Inverse norm conflict resolution

Daniel Kasenberg and **Matthias Scheutz**
Department of Computer Science
Tufts University
Medford, MA 02155

Abstract

In previous work we provided a "norm conflict resolution" algorithm allowing agents in stochastic domains (represented by Markov Decision Processes) to "maximally satisfy" a set of moral or social norms, where such norms are represented by statements in linear temporal logic (LTL). This required that the agent designer provide *weights* specifying the relative importance of each norm. In this paper, we propose an "inverse norm conflict resolution" algorithm for learning these weights from demonstration. This approach minimizes a cost function based on the relative entropy between a policy encoding the observed behavior and a policy representing optimal norm-following behavior. We demonstrate the effectiveness of the algorithm in a simple GridWorld domain.

Introduction

As artificial agents are increasingly being considered and employed in tasks requiring complex decision-making capabilities and social interactions with humans, providing such agents with the ability to learn, reason about, and obey human moral and social norms is increasingly being seen as a priority.

One popular approach to imbuing artificial agents with moral and social norms has been to represent these norms as *reward functions*. This allows these norms to be easily *obeyed* (via reinforcement learning and planning algorithms) and *learned* from behavior (via inverse reinforcement learning). Nevertheless, we have argued (Arnold, Kasenberg, and Scheutz 2017) that reward-based representation is insufficient because (1) Markovian reward functions cannot capture temporally complex moral and social norms; (2) reward functions are difficult to interpret; and (3) reward functions can be difficult to generalize to novel environments.

An alternative approach is to represent norms explicitly in some logical language. Representing norms in logic allows greater temporal complexity, greater interpretability, and greater generalizability to new worlds (provided that those worlds share a basic set of propositions). We have used linear temporal logic (LTL) to represent moral and social norms, and have demonstrated how an agent might learn such a norm from behavior (Kasenberg and Scheutz 2017). This approach applies in the same stochastic domains (namely, Markov Decision Processes) on which reward-based approaches apply.

One of the difficulties learning human moral and social norms is the fact that humans have complicated networks of such norms, and one person may possess norms that contradict each other, either in theory or in practice. Most previous approaches to value alignment deal with such "norm conflicts" only implicitly – in the event of norm conflicts, inverse reinforcement learning agents will settle upon a reward function that incentivizes what may in reality be a compromise between conflicting principles. Similarly, our previous approach to learning norms from behavior supposes that the "demonstrator" is driven by one monolithic norm.

In recent work on norm conflict resolution (Kasenberg and Scheutz 2018), in which we described an algorithm by which an agent in a Markov Decision Process could balance the demands of several, possibly conflicting, LTL statements representing moral and social norms. That algorithm required the agent designer to provide along with each norm a *weight* indicating the relative importance of that norm.

In this paper, we develop an algorithm for "inverse norm conflict resolution" – determining the relative importance of an agent's norms by observing that agent's behavior in norm conflicts. Our algorithm minimizes a cost function based on the relative entropy (or Kullback-Leibler divergence) between the observed behaviors and the optimal norm-following behavior.

We begin by describing Markov Decision Processes, LTL, and our norm conflict resolution algorithm; we then explain the inverse norm conflict resolution algorithm. We demonstrate this algorithm in a simple GridWorld domain, discuss the limitations of our approach and directions for future work, and conclude with a summary of our results.

Related work

This paper draws heavily upon our previous work on norm conflict resolution (Kasenberg and Scheutz 2018) (outlined in the "Preliminaries" section). This work in turn is inspired by previous approaches for planning with (single) temporal logic objectives in stochastic environments (Ding et al. 2011) and work on planning with partially satisfiable LTL objectives (Lahijanian et al. 2015, for example).

The present approach is related to our previous work on inferring temporal logic specifications from demonstrations. In particular, in (Kasenberg and Scheutz 2017) we formulated learning a temporal logic specification as a multi-objective optimization problem, which we solved by genetic programming. Subsequently (Vazquez-Chanlatte et al. 2017) formulated a similar problem, but used bounded specifications (where the truth values can be learned in finite time), and used lattice search to find specifications. Although both of these approaches learn temporally complex specifications from demonstration, both also assume a single monolithic objective.

Our work also bears some resemblance to inverse reinforcement learning (IRL). For example, (Ng and Russell 2000) first employ linear function approximation and learn a set of weights parametrizing a reward function. Other approaches (Boularias, Kober, and Peters 2011, for example) use relative entropy, which features in our approach.

