Generalized Inverse Planning: Learning Lifted non-Markovian Utility for Generalizable Task Representation

Sirui Xie[*], Feng Gao[*], Song-Chun Zhu
UCLA Center for Vision, Cognition, Learning and Autonomy
srxie@ucla.edu, f.gao@ucla.edu, sczhu@stat.ucla.edu

Abstract

In searching for a generalizable representation of temporally extended tasks, we spot two necessary constituents: the utility needs to be *non-Markovian* to transfer temporal relations invariant to a probability shift, the utility also needs to be *lifted* to abstract out specific grounding objects. In this work, we study learning such utility from human demonstrations. While inverse reinforcement learning (IRL) has been accepted as a general framework of utility learning, its fundamental formulation is *one* concrete Markov Decision Process. Thus the learned reward function does not specify the task independently of the environment. Going beyond that, we define a domain of generalization that spans *a set of* planning problems following a schema. We hence propose a new quest, *Generalized Inverse Planning*, for utility learning in this domain. We further outline a computational framework, *Maximum Entropy Inverse Planning* (MEIP), that learns non-Markovian utility and associated concepts in a generative manner. The learned utility and concepts form a task representation that generalizes regardless of probability shift or structural change. Seeing that the proposed generalization problem has not been widely studied yet, we carefully define an evaluation protocol, with which we illustrate the effectiveness of MEIP on two proof-of-concept domains and one challenging task: *learning to fold from demonstrations*.

Introduction

Humans learn underlying utility by observing others' behaviors. It is widely accepted that we humans have a Theory of Mind (ToM), assume others as bounded rational agents, and inversely solve for their utility to understand their planned behaviors (Baker, Saxe, and Tenenbaum 2009). The inferred utility is associated with some concepts, which together specify the task. Then in a similar context, this utility can generalize to incentivize us to act similarly. In this work, we study a formal definition of such generalization and a proper machinery to learn such utility.

The utility we want to study is different from the reward function in classical reinforcement learning. In their seminal book, Sutton and Barto (1998) distinguish planning from reinforcement learning as requiring some explicit *deliberation* using a world model. It is this deliberation that the utility we discuss here expects to capture.

Preliminary work. * indicates equal contribution.

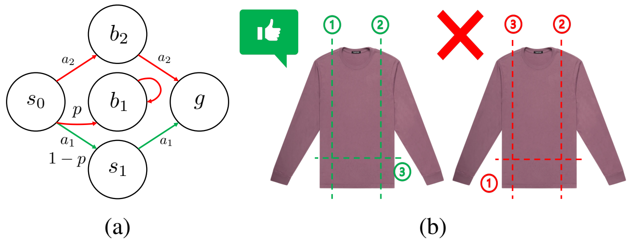

Figure 1: (a) An MDP on which the agent needs to reach g without hitting a b (highlighted with the green arrow). Any Markovian rewards that make this expected behavior optimal are coupled with p. (b) The *default* and *weird* sequences for folding a cloth. The former should have higher utility. The underlying utility should also generalize robustly.

Apart from these philosophical concerns, there are also computational issues in the setup of classical reinforcement learning. One of our key insights is that tasks specified with Markovian reward functions do not generalize over environments. Consider a didactic example from Littman et al. (2017), see Fig.1a. The desired behavior we want to specify is 'maximizing the probability of reaching the *goal* g without hitting a *bad state* b'. It is equivalent to a temporal description '*do not* visit a b *until* you reach the g', which unfortunately cannot be represented with Markovian rewards independently from the environment. Concretely, let's assume a discount of $\gamma = 0.8$ and a reward of +1 for reaching the goal. If $p = 0.1$, setting $r < -0.16$ to the bad state encourages the desired behavior. But if $p = 0.3$, this reward needs to be $r < -0.48$. Even though this example may seem contrived, it captures the essence of the limitation of Markovian rewards. There are more natural examples in our daily life. Imagine you are folding your clothes after laundry. Normally, *not until* you fold the left and right sleeves *do* you fold it in half from top to bottom, see Fig.1b. Searching your memory, you will probably realize it is the *default* order ever since you learned to fold clothes as a kid, no matter the one you fold is a T-shirt or a sweater, no matter it is your all-time favorite or a brand-new one. Remarkably, the utility you learned on this temporally extended task exhibits robust generalization. In other words, we have this cognitive capability to learn a task representation with a utility function that is independent of the environment.

In fact, a slip in the *transition probability* is only the mildest one in environmental shifts. In the example of cloth folding, *hoodies* and *T-shirts* have different *structures* in the underlying probabilistic graphical models (PGM). Concretely, this is because they have different numbers of edges in their contours thus different numbers of nodes in their object-oriented probabilistic graphs. Despite of this difference, the utility learning mechanism ought to be tolerant of the heterogeneous nature of demonstrations. The learned utility should also help when folding a *sweater* after seeing these demonstrations. From a classicist' perspective, this requirement goes beyond the formulation of RL, Markovian Decision Processes (MDP), where the language that specifies the structure of the MDP (which is essentially a PGM) is constrained to be *propositional*. For readers who are not familiar with classical AI or computational linguistics, propositional logic can be understood as a language without object-orientation. Its object-oriented counterpart is *first-order* or *relational* logic. Intuitively, utility associated with a relation, *i.e.,* "symmetric", can generalize better than than utility associated with a grounded description such as "the left and right sleeves are symmetric". In classical AI, this property is called *lifted* in the sense of not being grounded with specific entities.

How can machines learn utility that is both non-Markovian and lifted? The closest solution in the literature is Inverse Reinforcement Learning (IRL) (Abbeel and Ng 2004). However, IRL adopts the fundamental modeling of MDP. Given an MDP and a set of demonstrations from it, IRL learns a Markovian reward function by matching the *mean statistics* of states or state-action pairs. This learned reward function can only encourage the expected behavior in the *identical* MDP, because apparently it will fall into the trap of the didactic example above. Apart from that, utility learned with vanilla IRL also fails to generalize to a PGM with different structure. In this work, we propose a joint treatment for learning generalizable task representation from human demonstrations.

Our contributions are threefold:

- We characterize the domain of generalization in this utility learning problem by formally defining a *schema* for the *planning task* that represents a set of *planning problems*. The target utility should be learned in a subset of this domain and successfully generalize to other problem instances. This is in stark contrast to the formulation of IRL for which only one planning problem is of interest. We hence term the problem *generalized inverse planning*.
- We propose an energy-based model (EBM) for generalized inverse planning. In Statistics, energy-based models are also dubbed *descriptive models*. This is because the model is expected to match the minimal statistical description in data. It is this description that differentiates the proposed model, Maximum Entropy Inverse Planning (MEIP), from prior arts such as MaxEnt-IRL (Ziebart et al. 2008). Instead of matching the mean statistics in demonstrations under a Markovian assumption, MEIP matches the *ordinal statistics*, a description that sufficiently captures the temporal relations in non-Markovian planning. This model can be learned with Maximum Likelihood, sampled with Monte Carlo Tree Search (MCTS). To combine MEIP with a first-order *concept language*, we further introduce a boosting method to pursue abstract concepts associated with the utility.
- Seeing that the generalization problem in this work was barely studied systematically, we carefully design an evaluation protocol. Under this protocol, we validate the generalizability of utility learned with MEIP in two proof-of-concept experiments for environmental change and a challenging task, *learning to fold clothes*.

Background

Inverse Reinforcement Learning

Imitation learning, also known as learning from demonstrations, is a long-standing problem in the community of artificial intelligence. Earliest works most adopt the paradigm of Behavior Cloning (BC), directly supervise the policy at each step of provided sequences (Hayes and Demiris 1994), (Amit and Matari 2002). Atkeson and Schaal (1997) was the fist to consider the temporal drifting in sequences. Nonetheless, BC is always believed to be less transferable without generatively modeling the decision making sequences. Alternatively, Ng, Russell et al. (2000) proposed another paradigm, IRL, to inversely solve for the reward function in an MDP. The learned reward function is expected to help the agent learn a new policy, hopefully covers more states in the MDP. Together with some extensions such as Abbeel and Ng (2004) and Ratliff, Bagnell, and Zinkevich (2006), they set up the standard formulation of IRL.

