Discriminatively Learning Inverse Optimal Control Models for Predicting Human Intentions

Robotics Track

Sanket Gaurav
University of Illinois at Chicago
Chicago, Illinois
sgaura2@uic.edu

Brian Ziebart
University of Illinois at Chicago
Chicago, Illinois
bziebart@uic.edu

ABSTRACT

More accurately inferring human intentions/goals can help robots complete collaborative human-robot tasks more safely and efficiently. Bayesian reasoning has become a popular approach for predicting the intention or goal of a partial sequence of actions/controls using a trajectory likelihood model. However, the mismatch between the training objective for these models (maximizing trajectory likelihood) and the application objective (maximizing intention likelihood) can be detrimental. In this paper, we seek to improve the goal prediction of maximum entropy inverse reinforcement learning (MaxEnt IRL) models by training to maximize goal likelihood. We demonstrate the benefits of our method on pointing task goal prediction with multiple possible goals and predicting goal based activities in the Cornell Activity Dataset (CAD-120).

KEYWORDS

Goal Prediction; Intent Prediction; Inverse Reinforcement Learning; Maximum Likelihood Estimation

1 INTRODUCTION

Humans and robots work in close collaboration for many tasks [2, 9, 26–28, 33] or simultaneously pursue separate tasks in shared workspaces [4, 6, 21, 36]. To enable effective task completion in either setting, robots should be able to anticipate human intentions prior to the completion of the pursued task. Doing so enables a robot to plan compatible actions ahead of time that are more productive for collaborative tasks or with fewer resource conflicts in separate tasks. For example, self-driving vehicles that can predict pedestrians’ intentions and behaviors can navigate more safely and efficiently at intersections. However, improved methods for predicting human intentions are needed to support these examples of more synergistic decision making in autonomous systems.

Bayesian reasoning has been predominantly used to address the goal prediction task. Under this perspective, a predictive model of the trajectory of decisions given the goal is employed—along with a prior distribution over goals—to obtain the posterior distribution over goals. Numerous methods for the trajectory likelihood model have been employed [5, 10, 18, 23, 24, 40], ranging from simple goal-conditioned Markov models [13, 31] to inverse planning [5] and imitation learning methods [40]. Central to all of these methods is that the trajectory likelihood models are designed and optimized with sole consideration to trajectory prediction rather than goal prediction. While Bayes theorem holds for the true distributions of goal posteriors and trajectory likelihoods, it can produce error-prone goal posteriors when the likelihood model is noisily estimated from limited amounts of available data.

In this paper, we investigate training maximum entropy inverse reinforcement learning models [39] to maximize goal prediction likelihoods rather than trajectory likelihoods. In section 3, we develop our method for calculating the gradient for the likelihood of the final goal. By experimenting with an object reaching task with trained reward function from our new approach, we realize an average probability for the true goal given approximately 50% of the trajectory traveled that is not realized until 70% of the trajectory is traveled using the trajectory-based likelihood method [23]. Also, we also evaluate our method on the Cornell CAD-120 dataset [18].

The paper is organized as follows: we start with a summary of background information on decision processes, previous work on predicting human intention, the inverse optimal control formulation for imitation learning, and goal prediction using an inverse linear-quadratic regulation (LQR) formulation. Next, we describe in detail our algorithm for obtaining goal predictions from the MaxEnt IRL model trained using goal likelihood maximization rather than trajectory likelihood maximization. Next, we explain the experimental setup used to evaluate our proposed method. The result section summarizes the results obtained by our goal likelihood method versus the trajectory likelihood method and other baselines. Lastly, we provide conclusions and propose future work.

2 BACKGROUND AND RELATED WORK

2.1 Decision Processes and Goal Prediction

A wide variety of tasks can be represented using sequential decision process formulations. A Markov Decision Process (MDP) is defined\(^1\) as a tuple \((S, \mathcal{A}, \tau, R)\), where:

- state \(s\) is from a finite set of states \(s \in S\);
- action \(a\) is from a finite set of actions \(a \in \mathcal{A}\);
- \(\tau\) is the state transition probability from state \(s\) under action \(a\);
- \(R(s_t)\) is the reward or cost received by visiting state \(s_t\).

\(^1\)We denote random variables with uppercase letters, fixed variables in lowercase, and matrices in boldface uppercase.

A sequence of states and actions, \( s_1, a_1, s_2, a_2, s_3, \ldots, s_T \), is produced by applying a decision policy \( \pi(a_t|s_t) \) to the state transition dynamics of the decision process, \( \tau(s_{t+1}|s_t, a_t) \).

