Similarity-Based Representation Learning

Yi Liu
Andreea Bobu
Anca Dragan

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-78
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-78.html

May 9, 2023

Acknowledgement

I would like to first give my profound gratitude to Andreea Bobu, who has been an exceptional research mentor and a guiding force throughout my journey in the field of human-robot interaction. Her guidance and insight has been invaluable in shaping my understanding of research methodologies. I am truly grateful for her dedication and expertise. I would also like to extend my sincere gratitude to Professor Anca Dragan for accepting me into the lab and giving me the opportunity to experiment in the field. It has been truly incredible to be able to work with a titan in this branch of research. In addition I would like to thank Professor Ken Goldberg for accepting into his lab as an undergrad. It was a blessing to learn about motion planning and robots through working on projects at his lab.

Similarity Based Representation Learning

Yi Liu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

Professor Anca Dragan
Research Advisor

5/9/23

(Date)

⋆ ⋆ ⋆ ⋆ ⋆ ⋆ ⋆

Professor Sergey Levine
Second Reader

5/9/23

(Date)

Abstract

Similarity-Based Representation Learning

by

Yi Liu

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Anca Dragan, Chair

When robots optimize their behavior in an environment, they need to both learn a representation for what matters in the task – the task "features" – as well as how to combine these features into a single objective. The ability to learn meaningful representations from raw observations is crucial for efficient and effective reward and policy learning. This paper introduces a novel approach to representation learning, termed Similarity-Based Representation Learning (SIRL), motivated by the need to incorporate human feedback that generalizes to multiple tasks and multiple users. SIRL operates by querying the human on which two of three shown trajectories are more similar. By obtaining human feedback in this manner, we train a model using contrastive learning with triplet loss. This approach allows for the learning of robust representations that encode meaningful aspects of the environment, while discarding irrelevant information. We showcase the efficacy of our proposed SIRL framework through experiments on various benchmark tasks. Our results indicate that SIRL effectively learns representations that lead to improved performance in both preference-based reward learning and policy learning from human demonstrations. Moreover, we demonstrate the scalability and transferability of the learned representations, highlighting the potential of SIRL as a versatile and efficient tool for reinforcement learning in complex environments.

Acknowledgements

I would like to first give my profound gratitude to Andreea Bobu, who has been an exceptional research mentor and a guiding force throughout my journey in the field of human-robot interaction. Her guidance and insight has been invaluable in shaping my understanding of research methodologies and in determining new avenues to experiment with whenever I was stuck on an issue. I am truly grateful for her dedication and expertise. I would also like to extend my sincere gratitude to Professor Anca Dragan for accepting me into the lab and giving me the opportunity to experiment in the field. It has been truly incredible to be able to work with a titan in this branch of research. In addition I would like to thank Professor Ken Goldberg for accepting into his lab as an undergrad. It was a blessing to learn about motion planning and robots through working on projects at his lab. Lastly, it would have been impossible to reach where I currently am without the help of my family and it is their support that I am indebted to.

Contents

List of Figures

Chapter 1

Introduction

Imagine waking up in the morning and your home robot assistant wants to place a steaming mug of fresh coffee on the table exactly where it knows you will sit. Depending on the context, you will have a different preference for how the robot should be doing its task. Some days it carries your favorite mug close to the table to prevent it from breaking in the case of a slip (so that it will remain your favorite mug); some other days the steam from your delicious meal is difficult to handle for the robot's perception, so you'd want it to keep a large clearance from the table to avoid collisions. Similarly, some days you want the robot to keep your mug away from your laptop to avoid spilling on it; some other days the mug only has a small quantity of an espresso shot and so you'd rather the robot keep the mug close to the laptop to prevent clutter and leave the rest of the table open for you.

The reward function or policy that the robot learns changes due to variances in the tasks, having different users, or encountering different contexts that are not always part of the state the robot is using (e.g. holding the user's favorite mug and not just a regular mug). However, the *representation* on top of which the reward or policy is built, i.e the *features* that are important (like the distance from the table, being above the laptop, etc.), are shared. If the robot learns this representation correctly, it can use it to obtain the right behavior as the task, user, and context change.

Humans have the innate ability to compress information into more manageable representations and discard information that is irrelevant. Analogously, machine learning models develop a certain level of representation and comprehension of the input data provided to them. However, there is no guarantee that these models have accurately captured a representation that matches our internal interpretation of the scene. Representation learning approaches such as unsupervised learning that strives to learn features without human input may capture spurious correlations in the data and optimize features that are not generalizable to different tasks or users.

Meta-learning and multi-task learning methods [18, 30, 41] learn the representation from user input meant to teach the full reward, like preference queries or demonstrations. By contrast, we propose that if learning generalizable representations is the goal, then we should ask the user for input that is specifically meant to teach the representation itself, rather than asking for input meant to teach the full reward and hoping to extract a good

representation along the way.

Recent work in contrastive and unsupervised learning explicitly focus on learning good visual representations by training from (anchor, positive, negative) triplets generated via data augmentation techniques. Here, the contrastive loss induces a representation that makes visually similar anchors and positives map closer in the latent space and further from negatives. However, the notion of similarity is purely visual and driven by the data augmentation heuristics used. In contrast, we want our feature representations to align with people: because humans have adapted their environments to capture the full idiosyncrasies of completing tasks that they desire, these feature representations are implicit in their minds and so they are best equipped to help distill knowledge about them into the robot. As such, we introduce a novel type of human input to help the robot extract the person's feature representation of the tasks they might care about in the environment: *trajectory similarity queries*. A trajectory similarity query is a triplet of trajectories that the person answers by picking the two more similar trajectories. This results in an (anchor, positive, negative) triplet that can be used for training a feature representation. The intuition is that if two behaviors are similar, then their feature representation as seen by the person should also be similar. We call this process Similarity-Implicit Representation Learning (SIRL).

