Comparing Human-Centric and Robot-Centric Sampling for Robot Deep Learning from Demonstrations

Michael Laskey[1], Caleb Chuck[1], Jonathan Lee[1], Jeffrey Mahler[1],
Sanjay Krishnan[1], Kevin Jamieson[1], Anca Dragan[1], Ken Goldberg[1,2]

Abstract— **Motivated by recent advances in Deep Learning for robot control, this paper considers two learning algorithms in terms of how they acquire demonstrations. "Human-Centric" (HC) sampling is the standard supervised learning algorithm, where a human supervisor demonstrates the task by teleoperating the robot to provide trajectories consisting of state-control pairs. "Robot-Centric" (RC) sampling is an increasingly popular alternative used in algorithms such as DAgger, where a human supervisor observes the robot executing a learned policy and provides corrective control labels for each state visited. RC sampling can be challenging for human supervisors and prone to mislabeling. RC sampling can also induce error in policy performance because it repeatedly visits areas of the state space that are harder to learn. Although policies learned with RC sampling can be superior to HC sampling for standard learning models such as linear SVMs, policies learned with HC sampling may be comparable to RC for with emerging classes of highly-expressive learning models such as deep learning and hyper-parametric decision trees, which can achieve very low training error. We compare HC and RC using a grid world and a physical robot singulation task. In the latter the input is a binary image of a of objects on a planar worksurface and the policy generates a motion of the gripper to separate one object from the rest. We observe in simulation that for linear SVMs, policies learned with RC outperformed those learned with HC but that with deep models this advantage disappears. We also find that with RC, the corrective control labels provided by humans can be highly inconsistent. We prove there exists a class of examples where in the limit, HC is guaranteed to converge to an optimal policy while RC may fail to converge. These results suggest HC sampling may be preferable for highly-expressive learning models and human supervisors.**

I. Introduction

Learning from Demonstrations (LfD) is a well-established approach where a robot learns a policy from example trajectories provided by a human supervisor [15], [16], [22], [21], [23]. In principle, learning a state-feedback policy is a supervised learning problem–a regression from observed states to observed controls from the set of example trajectories. However, one challenge is if the robot cannot exactly replicate the policy of the supervisor, for example, due to modeling errors or insufficient data. In this case, the *training* error achieved on the example trajectories may not be indicative of the actual *execution* error when using the policy since the robot may visit a very different distribution of states. Algorithms such as DAgger have been proposed to address this problem, where after an initial policy is learned the supervisor observes the robot executing a learned policy and retro-actively provides corrective control labels for each state visited.

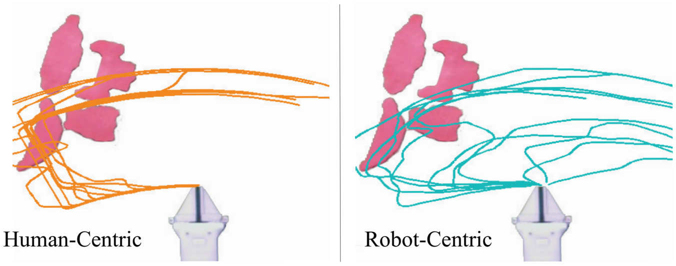


Fig. 1: HC and RC trajectories used to learn a robot singulation task: top view of 4 objects on a planar worksurface. Left: In HC, 10 trajectories where a human demonstrates the task by tele-operating a four-axis robot to separate one object from a connected set of objects. Right: In RC, after initial training, 10 trajectories of the sub-optimal robot policy are executed and a human provides corrective control labels for each. Note that the latter trajectories spend considerable time in areas of the workspace that will not be useful after the task is learned.

A recent trend in Machine Learning is to use more expressive models such Deep Neural Networks and Decision Trees [17]. Given enough training data, such models allow us to represent highly complex feedback policies and may help alleviate the discrepancy between training and execution error. This paper studies the relative merits of DAgger-like methods applied with a human supervisor in the context of model expressiveness.

We consider two classes of learning algorithms that differ in how they acquire demonstrations. "Human-Centric" (HC) sampling is the standard supervised learning algorithm, where a human supervisor demonstrates the task by tele-operating the robot to provide trajectories consisting of state-action pairs. [5]."Robot-Centric" (RC) sampling is an increasingly popular alternative used in algorithms such as DAgger, where a human supervisor observes the robot executing a learned policy and retro-actively provides corrective control labels for each state visited [21], [22], [15], [16], [12]. The intuition behind this is to provide examples to the robot in states it is likely to visit.

While RC methods have significant advantages, it can come at a cost in practice. One cost is the challenge posed to a human supervisor. In RC methods, a human provides corrective feedback, retroactively, to the robot without observing the effect of their suggested control. This can require additional cognitive effort on the human supervisor to predict how the world behaves with observing the outcome of their control.

A second cost is that RC sampling collects labels on states that occur due to previous errors. These states may

[1] Department of Electrical Engineering and Computer Sciences; {mdlaskey, calebchuck, jonathan_lee, jmahler, sanjaykrishnan, anca}@berkeley.edu

[2] Department of Industrial Engineering and Operations Research; goldberg@berkeley.edu

[1−2] University of California, Berkeley; Berkeley, CA 94720, USA
The AUTOLAB at UC Berkeley `automation.berkeley.edu`

require complex corrective actions that do not occur when the supervisor performs the task. For example, consider the task of singulating (i.e. separate) an object from a set of objects on a table (Fig. 1). Demonstrations involve moving forward and side to side to push obstacles out of the way. However, if the robot makes errors and ends up in a failure state (Fig. 1(right)), then more complex maneuvers will be required. A potential outcome of this effect is the robots policy can incur larger learning error and not converge to the supervisors policy.

The key question is whether the benefits of RC outweigh these practical challenges, if the robot's policy is an expressive model that can significantly reduce training error. First, we show in simulated examples, i.e., only comparing statistical efficiency, that when the model is highly expressive HC and RC are essentially at parity. By varying the expressiveness of the robots policy, we find on 100 randomly generated grid world environments that the performance gap between the two methods diminishes as the expressiveness is increased. Next, using a point mass control example, we find that RC sampling can fail to converge in cases when HC converges.

Finally, we illustrate the human factors challenge of implementing RC with a 10 person pilot study. In this experiment, each human used RC and HC sampling to train a Zymark robot to singulate an object. We observed a statistically significant gap in the average performance of the policies trained by the two approaches. For 60 trajectories, RC had a success rate of 40% compared with 60%, for HC. Our post analysis suggests participants had a hard time providing the retroactive feedback needed for RC sampling.

We conclude the paper with an initial theoretical analysis that attempts to explain our findings. First, we illustrate a counter-example where as the number of demonstrations increases, HC converges to an optimum with probability 1, while for the same model, RC has a non-zero probability of converging to a suboptimal solution. We emphasize that this result is not a contradiction of prior theory [22] as RC is only theoretically guaranteed in terms of hindsight regret. Finally, we analyze the effect of stochastic supervisor, which is common with humans. We show how this increases the overlap of the supervisor's and robot's distributions, which can reduce the penalty of using supervised learning.

