2020 29th IEEE International Conference on
Robot and Human Interactive Communication (RO-MAN)
August 31 - September 4, 2020 - Virtual Conference

Designing Context-Sensitive Norm Inverse Reinforcement Learning Framework for Norm-Compliant Autonomous Agents

Yue Guo[1], Boshi Wang[2], Dana Hughes[1], Michael Lewis[3], Katia Sycara[1]

Abstract— **Human behaviors are often prohibited, or permitted by social norms. Therefore, if autonomous agents interact with humans, they also need to reason about various legal rules, social and ethical social norms, so they would be trusted and accepted by humans. Inverse Reinforcement Learning (IRL) can be used for the autonomous agents to learn social norm-compliant behavior via expert demonstrations. However, norms are context-sensitive, i.e. different norms get activated in different contexts. For example, the privacy norm is activated for a domestic robot entering a bathroom where a person may be present, whereas it is not activated for the robot entering the kitchen. Representing various contexts in the state space of the robot, as well as getting expert demonstrations under all possible tasks and contexts is extremely challenging. Inspired by recent work on Modularized Normative MDP (MNMDP) and early work on context-sensitive RL, we propose a new IRL framework, Context-Sensitive Norm IRL (CNIRL). CNIRL treats states and contexts separately, and assumes that the expert determines the priority of every possible norm in the environment, where each norm is associated with a distinct reward function. The agent chooses the action to maximize its cumulative rewards. We present the CNIRL model and show that its computational complexity is scalable in the number of norms. We also show via two experimental scenarios that CNIRL can handle problems with changing context spaces.**

I. Introduction

Autonomous agents are increasingly assuming roles that involve interaction with humans, including domestic robots such as Roomba, assistive robots in hospitals or office environments, and self-driving cars. In order to let those agents interact with people in a natural way, it is vital for them to not only perform their domain tasks, but also to reason about various moral values, legal rules, and social conventions to gain human acceptability. Social norms [1], which denote guidelines of behaviors that are socially agreed upon, have formalized the prohibitions, permissions, and obligations of the agents when they are engaged in social activities.

Normative Markov Decision Process (NMDP) [2] is a framework that incorporates the social norms into reinforcement learning (RL), and solves the limitations of rule-based approaches by actively learning social rules without predefining them in advance [3], [4], [5]. The Modular Normative Markov Decision Process (MNMDP) in [6] addresses the scalability issue of reasoning over all combinations of norms by considering only activated norms which invoke an associated reward function and the corresponding precomputed optimal policy.

However, all the above approaches require manually designing the reward function that could induce norm-compliant behaviors. The developers need to have knowledge on how the reward function captures the desired values reflected in human beliefs when the agent acts socially, and also have to hand-tune the function parameters. As mentioned in [7], [8], one of the general challenges of using RL to develop norm-aware agents is the need to manually design appropriate reward functions, which requires the developers to specify the exact reward values that the agent should receive for norm-compliant or norm violation behaviors for all possible norms. To ensure that RL agents learn about the norms in addition to their domain task, developers need to craft appropriate reward functions for each norm's compliance. Inverse Reinforcement Learning (IRL) avoids hand-crafting rewards by learning from expert demonstrations, and thus could ameliorate the dilemma of designing appropriate reward function for normative behavior. The agent ideally learns from what a norm-compliant expert demonstrates, just like a child learning what is socially appropriate from well-behaved adults. Therefore, we claim that IRL is a powerful tool with infinite potential to learn from social norms, recover what experts value, and finally build a norm-compliant autonomous agent.

Additionally, norms are highly sensitive to environmental contexts [9], [10], which refer to factors that are independent from the domain task but affect which norms are active and prioritized, e.g. weather conditions, accidents, road blocks etc. for a self-driving car. We further assume that the context is independent of the domain task in this paper. One might argue that even though utilizing context as an indicator to guide the norm design and selection might be intuitive and promising, naively adding contexts into the state space could also ensure agents are norm-compliant, but this unfortunately is highly likely to result in an explosion of the state space, making such an approach infeasible for environments with large number of contexts.

To address the above issues, we propose a new IRL framework called Context-Sensitive Norm IRL (CNIRL) that extends the work on Modularized Normative MDP [6], and early work on context-sensitive RL in the cognitive science literature [11]. CNIRL treats states (related to the domain task) and contexts (related to the social conventions) separately, and assumes that the expert constantly determines the priority of every possible norm in the environment, where

[1]The authors are with School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {yueguo, danahugh, sycara}@andrew.cmu.edu

[2]The author is with the ShanghaiTech University, Shanghai, China. Email: wangbsh@shanghaitech.edu.cn

[3]The author is with the University of Pittsburgh, Pittsburgh, Pennsylvania, USA. Email: ml@sis.pitt.edu

each context-sensitive norm is associated with a different reward function over the domain state and the agent chooses the action to maximize cumulative rewards. The trained model recovers a set of reward parameters for each social norm in the environment, and also an activation function that can estimate the relevant importance for each social norm under different states and contexts, which serves as the priority of norms implicitly. By exploiting the fact that social norms are context-sensitive, our model scales well to the number of social norms and their context complexity that may be relevant to the given environment.

The paper is organized as follows: Section 2 discusses related work; Section 3 provides a brief background on IRL and Modularized Normative MDP; Section 4 presents our Context-Sensitive Norm Inverse Reinforcement Learning framework as well as the maximum likelihood estimation for optimizing the model parameters; Section 5 describes our simulation results; Section 6 concludes and provides possible directions for future work.