Work on multi-objective IRL especially resembles our approach. For example, (Babes et al. 2011) uses expectation-maximization to cluster observed trajectories and infers a separate reward function for each cluster. (Saitake and Arai 2016) learn from demonstration the weights of several objectives in order to generate a single Pareto-optimal policy.

Preliminaries

Markov Decision Processes

We are interested in agent behavior in stochastic domains. We will specify such domains as *Markov Decision Processes* (MDPs).

We define an MDP $\mathcal{M}$ as a tuple $\langle S, U, A, P, s_0, \gamma, \mathcal{L} \rangle$, where:

- S is a finite set of *states*;
- U is a finite set of *actions*;
- $A : S \to 2^U$ determines what actions are available in which states;
- P is the *transition function*, a probability distribution over S given a previous state and action, so that $P(s'|s, a)$ is the probability of transitioning to state s' given that the agent was previously in s and executed action a;
- $s_0 \in S$ is an initial state;
- $\gamma \in [0, 1)$ is a *discount factor*;
- $\mathcal{L} : S \to 2^\Pi$ is the *labeling function*, where Π is a set of atomic propositions; this maps each state s to the subset of Π that are true in s.

Note that the inclusion of $\mathcal{L}$ is nonstandard for MDPs, but is important to the task of linking them to temporal logic statements. Most MDP formulations also include a reward function R, but this will not be necessary for this paper.

An agent in a Markov Decision process begins in the initial state s_0. At each time step t, the agent chooses some action a_t from their current state s_t, and the environment transitions according to $P(\cdot|s, a)$ into the new state s_{t+1}.

A *trajectory* specifies the path of an agent through an MDP. In particular, a *finite* trajectory consists of a set of state-action pairs followed by a final state, e.g. $\tau = s_0, a_0, s_1, a_1, \cdots, s_T, a_T, s_{T+1}$. An infinite trajectory is an infinite sequence of state-action pairs, e.g. $\tau = s_0, a_0, s_1, a_1, \cdots$. Let $\mathrm{Traj}_\mathcal{M}$ be the set of finite trajectories in $\mathcal{M}$, and let $\mathrm{ITraj}_\mathcal{M}$ be the set of infinite trajectories.

We define a *policy* in an MDP as a mapping from finite trajectories to probability distributions over actions; that is, $\pi : \mathrm{Traj}_\mathcal{M} \to \Delta(A)$, where Δ is the probability simplex. That is, a policy specifies what the agent will do given its history, and the agent is allowed to choose randomly between actions. A policy π is *stationary* if it depends only on the agent's current state; that is, $\pi : S \to \Delta(A)$.

Linear temporal logic

The present work assumes that moral and social norms are represented in *Linear Temporal Logic (LTL)* (Pnueli 1977). LTL is a propositional logic that linearly encodes time. An LTL formula over the set of atomic propositions Π is generated according to the following syntax:

$$\begin{aligned}\phi ::= &\top \mid \bot \mid p, \text{where } p \in \Pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \\ &\phi_1 \to \phi_2 \mid \mathsf{X}\phi \mid \mathsf{G}\phi \mid \mathsf{F}\phi \mid \phi_1 \,\mathsf{U}\, \phi_2\end{aligned}$$

In this case, $\mathsf{X}\phi$ means "ϕ will be true in the next time step"; $\mathsf{G}\phi$ means "ϕ will be true at every present and future time step"; $\mathsf{F}\phi$ means "ϕ will be true at some present or future time step"; and $\phi_1 \,\mathsf{U}\, \phi_2$ means "ϕ_1 will be true until ϕ_2 is true".

The truth of each LTL statement is defined over an infinite sequence of *valuations* $\sigma_0, \sigma_1, \sigma_2, \cdots$, where $\sigma_i \subseteq \Pi$. This indicates a correspondence with Markov Decision Processes: for each infinite trajectory $\tau = s_0, a_0, s_1, a_1, \cdots$ of an agent behaving in an MDP, we can see that this trajectory induces an infinite sequence of valuations $\mathcal{L}(s_0), \mathcal{L}(s_1), \cdots$ over which any LTL formula ϕ can be evaluated.

Norm conflict resolution

We now briefly outline our norm conflict resolution approach in (Kasenberg and Scheutz 2018), which allows agents in MDPs to "maximally satisfy" moral and social norms represented in LTL.

We suppose that moral and social norms are represented by LTL statements. Consider an agent in a Markov Decision Process $\mathcal{M}$. The agent may possess several such norms $\phi_1, \cdots, \phi_N$. The goal of a norm-obeying agent is to satisfy these norms; ideally, the agent performs actions resulting in a trajectory $\tau \in \mathrm{ITraj}_\mathcal{M}$ such that $\tau \vDash \phi_1, \cdots, \tau \vDash \phi_N$.