II. Related Work

Below we summarize related work in HC and RC LfD and their theoretical insights.

HC Pormeleau et al. used HC to train a neural network to drive a car on the highway via demonstrations provided by a supervisor. To reduce the number of examples needed to perform a task, they synthetically generated images of the road and control examples [20]. A similar approach was used by Dilmann et al. to teach a robot to insert perform peg-in-hole using a neural network controller [8]. Schulman et al. used HC LfD for rope tying using kernelized interpolation as the policy class [24]. A survey of HC LfD techniques can be found in [5].

Ross et al. examined HC and showed error in this approach can grow, in the worst case, quadratically in the time horizon T [21]. The intuition behind this analysis is that due to training error the robot's policy will visit different states than the supervisor. Thus, the distribution it is trained on will be different than that it is tested on, which can lead to poor performance. The analysis also suggests that this worst case error can be controlled via training error. We are interested if expressive models are able to reduce training error enough to mitigate this problem.

We note there are examples [7], [10], [11], where RC methods still outperformed HC even though an expressive model was used. Despite an expressive model training error can be significant due to inaccurate feature representation and local minima in optimization. However, a model being able to represent a supervisor is a prerequisite for achieving low training error. It is also important to note these examples do not consider a human supervisor, which can affect relative performance of RC and HC.

RC RC has been used in numerous robotic examples, including flying a quadcopter through a forest where the state space is image data taken from an onboard sensor [23]. Other successful examples have been teaching a robotic wheelchair to navigate to target positions [14], teaching a robot to follow verbal instructions to navigate across an office building [9] and teaching a robot to grasp in clutter [15].

Ross et al. [22] analyze RC sampling as online optimization. They propose DAgger, an RC sampling algorithm, and show that the error for the robot's policy is linear in T for strongly convex losses (e.g. regularized linear or kernelized regression). However, the error also depends on the expected loss on the data collected during RC, which may be high due to observing complex recovery behaviors. We analyze this effect and show how it can prevent RC from converging to the supervisor's policy in Section V-A.

Kim et al. proposed to only query the supervisor in states that where robot is uncertain about which decision to make [14] . Laskey et al. extended this approach to high-dimensional states [16]. Laskey et al. [15] examined a hierarchy of supervisors (ranked by quality) to reduce the burden on an expert demonstrator. Zhang and Cho trained a classifier to detect when RC sampling would encounter dangerous states during policy execution and have a human take over [30].

Levine et al. identified that the RC sampling can force the robot into states that are harder to learn and proposed weighting the samples to correct for this. They also proposed forcing the supervisor to guide the robot's policy to better regions of the state space [18]. This approach is used for an algorithmic supervisor and assumes the supervisor can both be modified and the noise distribution is known. We are interested in maintaining the original assumption of Ross et al. [22], in which there is an unknown supervisor whose demonstrations cannot be modified. He et al. proposed changing the supervisor's example to be easier to learn for the robot's policy [12]. We are interested in changing the robot's expressiveness to make the supervisor easier to learn.

LfD Interfaces: Standard techniques for providing demonstrations to a robot are teleoperation, kinesthetic and waypoint specification [2], [3], [5]. Kinesthetic teaching is defined as moving the robot body via a human exerting force on the robot itself. Teleoperation uses an interface such as a joystick or video game controller to control the position of the robot end effector. Waypoint specification, or Keyframes, has a human select positions in the space that the robot needs to visit. These methods are forms of HC sampling because the human guides the robot through the task. We look specifically at teleoperation and compare it to RC's form of retroactive feedback.

III. Problem Statement and Background

The objective of LfD is to learn a policy that matches that of the supervisor on a specified task that demonstrations are collected on.

Modeling Choices and Assumptions: We model the system dynamics as Markovian, stochastic, and stationary. Stationary dynamics occur when, given a state and a control, the probability of the next state does not change over time.

We model the initial state as sampled from a distribution over the state space. We assume a known state space and set of controls. We also assume access to a robot or simulator, such that we can sample from the state sequences induced by a sequence of controls. Lastly, we assume access to a supervisor who can, given a state, provide a control signal label. We additionally assume the supervisor can be noisy.

Policies and State Densities. Following conventions from control theory, we denote by $\mathcal{X}$ the set consisting of observable states for a robot task, such as images from a camera, or robot joint angles and object poses in the environment. We furthermore consider a set $\mathcal{U}$ of allowed control inputs for the robot, which can be discrete or continuous. We model dynamics as Markovian, such that the probability of state $\mathbf{x_{t+1}} \in \mathcal{X}$ can be determined from the previous state $\mathbf{x}_t \in \mathcal{X}$ and control input $\mathbf{u}_t \in \mathcal{U}$:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t, \ldots, \mathbf{x}_0, \mathbf{u}_0) = p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$$

We assume a probability density over initial states $p(\mathbf{x}_0)$. The environment of a task is thus defined as a specific instance of a control and state space, initial state distribution, and dynamics.

Given a time horizon $T \in \mathbb{N}$, a trajectory τ is a finite sequence of T pairs of states visited and corresponding control inputs at these states, $\tau = ((\mathbf{x}_0, \mathbf{u}_0), ..., (\mathbf{x}_{T-1}, \mathbf{u}_{T-1}))$, where $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{u}_t \in \mathcal{U}$ for $t \in \{0, \ldots, T-1\}$.

A policy is a measurable function $\pi : \mathcal{X} \to \mathcal{U}$ from states to control inputs. We consider policies $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parametrized by some $\theta \in \Theta$. Under our assumptions, any such policy π_θ induces a probability density over the set of trajectories of length T:

$$p(\tau|\theta) = p(\mathbf{x}_0) \prod_{i=0}^{T-1} p(\mathbf{u}_t|\mathbf{x}_t, \theta) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$$

The term $p(\mathbf{u}_t|\mathbf{x}_t, \theta)$ indicates stochasticity in the applied control, which can occur due to noise in robot execution. While we do not assume knowledge of the distributions corresponding to: $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, $p(\mathbf{x}_0)$ or $p(\mathbf{x}_t|\theta)$, we assume that we have a stochastic real robot or a simulator such that for any state $\mathbf{x}_t$ and control $\mathbf{u}_t$, we can observe a sample $\mathbf{x}_{t+1}$ from the density $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. Therefore, when 'rolling out' trajectories under a policy π_θ, we utilize the robot or a simulator to sample the resulting stochastic trajectories rather than estimating $p(\mathbf{x}|\theta)$ itself.

Objective. The objective of policy learning is to find a policy that maximizes some known reward function $R(\tau) = \sum_{t=1}^{T} r(\mathbf{x}_t, \mathbf{u}_t)$ of a trajectory τ. The reward $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is typically user defined and task specific. For example in the task of grasping, the reward can be a binary measure of success.

In our problem we do not have access to the reward function itself. Instead, we only have access to a supervisor, π_{θ^*}, where θ^* may not be contained in Θ. A supervisor is chosen that can achieve a desired level of performance on the task.

