Learning User Models During Shared Autonomy

Shervin Javdani, J. Andrew Bagnell, Siddhartha S. Srinivasa
Robotics Institute, Carnegie Mellon University
{sjavdani, dbagnell, siddh}@cs.cmu.edu

***Abstract*—In *shared autonomy*, user input and robot autonomy are combined to control a robot to achieve a goal. One often used strategy considers the user and autonomous system as *independent* decision makers, combining the output of these two entities. However, recent work has shown that this can lead to poor performance, suggesting the need to incorporate user models into the autonomous decision maker. A key challenge for doing so is learning a good user model. Existing methods for learning user models assume autonomous assistance is not present, while prior work suggests that users behave differently in the presence of assistance. In this work, we propose a method for learning user behavior during shared autonomy. We present a cost minimization framework that utilizes this learned model to select assistance actions that minimize the cost incurred by the user. Finally, as altering the assistance strategy can change how the user behaves, we present a method for evolving the model and assistance strategy together.**

I. Introduction

Robotic teleoperation enables a user to achieve their goal by providing inputs into a robotic system. In *direct teleoperation*, user inputs are mapped directly to robot actions, putting the burden of control entirely on the user. However, input interfaces are noisy, and often have fewer degrees of freedom than the robot they control. This makes operation tedious, and many goals impossible to achieve. *Shared autonomy* seeks to alleviate these issues by combining manual teleoperation with autonomous assistance.

Dragan and Srinivasa [1] show that nearly all prior shared autonomy methods can be thought of as a *blending* between two independent sources - user inputs and an autonomous policy. Methods differ in the autonomous policy used, and how the blending is done. Potential field methods [2] combine direct teleoperation with a policy which pushes the robot away from obstacles and towards goals. Virtual fixtures [3, 4] combine direct teleoperation with an autonomous policy that projects the robot onto path constraints. Autonomous task completing systems [5, 6] use goal achieving policies, and give full control to either the user or autonomy. Linear blending [1] uses goal achieving policies, and blend by utilizing goal prediction confidence to smoothly switch between sources.

While blending is intuitive, Trautman [7] presents many examples where user inputs and the autonomous policy are reasonable in isolation, but their blended combination is not. These examples demonstrate a key shortcoming of the blending approach - the autonomous policy selects assistance actions *independent* of user inputs.

Although the autonomous policy itself does not usually consider user inputs, user models have been used to predict the user's goal and select an autonomous policy [4, 1, 8, 9].

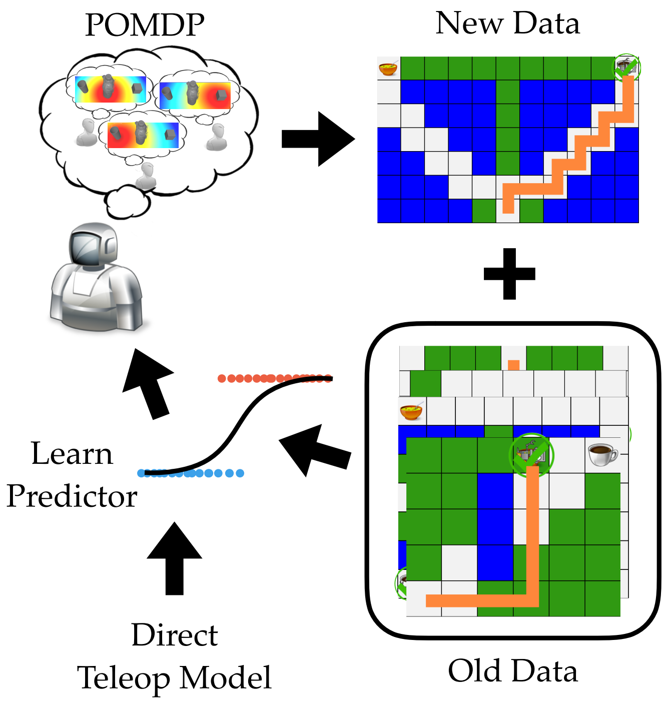

Fig. 1: Our algorithm pipeline. Given the current predictor, we utilize a POMDP based policy to select assistance actions under goal uncertainty. Once the shared autonomy system achieves the user goal, we add this data to all previous shared autonomy data. We use this entire dataset, along with a prior model of user behavior during direct teleoperation, to learn a new predictor. This induces a different POMDP based assistance policy, which we use at the next iteration. We repeat this process.