II. Related Work

Utilizing IRL to study norms has been recently discussed in the social norm community [12], but to our knowledge none of the work touches the context-sensitive nature of norms. Prior works have shown that norm activation is context-sensitive [9]. Authors in [13] and [14] use Dempster-Shafer Theory [15] to compute how context relates to norms. Context sensitive reinforcement learning work is sparse with some exceptions in the 2005s [11] [16], in which the authors propose Context-sensitive MDP that maps a context to an MDP that shares the same state space of tasks and action space with other MDPs but has a distinct reward function and transition function. However, that work considers neither social norms nor IRL. Inspired by the separation of the observable and context information of prior work, we propose a scalable model, called Context-sensitive Norm IRL (CNIRL) that considers both social norms and IRL and uses the separation of context and task related information in the state space. In this model, we can cluster expert demonstration trajectories that contain norms that are conditioned on the similar context, and develop the norm activation and reward function for each cluster.

A variety of work has dealt with clustering multiple intents. [17] first designs the Maximum Likelihood Inverse Reinforcement Learning (MLIRL) to handle the problem. It is then improved by [18] and [19] to generalize to the non-parametric case where the number of intents is not given. [20] first considers the locally consistent reward functions, where a single trajectory might contain multiple intents, and [21] works on both the non-parametric and locally consistent cases. Our work is especially inspired by the locally consistent reward functions, since this captures the intuition that the norm(s) that should be activated might change on one single trajectory due to changing contexts, and result in switching to other reward functions. Even though we have a complete different model that includes context, we share similar intuitions with the above works. However, as [8] points out, the requirement of collecting a large amount of human data as demonstrations, as well as the expensive computation for realistic tasks is a big concern for IRL. This is because demonstrations need to cover all the possible varied contextual factors, which might lead to combinatorial problems, by which CNIRL is not hindered as it only considers relevant context for the activated norms.

In addition, as mentioned in [6], the computation cost is intractable if we want to compute the optimal policy in the full Normative MDP system [2] that considers all the potential norms. [6] proposes a modularized approach, in which it only calls the reduced MDP with the current activated norms. More related works on modular MDP could be found in [22], [23], [24], [25]. Our method of Context-Sensitive Norm IRL is also a modular process: at each time step the agent only considers the norm(s) that may get activated given the observation of current context and the state, so that even though a number of corresponding MDPs conditioned on the norm selection have to be solved, each of them has a smaller size than the full NMDP with context. Those smaller MDPs are only different in terms of the parameters to calculate the reward functions, and thus would not make the storage explode but might be even faster in parallel computing. Hence, our innovation lies in the extension on incorporating a modified IRL algorithm as well as the consideration of context influence into the framework based on the MNMDP, in order to study social norms better.

III. Background

A. Inverse Reinforcement Learning

Markov Decision Process (MDP) is a discrete-time decision making mathematical framework where the agent behaves by taking an action each time and getting feedback from the environment. Inverse Reinforcement Learning (IRL) relies on this paradigm to find out the implicit reward function for the agent given expert demonstrations.

Definition 1 (Markov Decision Process)**.** *A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is the reward function, $\mathcal{T}(s, a, s') = p(s'|s, a)$ is the transition function, and $\gamma \in [0, 1]$ is the discount factor.*

A *policy*, $\pi(s, a) = p(a|s)$, specifies the probability of performing action a at state s. The Q-function $Q^\pi(s, a)$ is defined by the expected total discounted reward obtained starting from state s and performing action a:

$$Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')[\mathcal{R}(s') + \gamma \max_{a' \in \mathcal{A}} Q^\pi(s', a')] \quad (1)$$

Traditional RL finds the optimal policy π^*, the policy that maximizes the discounted cumulative reward received for any state. IRL instead considers the case of having an unknown reward function, and uses the expert demonstrations to recover it. The reward function $\mathcal{R}$ is assumed to be parameterized as a linear combination of predefined state features $\phi(s)$, with weights θ as the reward parameters. Hence, the reward of state s, $\mathcal{R}(s)$, is $\theta \cdot \phi(s)$. With K

different reward functions, $\mathcal{R}_k(s) = \theta_k \cdot \phi(s)$ for $k = 1, 2...K$, which gives the reward of state s under the kth reward function. Finding $\theta = \{\theta_1, ..., \theta_K\}$ solves the IRL problem. Furthermore, although a typical IRL uses the linear combination of state features to construct the reward map, it could be generalized to a non-linear one if needed [26], [27].

B. Normative Markov Decision Process

The Normative Markov Decision Process (NMDP) [2] is an extension of MDP, which models the decision process for the agents that are norm-aware.

Definition 2 (Normative Markov Decision Process)**.** *A Normative Markov Decision Process (NMDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma, \mathcal{N})$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, and γ are the same as Definition 1, but $\mathcal{R}$ is the reward for the domain task. $\mathcal{N}$ is the set of social norms, which is a tuple $(\nu, \Sigma, \phi_n, \phi_a, \phi_d, \sigma, \delta, \rho)$, where $\nu \in \{\mathcal{O}, \mathcal{P}, \mathcal{F}\}$ denotes the deontic modality, Σ is the set of norm applicable states, ϕ_n is the set of states in Σ where ν applies, ϕ_a, ϕ_d denote the activation and deactivation functions, σ is the sanction, δ represents the authority, and $\rho \in \mathbb{Z}^+$ represents priority.*

The recent work [6] proposes a scalable framework called Modular Normative Markov Decision Processs (MNMDP) that integrates the domain goal and norm reasoning by modulazring the full NMDP into smaller MDPs with a subset of norms. Instead of considering all the norms, MNMDP only updates the set of norms that get activated when the agent takes each action. It then follows the activated norms to decide the optimal policy at each time step. It further assumes that each norm is associated with a specific reward function, which is different from the domain task, to represent the sanction. For example, if the norm "avoid collision" is active, then in the corresponding reward function, all the states that the agent collides with others would have a low reward to address this prohibition. The priority is implicitly inferred based on the activation as well, by giving activated norms a higher rank, and non-activated norms a lower rank according to the activation function.