We measure the difference between controls using a surrogate loss $l : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ [22], [21]. The surrogate loss can either be an indicator function as in classification or a continuous measure on the sufficient statistics of $p(\mathbf{u}|\mathbf{x}, \theta)$. We measure total loss along a trajectory with respect to the supervisor's policy π_{θ^*} by $J(\theta, \tau) = \sum_{t=1}^{T} l(\pi_\theta(\mathbf{x}_t), \pi_{\theta^*}(\mathbf{x}_t))$.

HC In HC, the supervisor provides the robot a set of N demonstration trajectories $\{\tau^1, ..., \tau^N\}$ sampled from $p(\tau|\theta^*)$. This induces a training data set $\mathcal{D}$ of all state-control input pairs from the demonstrated trajectories. The goal is to find the θ^N that minimizes the empirical risk, or sample estimate of the expectation.

$$\theta^N = \underset{\theta}{\text{argmin}} \sum_{i=1}^{N} J(\theta, \tau_i). \quad (1)$$

RC Due to sample approximation and learning error, one potential issue with RC sampling is that θ^N differs from θ^*. Therefore it is possible that new states will be visited under θ^N that would never have been visited under θ^*. To account for this, prior work has proposed iterative solutions [22] that attempt to solve this problem by aggregating data on the state distribution induced by the current robot's policy.

Instead of minimizing the surrogate loss in Eq. 1, LfD with RC sampling [22], [16], [12] attempts to approximate the state distribution the final policy will converge to and minimize the surrogate loss on this distribution. A popular RC LFD algorithm is DAgger [22], which iterates two steps:

1) Step 1: The first step of any iteration k is to compute a θ_k that minimizes surrogate loss on the current dataset $\mathcal{D}_k = \{(\mathbf{x}_i, \mathbf{u}_i)|i \in \{1, \ldots, N\}\}$ of demonstrated state-control pairs (initially just the set $\mathcal{D}$ of initial trajectory demonstrations):

$$\theta_k = \arg\min_\theta \sum_{i=1}^{N} \sum_{t=1}^{T} l(\pi_\theta(\mathbf{x}_{i,t}), \mathbf{u}_{i,t}). \quad (2)$$

Note that equal weight is given to each example regardless of how likely it is under the current policy.

2) Step 2: The second step at iteration k, DAgger rolls out the current policy, π_{θ_k}, to sample states that are likely under $p(\tau|\theta_k)$. For every state visited, DAgger requests the supervisor to provide the appropriate control/label. Formally, for a given sampled trajectory $\tau = (\mathbf{x}_0, \mathbf{u}_0, ..., \mathbf{x}_T, \mathbf{u}_T)$, the supervisor provides labels $\tilde{\mathbf{u}}_t$, where $\tilde{\mathbf{u}}_t \sim \tilde{\pi}(\mathbf{x}_t) + \epsilon$, where ϵ is a zero mean noise term, for $t \in \{0, \ldots, T\}$. The states and labeled controls are then aggregated into the next data set of demonstrations $\mathcal{D}_{k+1}$:

$$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t)|t \in \{0, \ldots, T\}\}$$

Steps 1 and 2 are repeated for K iterations or until the robot has achieved sufficient performance on the task[1]

IV. Empirical Analysis

We first provide an empirical comparison of HC and RC LfD. We start with experiments in a Grid World environment, which enables us to vary the robot's policy class over a large number of randomly generated environments. Then we examine a linear dynamical system, which we use to exemplify a potential limitation in the ability of RC to match the supervisor's performance when it must learn complex recovery actions. Finally, we compare HC and RC on a real task. We perform a pilot study with 10 participants, who try to teach a robot how to singulate, or separate an object from clutter. We used DAgger as the example of an RC method in these experiments.

A. Policy Expressiveness

In this experiment, we hypothesize that on random problems with a perfect simulated supervisor, the performance gap of RC diminishes as the robot's policy becomes more expressive. The reason for this is if the robot has low training error than the robot can converge to the supervisor's distribution as more data is added.

In Grid World, we have a robot that is trying to reach a goal state, at which it receives $+10$ reward. It also receives -10 reward if it touches a penalty state. The policy must learn to be robust to the noise in the dynamics, reach the goal state and then stay there. The robot has a state space of (x, y) coordinates and a set of actions consisting of $\{$Left, Right, Forward, Backward, Stay$\}$ state. The grid size for the environment is 15×15. 15% of randomly drawn states are marked as a penalties. For the transition dynamics, $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, the robot goes to an adjacent state different from the one desired uniformly at random with probability 0.16. The time horizon for the policy is $T = 30$.

We use Value Iteration to compute an optimal supervisor. In all settings, we provided RC with one initial demonstration from HC sampling before iteratively rolling out its policy. This initial demonstration set from HC sampling is common in RC methods like DAgger [22].

We run all trials over 100 randomly generated environments. We measure normalized performance, where 1.0 represents the expected cumulative reward of the optimal supervisor.

Low Expressiveness: Fig. 2(a) shows a case when the robot's policy class is empirically not expressive enough to represent the supervisor's policy. We used a Linear SVM for the policy class representation, which is commonly used in RC [21], [22], [23]. RC LfD outperforms HC , which is consistent with prior literature [21], [22]. This outcome suggests that when the robot is not able to learn the supervisor's policy, RC has a large advantage. Note that neither method is able to converge to the supervisor's performance.

High Expressiveness: We next consider the situation where the robot's policy class is more expressive. We use decision trees with a depth of 100 to obtain a highly expressive function class.

As shown in Fig. 2(b), RC and HC both converge to the supervisor at the same rate. This suggests that the advantage of using RC LfD diminishes once the robot's policy is more expressive. When the robot is better able to learn the supervisor policy, it is no longer as beneficial to obtain demonstrations specifically on states visited by the robot's current policy. Both RC and HC methods converge to the supervisor's performance.

Noisy Supervisor: Real supervisors will not be perfect. Thus, we study the effects of noise on the performance of the two sampling techniques. This is particularly important for HC, because even an expressive learner might need many more demonstrations when the supervision is noisy, perhaps making samples on the current learned policy more useful. Here we consider the case where noise is applied to the observed label from the supervisor, thus the robot receives control labels that are randomly sampled with probability 0.3.

We use the same class of decision trees with depth 100. Due to the larger expressiveness of a decision tree it is more susceptible to overfitting. We then compare the performance of HC vs. RC. As shown, both methods are able to converge to the expected supervisor's normalized performance, suggesting they can average out the noise in the labels with enough data. Although it takes more data than without noise, both methods (HC and RC) converge at a similar rate to the true expected supervisor.

B. Algorithmic Convergence

In this experiment, we show an example in which HC converges to the supervisor's performance and RC does not. We construct an environment that is representative of our insight that in real problems the policy on *all* states (including recovery behaviors) is more complex than the policy on the supervisor's distribution.