Consider a finite-horizon MDP, which is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, h\rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the Markovian transition probability, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the Markovian transition probability and h is the horizon. Abbeel and Ng (2004) assume the Markovian reward to be linear to some predefined features $\phi(s) \in \mathbb{R}^k$ of states $s \in \mathcal{S}$ and derive the *feature expectation* as $\mu \in \mathbb{R}^k$. Specifically, the underlying unknown reward is assumed to be $R^*(s) = \omega^* \cdot \phi(s)$ with unknown parameter $\omega^* \in \mathbb{R}^k$. Given a set of demonstration trajectories $\Psi^E = \{\zeta_1^E, \zeta_2^E, ..., \zeta_m^E\}$ from this MDP with $R^*(s)$, $\zeta \in \mathcal{S}^h$, the value with the estimated parameter w is

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{h} R(s_i^j) = \frac{1}{m}\sum_{i=1}^{m} \omega \cdot \sum_{j=1}^{h} \phi(s_i^j) = \frac{1}{m}\sum_{i=1}^{m} \omega \cdot \mu(\zeta_i). \tag{1}$$

The algorithm of IRL always runs with an off-the-shelf reinforcement learning method to generate trajectories $\Psi^G = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ with the optimal value for the same MDP except that the reward $R(s)$ is the current estimation. We thus have two sets of trajectories. Assuming humans behave rationally when demonstrating, Abbeel and Ng then propose a learning objective to maximize the margin between these two sets. When the model converges, the *mean statistics* $\mu(\zeta)$ from Ψ^E and Ψ^G should be matched.

To account for humans' bounded rationality and imperfect demonstration, Ziebart et al. (2008) introduced a probabilistic modeling for IRL. They started from a statistical mechanics perspective: the IRL model above does not address the ambiguity that many reward functions can lead to the same feature count. They propose MaxEnt-IRL to maximize the entropy of distribution of trajectories to break the tie while matching the mean statistics. The distribution of a trajectory ζ_i in a deterministic MDP is thus

$$P(\zeta_i|\omega) = \frac{1}{Z(\omega)} \exp(\omega{\cdot}\mu(\zeta_i)) = \frac{1}{Z(\omega)} \exp(\omega{\cdot} \sum_{s_i^j \in \zeta_i} \phi(s_i^j)), \tag{2}$$

where $Z(\omega)$ is the partition function. For non-deterministic MDPs, there is an extra factor $P(\zeta_i)$ to account for the transition probability $\mathcal{T}$. In MaxEnt-IRL, the reward function is learned with Maximum Likelihood Estimate (MLE). Some variants such as Boularias, Kober, and Peters (2011) and Ortega and Braun (2013) learn by minimizing relative entropy (KL divergence).

We derive our model of Maximum Entropy Inverse Planning with the same principle. But in stark contrast to matching the mean statistics under the assumption of Markovian independence, MEIP matches the ordinal statistics *Kendall rank correlation* to account for the non-Markovian temporal relations. To some degree, we share the same spirit with Xu, Fern, and Yoon (2009) and Garrett, Kaelbling, and Lozano-Pérez (2016). But their models are discriminative, which are believed to be more data-hunger and less generalizable than *analysis by synthesis* (Grenander 1993).

Structured Utility Function

Prior attempts to make utility functions structured fall into two regimes: those that adopt First-order Logic (FOL) for lifting (Džeroski, De Raedt, and Driessens 2001) (Kersting and Driessens 2008) and those that adopt Linear Temporal Logic (LTL) for non-Markovian rewards (Li, Vasile, and Belta 2017), (Littman et al. 2017), (Icarte et al. 2018). Leveraging the expressive powers of language and axioms, these structured utility functions can be generalized over a *task domain*. We draw inspiration from them. There are works that also inversely solve for structured utility function from demonstrations (Munzer et al. 2015), (Vazquez-Chanlatte et al. 2018), but they only focus on one regime. To the best of our knowledge, our work is the first to provide a unified probabilistic model for both regimes. Our framework adopts a relational language for abstract and composable concepts and match ordinal statistics with a descriptive model. The learned structured utility function is expected to represent a task in *Generalized Planning*, to be elaborated below.

Generalized Inverse Planning

The planning we want to discuss in this work is generalized in the sense that its representation generalizes over a *domain* with multiple *planning problems* (Srivastava, Immerman, and Zilberstein 2011). It is thus fundamentally different from the formulation of reinforcement learning which only maximizes rewards on *one* specific MDP.

Definition 1 (Planning Domain) *A planning domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A} \rangle$ consists of a set of fluents $\mathcal{F}$ which are real-valued functions and a set of actions $\mathcal{A}$ which are associated with some entities or parameters or both.*

Obviously, fluents here are generalized predicates in classical symbolic planning, whose arity should be given in advance and value may vary with time. Actions may also be associated with preconditions and effects, specified by sets of fluent values. This definition makes the planning domain lifted to the first-order, abstracting out instances of objects, agents, and events, only describes their classes and relations. It ,therefore, generalizes to arbitrary numbers of instances. In the literature, STRIPS-style action language (Fikes and Nilsson 1971) provides such abstraction.

Definition 2 (Stochastic Planning Problem) *A stochastic planning problem $\Pi = \langle \mathcal{D}, \mathcal{E}, \mathcal{T}, R, \rho \rangle$ is given by a domain $\mathcal{D}$, a set of entities $\mathcal{E}$, a set of transition probabilities $\mathcal{T}$ for $\mathcal{A}$, a reward function $\mathcal{R}$ and a distribution ρ over initial states. A state is specified by a set of fluent values.*

Here we do not assume the transition probabilities $\mathcal{T}$ to be Markovian. It can thus be easily generalized to Partially Observable (Kaelbling, Littman, and Cassandra 1998) or semi-Markovian (Sutton, Precup, and Singh 1999). States are sets of fluent values, which may be grounded spatial relations, grounded temporal relations, or output of functions. The reward function R is preferred over `Goal` because the extensionality of the latter one might be hard to specify, particularly for inversion from demonstrations. As introduced earlier, Markovian rewards may not generalize over a domain independently to the environment, we assume R to be non-Markovian for each planning problem. A *plan* is a (sub)optimal trajectory of states for a planning problem.

Given a set of plans from different planning problems in the same domain, a *task* is their abstraction that generalizes these plans and captures their underlying optimality. It may abstract out numerical values in R. At the minimal level, it is the action or transition at each state that we care about. This principle was previously discussed in Martin and Geffner (2004), Natarajan et al. (2011), and Silver et al. (2020), which directly imitate the behavior with expressive policies whose hypothesis space is logically or programmatically constrained. But completely depending on the policy also constrained the capability of generalization. For example, the learned policy may not work or generalize well if the action set is (uncountably) infinite. To this end, an *ordering* in the state space is still needed for a task to represent the (possibly bounded) rationality in demonstrated plans (Khardon 1999).

Definition 3 (Planning Task) *A planning task is a tuple $\langle \mathcal{D}, \prec \rangle$, where $\mathcal{D}$ is the domain and $\prec$ is a partial ordering relation over states.*

This minimal algebraic description of a task for generalized planning. And *Generalized Inverse Planning* is to inversely solve for a representation of the task that maintains this algebraic structure from demonstrations.

Our proposal towards generalized inverse planning is a nested algorithm. In the inner loop, it learns a numerical representation of $\prec$ with maximum likelihood. In the outer loop it pursuits first-order concepts $\mathcal{F}$ with maximum a posteriori (MAP). We introduce from inside out in the next two sections and provide pseudo-code in Alg.1 and Alg.2.