In this paper, combined with IRL and contextual insights, we further extend the MNMDP model with the focus on recovering sanctions, i.e. a set of reward functions $\{\mathcal{R}_k\}$, parameterized by $\{\theta_1, \ldots, \theta_K\}$, and the activation functions for the K context-sensitive norms. Note that we do not aim to learn every component in NMDP presented above.

IV. Proposed Methods

A. Context-Sensitive Norm IRL

Our work focuses on how to construct and learn when one or more norms would get activated, as well as learn the optimal policy $\pi^*(a|s, c)$, when we are given norm-compliant expert demonstrations, which is a set $D = \{\tau_1, ..., \tau_n\}$, where n is the number of demonstrations and τ_i is the ith trajectory. A trajectory τ_i consists of T_τ state-context-action pairs $\{(s_1, c_1, a_1), ..., (s_{T_\tau}, c_{T_\tau}, a_{T_\tau})\}$. Here c_t is the contextual information in the environment at time t, which denotes the environmental factors that affect the norm, but are independent of the domain task and are not under the control of the agent. Context space $\mathcal{C}$ denotes the set of all possible values of c_t.

The new framework we propose is called Context-Sensitive Norm Inverse Reinforcement Learning (CNIRL), which is an extension based on MNMDP that has the reduction in computation complexity from normative MDP. We are inspired by the model of Context-sensitive MDP [11], [16] that utilizes the context information, so that we incorporate the norm and context information to form a new model Context-Sensitive MNMDP, and create efficient IRL methods to learn the appropriate construction of norms given expert demonstrations. It could further be viewed as a decomposition of the whole state space into the norm-related modules. We specifically focus on calculating the reward parameters θ_k and the norm activation $\phi_a(s, c)$, which maps from current state and context to an activation score, for each of the norms. The illustration of how a Context-Sensitive MNMDP makes decisions is depicted in Fig. 1.

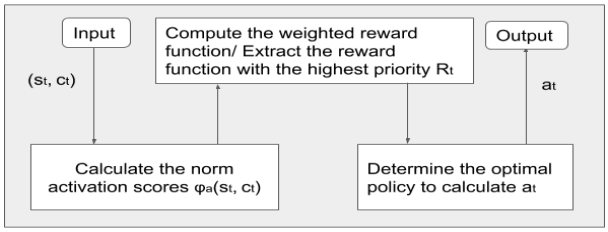


Fig. 1: Illustration of Context-Sensitive Modular Normative Markov Decision Process. At each time step t, (a) the agent first perceives the current state s_t and environmental context c_t; (b) the agent estimates each norm's activation score using the norm-activation function $\phi_a(s_t, c_t)$; (c) the agent finds the reward function R_t according to the active norm(s), and (d) based on its optimal policy, it picks the action a_t.

Definition 3 (Context-Sensitive MNMDP)**.** *A Context-Sensitive MNMDP is a tuple $(\mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \phi_a, \{\mathcal{R}_k\}_{k=0,..,K}, \gamma)$. Notice that S, A, T, γ and R_0 are the same as in definition 1, i.e. R_0 for the domain task, while $\{\mathcal{R}_k\}_{k=1,..,K}$ corresponds to the kth norm in $\mathcal{N}$ in definition 2. $\mathcal{C}$ is the context space and $\phi_a(s, c)$ maps a tuple of state and context pair (s, c) to a vector of activation scores for each of the K norms. i.e. $\phi_a(s, c) \succeq 0^{K+1}$ and $\sum_{k=0}^{K} \phi_a(s, c)_k = 1$. Under s and c, the agent acts with the optimal policy calculated by either the weighted reward function $\sum_{k=0}^{K} \phi_a(s, c)_k \mathcal{R}_k$, or the most potential reward function $\mathcal{R}_{\arg\max_k \phi_a(s,c)_k}$, specified by the scenario. After each $a \in \mathcal{A}$ is performed, s and c are again updated according to the environment.*

We assume that the expert demonstrations we learn from are compliant with the norms, and our goal is to learn the reward functions R and the activation function ϕ_a, which we represent as functions parameterized by θ and α, respectively. Whenever the θ and α are calculated, we use the traditional Boltzmann distribution to compute the optimal

policy [28]. Therefore, the probability of choosing an action is proportional to its exponential weighted optimal Q values, i.e. $p(a|s,c) \sim \exp(\beta \bar{Q}(s,a|c))$ where β is a parameter of the exponential distribution.

How do we achieve reduction of computational complexity? Indeed, we have eliminated the extra information of "context transition". Recall that if we want to take the contextual information into consideration, the dimension of state is augmented to include both the domain task and the context. Then the transition function has to consider how context would transit if an action is taken as well as how the domain state transits. However, this augmentation has no real performance improvement in the scenarios that the context is independent from the domain task, but simply increases the computation complexity due to the exponential growth of contextual information changes.

More formally, assume state has n dimensions, with each can take u values, action can take v values, and the context is a vector of m dimensions, with each can take w values. Then the sizes of the state space, action space, and context space are $|\mathcal{S}| = u^n, |\mathcal{A}| = v, |\mathcal{C}| = w^m$. If we augment the context into the state space, then the "augmented" state space would have size $|\mathcal{S}||\mathcal{C}| = u^n \cdot w^m$. Recall the Bellman iteration [29] in MLIRL which approximates the gradient of the reward parameters through fixed-point iterations. For each iteration the computation cost would be $O((|\mathcal{S}||\mathcal{C}|)^2|\mathcal{A}|) = O(u^{2n} \cdot w^{2m} \cdot v)$. Now our crucial assumption is that there are $|\mathcal{N}| < |\mathcal{C}|^2$ norms in the environment, each associated with a reward function over the domain state only, and can be invoked by the state and context information. In this model we need to perform the Bellman iteration for all norms, but with lower cost for each, which gives $O(|\mathcal{N}||\mathcal{S}|^2|\mathcal{A}|) = O(|\mathcal{N}| \cdot u^{2n} \cdot v) < O(u^{2n} \cdot w^{2m} \cdot v)$. The computation complexity is thus linear to the number of norms, and we avoid the exponential growth of the context in the transition function.