However, these models are generally learned from direct teleoperation, whereas experiments suggest that users change their behavior during shared autonomy [10, 11].

Some works have proposed minimizing a *user-robot cost function* [8, 10]. However, current implementations do not incorporate predictions of future user inputs into the policies, and do not model how user behavior changes when assisting.

More recently, Nikolaidis et al. [11] present a framework for minimizing cost while modelling how likely that user is to *adapt* to the autonomous assistance. In their framework, the user can either oppose or agree with the robot, where adaption corresponds to being more likely to agree. In contrast to our work, they assume a discrete set of joint user-robot policies.

Our aim is predict how users respond to assistance, and utilize these predictions to select assistance actions. If certain assistance actions cause users to incur greater cost, e.g. if they fight the system, we predict that incurred cost and penalize those actions. If other actions cause users to achieve their goal while incurring less cost, we prefer those actions.

We first review the cost minimization framework of Javdani et al. [10] assuming we have a model of user behavior. We next present a method for learning user behavior during assistance. Intuitively, we believe a model of user behavior during direct teleoperation serves as a good prior, and incorporate this notion into our method for learning a user model during shared autonomy. We show how this model can be used for selecting assistance actions. We discuss how to account for altering the assistance strategy in learning a user model during assistance. Finally, we discuss proposed experiments for testing this idea in a discrete shared autonomy game.

II. Assistance Action Selection

We present our user-robot cost minimization framework for a known goal. Note that we can extend this cost minimization to an unknown goal by following Javdani et al. [10] and using the QMDP method [12].

A. User-Robot Cost Minimization

Formally, let $s \in S$ be the robot state (e.g. position, velocity), and $a \in A$ be the actions (e.g. velocity, torque). We model the robot as a dynamical system with transition function $T : S \times A \rightarrow S$. The user supplies inputs $u \in U$ via an interface (e.g. joystick, mouse). These user inputs map to robot actions through a known deterministic function $\mathcal{D} : U \rightarrow A$, corresponding to the effect of *direct teleoperation*. We define a trajectory ξ_t as a sequence of states, user inputs, and actions, $\xi_t = \{s_0, u_0, a_0, \dots, u_{t-1}, a_{t-1}, s_t\}$.

We assume a known user-robot cost function $C : S \times U \times A \rightarrow \mathcal{R}$. This cost function can be hand-tuned or learned, e.g. through maximum entropy inverse optimal control (MaxEnt IOC) [13]. Together, the tuple (S, U, T, C^{u}) defines a Markov Decision Process (MDP).

Unlike standard MDP formulations, we cannot directly optimize for actions, as we do not decide user inputs. To select robot actions, we assume access to a stochastic user policy $\pi^{\text{u}}(\xi_t) = p(u_t|\xi_t)$, which provides a distribution over user inputs given the history of states, user inputs, and actions. In Sec. III, we discuss how we can learn this policy.

We similarly define the assistance policy $\pi^{\text{r}}(s) = p(a|s)$. This allows us to define the *value function*, or expected cost-to-go, given a user policy:

$$\begin{aligned} V^{\pi^{\text{u}}}_{\pi^{\text{r}}}(s_0) &= \sum_t \mathbb{E}_{s_t, u_t, a_t}[C(s_t, u_t, a_t)] \\ s_t &\sim T(s_{t-1}, a_{t-1}) \\ u_t &\sim \pi^{\text{u}}(\xi_t) \\ a_t &\sim \pi^{\text{r}}(s_t) \end{aligned}$$

We denote the optimal value function as the expected cost-to-go of the best policy, $V^{\pi^{\text{u}}}(s) = \min_{\pi^{\text{r}}} V^{\pi^{\text{u}}}_{\pi^{\text{r}}}(s)$.