In many domains, the decision processes for similar tasks differ only in small ways. We consider these differences being parameterized by a goal state \( g \) (where \( g \in G \), is set of all possible goals in the environment) that indicates the successful accomplishment of the goal when it is reached at final time step \( t_f \) (i.e., \( s_{t_f} = g \)). In contrast, if the goal state is not reached (i.e., \( s_{t_f} \neq g \)), a large cost (or negative reward) is incurred. More generally, the reward function can be parameterized by the goal \( g \) as: \( R_g(s_t) \).

In this paper, we also consider continuous-valued states and actions that can be modeled using a linear-quadratic regulation (LQR) formulation. In LQR, the dynamics of a system being investigated are represented by a linear relationship,

\[
s_{t+1} = A s_t + B a_t + \epsilon_t,
\]

where \( s_t \) denotes the state of the system at time \( t \), \( a_t \) denotes the action at time \( t \), \( \epsilon_t \) denotes some zero mean Gaussian noise, and \( A \) and \( B \) define the system dynamics. The state-action cost function,

\[
\text{cost}(s_t, a_t) = \begin{bmatrix} a_t \end{bmatrix}^T \begin{bmatrix} A \\ s_t \end{bmatrix} M \begin{bmatrix} a_t \\ s_t \end{bmatrix}, t < t_f,
\]

is a quadratic function that penalizes the dynamics of the system/control at each time step. We also incorporate a final state quadratic cost that penalizes the final state, \( s_{t_f} \), from deviating far from the desired goal \( g \) characterized by state \( s_g \),

\[
\text{cost}(s_{t_f}) = (s_{t_f} - s_g)^T M_f (s_{t_f} - s_g),
\]

where \( M \) and \( M_f \) are cost parameters. Similar to the MDP setting, the time-invariant state-action cost function and the final state cost can vary depending on the goal being pursued.

For the discrete MDP setting and the continuous LQR setting, the goal prediction task is defined as follows.

**Definition 2.1.** The **goal prediction task** seeks a probability distribution over potential goals given a partial sequence of states:

\[
P(s_{t_f}|g) = g(s_1, \ldots, s_t) \text{ for discrete decision processes and } P(G = g|s_1, \ldots, s_t) \text{ for continuous control processes.}
\]

In the discrete setting, the exact goal state is reached whereas the final state need only be sufficiently close to state \( g \) in the continuous setting.

### 2.2 Existing Goal Prediction Methods

Many goal prediction methods approach the goal prediction task using Bayesian reasoning. Given a generative, goal-conditioned model of the state sequence, \( P(s_{1:t}|g) \), the goal posterior is obtained using Bayes’ theorem:

\[
P_g(g|s_{1:t}) = \frac{P(s_{1:t}|g)P(g)}{\sum_{g' \in G} P(s_{1:t}|g')P(g')},
\]

where \( s_{1:t} \) is the partial trajectory of states from time step 1 to time step \( t \), \( g \) is the inferred goal, and \( g' \) of a goal from the set of pre-defined goals \( G \) in the environment.

A simple Bayesian approach for the discrete setting is the goal-conditioned Markov model [13, 31]. It estimates the next state given the current state and goal based on the empirical frequency,

\[
P(s_{t+1}|s_t, g) = \frac{\text{count}(s_{t+1}, s_t, g) + \alpha_{s_{t+1}, s_t, g}}{\text{count}(s_t, g) + \alpha_{s_t, g}},
\]

where \( \text{count}() \) is the number of occurrences in the training dataset and \( \alpha \) provide a set of optional pseudo-count values. The state trajectory likelihood of Eq. (4) is:

\[
P(s_{1:t}|g) = P(s_1) \prod_{t=1}^{t_f-1} P(s_{t+1}|s_t, g).
\]

**Predestination** [20] uses Bayes theorem to infer destinations from driving routes. It uses a history of driver destinations and driving behaviors to predict where the driver is heading (final destination). Similarly, comMotion [22] uses a set of previously visited destinations to predict a person’s destination using a Bayes classifier. More sophisticated trajectory likelihood modeling approaches treat the prediction tasks as the “inverse” of a planning process [5, 10, 14, 37]. For example, Baker, Tenenbaum & Saxe [5] use inverse planning, which assigns a probability distribution to different plans, to compute goal inferences. They investigated three different settings for goal prediction: single underlying goal, complex goals, and changing goals. In this paper, we consider the single underlying goal setting and leave extensions to the other settings as future work.