Maximum Entropy Inverse Planning

Descriptive Model and Maximum Entropy

We adopt descriptive modeling for inverse planning. Different from a discriminative model, which models a conditional probability, *i.e.* a classifier, a descriptive model specifies the probability distribution of the signal, based on an energy function defined on the signal through some descriptive feature statistics extracted from the signal (Wu et al. 2019). In the literature of modern AI, it is also known as energy-based model (LeCun et al. 2006). From the discussion above, it comes clear to us that given a set of plans from different problems in the same domain, the minimum statistical description to match should be ordinal to account for temporal relations. Therefore, different from the feature expectation in IRL, we match the Kendall ranking statistics for inverse planning given n plans from the same domain:

$$\tau = \frac{1}{n}\sum_{i=1}^{n} \frac{2}{(m_i)(m_i-1)} \sum_{j=1}^{m_i}\sum_{k=j+1}^{m_i} d(s_i^j, s_i^k), \quad (3)$$

where $d(s_i^j, s_i^k)$ scores the concordance or discordance of a ranking function g for temporally indexed states s_j^i and s_k^i where $j < k$ from a plan ζ_i for a problem Π_i:

$$d(s_i^j, s_i^k) = \begin{cases} +1 & g(s_i^k) - g(s_i^j) > 0 \\ -1 & g(s_i^k) - g(s_i^j) < 0 \\ 0 & g(s_i^k) - g(s_i^j) = 0 \end{cases}. \quad (4)$$

The range of Kendall τ is $[-1, 1]$, where $\tau = 1$ indicates a perfect match. However, normally there can be multiple distributions of plans and thus multiple ranking functions g that match this ordinal statistics. Similar to MaxEnt-IRL, we employ the principle of maximum entropy (Jaynes 1957) to resolve ambiguities in choosing distributions. Concretely, we maximize the entropy of the distribution over plans under the constraint that the Kendall τ can be matched between the demonstrations and generated plans:

$$\begin{aligned} \max \sum\nolimits_{\zeta} -P(\zeta|g)\log P(\zeta|g) \\ s.t. |P(\zeta|g)\tau(\zeta) - 1| < \epsilon, \\ \sum\nolimits_{\zeta} P(\zeta|g) = 1, \quad P(\zeta|g) \geq 0. \end{aligned} \quad (5)$$

Under the KKT condition, we can derive the Boltzmann form of this distribution from Eq.5's Langrangian:

$$P(\zeta|g,\lambda) = \frac{1}{Z(g,\lambda)}\exp(\lambda\cdot\tau(\zeta))P(\zeta), \quad (6)$$

where $Z(g,\lambda) = \sum_{\zeta}\exp(\lambda\cdot\tau(\zeta))P(\zeta)$ is the partition function, λ is the Langrangian multiplier and $P(\zeta)$ accounts for the the transition probability $\mathcal{T}$.

Utility Learning for Order Preserving

Comparing Eq.6 with Eq.2, it is easy to see the correspondence between $\lambda\cdot\tau(\zeta)$ and $\sum_{s^j\in\zeta} R(s^j)$. To further derive the utility function of plans from Eq.6, we first specify the concrete form of g. We assume for each state s, which is a set of grounded fluent values, there is a vector of fist-order concepts $f(s)$ that generalizes over the domain $\mathcal{D}$. We will introduce how this vector $f(s)$ can be learned in next section. Here we can simply assume it is given. And we can further assume that the ranking function $g(s)$ is piece-wise linear with respect to this concept vector $f(s)$. Being piece-wise linear is a general assumption because functions with this characteristics are in theory as expressive as artificial neural networks. Specifically, we discretize each entry $f_i(s)$ into M bins and attach a vector ω_i as a functional:

$$g_i(s) = \omega_i \cdot f_i(s) = \sum\nolimits_{j=1}^{M} \omega_i^j f_i^j(s). \quad (7)$$

Obviously, there is no need to separate λ and ω anymore. So we drop λ for the derivation below.

To further illustrate the utility function in MEIP, we can rewrite $\tau(\zeta)$ as:

$$\tau(\zeta) = \sum_{i=2}^{h}\frac{2}{(h)(h-1)}\sum_{j<i} d(s^j, s^i) = \sum_{i=2}^{h} R(s^i), \quad (8)$$

Therefore we have

$$R(s^i) = \frac{2}{(h)(h-1)}\sum_{j<i} d(s^j, s^i) \propto \sum_{j<i}\operatorname{sgn}(g(s^j) - g(s^i)). \quad (9)$$

It is easy to see that this reward function is non-Markovian, in stark contrast to $R(s) = \omega\cdot\phi(s)$ in MaxEnt-IRL.

We can solve for the ω by maximum likelihood (MLE) over given demonstrations, which according to Jaynes (1957) implies maximum entropy:

$$\mathcal{L}_\omega = \frac{1}{n}\sum\nolimits_i \log P(\zeta_i|\omega) = \frac{1}{n}\sum\nolimits_i \tau(\zeta_i) - \log Z(\omega). \quad (10)$$

Notice that $\log P(\zeta_i)$ is dropped as a constant. If τ is differentiable *w.r.t.* ω, consider Eq.10's gradient

$$\begin{aligned} \nabla_\omega\mathcal{L}_\omega &= \frac{1}{n}\sum\nolimits_i \nabla_\omega\tau(\zeta_i) - \nabla_\omega \log Z(\omega) \\ &= \frac{1}{n}\sum\nolimits_i \nabla_\omega\tau(\zeta_i) - \sum\nolimits_j P(\zeta_j|\omega)\nabla_\omega\tau(\zeta_j) \quad (11) \\ &= \frac{1}{n}\sum\nolimits_i \nabla_\omega\tau(\zeta_i) - \mathbb{E}_{P(\zeta_j|\omega)}[\nabla_\omega\tau(\zeta_j)], \end{aligned}$$

the second term $\mathbb{E}_{P(\zeta_j|\omega)}[\nabla_\omega\tau(\zeta_j)]$ can be approximated by sampling. Apparently, there is an contrastive view here: when maximizing the likelihood of demonstrations, we maximize their Kendall τ and minimize the Kendall τ of generated plans. The underlying intuition is that at convergence, for pairs of states $\langle s_i^m, s_i^n\rangle$ in the demonstrations, $g(s_i^m) < g(s_i^n)$ if $m < n$; for pairs $\langle s_j^m, s_j^n\rangle$ in generated plans, $g(s_j^m) \geq g(s_j^n)$ if $m < n$; for pairs with one state s_i^m in demonstrations and the other one s_j^m from generated plans, $g(s_i^m) \geq g(s_j^m)$. We would refer to ordinal relations listed here as $\hat{d}_k$, resembling groundtruth labels.

However, τ is not differentiable. To this end, we need to learn a classifier with $\langle g(s^m), g(s^n)\rangle$ to match the order described above, mimicking the optimization in Eq.11. One way is to directly relax $d(s^m, s^n)$ with a discriminator

$D(s^m, s^n) = \tanh(g(s^n) - g(s^m))$. To some extent, this approximation shares similar spirits with Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon 2016) and some variants (Finn et al. 2016). But different from them, we explicitly consider the temporal relation in the non-Markovian utility.

We can also learn this classifier with max-margin methods. Essentially, it is a Ranking SVM model (Liu 2011) taking both the demonstrations Ψ^E and the sampled plans Ψ as supervision. Consider there are K pairs of states $\langle s_k^m, s_k^n \rangle$ in total from $\Psi^E \vee \Psi$ expected to fulfill the ordinal relation $\hat{d}_k$ above, we have this Quadratic Optimization problem:

$$\begin{aligned} &\min_{\omega} \mathcal{L}_{\omega}^{SVM} = \frac{1}{2} \|w\|^2 + C \sum_{k}^{K} \xi_k, \quad s.t. \\ &\forall k : \xi_k \geq 0, \hat{d}_k \cdot (g(s_k^n) - g(s_k^m)) \geq 1 - \xi_k. \end{aligned} \tag{12}$$

It can be solved by off-the-shelf Gradient Descent (GD) or Quadratic Programming (QP). Notice that the number of constraints and slack variables grows quadratically in the size of each plan, we thus only consider the ordinal relations in each planning problem Π. But the parameters ω are shared for all problems in the domain $\mathcal{D}$. If we only consider the linear, primal form of the problem, there are also efficient methods for training (Joachims 2006).