Hence, since we treat the context as what we observe instead of what we predict, and make the decision conditioned on the context-sensitive norm, context can be effectively utilized as the indicator for selecting norms, and we can ensure computational reduction. This preserves the same spirit as in [6] to reduce computation, with one more step to tell which norm(s) is active depending on the state and the context, as well as how we could reasonably construct the norm from expert demonstrations.

B. Maximum Likelihood Estimation

We assume the reward $\mathcal{R}$ is a linear combination of θ, and the activation function ϕ_a could be captured by a network parameterized by α, and estimate both of them via MLE and gradient descent. Note that the math derivation for α does not limit ϕ_a to be any specific network. Further assuming the trajectories are generated independently, we can write the likelihood of the set of demonstrations D:

$$\begin{aligned} L(D;\alpha,\theta) &= \prod_{\tau \in D} p_{\alpha,\theta}(\tau) \\ &= \prod_{\tau \in D} \prod_{t=1}^{T_\tau} p(s_t|s_{t-1},a_{t-1}) p(c_t) p_{\alpha,\theta}(a_t|s_t,c_t) \end{aligned} \tag{2}$$

The log-likelihood of D is thus given by:

$$\begin{aligned} \log L(D;\alpha,\theta) = &\sum_{\tau \in D} \log p(s_1) + \sum_{\tau \in D} \sum_{t=1}^{T_\tau} \log p(c_t) \\ &+ \sum_{\tau \in D} \sum_{t=2}^{T_\tau} \log p(s_t|s_{t-1},a_{t-1}) \\ &+ \sum_{\tau \in D} \sum_{t=1}^{T_\tau} \log p_{\alpha,\theta}(a_t|s_t,c_t) \end{aligned} \tag{3}$$

We assume the MDP has a static initial state distribution, a static stochastic transition function, and the context variables are stationary and unaffected by agent actions (i.e., norms become active or inactive regardless of agent actions). Therefore, the first three terms are unaffected by the parameters defining reward function and activation function. Thus, when taking the gradient, we only need to consider the last term in Equation 3 which is parameterized by θ and α. For the reward parameter θ_k, expanding out the last term with the Boltzmann distribution yields

$$\begin{aligned} \nabla_{\theta_k} \log p_{\alpha,\theta}(a_t|s_t,c_t) = \beta \phi_a(s_t,c_t)_k (&\nabla_{\theta_k} Q^*_{\theta_k}(s_t,a_t) \\ &- \sum_{a'_t \in \mathcal{A}} p_{\alpha,\theta}(a'_t|s_t,c_t) \nabla_{\theta_k} Q^*_{\theta_k}(s_t,a'_t)) \end{aligned} \tag{4}$$

where $\nabla_{\theta_k} Q^*_{\theta_k}(s,a)$ for each $k = 0,\ldots,K$ can be estimated by a set of fixed-point equations similar to the Bellman-optimality equations [29] which is commonly used in Maximum Likelihood-based IRL methods:

$$\begin{aligned} \nabla_{\theta_k} Q^*_{\theta_k}(s,a) = &\nabla_{\theta_k} \mathcal{R}_k(s) \\ &+ \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) \sum_{a' \in \mathcal{A}} \pi^*_{\theta_k}(s',a') \nabla_{\theta_k} Q^*_{\theta_k}(s',a') \end{aligned} \tag{5}$$

Calculating the gradient with respect to α, we have:

$$\begin{aligned} \nabla_\alpha \log p_{\alpha,\theta}(a_t|s_t,c_t) = \beta \sum_{k \in K} (&Q^*_{\theta_k}(s_t,a_t) \\ &- \sum_{a'_t \in \mathcal{A}} p_{\alpha,\theta}(a'_t|s_t,c_t) Q^*_{\theta_k}(s_t,a'_t)) \nabla_\alpha \phi_a(s_t,c_t)_k \end{aligned} \tag{6}$$

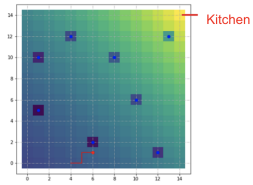
Notice that $(Q^*_{\theta_k}(s_t,a_t) - \sum_{a'_t} p_{\alpha,\theta}(a'_t|s_t,c_t) Q^*_{\theta_k}(s_t,a'_t))$ in equation (6) is exactly the conditional advantage under context c_t corresponding to the kth reward function. It measures how much better the action given by the policy is compared to the average result weighted by all the possible actions, and thus alpha is optimized to amplify the advantage of the observed expert's actions.

V. EXPERIMENTS

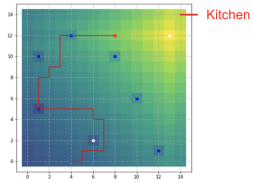
A. Scenario 1: Serve for a Cocktail Party

We first extend the scenario "Grab a Milk" in [8] to a *dynamic* setting which makes it more challenging. In this scenario, the agent serves as a waiter/waitress to navigate to the kitchen as its domain task, while at the same time, it carries sufficient drinks or snacks, and navigates to the guests when they want any, or avoids disturbing those guests that currently do not need any service on its way, as its pre-specified norms. [8] claims it challenging for IRL methods to solve similar tasks due to the exponentially growing state space of context information, and the number of expert demonstrations required. We show that our CNIRL is able to solve this extended scenario successfully, and make comparison to other benchmark algorithms.