### 2.3 Maximum Entropy Inverse Optimal Control

Inverse optimal control (also known as inverse reinforcement learning [1, 17, 25]) considers a Markov decision process without a reward function and learns the reward function that rationalizes demonstrated decision sequences [1]. Assumption a reward function linear in the state feature vectors parameterized by reward parameter \( \theta \), \( R(s_t) = \theta \cdot \phi(s_t) \), Abbeel & Ng [1] propose the apprenticeship learning approach based on Inverse Reinforcement Learning.
Learning [25]. They devise a strategy of matching feature expectations between expert’s policy (πE) and learner’s policy (π):

\[ \left\| \mathbb{E} \left[ \sum_{t=1}^{t_f} \phi(s_t)|\pi_E \right] - \mathbb{E} \left[ \sum_{t=1}^{t_f} \phi(s_t)|\pi \right] \right\|_\infty \leq \epsilon, \quad (5) \]

where \( \epsilon \) is the largest error allowed when approximately matching feature vectors. While useful for prescriptive behavior in imitation learning tasks, this approach is not as useful for prediction due to the ambiguities arising from many different mixtures of deterministic policies producing the same feature counts.

Ziebart et al. [39] employed the principle of maximum entropy [16] to resolve the ambiguity of mixing policies to match feature counts by selecting a probability distribution:

\[ P_\theta(s_{1:t_f}) = \frac{\sum_{s_{1:t_f}} \theta^T \phi(s_{1:t_f})}{Z^\theta} \quad (6) \]

where \( Z^\theta = \sum_{s_{1:t_f}} \exp(\theta^T \phi(s_{1:t_f})) \) is the partition function and \( s_{1:t_f} \) is the trajectory or path traveled from time step 1 through \( t_f \). The parameters \( \theta \) that maximize the trajectory log likelihood,

\[ \theta^* = \arg\max_\theta \sum_{s_{1:t_f} \in \Xi} \log P(s_{1:t_f}|\theta), \quad (7) \]

are employed by the model. Further, the gradient of \( Z^\theta \) (partition function) is established in Lemma 2.2.

**Lemma 2.2.** The gradient of the partition function, \( Z_\theta \), is:

\[ \nabla_\theta \log Z_{\theta_{s_{1:b}}} = -\mathbb{E} \left[ \sum_{t=1}^{t_f} \phi(s_t)|s_1=a, s_{t_f}=b \right] = -\mathbb{E} \left[ \phi(s_{a:b}) \right]. \]

**Proof.** Using the definition of \( Z^\theta \) from equation 6 we have,

\[ \nabla_\theta \log Z_{\theta_{s_{1:b}}} = \frac{1}{Z_{\theta_{s_{1:b}}}} \sum_{s_{1:t_f} \in \Xi, s_{1:t_f}=a, s_{t_f}=b} e^{-\theta^T \phi(s_{a:b})}(-\phi(s_{a:b})) = -\sum_{s_{1:t_f} \in \Xi, s_{1:t_f}=a, s_{t_f}=b} P(s_{a:b}) \phi(s_{a:b}) = -\mathbb{E} \left[ \sum_{t=1}^{t_f} \phi(s_t)|s_1=a, s_{t_f}=b \right] = -\mathbb{E} \left[ \phi(s_{a:b}) \right]. \]

Following from Lemma 2.2, the gradient of the trajectory log likelihood function for a set of trajectories and corresponding goals, denoted by \( \Xi \), is:

\[ \nabla_\theta \log \prod_{(s_{1:t_f}, g) \in \Xi} P(s_{1:t_f}|\theta, g) = \mathbb{E} \left[ \phi(S_{1:t_f})|g \right] - \phi(s_{1:t_f}). \quad (8) \]

Thus, when maximized, this gradient is zero and the expected feature counts must match the training data feature counts.

In this paper, we extend MaxEnt IRL to predict human intentions given partial trajectory by maximizing the true goal likelihood instead of the trajectory likelihood.

### 2.4 Inverse Linear-Quadratic-Regulation

Maximum entropy inverse reinforcement learning methods for MDPs have been extended to linear-quadratic regulation (LQR) settings to learn the \( M \) and \( Q \) coefficient matrices (reward parameters) from demonstrated behaviors using the principle of maximum causal entropy [38]. Under this model, computing the features \( \phi_{gi} \) of the partial trajectory \( s_{1:t_f} \) given the goal \( g_i \),

\[ \phi_{gi}(s_{1:t_f}) = \frac{t_{f-1}}{t_{f-1}} \sum_{t=0}^{t_{f-1}} a_t \begin{bmatrix} s_t \ s_t \end{bmatrix}^T \quad (9) \]

the expectation of the features \( \phi_{gi}(s_{1:t_f}) \) of the remaining trajectory \( s_{1:t_f} \) from the current position \( (i) \) to the goal \( (g_i) \),