Because of the stochasticity of the domain, the agent cannot in general guarantee a trajectory τ such that $\tau \vDash \phi_i$ for all $i \in \{1, \cdots, N\}$. Thus the problem of obeying all norms must be relaxed in some way. One obvious relaxation is to maximize the *probability* of satisfying $\phi_1, \cdots, \phi_N$; that is, finding some (nonstationary) policy

$$\pi^* = \arg\max_\pi \Pr_\pi(\{\tau : \tau \vDash \phi_1, \cdots, \tau \vDash \phi_N\}) \qquad (1)$$

This approach, however, is not ideal when the probability of the agent satisfying all its norms is zero. We refer to this scenario as a *norm conflict*.

When norm conflicts occur, simply aiming to maximize the probability of obeying all norms does not suffice to choose a policy (since when the maximum probability is zero, this is satisfied by every possible policy). Nevertheless, it is generally desirable for the agent to behave reasonably in the event of conflict. To do this, we relax the problem by defining a notion of "badness" of behavior trajectories: we define a notion of *violation cost*.

Our approach is to allow the agent to "ignore" some time steps t with respect to each norm, but to pay a cost each time this occurs.

More formally, consider some infinite trajectory $\tau = s_0, a_0, s_1, a_0, \cdots$. For any set of integers $N \subset \mathbb{Z}_{\geq 0}$, we can remove the set of time steps indexed by elements of N from the trajectory; we let $\tau \setminus N$ be the resulting trajectory. For example, if $N = \{0, 2\}$, then $\tau \setminus N$ is the trajectory obtained by removing time steps 0 and 2 from τ:

$$\tau \setminus \{0, 2\} = s_1, a_1, s_3, a_3, s_4, a_4, \cdots$$

We define the *violation cost* of some trajectory τ with respect to a norm ϕ to be the (discounted) minimum number of time steps that must be omitted in this way in order for τ to satisfy ϕ:

$$\mathrm{Viol}_{\phi}(\tau) = \min_{\substack{N \subseteq \mathbb{Z}_{\geq 0} \\ \tau \setminus N \vDash \phi}} \sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{t \in N} \tag{2}$$

Note that if $\tau \vDash \phi$, then $\mathrm{Viol}_{\phi}(\tau) = 0$ (so that obeying a norm incurs no violation cost). The use of the discount factor ensures that the violation cost of any trajectory τ is finite. This is necessary to allow comparison of trajectories that each violate a norm infinitely many times.

The critical assumption of this notion of violation cost is that the "badness" of behaviors that violate a norm depend primarily on the *duration* (in time steps) of the violation. For real moral norms this is likely not the case; we believe our approach to be sufficiently general to accommodate other notions of violation cost.

We define a *norm system* as a pair $(\Phi, \mathbf{w})$ where Φ is a tuple of LTL norms $\Phi = (\phi_1, \cdots, \phi_N)$ and $\mathbf{w} = (w_1, \cdots, w_N)^T$ is a corresponding weight vector. Each w_i is a positive real number representing the importance of the norm ϕ_i. It is these weights over which our inverse norm conflict resolution algorithm will be optimizing.

The violation cost of a trajectory τ with respect to a norm system $\mathcal{N}$ is simply the weighted sum of the violation costs of τ with respect to the constituent norms $\phi_1, \cdots, \phi_N$:

$$\mathrm{Viol}_{(\Phi, \mathbf{w})}(\tau) = \sum_{i=1}^{N} w_i \mathrm{Viol}_{\phi_i}(\tau) \tag{3}$$

Our aim in (Kasenberg and Scheutz 2018) is to find the (general) policy that minimizes the expected violation cost (over all trajectories). While this policy is not stationary in the underlying MDP, it *is* stationary in the *product* MDP $\mathcal{M}^{\otimes}$ between $\mathcal{M}$ and the DRAs of each norm $\mathcal{D}(\phi_1), \cdots, \mathcal{D}(\phi_N)$.

In order to maximally satisfy a norm system $\mathbf{N}$, the agent must perform some graph theoretic operations involving *end components*. This is discussed (along with some complications) in (Kasenberg and Scheutz 2018). Thois divides $S^{\otimes}$ into S_{bad}, the set of product states such that if $s \in S_{bad}$, no amount of suspending norms will allow the DRA states to become accepting. We let $S_{good} = S^{\otimes} \setminus S_{bad}$.