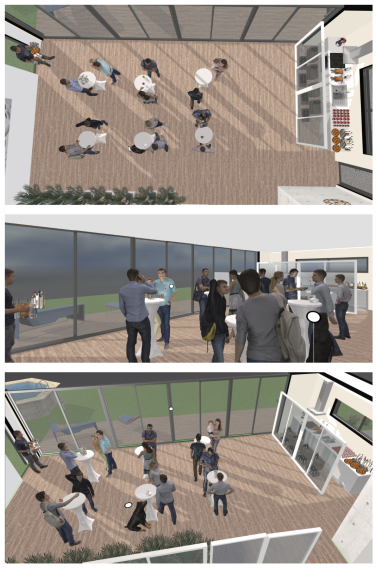
More specifically, in a 15 by 15 map, the agent starts from (4, 0) with the domain task to go to the goal position (14, 14) (the upper right corner) of kitchen (see Figure 2). There are a number of guests and they may want drink or snacks at *random* time steps. The agent is supposed to navigate to the guests whenever they are in need. For each time step, the domain state is the agent's location, and the context is each guest's status. If no guest is in need, the agent is supposed to go to the goal position and try to avoid all the guests; otherwise the agent is expected to always go serve the nearest guest in need. In this setting, the size of context space is *exponential* in the number of guests and their needs that change over time, but the number of reward functions grows linearly as the number of guests grows.



(a) Initially, no guest is in need (represented by blue dots), so the expert performs the domain goal to navigate to the kitchen while trying to avoid guests.



(b) Two guests want some snacks from the agent (represented by white dots). The expert goes towards the nearest guest to help while also trying to avoid other guests. After the agent touches them, they are marked to blue.



(c) Screenshots of a 3D simulation on an agent serving for a cocktail party.

Fig. 2: (a) and (b) are example expert behaviors. States with lighter colors have relatively high rewards and states with darker colors have lower ones. The red line shows the trajectory of the agent and red dot is its current position. (c) is a simulation for illustration.

[0]The implementation code: https://github.com/sophieyueguo/cnirl

Each reward function takes the form $\mathcal{R}_k(s) = \theta_k^T \phi(s)$, and we pre-determine $\phi(s)$ to be a binary vector consisting of whether the agent is at a guest position and the goal position, so the dimension of $\phi(s)$ is one plus the number of guests. We have multiple guests who begin to want snacks or drinks at time step $t \sim uniform(0, 40)$. 500 expert trajectories with maximum 40 time steps are generated. We set the architecture of ϕ_a to be 16-20-10 (number of units in different layers) with L_2 regularity factor of 10^{-5} on the weight matrices, where the input to the network consists of each status of a guest and the Euclidean distance to the agent.

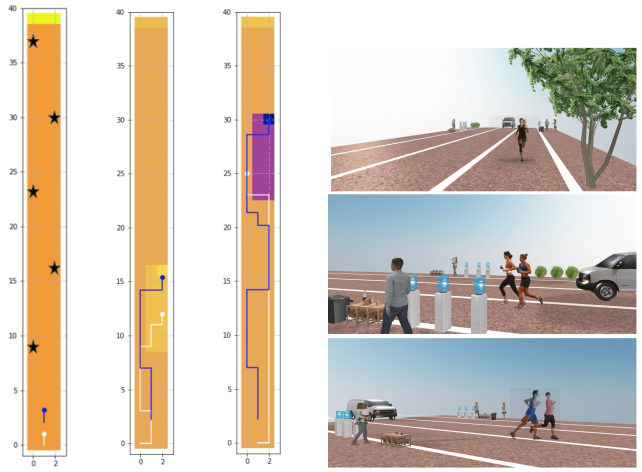
B. Scenario 2: Support a Marathon Runner

We have also developed a more challenging scenario called "Support a Marathon Runner", where the agent serves as a supply car that moves relatively slowly behind a marathon runner towards the goal location of the marathon. The runner would either take a break at the supply locations if she or he is tired, or pass by if otherwise. While the runner takes a break and stays in a supply location for a few seconds, the supply car is supposed to go to this location, but otherwise the supply car is expected to avoid this location to reduce the interruption since the runner does not need a break.

This scenario has the same "help" and "avoid" contextual information as in the previous scenario, but it is extended in terms of mobility of the humans that our agent want to help with. The supply car will only be considered as successfully helping the runner when it reaches the runner's location within the time that the runner stays there. However, the runner will only briefly stay, and move on towards the goal location no matter whether the supply car comes or not. On the other hand, "avoid" succeeds if the agent can tell the runner would only pass through the supply location, and does not go there.

As shown in Figure 3, in a world of 3 marathon tracks with 40 units of length, the runner starts from the middle lane at unit 2 and the car agent starts from the middle lane at unit 0. The domain task is to go to the goal location at unit 40 in any of the lanes. The runner has a fixed speed that is faster than the agent, and chooses the lane randomly when not considering passing by or staying in the supply locations. The duration of the runner to stay in the supply locations is 4 time steps. Whether the runner wants a "help" or "avoid" is the contextual information, and the number of all possible context values is exponential in the number of supply locations, but the number of reward functions grows linearly with respect to it.

The state s is the location of the agent, and the state feature $\phi(s)$ is a vector that denotes whether the state is near the supply location and if it is a goal location. We set 1 if it is the exact supply location, 0.5 if within visible range of 8 by 2 units, and 0 otherwise, resulting in its dimension being the number of supply locations plus one. 500 expert trajectories are generated with $\beta = 15$, and we set $\beta = 5$ when computing the parameters θ and α. The architecture of ϕ_a is the same as in scenario A.