\[ \mathbb{E} \phi_{gi}(S_{1:t_f}|g_i) = \sum_{i=1}^{t_{f-1}} \left( a_{i, s_{1:t_f}} a_{i, s_{1:t_f}}^T + a_{i, s_{1:t_f}} \right), \quad (10) \]

the expectation of the features \( \phi_{gi}(s_{1:t_f}) \) of the complete trajectory \( (S_{1:t_f}) \) from the starting point to the goal \( (g_i) \),

\[ \mathbb{E} \phi_{gi}(S_{1:t_f}|g_i) = \sum_{i=1}^{t_{f-1}} \left( a_{i, s_{1:t_f}} a_{i, s_{1:t_f}}^T + a_{i, s_{1:t_f}} \right), \quad (11) \]

can be achieved efficiently based on the fact that all marginal state probabilities are multivariate Gaussians with analytical expressions mean \( (\mu_{s_{1:t_f}}) \) and variance \( (\Sigma_{s_{1:t_f}}) \) for these expectations. Finally, the probability of the true goal \( (g_i) \) given the partial trajectory \( (s_{1:t_f}) \) is obtained using Bayes theorem as;

\[ P(g_i|s_{1:t_f}) \propto P(g_i|s_{1:t_f}) \prod_{i=1}^{t_{f-1}} \pi(a_t|s_t, g_i). \quad (12) \]

This predictive linear-quadratic regulator [23] for inverse optimal control is used to predict human intentions and trajectory forecasting. Promising results have been demonstrated on the Cornell Activity Dataset (CAD-120) [18]. In this paper, we have extended the technique used in [23] by training the MaxEnt IRL model by maximizing true goal likelihood.

### 3 APPROACH

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) is a widely used method to infer the true goal or intentions of a sequential decision maker given a partial trajectory by employing Bayesian reasoning. The reward parameters in the MaxEnt IRL setting are trained via maximizing the trajectory likelihood as shown in Equation 8. The trajectory likelihood models are designed and optimized solely with consideration to trajectory prediction rather than goal predictions. While Bayes theorem, the foundation of Bayesian reasoning, holds correctly for the true distributions of goal posterior and trajectory likelihoods, it can produce error-prone goal posteriors when the likelihood model noisily estimated from limited amounts of available data. To address this problem, in this section, we develop our approach for training the MaxEnt IRL model for goal prediction using goal likelihood maximization in place of the traditional trajectory likelihood maximization approach.
3.1 Goal Likelihood Maximization Formulation

To derive our optimization procedure, we first establish Lemma 3.1 for computing the gradient ($\nabla \theta$) of the log likelihood of a partial trajectory given a goal ($P_\theta(s_1:t_i|g_i)$) with respect to the reward parameter ($\theta$).

**Lemma 3.1.** The gradient for computing the probability of a partial trajectory ($s_1:t_i$) given the goal ($g_i$) can be separated into the sum of expectations and the feature vector,

$$\nabla \theta \log P_\theta(s_1:t_i|g_i) = -\phi_{g_i}(s_1:t_i) - E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right] + E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right].$$

**Proof.** Using the definition from equation 6, we have:

$$\log P(s_1:t_i|g_i) = -\text{cost}_g(s_1:t_i) + \log Z_{s_1:t_i} - \log Z_{s_1} - \log Z_{s_1}.$$

Taking the gradient with respect to the reward parameter $\theta$ and simplifying after using Lemma 2.2 proves Lemma 3.1. □

Next, using Lemma 3.1, we establish the maximum goal likelihood gradient for MaxEnt IRL given a partial sequence of states.

**Theorem 3.2.** The gradient for MaxEnt IRL for maximum goal likelihood given a partial trajectory decomposes into a sum of expectations, features and probabilities,

$$\nabla \theta \log P_\theta(g_i|s_1:t_i) = -\phi_{g_i}(s_1:t_i) - E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right] + E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right],$$

where $s_1:t_i$ is the partial trajectory from time step 1 to $t_i$, $g_i$ is the true goal and $g'$ are the possible goals ($G$) in the environment.