Sampling Method

We need to sample from the EBM to calculate parts in Eq.12 that correspond to the second term in Eq.11. Thus we need to have the distribution of trajectories from the EBM. In this work, we only consider planning problems where states are discrete. We leave the continuous state space as our future work. We can factorize the probability of a plan $\zeta_i = (s_i^0, s_i^1, ..., s_i^h)$ by conditioning:

$$P(\zeta_i|\omega) = P(s_i^0)P(s_i^1|s_i^0, \omega)P(s_i^2|s_i^1, s_i^0, \omega)... \tag{13}$$

We take its logarithmic form to connect with Eq.8 and Eq.9:

$$\log P(\zeta_i|\omega) = \log \rho(s_i^0) + \sum_{j=1}^{h} \log P(s_i^j|s_i^0...s_i^{j-1}, \omega), \tag{14}$$

where $\log P(s_i^j|s_i^0...s_i^{j-1}, \omega)$ takes both the action and transition probability into account. Since here we do not explicitly specify the action in $P(\zeta_i|\omega)$[1], we decompose it to be

$$\log P(s_i^j|s_i^0...s_i^{j-1}, \omega) = R_\omega(s_i^j) + \log P(s_i^j|s_i^0...s_i^{j-1}). \tag{15}$$

It then becomes clear that to sample from the descriptive model, we just sample proportionally to $\exp(\sum_j R_\omega(s_i^j))$:

$$P(\zeta_i|\omega) = P(s_i^0...s_i^j...s_j^h|\omega) \propto \exp(\sum_j R_\omega(s_i^j)). \tag{16}$$

[1]The main concern here is that grounded actions in plans from different Π may not generalize over $\mathcal{D}$. Bonet and Geffner (2018) discuss learning abstract actions for generalized planning. Here we do not enforce a relational form in the action space $\mathcal{A}$. We assume the equivalence between actions causing same transitions. We further assume there is an absorbing state *failure* for each state transition thus $\mathcal{T}$ is stochastic with $P(s_i^j|a_i^j, s_i^0...s_i^{j-1}) \neq 1$.

Algorithm 1: Maximum Entropy Inverse Planning

```
Input: A Set of Concepts $\Delta$ from the Concept
  Language; MCTS Convergence Conditions;
  Hyperparams.
Data: Human Demonstrations $\Psi^E = \{\zeta_i\}$
Result: Learned Parameters of Utility $\omega$
Init: $\omega$, value function $V$ in MCTS
while utility function not converged do
    while MCTS not stopped do
        MCTS rollout, generate $\zeta$
        Compute Kendall $\tau_\omega(\zeta)$ and $R(s)$ with Eq. 8
        Value iteration via MCTS with Kendall $\tau_\omega$
    end
    sample trajectories $\Psi = \{\zeta_j\}$ with MCTS
      according to converged values of branches
    update $\omega$ with $\Psi^E$ and $\Psi$ (See Eq. 11)
end
```

$\sum_j R_\omega(s_i^j)$ can be acquired from Monte Carlo Tree Search, for which the reward at each node is initialized with $R_\omega(s_i^j)$. After its convergence, we can sample trajectories according to the value of each branch.

It is worth noticing that the same sampling method can be used to plan for optimal utility when transferred to other problems Π_{new} in the same domain $\mathcal{D}$. So generally speaking, for sampled plans, there is a min-max view:

$$\min_\omega \max_\zeta \mathbb{E}_{P(\zeta|\omega)}[\tau(\zeta)]. \tag{17}$$

Learning First-Order Concepts

In the previous section, we introduced a descriptive model for inverse planning, given some concepts that generalize over the planning domain. In this section, we introduce a formalism for concepts with this characteristics and how we learn them as $\mathcal{F}$ in Generalized Inverse Planning.

Concept Language

As introduced in Sec.3, a planning domain $\mathcal{D}$ is defined in a lifted manner, abstract out instances of entities in specific problems. Therefore the computational form of the utility learned from Generalized Inverse Planning should also be lifted and be invariant to the variation of numbers of instances. In AI, the first-order logic with quantifiers and aggregators is a formalism to express this invariance. To this end, we employ a modified concept language (Donini et al. 1997) as a grammar for elements in $\mathcal{F}$ such that we can learn these first-order concepts in a top-down manner. Concept languages have the expressive power of subsets of standard first-order logic yet with a syntax that is suited for representing *classes* of entities. Adopting the terminology of FOL, each concept is represented by a first-order *formula*. However, in the original concept languages, concepts are assumed independent, which might not be the case for our utility function[2]. So the primary modification we introduce is to

[2]Recall that we assume the ranking function $g(s)$ to be (piecewise) linear in Eq.7, which requires concepts $f_i(s)$ to be indepen-

complete it as in FOL, explicitly accounting for formulas which take other formulas as *terms* with a syntax:

$$C \to C_a \mid P(C', C'', ...) \mid F(C', C'', ...),$$

where the $'$ notation highlights different concepts. P denotes predicates with binary value domains i.e. relations, F denotes functions with either real or discrete value domains.[3] They all have their own arities. $C_a \in \mathcal{C}_a$ are concepts represented by *atomic formulas* with a syntax:

$$C_a \to QV \mid AV,\ V \to FD \mid PD,\ D \to \text{ext}(P) \mid U,$$
$$Q \to \forall \mid \exists \mid \#,\ A \to \max \mid \min \mid \text{avg},$$
$$P \to P_p \mid \neg P' \mid P' \wedge P'' \mid P^p \mid P^*.$$

A is the set of aggregators and Q is the set of quantifiers. V is a set that is either the value domain of a fluent F or the truth domain of a predicate P. D is the domain of interest that can either be the extension of a certain predicate, which is denoted as ext(P), or the universe (U) of entities. The dimension of the domain D should match the arity of F and P when placed together in FD and PD. Among predicates P, P_p are *primitives*. Other predicates can be their constituents' negation $(\neg P')$ or conjunction $(P' \wedge P'')$. They can also be a result of permuting another predicate's arguments (P^p), if the arity is larger than 1 and arguments are from the same class. They can even be defined transitively (P^*), as in original concept languages. Apparently, $\mathcal{C}_a \subset \mathcal{C} \subset \mathbb{R}$.