We consider the example where a robot needs to learn to get to a location in a 2D continuous world domain. The robot is represented as a point mass with discrete time double integrator and linear dynamics.

The environment contains two sets of dynamics (1 and 2):

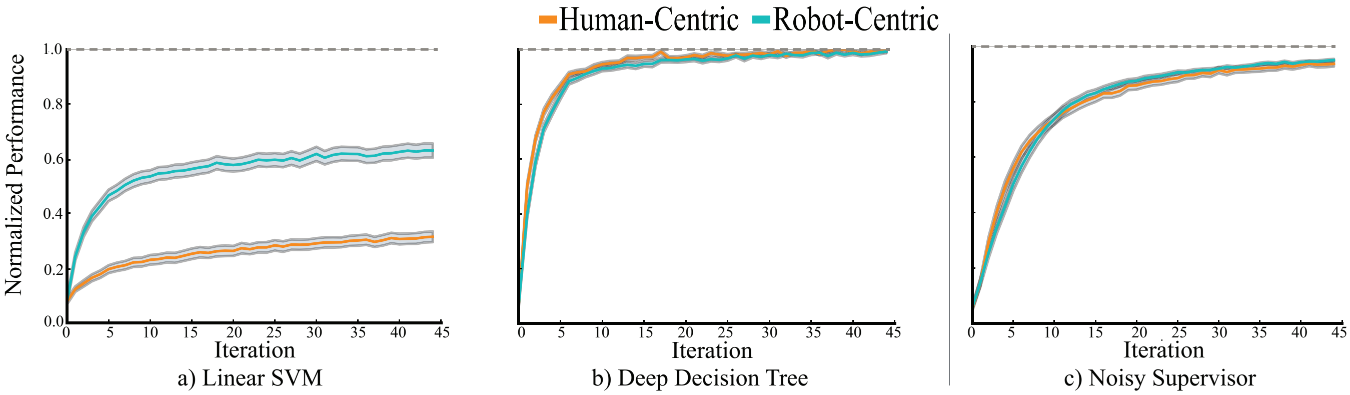
$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B_1\mathbf{u}_t + w$$

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B_2\mathbf{u}_t + w$$

where $w \sim \mathcal{N}(0, 0.1I)$. The state of the robot is $\mathbf{x} = (x, y, v_x, v_y)$ (i.e. the coordinates and the velocity). The control inputs to the robot are $\mathbf{u} = (f_x, f_y)$ (i.e. the forces in the coordinate direction). The matrix A is a 4×4 identity matrix plus the necessary two values to update the x, y state by the velocity with a timestep of 1. B_1 and B_2, correspond to a 4×2 matrix that updates only the velocity for each axis independently. This update corresponds to $\mathbf{v}_{t+1} = \mathbf{v}_t + \frac{1}{m}\mathbf{f}_t$, where m is the mass of the robot.

The dynamics for region 1 correspond to the point robot having a mass of $m = 1$ and in region 2 the point robot has a larger mass of $m = 4$. A target goal state lies at the origin $[0, 0]$ and the robot starts out at the point $[-15, -10]$ with a velocity of zero. The boundary for region 1 and 2 lies at $x = 12$ and $y = 12$, where region 2 lies $x > 12$ and $y > 12$.

[1] In the original DAgger the policy rollout was stochastically mixed with the supervisor, thus with probability β it would either take the supervisor's action or the robot's. The use of this stochastically mixed policy. However, Ross et al. recommend having humans provide feedback on slowed down videos of the robot executing its current policy [23]



a) Linear SVM

b) Deep Decision Tree

c) Noisy Supervisor

Fig. 2: We compare RC and HC LfD for low- and high-expressiveness policy classes (a and b respectively) over 100 randomly generated 2D gridworld environments, as a function of the amount of data provided to them. RC outperforms in the low-expressive condition, but the performance gap is negligible in the high-expressive condition, when the policy class contains the expected supervisor policy. We also examine the case of noisy supervisor labels (c), in which both techniques take more data to converge, but again perform similarly.

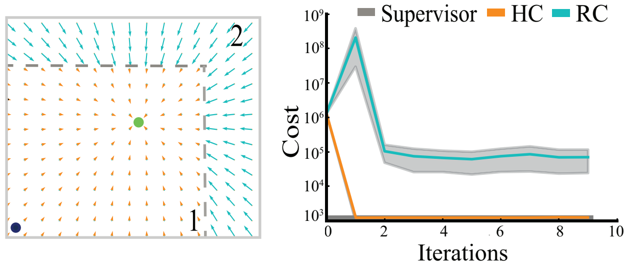


Fig. 3: Left: A 2D workspace where a point mass robot is taught to go to the green circle starting from the black circle. The world is divided into to two regions with different dynamics. The supervisor is computed via infinite horizon LQG for each region, which results in two different linear matrices in region 1 and 2. Right: RC fails to converges because it is attempting to learn a policy across the two regions, whereas HC remains in region 1 and converges to the supervisor performance.

An illustration is shown in Fig. 3. The time horizon for the task is $T = 35$.

The supervisor is a switching linear system, where each linear model is computed via the infinite horizon LQG for the specified dynamics. The robot gets feedback from the supervisor based on its current region. The robot's policy, π_θ, is a linear function which we find via least squares.

We run HC and RC in this setting and plot the performance in Fig. 3, averaged over 200 trials. HC is able to converge to the true supervisor's policy. However, the RC approach forces the robot to enter region 2. This prevents it from converging because it attempts to fit a single linear policy to two linear supervisors.

C. *Real-World Problem*

Next, we perform a pilot user study on a real robot to test performance in practice. Participants teach the robot to perform a singulation task (i.e., separate an object from its neighbors), illustrated in Fig. 4. A successful singulation means at least one object has its center located 10 cm or more from all other object centers.

We hypothesize that HC will match the supervisor more accurately than RC in practice, because: 1) RC sampling will cause the robot to try and learn more complex behavior, 2) participants will struggle with providing retroactive feedback.

The robot has a two-dimensional control space, $\mathcal{U}$, of base rotation and arm extension. The state space of the environment, $\mathcal{X}$, is captured by an overhead Logitech C270 camera, which is positioned to capture the workspace that contains all cluttered objects and the robot arm. The objects are red extruded polygons. We consider objects made of Medium Density Fiberboard with an average 4" diameter and 3" in height. We use a binary mask of the current image of the environment as the state representation, which captures positional information, but removes the background and is robust to lighting changes in the room. The policy is a deep neural network with the architecture from [15]. The network is trained using TensorFlow [1] on a Telsa K40 GPU.

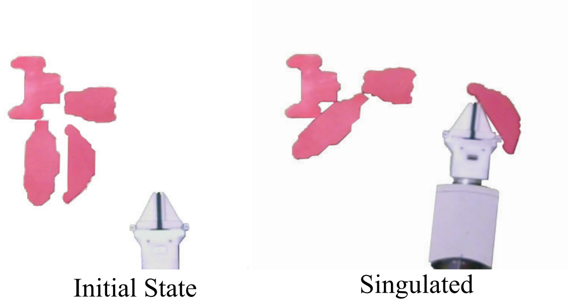


Fig. 4: Left: An example initial state the robot observes. The initial state can vary the relative position of the objects and pose of the pile. Right: A human is asked to singulate the object, which is to have the robot learn to push one object away from its neighbors. A successful singulation means at least one object has its center located 10 cm or more from all other object centers.