**Proof.** Taking the gradient with respect to $\theta$ of the goal log likelihood, after expanding using Equation 4:

$$\nabla \theta \left( \log P_\theta(s_1:t_i|g_i) + \log P(g_i) - \sum_{g' \in G} P(g'|s_1:t_i) \nabla \theta \log P(g_i) \right)$$

\[
= \nabla \theta \log P_\theta(s_1:t_i|g_i) + \nabla \theta \log P(g_i) - \log \sum_{g' \in G} P(g'|s_1:t_i) \nabla \theta \log P(g_i) P(g')
\]

\[
= -\phi_{g_i}(s_1:t_i) - E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right] + E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right] - \sum_{g' \in G} P(g'|s_1:t_i) \left( -\phi_{g_2}(s_1:t_2) - E \phi_{g_2}(S_2:t_2|g_2) g_2 \right) + E \left[ \phi_{g_i}(S_1:t_i|g_i) g_i \right],
\]

where: (a) follows from properties of the gradient applied to logarithms and the definition of the goal posterior and (b) is obtained after employing Lemma 3.1. □

This gradient trivially equals zero when the goal predictions are perfectly correct (i.e., $P(f_i = g_i|s_1:t_i) = 1$). However, this is often difficult to achieve when training from noisy data. In general, optimizing the reward parameter to maximize goal likelihood is non-concave. However, we can obtain a reasonable local maxima by changing the starting conditions and other factors. For example, initializing the reward parameter optimization at the maximum trajectory likelihood parameters guarantees no worse parameters than the trajectory-based approach.

**Algorithm 1 Learning IOC model for goal prediction**

Input: The reward parameter $\theta$; Set of training trajectories reaching goals $\Xi$; Set of Goals $G$.

Output: The optimized/learned reward parameter $\theta$.

1. for $(s, g_i, t_i) \in \Xi$ do
2. Extract partial trajectory $s_1:t_i$
3. \(\nabla \theta \leftarrow -\phi_{g_i}(s_1:t_i) - \sum_{g' \in G} \phi_{g_i}(S_1:t_i|g_i) g_i + \sum_{g' \in G} \phi_{g_i}(S_1:t_i|g_i) g_i\)
4. for $g' \in G$ do
5. \(\nabla g' \leftarrow \phi_{g'}(s_1:t_i) + \sum_{g'' \in G} \phi_{g''}(S_1:t_i|g'') g'' - \sum_{g'' \in G} \phi_{g''}(S_1:t_i|g'') g''\)
6. Compute $P(g'|s_1:t_i)$
7. \(\nabla \theta \leftarrow \theta + \eta \nabla \theta\)
8. end for
9. $\theta \leftarrow \theta + \eta \nabla \theta$
10. end for
11. return $\theta$

The learning procedure (Algorithm 1) takes as input an initial reward parameter, a set of training trajectories, and a set of possible goals in the space. It iterates over randomly selected training trajectories, extracting the partial trajectory from the selected trajectory, i.e., $s_1:t_i$, and then constructs the full gradient from its components in step 3, step 5, and step 7. Step 3 computes the difference in expected features for the true goal. Step 5 computes the same differences for each possible goal and then Step 7 weights these by the goal probabilities. Lastly, Step 9 applies a gradient step weighted by $\eta$ to improve towards locally optimal reward parameters $\theta^*$ using expectations computed for the true goal and all other goals. In practice, more sophisticated gradient-based updates [11, 34] can be employed. The algorithm repeats steps 2 through 8 (with decreasing learning weights) for all of the training trajectories until approximately converging to a locally optimal point.

We note the contrast from previous goal prediction methods using MaxEnt IRL trained by maximizing the likelihood over the trajectory to train the reward parameter as explained in Equation (8). Critically, the likelihood of the correct goal given a partial sequence of actions is inferred using Bayesian reasoning. This produces a mismatch between the training and application objective and can produce error-prone goal likelihoods. Thus, the most significant advantage of training using the proposed method (maximum goal likelihood) is that we maximize the likelihood over the true goal, which correctly matches the application objective.

3.2 Extension to Linear-Quadratic Regulation

Algorithm 1 provides a general algorithm for MaxEnt IRL trained to maximize goal prediction for the case of discrete state/action decision processes. We can extend this general method to other settings/controllers to match other real-life scenarios. In this paper, we use inverse LQR to conduct our experiments and we have the
we used the Armadillo C++ linear algebra library for fast linear computations [29].

4 EXPERIMENTAL SETUP

In this section, we explain our experimental setup used for evaluating our proposed Algorithm 1 from Section 3 for the inverse LQR setting. We have used two real-life datasets to evaluate our proposed method.

4.1 Goal Pointing Task Datasets

For our first set of experiments, we have used an existing dataset of pointing tasks [30]. The data was collected using a Baxter robot from Rethink Robotics and a Microsoft Kinect camera. For the training data, 10 balls were hung from the ceiling (5 on both sides of the Baxter robot), and a teleoperator was asked to stand in front of the Kinect Camera (input sensor). The teleoperator was asked to reach the displayed ball number on Baxter’s head-mounted display from a neutral position. Another operator moved Baxter’s corresponding arm in zero gravity mode from a neutral position to the displayed goal in synchronization with the human arm motion. This Kinect-Baxter correspondence data was used to train a linear regression correspondence model for robotic teleoperation. Further, we used the training sequence to extract states and actions for inverse LQR system and trained cost functions $M$ and $M_f$.