Concept Pursuit

To consider the combination over the full bank of concepts $\mathcal{C}$ would be computationally intractable. A Bayesian treatment of concept induction can provide a principled way to incorporate the prior of Occam's razor to probabilistic grammars. Let us denote the selected K concepts with $\Delta_k = \{f_0, f_1, ..., f_{k-1}\}$, where f_i coincides Eq.7, the posterior of the utility function $\langle \Delta_k, \Omega_k \rangle$ given demonstrated plans $\Psi = \{\zeta_0, \zeta_1, ...\zeta_{n-1}\}$ would be

$$P(\Delta_k, \Omega_k|\Psi) = \frac{P(\Psi|\Delta_k, \Omega_k)P(\Delta_k)P(\Omega_k)}{P_k(\Psi)}. \quad (18)$$

To obtain the MAP estimate efficiently, we adopt stepwise greedy search over $\mathcal{C}$:

$$\begin{aligned} C_+, \omega_+ &= \arg\max \log P(\Delta_+, \Omega_+|\Psi) - \log P(\Delta_k, \Omega_k|\Psi) \\ &\approx \arg\max KL(P_+(\zeta)|P_k(\zeta)) + \log P(C) - \log P_+(\Psi)/P_k(\Psi) \\ &\approx \arg\max |\mathbb{E}_\Psi[\tau_{C,\omega}(\zeta_i)] - \mathbb{E}_{P_+}[\tau_{C,\omega}(\zeta_j)]| + \log P(C). \end{aligned} \quad (19)$$

Specifically, concepts in $\mathcal{C}$ are first sorted by their complexity, reflecting the prior $P(C)$. Note that concepts are mutually exclusive if they share the same V and only differentiate at A or Q, so they are stored in the same slot in the sorted list. The levels of complexity are naturally discretized by the number of *primitive* fluents or predicates involved. We start from the simplest level. At each level, the concept that brings the largest margin is added to Δ in a step-wise greedy manner. The selection terminates at the current level when the marginal benefit in the posterior is below a threshold. Then we move on to the next level. Since some complex concepts may have information overlap with simpler ones *e.g.* $\forall(P_1 \wedge P_2)D$ vs $\forall P_1 D \vee \forall P_2 D$, the simpler ones are replaced from Δ when the complex are added. The first term in Eq.19 is equivalent to the MLE in Eq.11 and can be solved with MEIP. Therefore the complete algorithm is nested.

This derivation leads us to a boosting method (Friedman 2001). A similar greedy search strategy was proposed for feature selection in IRL by Bagnell et al. (2007). But different from them, we further assume the increase from $\log P(C)$ is always more significant than the former term, such that we only need to consider a subset of $\mathcal{C}$ with the lowest complexity at a time.

dent from each other. Utility functions whose concepts are not independent will never be expressed without this modification.

[3]They may have neural network equivalents structuralized as Multi-Layer Perceptrons (MLP) or Graph Neural Nets (GNN).

Experiments

Evaluation Protocol

To help readers better understand the generalization problem studied in this work, we provide a systematic introduction of our evaluation protocol. Note that even though it is the *learned utility* that is to be evaluated, it cannot be evaluated without planned behaviors. This because the optimal utility can not be trivially defined for most tasks where Generalized Inverse Planning is meaningful, a issue that originally motivated the proposal of IRL (Abbeel and Ng 2004)

1. The agent first learns the utility in the environment where the demos come from;
2. We then transfer this agent to another environment, which is built under the same *schema* thus in the same *planning task*. This new environment is a result of probability shift or structural change or both;
3. Given the symbolic world model of the new environment, the agent optimizes for a policy that maximizes the reward function with model-based methods such as MCTS;
4. Test if the behavior of the agent in this new environment is consistent with the demo in terms of temporal relations.

We adopt the following two sets of evaluation metrics, depending on the diversity of the desired behavior:

1. For simple tasks the ground-truth behavior is only one single sequence, such as in the didactic example $S_0 \to S_1 \to g$ is the only desired sequence, we evaluate the learned utility by a Monte Carlo estimate of the probability of the desired sequence executed by an MCTS agent's planned behavior in the new environment.
2. Most of the time, the ground-truth optimal utility is not clear to us, especially its numerical value. But we can extract the ground-truth concepts and their ordering from the demos. We evaluate the learned utility with the objective of IRL or inverse planning: the matching in the statistics. Specifically, given the planned behaviors and the demonstrated ones, we measure their *mean matching* to check if the learned utility can attain a *similar* behavior from the perspective of IRL; We measure their *Kendall* τ (ordinal matching) to check if the learned utility can attain a *similar* behavior from the perspective of inverse planning.

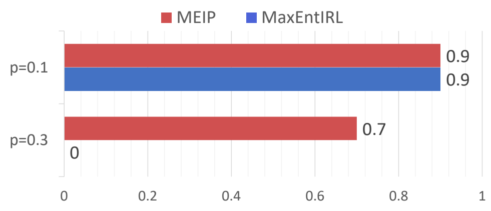

Figure 2: The probability of 'reaching the goal state without hitting a bad state' in the training environment ($p = 0.1$) and the testing ($p = 0.3$) for agents whose utility is learned with MEIP and MaxEnt-IRL respectively. $1 - p$ is the optimal.

Experiment 1: Probability Shift

We first conduct an experiment with the didactic example from (Littman et al. 2017) introduced above to illustrate utility learned with MEIP can generalize regardless of probability shift. Here probability shift refers to a change in a distribution in $\mathcal{T}$. The desired behavior is '*do not* visit a bad state *until* you reach the goal'. We recommend readers to review Fig.1a for the problem setup. When learning the utility, the agent is provided with demonstrations collected from the MDP with $p = 0.1$. And it is also provided a black-box model to simulate this MDP during MCTS. The learned utility is then tested in another MDP, with $p = 0.3$. In both environments, we evaluate agents' behaviors by estimating the probability of the desired sequence $S_0 \to S_1 \to g$. The result is shown in Fig.2: Although both agents with MEIP and MaxEnt-IRL behave perfectly with $p = 0.1$, only the agent with MEIP still performs perfectly with $p = 0.3$. The MaxEnt-IRL agent discards temporal relations in utility and only matches mean statistics $\mu(\zeta)$, thus prefers $S_0 \to b_1$ in the first transition after a probability shift.

We also conducted an empirical study to explore the boundary of generalization of both MEIP and MaxEntIRL. If $p < 0.24$, the difference between MEIP and MaxEntIRL would be insignificant since the reward learned from MaxEntIRL can also discourage agents from taking a_2. We also notice that if $p > 0.85$, which induces an extremely high probability for agents to move to b_1 after taking a_1, neither MEIP nor MaxEntIRL learns meaningful utility.

Experiment 2: Structural Change

Structural change happens when the utility is transferred between two environments whose underlying PGMs for the transition $\mathcal{T}$ have the same fluent nodes $\mathcal{F}$ but different causal structures or different numbers of nodes. Note that causal structures of environments always implicitly enforce ordering in demonstration sequences. If the ordinal information in these sequences only reflects causality, there should be no difference between MEIP and IRL. However, we humans are cultural creatures. There are lots of things we do in an order not because they are the only feasible ways, but due to certain social conventions, *e.g.* the traditional order in a wedding ceremony, the stroke order in hand-writing, *etc.* We acquire social utility by following these conventions. It is under these situations does MEIP differentiate from IRL.

Consider the scenario in Fig.3 that hypothetically took

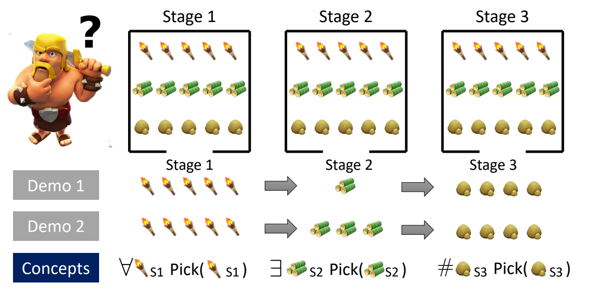

Figure 3: There are 3 stages with abundant preps. Alex was shown demos by the chief who followed a prescribed order to fetch different numbers of **torch, bamboo, clay**.

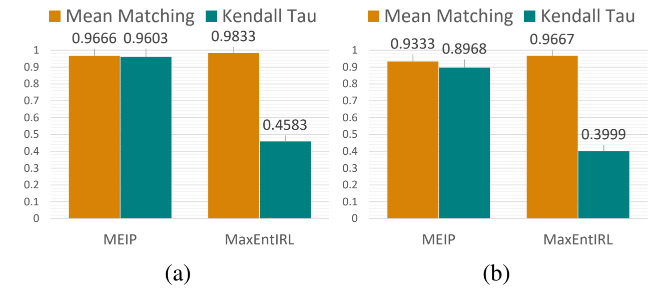

Figure 4: (a) Ordinal and mean matching when agents are given world models that do not enforce the demonstrated temporal ordering. (b) Results after an extra change in entity quantity. These results are the average of trajectory samples from 20 convergences of MCTS with the learned utility.

place in early history when utterance was not at all easy for our ancestors. Alex was a new-comer to a tribe. He was invited to a ritual host by the chief of the tribe. The ritual went like this: There were 3 stages in total. In stage 1, Alex saw the chief took out **all torches**. After stage 2, Alex was showed **1 or 3 bamboos**. Eventually, the chief fetched **4 pieces of clay** from stage 3. As a member of this tribe, Alex need to understand the process of this ceremony. And the metric to test his understanding is how he would imagine himself hosting it, presuming some environmental changes.