(a) (b) (c) (d)

Fig. 3: An example of the expert's demonstrations in "Support a Marathon Runner". States with light colors, such as yellow, have relatively high rewards, and those with dark colors, such as blue and purple, have lower ones. The blue dot is the current location of the runner, and the blue line is her or his trajectory. The white dot and line are for the supply car agent. When the runner is not in the supply locations, the agent only goes towards the domain goal as it moves towards the end of the track as shown in (a), while in (b) and (c) it goes towards or roundabout the supply locations. The supply locations are located in the cells marked with black stars, and they are only shown in (a) for simplicity. (d) gives a 3D simulation on supporting a marathon runner.

C. Comparison with Expert Behaviors

In both static and moving scenarios, we first evaluate the learned optimal behavior calculated from the recovered reward function by comparing it with the norm-compliant expert's behavior. For the scenario of serving a guest, we calculate the average number of times the agent navigates to guests in need or not during one task. For every 10 iterations of training, we sample 500 trajectories using the trained parameters and test on the performance. The results are compared with those of the expert in Fig.4 (a).

Figure 4 (a) shows that the average number of helping steadily increases during training. Although at first the agent doesn't pay attention to avoid guests that are not in need, possibly because the agent tries to approach all guests to learn helping better, this is taken into account after a number of iterations. The agent's behavior is ultimately aligned with the expert's, which justifies our method's capability of learning norm-compliant behaviors under changing contexts.

In the scenario of supporting a marathon runner, for every 10 episodes of training, we sample 500 trajectories and test the same runner context for the expert and the trained agent. We calculate the number of times that the runner actually needs or does not need help, as well as whether the agent has come to the supply location or not. The ratio yields the success rates of helping (Figure 4 (b)) and failure rates of avoiding interruption (Figure 4 (c)).

It is clear from Figure 4 (b) that the agent is finally able to achieve the same success rate with the expert, with steadily increasing performance and occasional drawbacks. Figure 4 (c) shows that the failure rate starts from 0 at the beginning of the training when the agent has no knowledge of whether to help or avoid the runner. The agent gradually learns how to approach the runner, but is still uncertain in distinguishing between "help" and "avoid". Whenever it mistakes the need of avoiding for the need of helping, the failure occurs. It takes a few hundred episodes to converge to a relatively low rate of failure, where the agent can avoid going to the supply location when the runner has no need of being supplied. The success and failure rate results demonstrate that CNIRL is able to handle the harder case where there are two moving objects (the supply car agent and the runner) along with dynamically changing context.

D. Comparison with Other Benchmark IRL Algorithms

We implement three benchmark IRL algorithms, Bayesian IRL (BIRL) [30], Maximum Entropy IRL (MaxentIRL) [31], and Maximum Likelihood IRL (MLIRL) [32] for both the cases of serving for a cocktail party and supporting a marathon runner, and make comparisons with our CNIRL. First we show the performances of all algorithms on the cases where only one object is present, and then we show the run-time of them with respect to the varied number of objects. Finally we compare the performances of multiple objects solved by the algorithms being able to handle the resulting context complexity.

Figure 5 gives the performances by recording the probability of successfully helping a guest or the runner when needed and the probability of failing to avoid when not needed over the training process. The tasks for two scenarios only contain one guest or one supply location, and thus also serve as a sanity check for the three benchmark IRL algorithms since the context complexity is relatively small. According to the plots in (a) and (b), all algorithms can achieve the perfect 100% successful rate to help the guest or the runner when they are in need. For the cases where the agent is expected to avoid the guest or the runner as they do not want the agent to approach them, it is easier to learn serving for the party than supporting the runner. In (c), all the four algorithms are able to decrease the failure rate, with MaxentIRL performs relatively worse, while in (d), the failure rate to avoid the runner does not decrease for MLIRL or MaxentIRL due to their inability to differentiate avoiding from helping (they are still far lower than the successfully help rate), while the activation function makes CNIRL perfectly do so, and thus keeps a zero failure rate throughout the training. The trend of BIRL in Figure 5 (c) is similar to the CNIRL one in Figure 4 (c) and has the same reason. Additionally, the 200 test cases for both of the scenarios remain the same over the training for all algorithms, and thus we claim that CNIRL, along with other benchmark IRL algorithms, is able to solve these two tasks with simple context complexity.

Next, we show the run-time and the log value of run-time of each iteration performed on our computer (Intel(R)

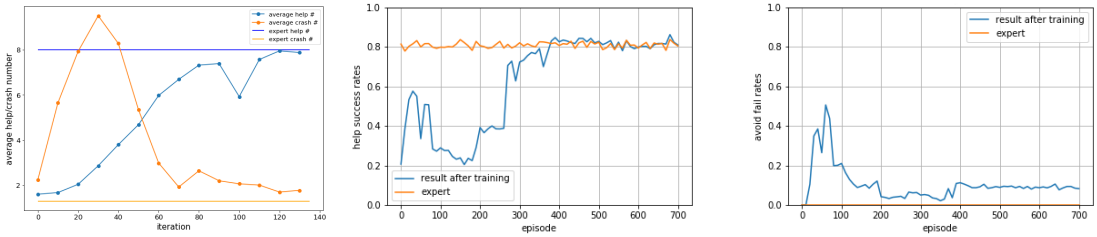


Fig. 4: (a) Average number of times the agent helps guests over training time, compared with the expert. (b) The success rates of offering help when the runner needs in the supply locations. (c) The failure rates of avoiding the supply locations and interrupting the runner when the runner does not need a break in the supply locations. Notice that the expert has a constant 0.0 failure rate that might be hard to view.