The 10 hanging balls from the ceiling were then shuffled to new positions (different from the training set-up) for the testing phase. The 18 teleoperators were asked to teleoperate the Baxter robot’s arm by standing in front of the Kinect camera from a neutral position to reach the goal that was displayed on the Baxter head-mounted screen. This process was repeated for each goal and three different control assistance methods ((i) Sigmoid assist, (ii) Step assist, and (iii) No assist), for details please refer to [30]. The three control assistance methods were also repeated twice in random order to maintain consistency. Thus, each person performed 60 trajectory sequences of reaching the displayed goal.

In total, the dataset consisted of 1080 goal-reaching trajectories. Figure 2 explains the steps of test data collection. The dataset contains the Kinect skeleton values, the Baxter end-effector position matrix updates. The three control assistance methods were also repeated twice in random order to maintain consistency. Thus, each person performed 60 trajectory sequences of reaching the displayed goal.

4.2 Cornell Activity Dataset (CAD-120)

For our second set of experiments, we employed our Algorithm 1 to train reward parameters on the publicly available Cornell Activity Dataset (CAD-120) to strengthen our claim. This dataset consists of 120 depth camera video of daily activities. There are ten high-level activities: making cereal, taking medicine, stacking objects, unstacking objects, microwaving food, picking objects, cleaning objects, carrying food, organizing objects and eating a meal. These activities are further divided into ten sub-activities: reaching, moving, pouring, eating, drinking, opening, placing, closing, cleaning and null. For example, the task of making cereal can be broken down: reaching (cereal box), moving (cereal box on top of bowl), pouring (from cereal box to a bowl), moving (cereal box to the previous position) and null (moving the hand back).

In this study, we have divided the trajectories based on the above 10 sub-activities. We disregarded null sub-activity as it has an undefined goal or intention. First, we extracted goals for each of the trajectory in the sub-activity. Second, we trained the cost functions $M$ and $M_f$ for each of these sub-activities separately. We withheld...
The end-effector consists of two coordinate frames, one associated with the robot and the other with the robot arm, which is calculated using forward kinematics [32]. The possible goals [23] are inferred given the observed partial trajectory of the end-effector in real time. The process is clearly depicted in Figure 1 and Equation (12). These goal state probabilities are conditioned on the state representation to incorporate linear features into the quadratic cost function in Equation (2). Additionally, goal state $i$ of the end-effector is represented using only the goal’s translational position,

$$g_i = [x_{g_i}, y_{g_i}, z_{g_i}, 0, 0, 0, 0, 0, 0]^{T}. \quad (15)$$

To compute goal predictions along the test trajectories, we train the reward parameters $\mathbf{M}$ and $\mathbf{M}_t$ using our proposed method (maximum goal likelihood) as described in Algorithm 1 on the training data. From these trained cost matrices, the probabilities of different possible goal states are inferred given the observed partial trajectory of the end-effector in real time. The process is clearly depicted in Figure 1 and Equation (12). These goal state probabilities are $P(g_i|s_{1:T})$ and the probability of the most likely intended goal of the partial trajectory, $l$, is,

$$I = \max_i P(g_i|s_{1:T}). \quad (16)$$

4.3 Estimating the Reward Parameters

The inverse LQR model used in this paper has two separate reward/cost parameter matrices $\mathbf{M}$ and $\mathbf{M}_t$ to train. To provide the strongest guarantees, we first train the reward parameters using the maximum trajectory likelihood method as explained in Equation (8) on the training data for both datasets. Then we use these trained reward parameters to initialize Algorithm 1 to learn our proposed method for maximizing goal likelihood on the training data.

We have used accelerated stochastic gradient descent with an adaptive learning rate [11, 34] and L1 regularization on both parameters simultaneously. This regularized approach prevents overfitting over the demonstrated trajectories of the datasets used in this paper. In the next section, we would describe goal predictions using inverse LQR controller on the test data for both datasets.

4.4 Goal Prediction via Inverse LQR

Following the existing formulations employed for maximum trajectory likelihood methods [30], a goal is defined as a location in $x_g, y_g, z_g$ translational space that we want the robot arm end-effector to approximately reach. The end-effector is the endpoint of the robot arm, which is calculated using forward kinematics [32]. The end-effector consists of $x_t, y_t, z_t$ translational and $x_r, y_r, z_r, \omega_r$ quaternion angles as rotational dimensions referenced from the associated robot’s coordinate frame. We have considered only translational dimensions for goal positions.

