no change" might provide enough local gradient information for the system to converge. More detailed feedback such as critiquing and direct modifications could also be used.

Additionally, it may be possible to use successful demonstrations in conjunction with failed ones to guide the system to self educate. A robot could start by replicating the human, and then vary its behavior within the observed error bounds. Doing so may give the robot a better sense of where the policy breaks down and help it be more robust to changes.

Thus, we see LfF as a potential 'afterburner' supplement to already existing LfD techniques. While learning from a perfect demonstration may be the ideal, collecting that data will become more difficult as task complexity increases. Current techniques exist that can use suboptimal demonstrations and improve their performance with further interaction, but do not treat the known bad examples as such. By modeling this fact explicitly, a robot may be able to better leverage the data it is given, and decrease the total amount of information needed from the user by not repeating the same mistakes.

9 Conclusion

In this article, we argue that data from failed human demonstrations of a task should not be discarded. Instead, we show that it is possible to build models from this data that can guide a robot system to discover a successful way to perform a novel task. In higher dimensions, however, more information may be needed to achieve good performance.

Acknowledgements This work was supported by the European Commission under contract number FP7-248258 (First-MM).

References

1. Abbeel P, Coates A, Quigley M, Ng AY (2006) An application of reinforcement learning to aerobatic helicopter flight. In: Neural inf proc systems
2. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483
3. Billard A, Calinon S, Dillmann R, Schaal S (2008) Survey: robot programming by demonstration. Handbook of robotics. MIT Press, Cambridge
4. Chernova S, Veloso M (2007) Confidence-based policy learning from demonstration using Gaussian mixture models. In: Intl joint conf on autonomous agents and multi-agent systems
5. Dayan P, Hinton G (1997) Using expectation-maximization for reinforcement learning. Neural Comput 9(2):271–278
6. Deisenroth MP, Rasmussen CE (2011) Pilco: a model-based and data-efficient approach to policy search. In: Intl conf on machine learning
7. Dong S, Williams B (2011) Motion learning in variable environments using probabilistic flow tubes. In: Intl conf on robotics and automation
8. Gams A, Do M, Ude A, Asfour T, Dillmann R (2010) On-line periodic movement and force-profile learning for adaptation to new surfaces. In: Intl conf on humanoid robots
9. Grimes DB, Chalodhorn R, Rao RPN (2006) Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Robotics: science and systems
10. Grollman DH, Billard A (2011) Donut as I do: Learning from failed demonstrations. In: Intl conf on robotics and automation
11. Grollman DH, Jenkins OC (2007) Dogged learning for robots. In: Intl conf on robotics and automation
12. Hersch M, Guenter F, Calinon S, Billard A (2008) Dynamical system modulation for robot learning via kineshetic demonstrations. Trans Robot, 1463–1467
13. Hu X, Xu L (2004) Investigation on several model selection criteria for determining the number of cluster. Neural Inf Process - Lett Rev 4(1):1–10
14. Kober J, Peters J (2010) Policy search for motor primitives in robotics. Mach Learn 84(1–2):171–203
15. Kober J, Mohler B, Peters J (2008) Learning perceptual coupling for motor primitives. In: Intl conf on intelligent robots and systems
16. Kronander K, Khansari Zadeh SM, Billard A (2011) Learning to control planar hitting motions in a monigolf-like task. In: Intl conf on intelligent robots and systems
17. Kuniyoshi Y, Inaba M, Inoue H (1994) Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Trans Robot Autom 10(6):799–822
18. Meltzoff AN (1995) Understanding the intentions of others: reenactment of intended acts by 18-month-old children. Dev Psychol 31(5):838–850
19. Mtsui T, Inuzuka N, Seki H (2002) Adapting to subsequent changes of environment by learning policy preconditions. Int J Comput Inf Sci 3(1):49–58
20. Neal R, Hinton GE (1998) A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Learning in graphical models
21. Pastor P, Kalakrishnan M, Chitta S, Theodorou E, Schaal S (2011) Skill learning and task outcome prediction for manipulation. In: Intl conf on robotics and automation
22. Ramachandran D, Amir E (2007) Bayesian inverse reinforcement learning. In: Intl joint conf on artificial intelligence
23. Sung HG (2004) Gaussian mixture regression and classification. PhD thesis, Rice
24. Thomaz AL, Breazeal C (2008) Experiments in socially guided exploration: lessons learned in building robots that learn with and without human teachers. Connect Sci 20(2–3):91–110
25. Want SC, Harris PL (2001) Learning from other people's mistakes: causal understanding in learning to use a tool. Child Dev 72(2):41–443

Daniel H. Grollman is a postdoctoral fellow at the LASA Laboratory at EPFL. He received his B.S. (2003) in Electrical Engineering and Computer Science from Yale University, and his Sc.M. (2005) and Ph.D. (2010) in Computer Science from Brown University.

Aude G. Billard is Associate Professor and head of the LASA Laboratory at the School of Engineering at EPFL. She received her B.Sc. (1994) and M.Sc. (1995) in Physics from EPFL, and a Ph.D. in Artificial Intelligence from the University of Edinburgh (1998).