The first evaluation metric, mean matching, tests whether the learned utility is associated with the correct set of concepts and thus can be transferred to another environment with different quantities of objects. In the demonstrated setting, there are 5 torches, 5 bamboos and 5 pieces of clay in each stage. After the structural change, there are 6 pieces of each objects. The ground-truth concepts of these demonstrations are $\forall\ _{S1}\ Pick(\ _{S1})$, $\exists\ _{S2}\ Pick(\ _{S2})$, $\#\ _{S3} Pick(\ _{S3})$. In plain English, the learner needs to fetch all torches from stage 1, any number of bamboo from stage 2 and 4 of pieces of clay from stage 3. We estimate the mean statistics of matching these concepts in sequences planned with utility learned by MEIP and MaxEnt-IRL. Both agents successfully discover the correct set of concepts with the help of our concept language (Fig.4a). These concepts empower them to generalize learned utility to environments with different quantities of entities (Fig.4b).

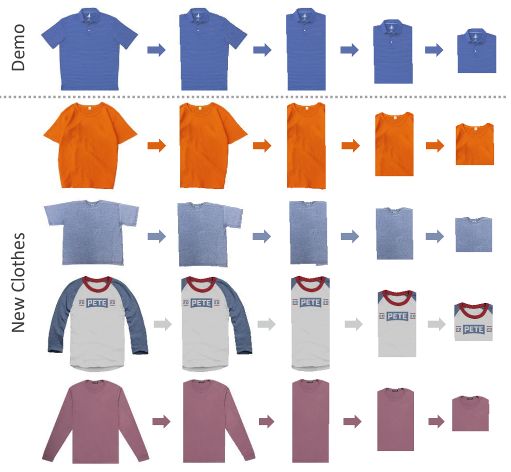

Figure 5: Qualitative results of transferring the learned utility to folding different clothes. Note that the shape of `polygons` and the number of `edge` segments in the testing clothes are substantially different from the one in demos.

The second metric is for ordinal relations in sequences. We estimate the Kendall τ of planned sequences with the ground-truth ordering $S1 \rightarrow S2 \rightarrow S3$. To test this generalization, structural change needs to alter causality. And this alternation is done by controlling the world model provided to agents. When the transition in the world model directly enforces the required ordering, both agents have $\tau = 1$ when planning with this world model. However, after changing to a transition without causality constraint on ordering, namely, agents can choose to enter any stage in any order, only the MEIP agent can attain the desired ordering ($\tau = 96.03\%$) with the learned utility. We also tried a controlled setting where we provide a world model that does not have causality constraints to both agents. The contrast between the planned behaviors of the MEIP agents and the MaxEntIRL agent remains the same. These results justify that MEIP can learn not only correct concepts but also the desired temporal order. On the other hand, MaxEnt-IRL fails to capture the ordinal information. As illustrated in Fig.4, mean matching does not enforce ordinal utility learning.

Experiment 3: *Learning to Fold*

Another task that reflects the ubiquity of this social utility in our daily life is the running example we start from the introduction, cloth folding. If the utility was only associated with the final state that the cloth becomes a square whose area is within a range, there would not be those exemplar sequences you recall when you hear this task. We conduct an experiment to learn the utility of cloth folding in a visually and geometrically authentic simulator.

To represent the cloth to be folded under our formulation, we adopt a grammar, Spatial And-Or-Graph (S-AOG) (Zhu and Mumford 2007) for the image input. Details of this grammar, as well as other technical details, can be found in the supplementary. Given the visual input of a cloth, the agent should *parse* it with this grammar to acquire a

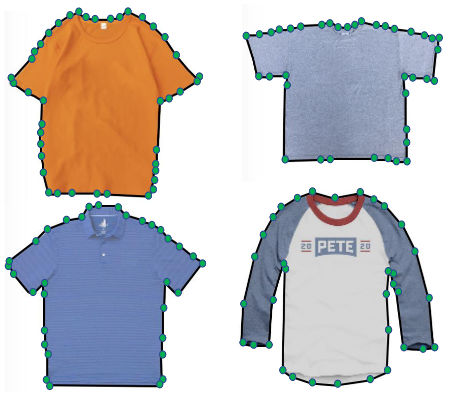

Figure 6: Nodes of the underlying structures, illustrated as `edges` and `vertexes` of different clothes. They are significantly different in different clothes.

first-order representation. For the sake of simplicity, we ignore the uncertainty from perception. We also endow the agent with an action model, assuming it can imagine geometric transformations like a toddler. As such, the agent has sufficient knowledge of each problem instance to rollout with MCTS. In our experiments, we collect 15 "good folds" from human demonstrators. When presented with these folding sequences, the agent with MEIP learns the utility and concepts that successfully generalize to unseen clothes, see Fig.5. Even though all shirts (or sweater) look similar on their appearances, their underlying structures are significantly different in the number, location, and orientation of `edges` and `vertexes`, see Fig.6.

Concluding Remarks

In this work we propose a new quest for learning generalizable task representation, especially its utility, from demonstrations. This problem lies outside of the regime of inverse reinforcement learning and thus dubbed *Generalized Inverse Planning*. We then outline *the computational principles in the cognitive process* (Marr and Poggio 1976) of lifted non-Markovian utility learning, which we model as Maximum Entropy Inverse Planning (MEIP). Comparing to existing inverse reinforcement learning methods, our model learns a task representation that generalizes regardless of probability shift and structural change in the environment. To highlight this contribution, we exclude irrelevant representation learning by adopting classical assumptions and representation in planning, *i.e.* grounded semantics of entities and relations are given *a priori*, as well as an action model over them. This kind of assumption was also made in original works of IRL (Abbeel and Ng 2004), (Ratliff, Bagnell, and Zinkevich 2006),(Ziebart et al. 2008), (Munzer et al. 2015). To disclaim, we are aware that such assumption on priors might be regarded as too strong in modern days since there are some recent progress in deep reinforcement learning that successfully induces them with a weaker assumption of relational inductive biases (Zambaldi et al. 2019). Our model is general enough to be extended to those neural networks. It would an interesting future work to investigate their synergy.

Acknowledgments

The authors thank Prof. Ying Nian Wu and Dr. Yixin Zhu of UCLA Statistics Department, Prof. Guy Van den Broeck of UCLA Computer Science Department for useful discussions. The work reported herein was supported by ONR MURI grant N00014-16-1-2007, ONR N00014-19-1-2153, and DARPA XAI N66001-17-2-4029.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on Machine learning*, 1.

Amit, R.; and Matari, M. 2002. Learning movement sequences from demonstration. In *Proceedings 2nd International Conference on Development and Learning. ICDL 2002*, 203–208. IEEE.

Atkeson, C. G.; and Schaal, S. 1997. Robot learning from demonstration. In *ICML*, volume 97, 12–20. Citeseer.

Bagnell, J.; Chestnutt, J.; Bradley, D. M.; and Ratliff, N. D. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, 1153–1160.

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition* 113(3): 329–349.

Bonet, B.; and Geffner, H. 2018. Features, projections, and representation change for generalized planning. *arXiv preprint arXiv:1801.10055* .

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.

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Nutt, W. 1997. The complexity of concept languages. *Information and Computation* 134(1): 1–58.