Due to the stationarity of the optimal policy in $\mathcal{M}^{\otimes}$, the optimal expected violation cost beginning from an individual product state $(s, q_1, \cdots, q_N)$ and immediately performing action a satisfies the following Bellman-like equation:

$$\mathrm{Viol}_{(\Phi, \mathbf{w})}((s, q_1, \cdots, q_N), a) =$$
$$\begin{cases} \sum\limits_{s' \in S} P(s'|s, a) \min_{\tilde{a} \in \{0,1\}^n} \sum\limits_{i=1}^{N} \Big(w_i \tilde{a}_i & \\ \quad + \gamma \min_{a \in A} \mathrm{Viol}_{(\Phi, \mathbf{w})}((s', q_1', \cdots, q_n'), a) \Big) & \text{if } s \in S_{good} \\ \frac{1}{1-\gamma} \sum\limits_{i=1}^{N} w_i & \text{otherwise} \end{cases}$$

where

$$q_i' = \begin{cases} q_i & \text{if } \tilde{a}_i = 1 \\ \delta_i(q_i, \mathcal{L}(s')) & \text{otherwise} \end{cases}$$

This equation may require some interpretation. $\tilde{a}_i$ represents whether the norm ϕ_i will be "suspended" (the current time step omitted) after seeing the new state s'. If the agent does decide to suspend ϕ_i ($\tilde{a}_i = 1$), the DRA $\mathcal{D}(\phi_i)$ stays in its state q_i; if the agent chooses not to do so ($\tilde{a}_i = 0$) then $\mathcal{D}(\phi_i)$ transitions normally. The agent minimizes not only over which action it performs, but also over its choice of which norms to suspend at any given time step.

This is a Bellman-like equation, and so we can compute the optimal expected value from any product state using value iteration, with the following update equation:

$$\begin{aligned} &\mathrm{Viol}^{(k+1)}_{(\Phi, \mathbf{w})}((s, q_1, \cdots, q_N), a) \leftarrow \\ &\qquad \sum_{s' \in S} P(s'|s, a) \min_{\tilde{a} \in \{0,1\}^n} \sum_{i=1}^{N} \Big(w_i \tilde{a}_i \\ &\qquad\qquad + \gamma \mathrm{Viol}^{(k)}_{(\Phi, \mathbf{w})}((s', q_1', \cdots, q_n')) \Big) \end{aligned} \tag{4}$$

States in $S^{\otimes}_{bad}$ have their violation cost initialized to the maximal violation cost $\frac{1}{1-\gamma} \sum\limits_{i=1}^{N} w_i$ and *not* updated during value iteration.

Once the optimal expected violation cost from each product state is computed by running (4) to convergence, the optimal set of actions from each product state $s^{\otimes}$ can be computed by $\mathrm{argmin}_{a \in A} \mathrm{Viol}_{(\Phi, \mathbf{w})}(s^{\otimes})$. This yields an *optimal product-space policy* $\pi^{\star}_{(\Phi, \mathbf{w})}$ that uniformly selects one of the optimal actions from the observed product state.

Finally, at each time step t the agent must use its history to determine its current product state $s_t^{\otimes}$. This can be thought of as a sort of filtering, and can be done in time constant in the number of time steps. At each t, the agent computes $s_t^{\otimes}$ and selects an action a with probability $\pi^{\star}_{(\Phi,\mathbf{w})}(s_t^{\otimes}, a)$.

Expected violation cost can also be computed for an individual policy, by replacing the $\min_{a\in A}(\cdot)$ in (4) with an expectation over the policy ($\sum\limits_{a\in A} \pi(s,a) \times (\cdot)$). We will denote by $\mathrm{Viol}^{\pi}_{\phi}$ the expected violation cost of the policy π with respect to the norm ϕ.

It will be helpful to define the state-action violation cost *vector* as the vector whose entries are the violation cost of the optimal policy $\pi^{\star}_{(\Phi,\mathbf{w})}$ with respect to each individual norm in ϕ:

$$\mathbf{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a) = \begin{bmatrix} \mathrm{Viol}^{\pi^{\star}_{(\Phi,\mathbf{w})}}_{\phi_1}(s^{\otimes}, a) \\ \vdots \\ \mathrm{Viol}^{\pi^{\star}_{(\Phi,\mathbf{w})}}_{\phi_N}(s^{\otimes}, a) \end{bmatrix}$$

Note that $\mathrm{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a) = \mathbf{w}^T \mathbf{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a)$.