Note that in prior work [10], this value function was approximated by assuming the user would not supply more inputs:

$$V^{\pi^{\text{u}}}_{[10]}(s_0) = \min_{\pi^{\text{r}}} \sum_t \mathbb{E}_{s_t, a_t}[C(s_t, 0, a_t)]$$

This assumption was made for computational purposes - rolling out the user policy while selecting assistance actions is computationally difficult. However, if we wish to incorporate the user model into action selection - for example, to avoid fighting the user - we must relax this assumption.

Instead, we approximate by rolling out our policy and predictor for a short horizon, and utilize a heuristic thereafter:

$$V^{\pi^{\text{u}}}(s_0) \approx \min_{\pi^{\text{r}}} \sum_t^T \mathbb{E}_{s_t, u_t, a_t}[C(s_t, u_t, a_t)] + \widetilde{V}(s_T)$$

Where $\widetilde{V}$ is some estimate of the cost-to-go, e.g. assuming the robot will take over.

III. Learning the User Policy

Most shared autonomy works have focused on utilizing predictors to infer a distribution over the user's goal [4, 1, 8, 9]. These methods learn a predictor during *direct teleoperation*, and apply it during shared autonomy [1, 8, 9, 10]. Implicitly, this assumes that users do not change their behavior when assistance is provided. However, recent studies suggest that users alter their behavior during assistance [10, 11].

A. Learning User Adaptation

The way in which users respond to assistance actions provides information about the effectiveness of those actions. If the user fights certain assistance actions, we should actively avoid those assistance actions. On the other hand, if we can predict that some assistance actions will enable users to achieve their goal while incurring less cost, we should prefer those actions.

Intuitively, we believe that user behavior during assistance will resemble that of direct teleoperation. Let p^{me} be a predictor of direct teleoperation behavior, e.g. learned through maximum entropy inverse optimal control (MaxEnt IOC) [13]. To learn a predictor with assistance, we employ the principle of *minimum cross-entropy* [14], learning a predictor that matches the observed data while minimizing the Kullback-Leibler (KL) divergence to this prior distribution. This has the additional benefit of leveraging existing work on user predictions during direct teleoperation.

Let f^{ξ}_u be some features of user input u and trajectory so far ξ. Let $\overline{f}^{\xi}_u$ be the average feature observed in the data:

$$\begin{aligned} \arg\min_{p^{\text{kl}}} \ & \text{KL}(p^{\text{kl}} \| p^{\text{me}}) \\ \text{s.t.} \sum_{\xi \in \text{Data}} & p(\xi) \sum_u p^{\text{kl}}(u|\xi) f^{\xi}_u = \overline{f}^{\xi}_u \end{aligned}$$

That is, the average feature of the data $\overline{f}^{\xi}_u$ should match the expected feature predicted by our learned distribution p^{kl} on the trajectories observed in the data.

For computational purposes, we follow Nikolaidis et al. [11] and utilize a bounded memory model, incorporating features of only a short history.

B. Iterating Learning and Policy Updates

The above learning problem assumes that that the training and testing distributions are iid - that is, the histories $\xi \in$ Data is the same as the histories we will see during testing. However, updating our model of the user causes our shared autonomy policy to change, and therefore the histories to be different, violating this assumption.

This common problem in reinforcement learning is addressed by Ross et. all [15, 16] with the DAgger algorithm. The solution is intuitively simple - iteratively update your policy, get a new set of data with the current policy, and train the predictor with all data, including data from previous policies. See fig. 1. In addition to enabling stronger theoretical guarantees in this setting, this has the empirical benefit of continuously adapting to the user's behavior during assistance.

IV. Planned Experiments

We have implemented these methods for a discrete grid-world scenario with modal control [17]. Briefly, modal control addresses the problem of controlling high degree of freedom systems with lower degree of freedom inputs by defining a discrete set of control modes, each of which controls a subset of the robot degrees of freedom. See fig. 2. We are currently preparing testing of this system for mechanical turk.