Following the approach outlined for the inverse LQR setting [23], the authors of [30] assume the linear dynamics of Equation (1), in which the state of the end-effector is defined as,

$$s_t = [x_t, y_t, z_t, x_t, y_t, z_t, \hat{x}_t, \hat{y}_t, \hat{z}_t, 1]^{T}, \quad (13)$$

and end-effector actions as

$$a_t = [\hat{x}_t, \hat{y}_t, \hat{z}_t]^{T}, \quad (14)$$

where $(\hat{x}_t, \hat{y}_t, \hat{z}_t)$ are velocities, $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ are accelerations, and a constant of 1 is added to the state representation to incorporate linear features into the quadratic cost function in Equation (2). Additionally, goal state $i$ of the end-effector is represented using only the goal’s translational position,

$$g_i = [x_{g_i}, y_{g_i}, z_{g_i}, 0, 0, 0, 0, 0, 0]^{T}. \quad (15)$$

To compute goal predictions along the test trajectories, we train the reward parameters $\mathbf{M}$ and $\mathbf{M}_t$ using our proposed method (maximum goal likelihood) as described in Algorithm 1 on the training data. From these trained cost matrices, the probabilities of different possible goal states are inferred given the observed partial trajectory of the end-effector in real time. The process is clearly depicted in Figure 1 and Equation (12). These goal state probabilities are $P(g_i|s_{1:T})$ and the probability of the most likely intended goal of the partial trajectory, $l$, is,

$$I = \max_i P(g_i|s_{1:T}). \quad (16)$$

4.5 Prior Distribution

The inverse LQR goal prediction method is a Bayesian inference method that benefits significantly from a prior distribution over the possible goals [23]. In the previous trajectory likelihood maximization experiment [30], they used a distance prior similar to the one used in previous work [23],

$$P(g_i|s_t) \propto e^{-\beta \text{dist}(s_t, g_i)}, \quad (17)$$

where $\text{dist}(s_t, g_i)$ is a function that computes the Euclidean distance between the spatial coordinates of $s_t$ and $g_i$, and $\beta$ is an adjustable coefficient that increases the importance of distance on
the distribution. As \( \text{dist}(s_t, g_i) \) decreases, \( P(g_i|s_t) \) increases effectively making closer targets more probable. We have used the same formulation for most intended goal prediction for both experiments in this paper.

### 4.6 Baselines

To compare our goal likelihood method on goal pointing task data from the two datasets, we use the nearest target (predicting the nearest goal as the true goal along the trajectory points) prediction. It is the simplest baseline for goal prediction, and all methods should be expected to perform better than it. We additionally use logistic regression [8] as the discriminative method comparison baseline. We also compare with the previous approach of constructing a model using trajectory likelihood maximization [23]. For CAD-120 dataset, in addition to comparing to the trajectory likelihood method, we also compared our method with ATCRF [19].

### 4.7 Evaluation Metrics

To evaluate our proposed method against the existing trajectory maximum likelihood method, we use two evaluation metrics. First, we compute the logarithmic loss for true goal probability across the whole trajectory. The logarithm loss has been plotted for both methods at various fractions of the trajectory covered in Figure 3-a on pointing task dataset and Figure 3-d on CAD-120. Second, we compute the accuracy of our proposed method and other baselines across different fractions of the trajectory in predicting the true goal. We have also reported precision and recall for both methods. Tables 1 and 2 report the results for both datasets.

## 5 RESULTS AND DISCUSSION

The proposed optimization of the reward parameter to maximize goal likelihood involves maximizing a non-concave function. This prevents any guarantees of convergence to a global optimum. However, still, we can reach some local maximum that provides a better result than previous trajectory-based optimization methods. We have experimented with three different starting points to train the cost function \( M \) and \( M_f \): (1) initial values of all 0; (2) pre-trained initial values using the optimization objective of past work (i.e., trajectory likelihood maximization); and (3) randomized starting points. We find convergence to very similar parameters with all three of the different starting points, indicating that we can reach a stable local maxima without strong sensitivity to the initial values.
We have tested our method on two different real-life datasets involving human and robot goal-directed movements. Figure 3-a illustrates the logarithmic loss of the correct goal prediction given a partial trajectory computed across the fraction of the trajectory for the pointing task dataset. The black color represents the trajectory likelihood method and the green color represents the proposed goal likelihood approach. It is evident from Figure 3-a that the goal likelihood maximization method’s logarithmic loss decreases faster and reaches the true goal probability in approximately 50% of the trajectory. On the other hand, the trajectory likelihood maximization method achieves the same performance at 70% of the trajectory. In both settings, we have used a distance prior, so the probability distribution rapidly increases from a uniform distribution as the true goal may be farther from the neutral position than other targets.