We can use value iteration to compute $\mathbf{Viol}_{(\Phi,\mathbf{w})}$ directly (rather than computing the scalar violation cost $\mathrm{Viol}_{(\Phi,\mathbf{w})}$). The computation is straightforward, but the mathematical description is not; we omit the details of the computation.

Inverse norm conflict resolution

The problem of interest in this paper is: given some tuple $\Phi = (\phi_1, \cdots, \phi_N)$ of norms represented in LTL and some set of (finite) behavior trajectories $\tau^{(1)}, \cdots, \tau^{(m)}$, compute some vector of weights $\mathbf{w} = (w_1, \cdots, w_N)^T$ that "best explain" the observed behavior.

We shall constrain the weights to sum to one (since multiplying all weights by a constant will not change the behavior). We will also require all weights to be non-negative.

We now clarify the notion of "best explain". The weights $w_1, \cdots, w_N$ inferred should be such that the *observed* behavior should be as similar as possible to the behavior of the agent optimally satisfying the norm system $(\Phi, \mathbf{w})$

The similarity between the observed behavior and optimal norm-folllowing behavior may be defined in terms of the *relative entropy*, or *Kullback-Leibler divergence* (Kullback and Leibler 1951), between the observed product-space policy π^o and the optimal product-space policy given the weights $\pi^{\star}_{\mathcal{N}_{\mathbf{w}}}$ (where $\mathcal{N}_{\mathbf{w}} := ((w_1, \phi_1), \cdots, (w_N, \phi_N))$). The relative entropy between two probability distributions P and Q is defined by

$$D(P \mid\mid Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{5}$$

The relative entropy $D(P \mid\mid Q)$ can be thought of as a measure of the "distance" between P and Q (although D does not qualify as a distance function, since in general $D(P \mid\mid Q) \neq D(Q \mid\mid P)$).

Recall that a policy defines a probability distribution over actions for each state. Thus, at each state s, we can compute the relative entropy $D(\pi_1(s,\cdot) \mid\mid \pi_2(s,\cdot))$ between two stationary policies π_1 and π_2.

The two quantities will be the *observed product-space policy* π^o, and a *Boltzmann policy* defined based on the violation cost.

The observed product-space policy π^o is computed from the trajectories. In particular, each trajectory is lifted into a *product-space* trajectory by mapping each state s_t to a product state $(s_t, q_1, \cdots, q_N)$ considering the agent's history. Details about this process can be found in (Kasenberg and Scheutz 2017). After lifting trajectories to the product space, construct the "permissible action map" $A^* : S^{\otimes} \to 2^A$ as follows:

- If a given product state $s^{\otimes}$ occurs in any product-space trajectory, $A^*(s^{\otimes})$ is the set of all actions the demonstrator performed at least once in $s^{\otimes}$.
- If a given product state $s^{\otimes}$ never occurs in any observed trajectory, $A^*(s^{\otimes}) = A$.

Since the demonstrator is assumed to be acting optimally with respect to its norms, the apprentice assumes that if the demonstrator performed a in product state $s^{\otimes}$, then a is permissible from $s^{\otimes}$; A^* maps each product state to the set of actions assumed to be permissible in that product state.

The observed product-space policy is then the uniform policy over the set of all permissible actions:

$$\pi^o(s^{\otimes}, a) = \begin{cases} \frac{1}{|A^*(s^{\otimes})|} & \text{if } a \in A^*(s^{\otimes}) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The Boltzmann policy is a policy that strongly favors actions with higher long-term reward. In this case, the policy favors actions with lower expected violation cost. In this way it approximates the optimal policy given $(\Phi, \mathbf{w})$:

$$\begin{aligned} \pi^B_{(\Phi,\mathbf{w})}(s^{\otimes}, a) &= \frac{\exp\{-\beta \mathrm{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a)\}}{\sum\limits_{a'\in A} \exp\{-\beta \mathrm{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a')\}} \\ &= \frac{\exp\{-\beta \mathbf{w}^T \mathbf{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a)\}}{\sum\limits_{a'\in A} \exp\{-\beta \mathbf{w}^T \mathbf{Viol}_{(\Phi,\mathbf{w})}(s^{\otimes}, a)\}} \end{aligned} \tag{7}$$

We use the Boltzmann policy instead of the optimal policy because (1) it is differentiable in the violation cost, whereas the optimal policy is usually not, and (2) it gives nonzero probability to all actions, and relative entropy is undefined when its second argument is zero. As $\beta \to \infty$, the Boltzmann policy approximates the optimal policy arbitrarily well.