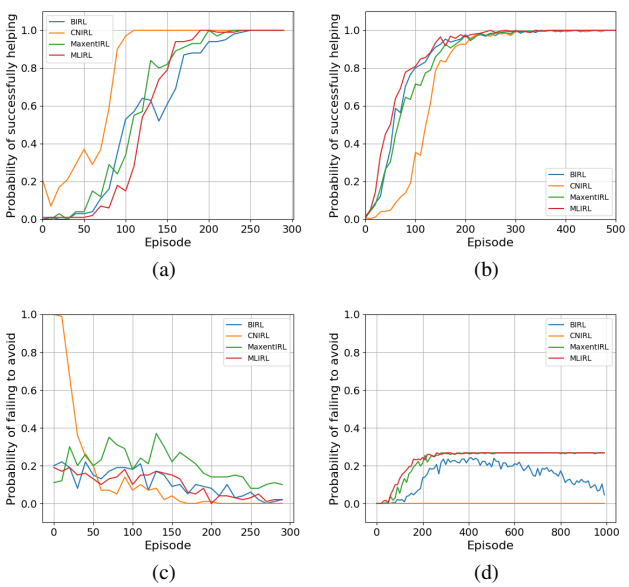


Fig. 5: Performances of the four IRL algorithms for the case where there is one object. (a) and (c) are for serving a cocktail party, while (b) and (d) are for supporting a marathon runner. (a) and (b) are probabilities of successfully helping the guest or the runner in need over training, while (c) and (d) are those of failing to avoid them when not.

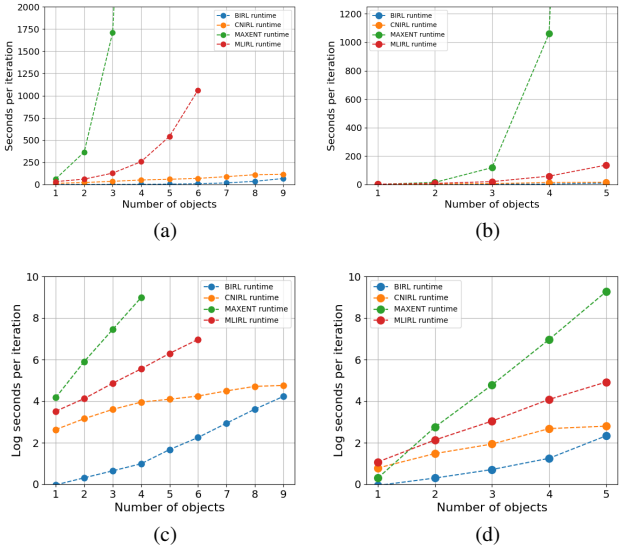


Fig. 6: Run-time per iteration and the log value of run-time per iteration of four IRL algorithms with respect to varied number of objects. (a) and (c) are for serving a cocktail party, while (b) and (d) are for supporting a marathon runner. (a) and (b) are seconds used per iteration, while (c) and (d) are the log values of them respectively.

Xeon(R) CPU E5-2660 v3 @ 2.60GHz with 128 GB RAM) for the four algorithms in Figure 6. According to the results, BIRL, MaxentIRL, and MLIRL all show exponential growth of run-time with respect to the increasing number of objects for both scenarios. Among them, BIRL has the shortest run-time per iteration because it utilizes policy iteration and avoids the expensive value iteration, but the exponential growth of run-time is still unavoidable. However, CNIRL clearly demonstrates a linear growth in run-time, and even it uses value iteration, the growth of run-time is not exponential as in MaxentIRL or MLIRL. Moreover, even though CNIRL is slower than BIRL, the trend shows it is expected to catch up with it with more guests or supply locations.

Due to the high cost of run-time shown in Figure 6 for MLIRL and MaxentIRL, we only compare the performances of CNIRL and BIRL with 5 guests or supply locations. As demonstrated in Figure 7, (a) and (c) indicates the ability of both BIRL and CNIRL to solve the task of serving 5 guests in the cocktail party, both with 200 training trajectories and the same testing conditions. (b) and (d) shows CNIRL is able to learn to help the runner in need while at the same time maintains a relatively low probability to interrupt when the runner is not in need, however, this is difficult for BIRL to learn. With the same 500 training trajectories, BIRL performs worse when learning to help, and has not reached the point that it increases the probability of failing to avoid to learn, i.e., it never catches up with the runner to learn differentiating the help or avoid, resulting in a constant failure rate in (d). We also compare the performances with only 100 training trajectories, and we could witness the improvement with respect to more training trajectories, but BIRL is still inferior than CNIRL on this task.

VI. Conclusion and Future Work

We propose CNIRL, a scalable framework for developing norm-compliant agents through learning from norm-compliant expert demonstrations with changing contexts. In the future, we are interested in incorporating ownership

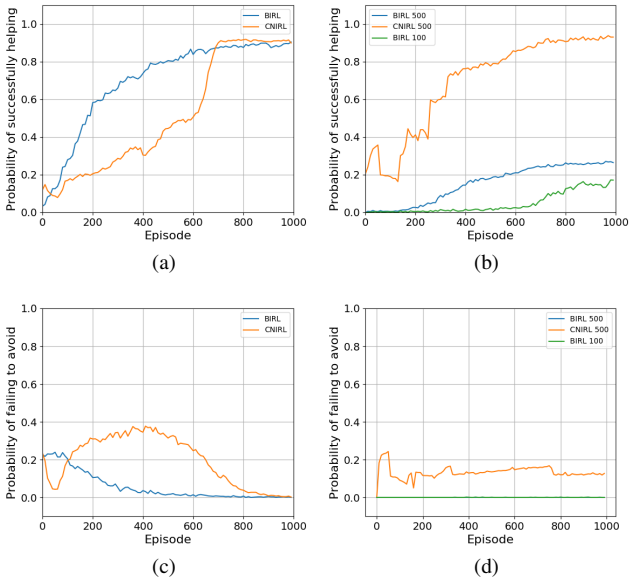


Fig. 7: Performances of BIRL and CNIRL for the case where there are five objects. (a) and (c) are for serving a cocktail party that has 5 guests, while (b) and (d) are for supporting a marathon runner that has 5 supply locations. (a) and (b) are probabilities of successfully helping the guests or the runner in need over training, while (c) and (d) are those of failing to avoid the guests or the runner when not in need.

of norms into context[33], differentiating bad examples for more structured prohibition norms, and thus further lowering the difficulty of getting trajectories to train the agent.