To illustrate the behavior of the goal prediction methods, we select a trajectory from pointing task test data and plot the probability distribution across five goals along the trajectory length in Figure 3-b and c. The plot of the resulting distribution in Figure 3-b corresponds to the trajectory likelihood method and Figure 3-c corresponds to our proposed goal likelihood maximization method. We can see that our goal likelihood maximization method performs better than the trajectory likelihood maximization method. Our proposed method realizes a high probability prediction for the true goal much earlier than the previous trajectory likelihood maximization method with a smoother transition across different goal probabilities.

Figure 3-d shows the logarithmic loss of goal prediction along the trajectory for reaching a goal from the CAD-120 dataset. The trajectory likelihood maximization method is represented by the black color and our proposed goal likelihood maximization method is shown in green. The plot clearly shows that our goal likelihood maximization method predicts the true goal (approximately 60%) much earlier in the trajectory than the trajectory likelihood method (approximately 80%).

In Table 1, we report the accuracy, precision, and recall for goal prediction for three methods, i.e., the nearest goal predictor, trajectory likelihood maximization model, and the goal likelihood maximization model. Both the previous (trajectory likelihood) and proposed (goal likelihood) models perform significantly better than the simplest baseline method, i.e., the nearest goal baseline. At 40% and 60% of the trajectory, our proposed goal likelihood-based method outperforms the trajectory likelihood-based method by a noticeable margin. The result also matches with our log loss metrics as shown in Figure 3-a. We also compare our results with a logistic regression model [8] as the generative method baseline. The reported goal prediction accuracy of 57.9% is obtained from a partial trajectory of length 60 time-steps. The average range of the trajectories of pointing task dataset is 110 time-step. So, at 60% of the trajectory length, we found that our proposed method predicts the true goal with an accuracy of 92.8%, which is significantly better than logistic regression.

Table 2 shows the performance results of the experiment conducted on the CAD-120 dataset. We compared the performance of our proposed goal-based method with other baselines based on trajectory likelihood maximization and the ATCRF model (only result for 100% is available). We can see that the trajectory likelihood method achieves comparable accuracy at 40% of the trajectory what is not realized until 100% of the sequence is observed using the ATCRF model. The result of the ATCRF method is on the unmodified CAD-120 dataset, which consists of null sub-activities, which prevents it from achieving 100% accuracy even when observing the complete trajectory. From the beginning of the trajectory, our proposed goal-based method outperforms the trajectory-based method by a considerable margin, which matches the log loss results shown in Figure 3-d and achieves 100% accuracy in prediction at 80% of the trajectory.

Thus, these experiments strongly support our claim that by re-training the MaxEnt IRL approach using goal likelihood maximization for goal predictions, we can achieve better and faster goal prediction than existing methods—specifically those based on trajectory likelihood maximization. As this is an important sub-problem for planning symbiotic robot behavior, we believe these improvements will help increase the productivity of human-robot collaborative tasks when used appropriately.

6 CONCLUSION AND FUTURE WORK
In this paper, we have proposed training inverse reinforcement learning models that were initially designed for policy estimation, to instead be optimized for goal prediction. We derived the gradient for optimizing goal likelihoods under the general discrete maximum entropy inverse reinforcement learning (MaxEnt IRL) setting and under the continuous inverse linear-quadratic regulation (LQR) setting. We demonstrated that our goal likelihood maximization method provides significant improvements for goal prediction compared to previous methods based on trajectory likelihood maximization in practice. Thus, with our new approach, we can more accurately infer intended goals farther in advance than previous approaches, enabling robots to know human intentions to make more compatible decisions.

As future work, we will test our method on real-world human–robot tasks like assisting robotic teleoperation [30]. These tasks often involve additional complications that should also be modeled to improve goal prediction. For example, though we have assumed that the robot’s workspace is free of obstacles in this paper, many real-world robotic workspaces contain numerous obstacles. We plan to extend our goal prediction optimization approach to the hybrid, two-level imitation learning method [7] that incorporates discrete waypoints at the top level and employed LQR predictions conditioned on the waypoints at the bottom level. We believe that through arm motion demonstrations of obstacle avoidance during training, the cost function can be learned to reason about arm movements around obstacles in testing environments. Further, in this paper, we have assumed that the goals are static in the environment. We will relax this assumption by allowing goals to change over time without being reached [5, 12].

ACKNOWLEDGMENTS
We thank Mathew Monfort for sharing his linear quadratic regulator for inverse optimal control code with us for modification. This research is supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2016-158710 and NSF grant #1652530.
REFERENCES