Since this is defined over individual states and we need a measure over the entire product MDP $\mathcal{M}^{\otimes}$, we propose treating the relative entropy as a cost function defined over each state:

$$C_{\mathbf{w}}(s^{\otimes}) = D(\pi^o(s^{\otimes}, \cdot) \mid\mid \pi^B_{(\Phi,\mathbf{w})}(s^{\otimes}, \cdot)) \tag{8}$$

We shall then define the objective function in terms of

$$\mathrm{Obj}(\mathbf{w}) = \mathbb{E}_{\pi^o}\left[\sum_{t=0}^{\infty} \gamma^t C_{\mathbf{w}}(s_t^{\otimes})\right] \tag{9}$$

Algorithm 1 Inverse norm conflict resolution

```
function InverseNCR
    Instantiate $\mathcal{M}^\otimes$ and follow steps in (Kasenberg and Scheutz 2018) to compute $S^\otimes_{bad}$
    Compute the observed product-space policy $\pi^o$
    repeat
        for $s^\otimes \in S^\otimes \backslash S^\otimes_{bad}$ do
            for $a \in A$ do
                Compute $\mathbf{Viol}^{(k+1)}_{(\Phi,\mathbf{w})}(s^\otimes, a)$ using previous estimates $\mathbf{Viol}^{(k)}_{(\Phi,\mathbf{w})}(\cdot,\cdot)$
            end for
            $V^{(k+1)}(s^\otimes) \leftarrow C_{\mathbf{w}}(s^\otimes) + \gamma \sum_{a\in A} \pi^o(s^\otimes,a) \sum_{s^{\otimes\prime}\in S^\otimes} P^\otimes(s^{\otimes\prime}|s^\otimes,a) V^{(k)}(s^{\otimes\prime})$
            $\mathbf{\Delta}^{(k+1)}(s^\otimes) \leftarrow \tilde{\nabla}_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes) + \gamma \sum_{a\in A} \pi^o(s^\otimes,a) \sum_{s^{\otimes\prime}\in S^\otimes} P^\otimes(s^{\otimes\prime}|s^\otimes,a) \mathbf{\Delta}^{(k)}(s^{\otimes\prime})$
        end for
        $\hat{\mathbf{w}} \leftarrow \mathbf{w}^{(k)} - \eta \mathbf{\Delta}^{(k+1)}(s_0^\otimes)$
        $\mathbf{w}^{(k+1)} \leftarrow$ Project $\hat{\mathbf{w}}$ onto $\left\{\mathbf{w} : \sum_{i=1}^{N} w_i = 1, w_i \geq 0\right\}$
        $k \leftarrow k + 1$
    until $\mathbf{w}$, $V$, $\mathbf{\Delta}$, $\mathbf{Viol}_{(\Phi,\mathbf{w})}$ all converge
    return $\mathbf{w}^{(k)}$
end function
```

Note that $\mathrm{Obj}(\mathbf{w}) = V_{\mathbf{w}}(s_0^\otimes)$ where $s_0^\otimes$ is the initial state of $\mathcal{M}^\otimes$, and $V_{\mathbf{w}}$ is a function satisfying the following Bellman equation:

$$V_{\mathbf{w}}(s^\otimes) = C_{\mathbf{w}}(s^\otimes) + \gamma \sum_{a\in A} \left(\pi^o(s^\otimes,a) \sum_{s^{\otimes\prime}\in S^\otimes} P^\otimes(s^{\otimes\prime}|s^\otimes,a) V_{\mathbf{w}}(s^{\otimes\prime}) \right) \quad (10)$$

Equation (10) can be used as an update equation, and so for any vector of weights $\mathbf{w}$, this objective can be computed via value iteration.

In order to optimize weights, we note that the gradient of (10) can be taken with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} V_{\mathbf{w}}(s^\otimes) = \nabla_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes) + \gamma \sum_{a\in A} \pi^o(s^\otimes,a) \sum_{s^{\otimes\prime}\in S^\otimes} P^\otimes(s^{\otimes\prime}|s^\otimes,a) \nabla_{\mathbf{w}} V_{\mathbf{w}}(s^{\otimes\prime}) \quad (11)$$

We approximate the value of $\nabla_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes)$ by

$$\tilde{\nabla}_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes) = \beta \sum_{a\in A} \pi^o(s^\otimes,a) \left(\mathbf{Viol}_{(\Phi,\mathbf{w})}(s^\otimes,a) - \sum_{a'\in A} \mathbf{Viol}_{(\Phi,\mathbf{w})}(s^\otimes,a') \pi^B_{(\Phi,\mathbf{w})}(s^\otimes,a') \right) \quad (12)$$

Equation (12) does not exactly compute $\nabla_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes)$, since it ignores the dependence on $\mathbf{Viol}_{(\Phi,\mathbf{w})}$ on $\mathbf{w}$ and thus the terms $\nabla_{\mathbf{w}} \mathbf{Viol}_{(\Phi,\mathbf{w})}$. we find in practice that the approximate form works well.