Džeroski, S.; De Raedt, L.; and Driessens, K. 2001. Relational reinforcement learning. *Machine learning* 43(1-2): 7–52.

Fikes, R. E.; and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3): 189 – 208.

Finn, C.; Christiano, P.; Abbeel, P.; and Levine, S. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852* .

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232.

Garrett, C. R.; Kaelbling, L. P.; and Lozano-Pérez, T. 2016. Learning to Rank for Synthesizing Planning Heuristics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 3089–3095.

Grenander, U. 1993. *General pattern theory: A mathematical study of regular structures Oxford mathematical monographs*. Oxford University Press: Clarendon.

Hayes, G. M.; and Demiris, J. 1994. *A robot controller using learning by imitation*. University of Edinburgh, Department of Artificial Intelligence.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in neural information processing systems*, 4565–4573.

Icarte, R. T.; Klassen, T.; Valenzano, R.; and McIlraith, S. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, 2107–2116.

Jaynes, E. T. 1957. Information theory and statistical mechanics. *Physical review* 106(4): 620.

Joachims, T. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101(1-2): 99–134.

Kersting, K.; and Driessens, K. 2008. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In *Proceedings of the 25th international conference on Machine learning*, 456–463.

Khardon, R. 1999. Learning action strategies for planning domains. *Artificial Intelligence* 113(1-2): 125–148.

LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; and Huang, F. 2006. A tutorial on energy-based learning. *Predicting structured data* 1(0).

Li, X.; Vasile, C.-I.; and Belta, C. 2017. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3834–3839. IEEE.

Littman, M. L.; Topcu, U.; Fu, J.; Isbell, C.; Wen, M.; and MacGlashan, J. 2017. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341* .

Liu, T.-Y. 2011. *Learning to rank for information retrieval*. Springer Science & Business Media.

Marr, D. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. USA: Henry Holt and Co., Inc. ISBN 0716715678.

Marr, D.; and Poggio, T. 1976. From understanding computation to understanding neural circuitry .

Martin, M.; and Geffner, H. 2004. Learning generalized policies from planning examples using concept languages. *Applied Intelligence* 20(1): 9–19.

Munzer, T.; Piot, B.; Geist, M.; Pietquin, O.; and Lopes, M. 2015. Inverse reinforcement learning in relational domains. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Natarajan, S.; Joshi, S.; Tadepalli, P.; Kersting, K.; and Shavlik, J. 2011. Imitation learning in relational domains: A functional-gradient boosting approach. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Ng, A. Y.; Russell, S. J.; et al. 2000. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, 2.

Nitzberg, M.; and Mumford, D. B. 1990. *The 2.1-D sketch*. IEEE Computer Society Press.

Ortega, P. A.; and Braun, D. A. 2013. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469(2153): 20120683.

Ratliff, N. D.; Bagnell, J. A.; and Zinkevich, M. A. 2006. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, 729–736.

Silver, T.; Allen, K. R.; Lew, A. K.; Kaelbling, L. P.; and Tenenbaum, J. 2020. Few-Shot Bayesian Imitation Learning with Logical Program Policies. In *AAAI*, 10251–10258.

Spelke, E. S.; and Kinzler, K. D. 2007. Core knowledge. *Developmental science* 10(1): 89–96.

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2011. A new representation and associated algorithms for generalized planning. *Artificial Intelligence* 175(2): 615–647.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2): 181–211.

Vazquez-Chanlatte, M.; Jha, S.; Tiwari, A.; Ho, M. K.; and Seshia, S. 2018. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems*, 5367–5377.

Wu, Y. N.; Gao, R.; Han, T.; and Zhu, S.-C. 2019. A tale of three probabilistic families: Discriminative, descriptive, and generative models. *Quarterly of Applied Mathematics* 77(2): 423–465.

Xu, Y.; Fern, A.; and Yoon, S. 2009. Learning Linear Ranking Functions for Beam Search with Application to Planning. *Journal of Machine Learning Research* 10(7).

Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; et al. 2019. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*.

Zhu, S.-C.; and Mumford, D. 2007. *A stochastic grammar of images*. Now Publishers Inc.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, 1433–1438. Chicago, IL, USA.

Algorithmic Details

Maximum Entropy Inverse Planning (MEIP)

For the purpose of generalization, the ranking function $g_\omega(x)$ of utility is designed to be a piece-wise linear function. When $x \in \mathbb{R}^1$, the form of the function is:

$$g(x) = \begin{cases} k_0 x + b_0 & x \in [x_0, x_1] \\ k_1 x + b_1 & x \in [x_1, x_2] \\ ... \\ k_n x + b_n & x \in [x_n, x_{n+1}]. \end{cases}$$

Cases where x has higher dimensions can be derived accordingly. To make sure the continuity, we further add constraint on the connection between two consecutive bins, i.e. $k_i x + b_i = k_{i+1} x + b_{i+1}$.

The training process of MEIP follows the principle of *analysis by synthesis*. According to Eq.11 there are two parts to be optimized, one for demonstrations Ψ^E and one for sampled plans Ψ. Specifically, the learning is implemented with a Ranking SVM, see Eq.12. Ordinal relations between pairs are specified in the main text after Eq.11. For a $\prec$ pair $\langle s_1, s_2 \rangle$, we set $g(s_2) - g(s_1) \geq 1 - \xi$ for the Ranking SVM.

The sampling process is based on a Monte Carlo Tree Search (MCTS). First, we initialize the value function V of the MCTS. We keep doing rollout to compute $\tau_\omega(\zeta)$ for trajectories ζ until value functions in MCTS converges. After that, we choose to sample a small number of trajectories Ψ according to the values on the branches of the Monte Carlo Tree. These trajectories, together with the human demonstrations Ψ^E, are used to update the utility function.

Concept Pursuit

As mentioned in the main text, we generate first-order concepts from a concept language. However, the size of this language could be infinitely large. Thus we adopt the principle of Occam's razors. We herein describe the algorithm for concept pursuit in algorithm 2.

There are three main steps in the pursuit process. First, we sort the concepts according to the complexity. The measurement of complexity is based on the number of predicates (or functions) in a concept. The more predicates (functions) in a concept, the higher complexity it is. This discrete nature introduces *levels* in the complexity.

The second step is to exclude concepts that are not relevant at all. We can select those relevant concepts by simply counting the number of violations in the state pairs. The last and most important step is to select concepts according to Eq. 19 for the utility we learn. Each concept is added to the selected set Δ in a step-wise greedy manner. Note that concepts that share the same predicate (or function) and the same domain but different quantifiers are mutually exclusive.

Details for Experiment 2

Concept Space in Ritual Learning

In the ritual learning experiment, we only consider the concepts that are generated by the rule QPD in which Q is the

Algorithm 2: Concept Pursuit

```
Input: Sorted Concept Set $\mathcal{C}$; Threshold $\epsilon$
Data: The Set of State Pairs : $\{\langle s_i, s_j\rangle, ...\}$
Result: Selected Concept Set $\Delta$; Optimized
        Parameters $\omega$
Init: Selected Concepts Set $\Delta = \{\}$
foreach concept subset $\mathcal{C}_{l_i} \subset \mathcal{C}$ do
    foreach first-order concept $c \in \mathcal{C}_{l_i}$ do
        violation count $v_c \leftarrow 0$
        foreach state pair $\langle s_i, s_j\rangle$ do
            if $c(s_j) \neq c(s_i)$ then
                $v_c \leftarrow v_c + 1$
            end
        end
        if $v_c = 0$ then
            remove $c$ from $\mathcal{C}_{l_i}$
        end
    end
end
foreach concept subset $\mathcal{C}_{l_i} \subset \mathcal{C}$ do
    while True do
        $\{\delta_{tmp}\} = \{\}$
        foreach $c \in \mathcal{C}_{l_i}$ do
            compute $\delta_c$ by calling algorithm 1 with
              $\Delta + c$
            add $\delta_c$ to $\{\delta_{tmp}\}$
        end
        if $max(\{\delta_{tmp}\}) \leq \epsilon$ then
            break
        end
        else
            $c_{max} = argmax(\{\delta_{tmp}\})$
            $\Delta \leftarrow \Delta + c_{max}$
            update $\omega$
        end
    end
end
```

quantifier, P is the predicate, and D is the domain. In the ritual learning experiment, the domain $D = (O, S)$ in which O is the set of object types and S is the set of stages. A primitive concept in this experiment can be expressed as:

$$\forall|\exists|\#\texttt{picked}(o \in O, s \in S)$$

The *primitive predicate*, `picked`, describes whether the host carries some type of object from a specific stage. As introduced in the main text, *primitive concepts* represented by *atomic formulas* with the primitive predicate are *terms* for formulas of more complex concepts. Here these more complex concepts are encodings of the interrelation between primitive concepts.