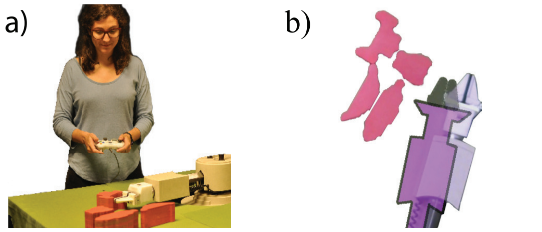


Fig. 5: Two ways to provide feedback to the robot. a)In HC sampling, the human teleoperates the robot and performs the desired task. For the singulation task, the human supervisor used an Xbox Controller. b) In RC sampling, the human observes a video of the robot's policy executing and applies retroactive feedback for what the robot should have done. In the image shown, the person is telling the robot to go backward towards the cluster.

The robot is moved via positional control implemented with PID. Similar to [16], the control space $\mathcal{U}$ consists of bounded changes in rotation and translation. The control signals for each degree of freedom are continuous values

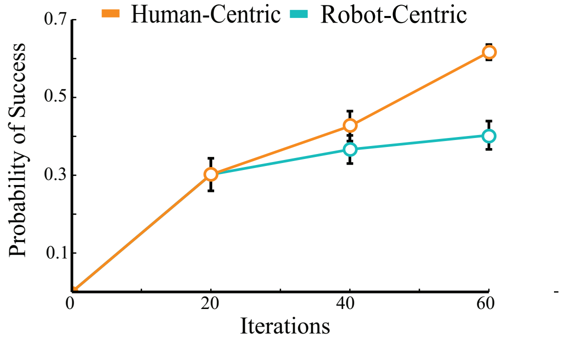


Fig. 6: Average success at the singulation task over the 10 human subjects as a function of number of demonstrations. Each policy is evaluated 30 times on the a held out set of test configurations. The first 20 rollouts are from the supervisor rolling out there policy and the next 40 are collected via retro-active feedback for RC and tele-operated demonstrations for HC. HC LfD shows a 20% improvement in their success at the end.

with the following ranges: base rotation, $[-1.5^\circ, 1.5^\circ]$, arm extension $[-1cm, 1cm]$.

During training and testing the initial state distribution $p(\mathbf{x}_0)$ consisted of sampling the translation of the cluster from a multivariate isotropic Gaussian with variance of 20cm and the rotation was selected uniformly from the range $[-15^\circ, 15^\circ]$. The relative position of the 4 objects is chosen randomly. To help a human operator place objects in the correct pose, we used a virtual overlay over the webcam.

We selected 10 UC Berkeley students as human subjects. The subjects were familiar with robotics, but not the learning algorithms behind the techniques. They first watched a trained robot perform the task successfully. They then practiced providing feedback through RC sampling for 5 demonstrations and HC sampling for 5 demonstrations. Next, each subject performed the first 20 demonstrations via HC sampling. They then performed 40 HC demonstrations and 40 RC demonstrations in a counter-balanced order. In RC, we chose $K = 2$ iterations of 20 demonstrations each. The experiment took 2 hours per person on average.

In HC, we asked participants to provide 60 demonstrations to the robot using an Xbox Controller, as shown in Fig. 5a. In RC, participants provided 20 initial demonstrations via the Xbox Controller, and then provided retroactive feedback for $K = 2$ iterations of 20 demonstrations each.

Retroactive feedback is provided through a labeling interface similar to our previous work [15], illustrated in Fig 5b. In this interface, we showed a video at half speed of the robot's rollout to the participant. They then use a mouse to provide feedback in the form of translation and rotation. We provide a virtual overlay so that they can see the magnitude of their given control. A video that illustrates this setup and the different sampling approaches can be found at `https://berkeleyautomation.github.io/lfd_icra2017/`.

In Fig. 6 , we show the average performance of the policies trained with RC and HC LfD. Each policy is evaluated on a holdout set of 30 initial states sampled from the same distribution as training. The policies learned with RC have approximately 40% probability of success versus 60% for HC. This suggests that HC may outperform RC when supervision is provided by actual human demonstrations.

	Surrogate Loss on Test Set	
Algorithm Type	**Translation (mm)**	**Rotation (rad)**
HC LfD	2.1 ± 0.2	0.009 ± 0.001
RC LfD	3.4 ± 1.0	0.014 ± 0.003

TABLE I: The average surrogate loss on a held out set of 10 demonstrations from the the total 60 demonstrations collected for each 10 participants. The confidence intervals are standard error on the mean, which suggest that RC LfD obtains a higher surrogate loss in both degrees of freedom, forward and rotation.

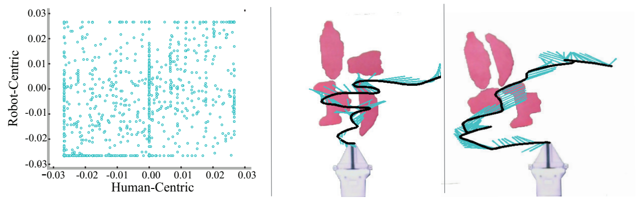


Fig. 7: Results from the post analysis examining how well retroactive feedback matched teleoperation. The scatter plot shows the normalized angle of the control applied for both HC (teleoperation) and RC (retroactive). The large dispersion in the graph indicates that the five participants had a difficult time matching their retroactive and teleoperated controls. Two example trajectories are also shown. The black line indicates the path from teleoperation and the teal line is the direction and scaled magnitude of the feedback given. If they matched perfectly, the teal line would be tangent to the path.

D. *Understanding RC Performance in Practice*

To better understand why RC performed worse than HC, we looked for the two causes in our hypothesis: policy complexity, and human supervision difficulty.

Policy Complexity: We first analyzed the complexity of behaviors collected with RC sampling. In Fig. 1c, we show trajectories collected on a single initial state during the study. As illustrated, HC sampling is concentrated around a path needed to singulate the bottom right object. However, RC sampling places the robot in a wide variety of states, some of which would require the robot to learn how to move backward or rapidly change direction to recover.

To better analyze this, we examined the surrogate loss on a test set of 10 randomly selected trajectories from each supervisor's dataset. As shown in Table 1, we observed the average test error over the policies trained with 60 demonstrations in both degrees of freedom (i.e., translation and rotation) is significantly higher. This indicates that the RC policies had a harder time generalizing to unseen labels in their aggregate dataset on average, which may be due to the complexity of the corrective actions.