Equations (8) and (12) apply to product states which *occur* in the observed trajectories. Rather than speculate about the behavior of the agent in unobserved product states, we assume that the KL divergence between the observed policy and the Boltzmann policy in these states is zero; we thus set both $C_{\mathbf{w}}(s^\otimes)$ and $\tilde{\nabla}_{\mathbf{w}} C_{\mathbf{w}}(s^\otimes)$ in all such states.

Equation (11) is a fixed-point equation, and thus we can use a fixed-point iteration to compute the gradient $\nabla_{\mathbf{w}} V_{\mathbf{w}}(s)$ as well. We can thus use projected gradient descent to optimize over the weights (where the projection is done to enforce the constraints over $\mathbf{w}$).

Rather than running (4) and (11) to convergence every time the weights are updated, we can make the computation more efficient by performing only *one* iteration of updates according to (4) and (11) between gradient descent steps. The result is Algorithm 1.

Example: GridWorld

We tested our approach in a simple 3 × 3 GridWorld to demonstrate its effectiveness. In particular, the agent begins in the bottom-left cell. The agent has two goals: to spend as much time as possible in the top-left "good cell", and to avoid the bottom-left "bad cell" as much as possible. The agent had actions $north$, $south$, $east$, and $west$, which in this case succeed 80% of the time (and otherwise move the agent in one of the other three directions). Any movement that would cause the agent to leave the grid instead results in the agent "hitting a wall" and remaining in its current cell.

We represented the two goals using a proposition $goodCell$ to indicate when the agent was in the good cell, and $badCell$ when the agent was in the bad cell, and providing the agent with the two norms $\mathsf{G}\ goodCell$ and $\mathsf{G}\ \neg badCell$ with weights w_1 and w_2 respectively. Demonstrations were obtained using our algorithm in (Kasenberg and Scheutz 2018).

The resulting pattern of behavior depends on the relative weights of the two norms. If w_2 is low (less than roughly $1.82w_1$ - Regime 1) the agent travels directly toward the good cell, passing through the bad cell on the way. If $1.82w_1 < w_2 < 9.5w_1$ (Regime 2), the agent attempts to take the path leading through the center cell. If $9.5w_1 < w_2 < 14w_1$ (Regime 3), the agent takes the path along the right-hand-side. If $w_2 > 14w_1$ or $w_1 = 0$ (Regime 4), the agent ignores the good cell completely and focuses on avoiding the bad cell.

We picked one set of weights from each of the regimes R1-R4. Each demonstration consisted of ten episodes of ten time steps each. We then ran Algorithm 1 on the resulting trajectories. The set Φ of norms included the actual norms $\mathsf{G}\ goodCell$ and $\mathsf{G}\ \neg badCell$ as well as the "distraction" norms $\mathsf{G}\ \neg goodCell$ (with weight w_3) and $\mathsf{G}\ badCell$ (with weight w_4). In each case we chose $\eta = 10^{-5}$, $\beta = 150$, and ran until (a) the change in all values was less than 10^{-6}, or (b) for 10000 iterations.

The results are shown in Table 1. In cases R1 to R3 the weights of the "distraction" norms are very close to zero.

Table 1: Results of inverse norm conflict resolution in GridWorld

Regime	Actual weights	Policy	Recovered weights	Recovered policy
R1	(0.5, 0.5)	↑ ← ← ↑ ↑ ↑ ↑ ↑ ↑	(0.668, 0.332, 0, 0)	↑ ← ← ↑ ↑ ↑ ↑ ↑ ↑
R2	(0.25, 0.75)	↑ ← ← ↑ ↑ ↑ → ↑ ↑	(0.324, 0.673, 0.002,0)	↑ ← ← ↑ ↑ ↑ → ↑ ↑
R3	(0.08, 0.92)	↑ ← ← ↑ ↑ ↑ → → ↑	(0.082, 0.898, 0, 0.019)	↑ ← ← ↑ ↑ ↑ → → ↑
R4	(0.06, 0.94)	→ → ↑ ↑ → ↑ → → ↓	(0, 0.934, 0.066, 0)	→ → → → → ↓ → → →

Furthermore, in each such case the weights of w_1 and w_2 are in the right regimes, and the recovered weights cause the agent to follow the demonstrator's policy precisely (even though, for example, in R1 the agent never saw the demonstrator's actions in the bottom-right cell).