ACKNOWLEDGMENT

This work has been supported by NSF Award #1724222 and AFOSR/AFRL FA9550-18-1-0251.

REFERENCES

[1] G. Brennan, L. Eriksson, R. E. Goodin, and N. Southwood, Explaining norms. Oxford University Press, 2013.

[2] M. S. Fagundes, S. Ossowski, J. Cerquides, and P. Noriega, "Design and evaluation of norm-aware agents based on normative markov decision processes," International Journal of Approximate Reasoning, vol. 78, pp. 33–61, 2016.

[3] J. Oh, F. Meneguzzi, K. Sycara, and T. J. Norman, "Prognostic normative reasoning," Engineering Applications of Artificial Intelligence, vol. 26, no. 2, pp. 863–872, 2013.

[4] M. S. Fagundes, S. Ossowski, M. Luck, and S. Miles, "Using normative markov decision processes for evaluating electronic contracts," AI Communications, vol. 25, no. 1, pp. 1–17, 2012.

[5] M. S. Fagundes, S. Ossowski, J. Cerquides, and P. Noriega, "Design and evaluation of norm-aware agents based on normative markov decision processes," International Journal of Approximate Reasoning, vol. 78, pp. 33–61, 2016.

[6] V. Krishnamoorthy, W. Luo, M. Lewis, and K. Sycara, "A computational framework for integrating task planning and norm aware reasoning for social robots," in 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). IEEE, 2018, pp. 282–287.

[7] C. E. Sezener, "Inferring human values for safe agi design," in International Conference on Artificial General Intelligence. Springer, 2015, pp. 152–155.

[8] Y.-H. Wu and S.-D. Lin, "A low-cost ethics shaping approach for designing reinforcement learning agents," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[9] B. F. Malle, M. Scheutz, and J. L. Austerweil, "Networks of social and moral norms in human and robot agents," in A world with robots. Springer, 2017, pp. 3–17.

[10] H. Aarts and A. Dijksterhuis, "The silence of the library: environment, situational norm, and social behavior." Journal of personality and social psychology, vol. 84, no. 1, p. 18, 2003.

[11] C. Balkenius and S. Winberg, "Cognitive modeling with context sensitive reinforcement learning," Proceedings of AILS, vol. 4, pp. 10–19, 2004.

[12] D. Kasenberg, T. Arnold, and M. Scheutz, "Norms, rewards, and the intentional stance: Comparing machine learning approaches to ethical training," in Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. ACM, 2018, pp. 184–190.

[13] V. Sarathy, M. Scheutz, Y. N. Kenett, M. Allaham, J. L. Austerweil, and B. F. Malle, "Mental representations and computational modeling of context-specific human norm systems." in CogSci, vol. 1, 2017, pp. 1–1.

[14] V. Sarathy, M. Scheutz, and B. F. Malle, "Learning behavioral norms in uncertain and changing contexts," in 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, 2017, pp. 000 301–000 306.

[15] G. Shafer, A mathematical theory of evidence. Princeton university press, 1976, vol. 42.

[16] C. Balkenius and J. Morén, "A computational model of context processing," in 6th international conference on the simulation of adaptive behaviour, 2000.

[17] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 897–904.

[18] J. Choi and K.-E. Kim, "Nonparametric bayesian inverse reinforcement learning for multiple reward functions," in Advances in Neural Information Processing Systems, 2012, pp. 305–313.

[19] B. Michini and J. P. How, "Bayesian nonparametric inverse reinforcement learning," in Joint European conference on machine learning and knowledge discovery in databases. Springer, 2012, pp. 148–163.

[20] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Inverse reinforcement learning with locally consistent reward functions," in Advances in neural information processing systems, 2015, pp. 1747–1755.

[21] A. Šošić, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning via nonparametric subgoal modeling," in 2018 AAAI Spring Symposium Series, 2018.

[22] E. Uchibe, M. Asada, and K. Hosoda, "Behavior coordination for a mobile robot using modular reinforcement learning," in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96, vol. 3. IEEE, 1996, pp. 1329–1336.

[23] N. Sprague and D. Ballard, "Multiple-goal reinforcement learning with modular sarsa (0)," 2003.

[24] S. J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 656–663.

[25] C. A. Rothkopf and D. Ballard, "Credit assignment in multiple goal embodied visuomotor behavior," frontiers in Psychology, vol. 1, p. 173, 2010.

[26] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in Advances in Neural Information Processing Systems, 2011, pp. 19–27.

[27] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," arXiv preprint arXiv:1507.04888, 2015.

[28] G. H. John, "When the best move isn't optimal: Q-learning with exploration," in AAAI. Citeseer, 1994, p. 1464.

[29] G. Neu and C. Szepesvári, "Apprenticeship learning using inverse reinforcement learning and gradient methods," arXiv preprint arXiv:1206.5264, 2012.

[30] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning." in IJCAI, vol. 7, 2007, pp. 2586–2591.

[31] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in Aaai, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[32] M. C. Vroman, "Maximum likelihood inverse reinforcement learning," Ph.D. dissertation, Rutgers University-Graduate School-New Brunswick, 2014.

[33] Z.-X. Tan, J. Brawer, and B. Scassellati, "That's mine! learning ownership relations and norms for robots," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 8058–8065.