Human Supervision. We next hypothesized that the participants could have had trouble providing retroactive feedback that is consistent with their true policy. To test this, we asked 5 of the participants to provide 5 demonstrations via teleoperation. Then we asked them to match their controls via the RC labeling interface. If the person has a hard time matching their own control on a trajectory they executed, it could imply they would also have a challenging time correcting a robot's policy on a robot's trajectory.

We measured the correlation between the controls applied via retroactive feedback and teleoperation. When calculated over all the participants and trajectories, the Pearson Correlation Coefficient in rotation and translation was 0.60 and 0.22, respectively. A smaller correlation coefficient suggests that it is harder for people to match their control in teleoperation.

In Fig. 7 we plot the control angle from HC sampling versus the control angle from RC sampling, showing that the two are not correlated. We also show two trajectories that a participant teleoperated with their retroactive labels overlayed, both of which suggest disagreement between the teleoperated and retroactive controls. Overall, our analysis suggests that RC sampling can lose the intent of the human supervisor.

V. Theoretical Analysis

To understand the empirical results from above, we develop a theoretical model of RC and HC LfD. In this section, we first show the existence of environments where HC converges to the supervisor but RC may not. Then we present a new analysis of the accumulation of error for HC .

A. Algorithm Consistency

The analysis of RC LfD in [22] is performed in the context of the regret framework of online learning. In online learning [25], at each iteration $k \in \{1, \dots, N\}$ the algorithm chooses an action $\theta_k \in \Theta$ (e.g. a policy to roll out), and observes loss $f_k(\theta_k)$. One can derive an upper bound on the cumulative loss incurred by the algorithm relative to taking just the best single action in Θ:

$$\sup_{\theta \in \Theta} \sum_{k=1}^{N} f_k(\theta_k) - f_k(\theta).$$

In the context of robotics and specifically in the case of RC LfD, taking an action θ_k at iteration k is rolling out a policy dictated by θ_k which induces a series of states $\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,T-1}$ (i.e. a trajectory with time horizon T) and the loss is evaluated with respect to *these* states. Because these particular states could have been produced by a poor policy due to initialization, they could have little relevance to the task of interest and consequently, it may not be helpful to compare to the policy that performs best on these particular states. What is important is the absolute performance of a policy on the task of interest. In the notation of Ross et al. [22] this notion of relative regret is encoded in the ϵ_N error term that appears in the bounds and is defined relative to the particular set of rollouts observed by the RC LfD policy.

As an example for why low-regret may be uninformative, consider a car-racing game where the car has constant speed and the policy only decides to go straight, left, or right at the current state. Suppose at some point in the race the car encounters a “fork” in the road where one path is a well-paved road and the other path is a muddy dirt road with obstacles. The car will finish the race faster by taking the paved road, but due to either the stochastic dynamics or imperfection of the supervisor it is possible that after just a small number of supervisor demonstrations given to the robot to initialize RC LfD, a poor initial policy will be learned that leads the car down the dirt road versus the paved road. When this policy is rolled out the supervisor will penalize the decision at the time point of taking the dirt versus paved road. But if the policy’s actions agree with the supervisor once the car is on the dirt road (i.e., making the best of a bad situation) this policy will incur low-regret because the majority of the time the policy was acting in accordance with the supervisor. Thus, in this example a globally bad policy will be learned (because it took the dirt road instead of the paved road) but relative to the best policy acting on the dirt road, it performs pretty well.

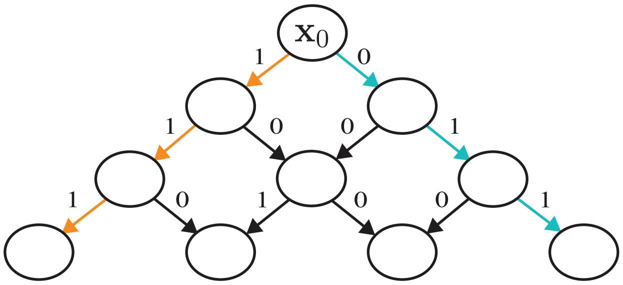


Fig. 8: A Directed Acyclic Graph, where a robot is being taught by a supervisor to descend down and achieve maximum cumulative reward, which is shown via the numbers on each branch. Each node has the action the supervisor would select, either left or right, $\{L, R\}$. The HC method converges to the Orange path, which is optimal. While the RC method may converge to the Teal path due to process noise, because it tries to learn on examples from that side of the tree.

In effect, RC LfD is performing a greedy strategy for optimization and has the potential to get stuck in local minima. The next theorem shows a class of examples in which an initial policy can be different enough from the optimal policy that the states visited by the initial policy – even with corrective feedback from the supervisor – are not informative or relevant enough to the task to guide RC LfD to the optimal policy. We show this for the DAgger algorithm with $\beta = 0$ for RC sampling, however a similar statement can be made with stochastic mixing.

Theorem 5.1: For a given MDP environment (state space, action space, dynamics mapping state-action pairs to states, and a loss function) and policy class Θ, let θ_{HC}^N be the policy learned from N supervisor demonstrations, and let θ_{RC}^N be the policy learned by the RC procedure described in Section III that is initialized with m supervisor demonstrations. Then there exists an environment and policy class Θ such that

- θ^* is the unique minimizer of $E_{p(\tau|\theta)}[J(\theta)]$,
- $\lim_{N\to\infty} \theta_{HC}^N = \theta^*$ with probability 1,
- $\lim_{N\to\infty} \theta_{RC}^N \neq \theta^*$ with probability $\geq ce^{-m}$

for some universal constant c. In other words, even with infinite data RC may converge to a suboptimal policy while HC converges to the best policy in the class.

Proof: Let $\{\{(\mathbf{x}_{k,t}, \mathbf{u}_{k,t}\}_{t=0}^3\}_{k=1}^N$ denote N supervisor trajectories. Consider an environment with a deterministic initial state at the root of the DAG of Figure 8 so that $p(\mathbf{x}_{k,0} = \texttt{root}) = 1$ for all k. The space of policies are constant functions $\Theta = \{L, R\}$ where if $\theta = L$ then regardless of the state, the control input $\mathbf{u}_{k,t}$ will be to take the left child (and analogously for $\theta = R$ taking the right child). For any state in the DAG with children, let $\phi(\mathbf{x}_{k,t}, \theta)$ denote the left child of $\mathbf{x}_{k,t}$ if $\theta = L$ and the right child otherwise. The dynamics are described as follows: for some $\mu \in (0, 1/4]$ to be defined later, if $\theta = L$ and $\mathbf{x}_{k,t} = \texttt{root}$ then $p(\mathbf{x}_{k,1} = \phi(\mathbf{x}_{k,0}, R)) = \mu$ and $p(\mathbf{x}_{k,1} = \phi(\mathbf{x}_{k,0}, L)) = 1 - \mu$, but if $\theta = R$ then the right child is chosen with probability 1. If $\mathbf{x}_{k,t} \neq \texttt{root}$ then $\mathbf{x}_{k,t+1} = \phi(\mathbf{x}_{k,t}, \theta)$.