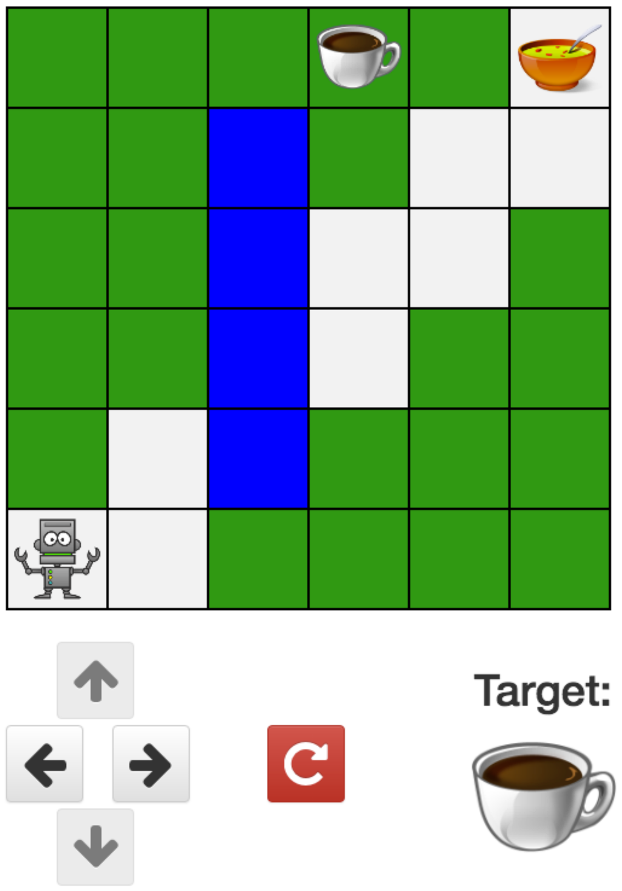

Fig. 2: Our proposed experiment for modal control. Users must navigate the robot to the specified goal, which the system does not know apriori. The grid includes fast-moving squares (white), slow-moving squares (green), and walls (blue). Users have two control modes: left-right and up-down, and can switch modes by rotating the robot. In order to assist the user, the system can automatically switch modes.

References

[1] A. Dragan and S. Srinivasa, "A policy blending formalism for shared control," *IJRR*, 2013.

[2] J. W. Crandall and M. A. Goodrich, "Characterizing efficiency on human robot interaction: a case study of shared–control teleoperation," in *IEEE/RSJ IROS*, 2002.

[3] S. Park, R. D. Howe, and D. F. Torchiana, "Virtual fixtures for robotic cardiac surgery," in *Med. Image. Comput. Comput. Assist. Interv.*, 2001.

[4] M. Li and A. M. Okamura, "Recognition of operator motions for real-time assistance using virtual fixtures," in *HAPTICS*, 2003.

[5] A. H. Fagg, M. Rosenstein, R. Platt, and R. A. Grupen, "Extracting user intent in mixed initiative teleoperator control," in *AIAA*, 2004.

[6] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transa. Ind. Electron.*, 2005.

[7] P. Trautman, "Assistive planning in complex, dynamic environments: a probabilistic approach," in *HRI Workshop Hum. Rob. Team.*, 2015.

[8] K. K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Auton. Robots*, vol. 35, 2013.

[9] H. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," in *RSS*, 2013.

[10] S. Javdani, S. Srinivasa, and J. A. D. Bagnell, "Shared autonomy via hindsight optimization," in *RSS*, 2015.

[11] S. Nikolaidis, A. Kuznetsov, D. Hsu, and S. Srinivasa, "Formalizing human-robot mutual adaptation via a bounded memory based model," in *ACM/IEEE HRI*, 2016.

[12] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *ICML*, 1995.

[13] B. D. Ziebart, A. Maas, J. A. D. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[14] J. E. Shore and R. W. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy." *IEEE Trans. Info. Theory*, vol. 26, pp. 26–37, 1980.

[15] S. Ross, G. Gordon, and J. A. D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[16] S. Ross and J. A. D. Bagnell, "Agnostic system identification for model-based reinforcement learning," in *ICML*, 2012.

[17] L. Herlant, R. Holladay , and S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *ACM/IEEE HRI*, 2016.