Experimental Details

The Environment is designed to be an analogy to a ritual. There are different stages in the environment. Both the agent and the demonstrator are required to choose a certain stage before they can advance to pick up objects in it. After locking down a specific stage, one will be asked to choose one type of object and pick up the $[0, +\infty)$ of them ($+\infty$ means all). None of the stages can be visited more than once. The ritual will be terminated after all stages are visited.

Human Demonstrations consist of at least 3-5 sequences. A sequence consists of an ordered descriptions of objects that the demonstrator obtains at each stage. The demonstrator can only choose to pick one type of object at one stage without limitation on the quantity. Examples of demonstrations are $5\times$ $_{S1} \rightarrow 3\times$ $_{S2} \rightarrow 4\times$ $_{S3}$ or $1\times$ $_{S1} \rightarrow 2\times$ $_{S2} \rightarrow 2\times$ $_{S3}$. There is no doubt that we can have a longer demo if it is legal in the environment, although each demo only have 3 stages in our experiment. Note that all demos must contain exactly the same set of specific concepts.

Hyper-parameters of this experiment are listed as the following. MCTS converge condition: terminate after 3000 iterations. Size of sample trajectories $\Psi = \{\zeta_j\}$: 5. Upper confident bound coefficient: 1.

Details for Experiment 3

Representation for Visual Input of Clothes

We adopt a stochastic grammar, Spatial And-Or-Graph (S-AOG) (Zhu and Mumford 2007) for the visual input of all clothes. The design of this grammar follows Gestalt Laws in vision (Marr 1982). Here we informally summarize its *production rules*. A `cloth` is an *And* node that *produces* a set of `polygons`. The number of `polygons` may change after being folded. Since some `polygons` may be occluded in the visual input, we adopt a 2.1D representation (Nitzberg and Mumford 1990). The 2.1D representation is a layer representation. In our case, the order of the layer is consistent with the folding order. All `polygons` belong to the same class with a template set of *fluents*. They produce a set of line segments, `edges`. Different configurations of `edges` in one `polygon` consist an *Or* node. All `edges` also belong to the same class. Each `edge` is associated with two `vertexes` as its attribute. Each `vertex`is specified by its coordinate. Classes introduced above are regarded as domains in the concept language.

The full fluent set for this grammar is designed following axioms in Euclidean geometry, as showed in Table 1. Edge s, vertices, surfaces and their relations are also believed to be our core knowledge developed in early age (Spelke and Kinzler 2007). Fluents for `edges` are categorized into functions/relations between `edges` *e.g.* parallel, and functions/relations between one `edge`and one `vertex` of another `edge`*e.g.* distance. Other fluents, such as `Logo` and `Neck`, are for `polygons`, which are visual features. With these classes and fluents, we can generate *concepts* from the concept language.

During planning, the visual input of the cloth from each situation is parsed into a *parse graph* (*pg*) of the S-AOG. As shown in Figure 7, *pg* is a hierarchical representation in which the terminal node is an `edge`with two `vertexes` and non-terminal nodes are `polygons`. Clothes with the simplest structures, such as a shirt, are initially parsed into three

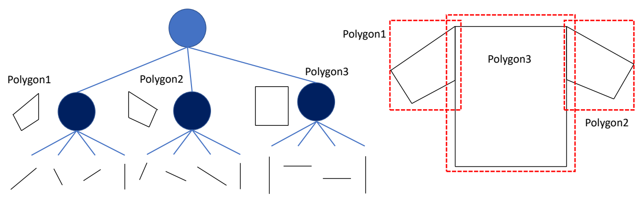

Figure 7: A minimal parse graph of the S-AOG

Name	Arity	Type
`Edge Length`	Unary	function
`Logo`	Unary	function
`Neck`	Unary	function
`Vt2Vt Distance`	Binary	function
`Vt2Edge Distance`	Binary	function
`Parallel`	Binary	predicate
`Perpendicular`	Binary	predicate
`Vertex on Edge`	Binary	predicate
`Edge on Edge`	Binary	predicate
`Vertex in Polygon`	Binary	predicate

Table 1: Fluent space for learning to fold

polygons and their own affiliated edges.

Experimental Details

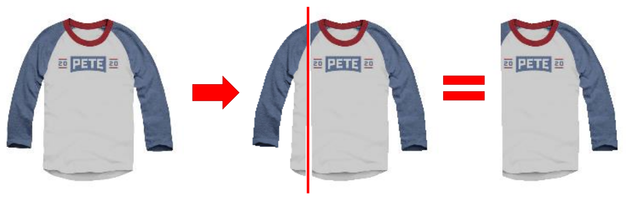

Figure 8: The action is a folding line in the simulator

The Environment is a simulator for folding. The demonstrator is asked to draw a folding line that splits a polygon into two new polygons. In Figure 8, we show an example of a legal folding line in the simulator. The polygon is separated by the folding line and the small one will be flipped to the back of the larger one after each fold.

Human Demonstrations are given by folding a demo shirt (or sweater). In a demo, states of the shirt are recorded, serialized as a sequence. A fold is not reversible therefore the demonstrator needs to consider the final state at every step. If the demonstrator made a "bad fold", it could have a significant impact on the final state. Unlike the previous experiment, the demonstrator does not have a prescribed concept set during the demo process. Instead, the demonstrator needs to conduct a folding sequence that can lead to good final states which meet their own criteria of "good folds" based on *default* sequences in their real-life habits. We collected 15 sequences as demonstrations.

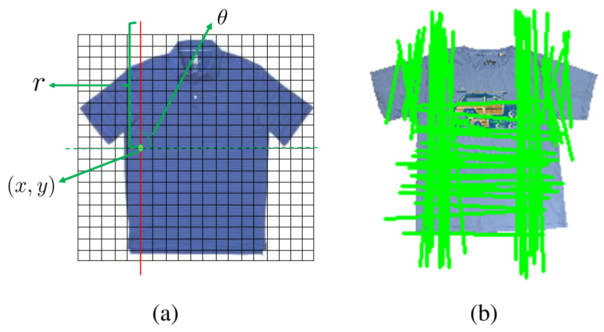

Figure 9: (a) The discretized action space for folds. (b) Samples of folds that with high probabilities.

The Action Space of a folding line is discretized to make MCTS applicable. It is defined as a tuple $a = (x, y, r, \theta)$ and each parameter is discretized accordingly. As illustrated in Figure 9a, (x, y) is coordinate of a point on the grid. The grid is a discretization of the bounding box of a shirt. r is the radius of the folding line and θ is the angle. Note that some folding lines may be redundant, therefore we need to check the uniqueness of each folding line.

Even though we have discretized action space, it is yet too large for MCTS with limited computational resources. Thus, it is necessary to have a reasonable number of legal folds for the MCTS. The solution is to learn a probabilistic distribution over the action space. The folds that are similar to demo folds will be associated with high probabilities. We assume that each parameter in a follows a normal distribution around some exemplars. See Figure 9b.