Assume that the supervisor $\pi^* : \mathcal{X} \to \{L, R\}$ acts greedily according to the given rewards given in the DAG so that $\pi^*(\mathbf{x}_{k,t}) = L$ if the reward of the left child exceeds the right child and $\pi^*(\mathbf{x}_{k,t}) = R$ otherwise. Finally, define the state loss function $\ell(\cdot,\cdot)$ as the $0/1$ loss so that after N trajectories, the loss is given as $J_N(\theta) = \sum_{k=1}^N \sum_{t=0}^2 \mathbf{1}\{\pi^*(\mathbf{x}_{k,t}) \neq \theta\}$ for all $\theta \in \{L,R\}$. Note that $\widehat{\theta}_N = \arg\min_{\theta\in\{L,R\}} J_N(\theta)$ is equivalent to looking at all actions by the supervisor over the states and taking the majority vote of L versus R (i.e., the states in which these actions are taken has no impact on the minimizer). Note that we have not yet specified how the states $\mathbf{x}_{k,t}$ were generated.

We can compute the true loss when the trajectories are generated by $\theta \in \{L,R\}$. Let the empirical distribution of observed states under a fixed action $\theta \in \{L,R\}$ be given by $p(\tau \mid \theta)$, then

$$\begin{aligned} E_{p(\tau|\theta=L)} J(\theta = L) &= p(\mathbf{x}_{k,1} = \phi(\mathbf{x}_{k,0}, L)) \cdot 0 + \\ &\quad p(\mathbf{x}_{k,1} = \phi(\mathbf{x}_{k,0}, R)) \cdot 2 = 2\mu \\ E_{p(\tau|\theta=R)} J(\theta = R) &= 1 \end{aligned}$$

which implies that $\theta^* = L$ and performs strictly better than R whenever $\mu < 1/2$.

It follows from the stochastic dynamics that the demonstrated action sequence by the supervisor equals $\{L,R,R\}$ with probability μ and $\{L,L,L\}$ with probability $1-\mu$. After m supervisor sequences, the expected number of L actions is equal to $\mu + 3(1-\mu) = 3 - 2\mu$ while the expected number of R actions is equal to just 2μ. By the law of large numbers, if only supervisor sequences are given then $\arg\min_{\theta\in\{L,R\}} J_m(\theta) \to L = \theta^*$ as $m \to \infty$ since $3 - 2\mu > 2\mu$ for all $\mu < 1/2$. In other words, $\theta^N_{HC} \to \theta^*$ as $N \to \infty$.

We now turn our attention to RC's policy. Note that if after m supervisor action sequences the number of observed R's exceeds the number of observed L's, then RC policy will define $\theta^m_{RC} = R$. It is easy to see that a policy rolled out with $\theta = R$ will receive the supervisor's action sequence $\{L,R,R\}$ and thus R will remain the majority vote and consequently $\theta^N_{RC} = R$ for all $N \geq m$. What remains is to lower bound the probability that given m supervisor demonstrations that $\theta^m_{RC} = R$.

For $k = 1,\ldots,m$ let $Z_k \in \{0,1\}$ be independent Bernoulli(μ) random variables where $Z_k = 1$ represents observing the supervisor sequence $\{L,R,R\}$ and $Z_k = 0$ represents observing $\{L,L,L\}$. Given m supervisor sequences, note that the event $\arg\min_{\theta\in\{L,R\}} J_m(\theta) = R$ occurs if $\frac{1}{m}\sum_{k=1}^m Z_k > 3/4$. Noting that $\sum_{k=1}^m Z_k$ is a binomial(m,μ) random variable, the probability of this event is equal to

$$\sum_{k=\lfloor 3m/4\rfloor+1}^{m} \binom{m}{k} \mu^k (1-\mu)^{m-k} \geq \Phi\left(\frac{\lfloor 3m/4\rfloor + 1 - \mu m}{\sqrt{m\mu(1-\mu)}}\right)$$

where we have used Slud's inequality [26] to lower bound a binomial tail by a Gaussian tail. Setting $\mu = 1/4$ we can further lower bound this probability by $\Phi\left(\frac{m+2}{\sqrt{3m/4}}\right) \geq ce^{-m}$ for some universal constant c. Consequently, with probability at least ce^{-m} we have that $\lim_{N\to\infty} \theta^N_{RC} = R \neq \theta^*$. ■

We note there exists techniques to correct for this problem. One approach is to consider the value of each action and select actions that lead to higher reward during roll-out [**?**]. However, this assumes the robot's has access to a reward function, which may be challenging if the reward function is very delayed and uninformative. Another solution is to increase the model expressiveness of the robot's policy class. However, this could require more data than HC methods [27].

B. Bound on Error for HC LfD

In the HC LfD setting a robot is trained on the states visited by the supervisor. However, at run time the robot may encounter a different distribution of states due to not perfectly matching the supervisor's policy. Ross et al. showed that given a time horizon, T, the worst case error scales quadratically (i.e., $O(T^2 E_{p(\mathbf{x}|\theta^*)} l(\theta^N))$ when executing the robot's policy [21]. Note that according to the notation of Ross et al., $TE_{p(\mathbf{x}|\theta^*)} l(\theta^N) = E_{p(\tau|\theta^*)} J(\theta^N, \tau)$. We present a new analysis for a class of stochastic policies that yields a rate of $O(T\sqrt{TE_{p(\mathbf{x}|\theta^*)} l(\theta^N)})$.

Define the surrogate loss as the squared euclidean norm, or $l(\pi_\theta(\mathbf{x}), \pi_{\theta^*}(\mathbf{x})) = ||\pi_\theta(\mathbf{x}_{i,t}) - \pi_{\theta^*}(\mathbf{x}_{i,t})||_2^2$. We assume that the controls are bounded and thus can be normalized such that the $l \in [0,1]$. We are interested in the situation where the supervisor and robot policy are stochastic with a Normal Distribution (i.e. $p(\mathbf{u}|\pi_\theta(\mathbf{x})) = \mathcal{N}(\pi_\theta(\mathbf{x}), \sigma I)$. We note these assumptions imply $\pi_\theta(\mathbf{x})$ and $\pi_{\theta^*}(\mathbf{x})$ exist in a bounded controls space, but the stochastic policies may have support outside of this region. The interest in stochastic policies instead of deterministic is two-fold: 1) human supervisor can be noisy in nature and 2) due to noise in robot execution, such as cable coupling, the intended and actual control may differ in a stochastic way [19]. To be concise we refer to $J(\theta,\tau)$ as $J(\theta)$, dropping the reference to a trajectory.