Although the correct regime is recovered in case R4, the weight computed for $\mathsf{G}\ \neg goodState$ is nonzero, and the agent's policy is not correct in all states. We believe that this may be caused by our choice of β. As we described in the previous section, the extent to which $\pi^{B}_{(\Phi,\mathbf{w})}$ matches the optimal policy π^* depends on the value of β. In practice we find that increasing β (and correspondingly decreasing η) yields better results (at the cost of slower convergence).

Note that while the algorithm determines the correct regime, the exact given weights are not recovered. This is expected, since the weights are only accessible to the agent via the demonstrations. The more varied the demonstrations, the more accurately the weights can be recovered.

Discussion and future work

Inverse norm conflict resolution allows an agent to estimate from demonstrations the relative importance of various moral and social norms (represented in temporal logic). These weights can help the agent to behave appropriately in novel norm conflicts. While this approach requires a finite set of possible norms to be provided, we have shown in (Kasenberg and Scheutz 2017) how a (single) temporal logic statement may itself be learned from behavior. The combination of these two approaches hints at a mechanism for co-learning multiple norms and their corresponding weights.

One crucial priority is improving the computational complexity of this approach. Our algorithm runs in time exponential in the total number of norms (as does the norm conflict resolution algorithm underlying it). This complexity renders the actual implementation of our approach on real systems impractical. We thus emphasize the importance of research into tractable norm conflict resolution algorithms.

While our approach has only been concretely implemented in linear temporal logic (LTL), we believe that similar approaches may be applicable to other formal languages, and other notions of violation cost. Precisely which logics are amenable to this sort of approach (as well as which notions of violation cost) is a topic for future work. One priority would be to incorporate explicit deontic operators, so as to facilitate more sophisticated reasoning.

The task of learning the importance of norms by observing norm conflicts raises the intriguing possibility of using *active learning* for such purposes. Agents using such algorithms may be able to pose *new moral dilemmas* to another agents or human, and use the resulting information to better understand that agent's values. Using norms explicitly represented in a logical language facilitates this, potentially allowing agents to generate new possible worlds with bizarre dynamics and properties, but which may help to elucidate the underlying norms.

Conclusion

In this paper, we provided an algorithm (inverse norm conflict resolution) for determining the relative importance of a set of moral and social norms to a demonstrator by observing the demonstrator's behavior in norm conflicts. This algorithm contributes towards the goal of artificial agents that can learn and obey human moral and social norms, even when those norms conflict.

References

Arnold, T.; Kasenberg, D.; and Scheutz, M. 2017. Value alignment or misalignment – what will keep systems accountable? In *Proceedings of the 3rd International Workshop on AI, Ethics, and Society.*

Babes, M.; Marivate, V.; Subramanian, K.; and Littman, M. L. 2011. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 897–904.

Boularias, A.; Kober, J.; and Peters, J. 2011. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 182–189.

Ding, X. C.; Smith, S. L.; Belta, C.; and Rus, D. 2011. MDP optimal control under temporal logic constraints. *Proceedings of the IEEE Conference on Decision and Control* 532–538.

Kasenberg, D., and Scheutz, M. 2017. Interpretable apprenticeship learning with temporal logic specifications. In *Proceedings of the 56th IEEE Conference on decision and control (CDC 2017)*.

Kasenberg, D., and Scheutz, M. 2018. Norm conflict resolution in stochastic domains. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics* 22(1):79–86.

Lahijanian, M.; Almagor, S.; Fried, D.; Kavraki, L. E.; and Vardi, M. Y. 2015. This Time the Robot Settles for a Cost: A Quantitative Approach to Temporal Logic Planning with Partial Satisfaction. In *The Twenty-Ninth AAAI Conference (AAAI-15)*, 3664–3671.

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*. Citeseer.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57.

Saitake, R., and Arai, S. 2016. Parameter estimation of multi-objective reinforcement learning to reach arbitrary pareto solution. In *IEEE International Conference on Agents (ICA)*, 110–111. IEEE.

Vazquez-Chanlatte, M.; Jha, S.; Tiwari, A.; and Seshia, S. A. 2017. Specification inference from demonstrations. *arXiv preprint arXiv:1710.03875*.