Theorem 5.2: Given a policy π_{θ^N}, the following inequalities holds

$$E_{p(\tau|\theta^n)} J(\theta^N) \leq T\sqrt{\frac{1}{4\sigma} E_{p(\tau|\theta^*)} J(\theta^N)} + E_{p(\tau|\theta^*)} J(\theta^N)$$

Proof: For convenience we write $E_{p(\tau|\theta)} = E_\theta$ and $l(\theta,\mathbf{x}) = l(\theta)$. The proof follows by first deriving an upper bound on the worst case difference between the two quantities $E_{\theta^N} J(\theta^N) - E_{\theta^*} J(\theta^N)$. Then we leverage the intuition that if one is minimizing $E_{\theta^*} J(\theta^N)$, they are also decreasing the distance between the robot and supervisor's distributions.

$$\begin{aligned} & E_{\theta^N} J(\theta^N) - E_{\theta^*} J(\theta^N) && (3) \\ & = T(\frac{1}{T} E_{\theta^N} J(\theta^N) - \frac{1}{T} E_{\theta^*} J(\theta^N)) && (4) \\ & \leq T||p(\tau|\theta^N) - p(\tau|\theta^*)||_{TV} && (5) \\ & \leq T\sqrt{\frac{1}{2} D_{KL}(p(\tau|\theta^*), p(\tau|\theta^N))} && (6) \end{aligned}$$

Line 5 leverages the fact that the worst case loss is bounded by 1 and the definition of Total Variational distance. Line 6 uses Pinsker's inequality [29].

$$= T\sqrt{\frac{1}{2}E_{p(\theta^*)}\log\frac{p(\tau|\theta^*)}{p(\tau|\theta^N)}} \tag{7}$$

$$= T\sqrt{\frac{1}{2}E_{p(\theta^*)}\sum_{t=1}^{T}\log\frac{p(\mathbf{u}_t|\mathbf{x}_t,\theta^*)}{p(\mathbf{u}_t|\mathbf{x}_t,\theta^N)}} \tag{8}$$

$$= T\sqrt{\frac{1}{4\sigma}E_{p(\theta^*)}\sum_{t=1}^{T}||\mathbf{u}_t-\pi_{\theta^N}(\mathbf{x}_t)||_2^2 - ||\mathbf{u}_t-\pi_{\theta^*}(\mathbf{x}_t)||_2^2} \tag{9}$$

$$\leq T\sqrt{\frac{1}{4\sigma}E_{p(\theta^*)}\sum_{t=1}^{T}||\pi_{\theta^*}(\mathbf{x}_t)-\pi_{\theta^N}(\mathbf{x}_t)||_2^2} \tag{10}$$

$$= T\sqrt{\frac{1}{4\sigma}E_{p(\theta^*)}J(\theta^N)} \tag{11}$$

$$= T\sqrt{T\frac{1}{4\sigma}E_{p(\mathbf{x}|\theta^*)}l(\theta^N)} \tag{12}$$

Line 7,8 and 9 apply the definition of the KL-divergence, the markov chain and the normal distribution over $p(\mathbf{u}_t|\mathbf{x}_t,\theta)$. Line 10 applies the triangle inequality to upperbound by the defined surrogate loss. Line 11 applies the assumed definition of $J(\theta)$. Line 12 uses the notation of Ross et al. [22].

The intuition is that difference between the two distributions can be controlled via the surrogate loss on the expected supervisor. This confirms that the closer the robot's policy matches the supervisor's policy on the supervisor's distribution, the smaller the total variational difference between the resulting two distributions will be. ■

The above analysis demonstrates how the constant T affects the bound in error. However, it is important to note that this is not the only variable that plays a role in performance. The size of the function class and number of demonstrations needed are also important in determining how large $E_{p(\mathbf{x}|\theta^*)}l(\theta^N)$ is.

VI. Discussion and Future Work

Motivated by recent advances in Deep Learning from Demonstrations for robot control, this paper reconsiders HC and RC sampling in the context of human supervision, finding that policies learned with HC sampling perform equally well or better with classes of highly-expressive learning models and can avoid some of the drawbacks of RC sampling. We further provide new theoretical contributions on the performance of both RC and HC methods.

It is important to note that there are challenges to using highly- expressive policies. When training complex models such as Recurrent Neural Networks or Differentiable Neural Computers, it may be difficult to achieve low training error and thus RC methods can lead to better performance [7], [10]. Furthermore, poor selection of feature representation or state space could result in violation of the markovian assumption and incur large training error.

Lastly, an increase in demonstrations is needed for more expressive function classes. Thus, we recommend not necessarily using the largest function class possible, but instead performing structured risk minimization [27] to determine what size best represents the supervisor's policy. Understanding how much data is needed to learn a function, is known as sample complexity analysis [4], [6], [13], [28]. In future work, we will explore these effects and how sample complexity grows with variance in the initial state and dynamics distributions.

VII. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the UC Berkeley's Algorithms, Machines, and People Lab, BAIR, BDD, and the CITRIS "People and Robots" (CPAR) Initiative: `http://robotics.citris-uc.org`. KJ is supported by ONR awards N00014-15-1-2620 and N00014-13-1-0129. JM is the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. We acknowledged David Gealy and Menglong Guo for their help with the hardware setup . We also thank Florian Pokorny, Drew Bagnell and Roy Fox for valuable insight.

References

[1] "Tensor flow," https://www.tensorflow.org/.

[2] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[3] B. Akgun, K. Subramanian, and A. L. Thomaz, "Novel interaction strategies for learning from teleoperation." in *AAAI Fall Symposium: Robots Learning Interactively from Human Teachers*, vol. 12, 2012, p. 07.

[4] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[5] 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.

[6] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 463–482, 2002.

[7] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[8] R. Dillmann, M. Kaiser, and A. Ude, "Acquisition of elementary robot skills from human demonstration," in *International symposium on intelligent robotics systems*. Citeseer, 1995, pp. 185–192.

[9] F. Duvallet, T. Kollar, and A. Stentz, "Imitation learning for natural language direction following through unknown environments," in *ICRA*. IEEE, 2013, pp. 1047–1053.

[10] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[11] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *NIPS*, 2014, pp. 3338–3346.

[12] H. He, J. Eisner, and H. Daume, "Imitation learning by coaching," in *Advances in Neural Information Processing Systems*, 2012, pp. 3149–3157.

[13] S. M. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Advances in neural information processing systems*, 2009, pp. 793–800.

[14] B. Kim and J. Pineau, "Maximum mean discrepancy imitation learning." in *Robotics Science and Systems*, 2013.

[15] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE).*, 2016.

[16] M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 462–469.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[18] S. Levine and V. Koltun, "Variational policy search via trajectory optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 207–215.

[19] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel *et al.*, "Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2014, pp. 532–539.

[20] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," Carnegie-Mellon University, Tech. Rep., 1989.

[21] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 661–668.

[22] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2010.

[23] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *ICRA, 2013 IEEE*. IEEE.

[24] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research*. Springer, 2016, pp. 339–354.

[25] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[26] E. V. Slud, "Distribution inequalities for the binomial law," *Ann. Probab.*, vol. 5, no. 3, pp. 404–412, 06 1977. [Online]. Available: http://dx.doi.org/10.1214/aop/1176995801

[27] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in Neural Information Processing Systems*, 1992, pp. 831–838.

[28] ——, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[29] S. Verdú, "Total variation distance and the distribution of relative information." in *ITA*. Citeseer, 2014, pp. 1–3.

[30] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.