In this thesis, we show SIRL can learn a generalizable representation that allows for downstream policy and reward learning that outperforms baselines in multiple environments. Furthermore, we demonstrate SIRL can learn a representation that captures the ground truth features in the environment.

The thesis is organized as follows: In Chapter 2, we provide the context and topics surrounding representation learning and its uses. In Chapter 3, we introduced similarity-based representation learning, the loss function for training, and the evaluation measures. In Chapter 4, we present the experimental setup in simulated environments, including the baselines and training process. In Chapter 5, we investigate the simulated results of similarity-based representation learning and the baselines and show the benefits of using similarity as a method of representation learning. In Chapter 6, we provide results for a user study and show our method to work better even with real human data. In Chapter 7, we summarize our results and findings and look at future directions of our work.

Chapter 2

Background

In this section, we present a detailed overview of the background and relevant materials related to our method.

2.1 Reinforcement Learning

The development of sophisticated artificial intelligence systems that are capable of learning and making decisions autonomously has been a long-standing goal in the field of AI research. Reinforcement learning (RL) has emerged as a promising structure for achieving this objective due to its ability to adapt and optimize an agent's behavior in complex and dynamic environments. RL is a computational approach where the agent learns to perform tasks based only on a reward signal that is defined by the human. Environments are assumed to follow a Markov Decision Process (MDP), a discrete-time stochastic process where outcomes are based on randomness within the environment and the decisions of an agent. A MDP is defined as a tuple (S, A, P, R), where S represents the set of states, A denotes the set of actions, P defines the transition probabilities, and R is the reward function (Fig 2.1). An example is Pac-man where the agent has to avoid ghosts while eating all of the pellets. The reward is defined by the score that the agent receives, which is based on the number of pellets consumed and the capture of ghosts when they are vulnerable.

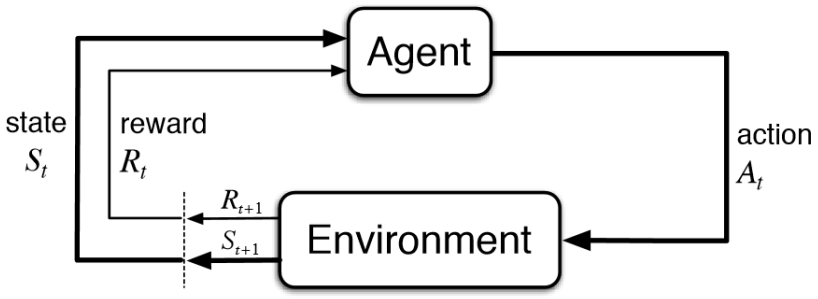


Figure 2.1: Given the state information from the environment, the agent applies an action, receiving the next state of the environment and a reward value.

One approach to reinforcement learning is q-learning [28]. Q-learning is a model-free, value-based reinforcement learning algorithm that aims to optimize an agent's decision-making process in a given environment by estimating the expected cumulative reward for each state-action pair. The core of Q-learning lies in a neural network that predicts Q-values for each combination of state and action. During test time, the agent calculates the q-value for each action at the current state, selecting the action that maximizes this q-value. After executing the action, the agent receives a reward and transitions to a new state, and the Q-value for the taken action is updated using the Bellman equation.

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \tag{2.1}$$

where $R(s,a)$ is the reward at the current state-action pair and γ is the discount factor for discounting future rewards.

For many environments, Q-learning is enough to find the optimal policy; however, for tasks such as performing a back flip (Fig 2.2) or assisting a human, it is difficult to define a reward function and a tremendous amount of effort is necessary to prevent the agent from "reward hacking", finding policies that maximize the cumulative reward but is not actually what the human wants.

Imitation learning presents an alternative solution to reinforcement learning by enabling an agent to acquire skills through observing expert demonstrations, thus side stepping the necessity of manually defining reward functions. In this context, the agent learns a policy that maps states to actions by emulating the expert's exhibited behavior. This approach proves beneficial in situations where learning from scratch is difficult, or where exploration entails significant costs or risks. By harnessing the expert's knowledge, imitation learning mitigates the agent's reliance on exploration. However, imitation learning has considerable faults, one of which is the inability to generalize to states that were not in the expert's demonstrations.

Figure 2.2: Imitation learning for performing a backflip. Defining a reward function for a policy to follow in this environment is difficult and require a tremendous amount of engineering.

2.2 Reward Learning

Instead of learning a policy directly from human demonstrations, we can alternatively first learn a human's reward function and then optimize a policy for the learned reward function. One method is through human preferences where humans answer queries about which trajectory or state-action pair is more preferred. Machine learning models can then leverage preference data to derive a reward function. However, a limitation of reward learning through preferences is its applicability to single-task learning. Environments

involving robot arms can have multiple tasks such as picking up blocks, moving blocks, or stacking blocks, necessitating a different human reward model for each task.

To learn multiple models of human reward functions, prior work has proposed clustering unlabeled demonstrations and learning a different reward function for each cluster [3, 10, 15]; however, these methods require a large number of demonstrations and do not adapt to new reward functions. Meta-learning approaches [17] seek to learn a reward function initialization that enables fast fine-tuning at test time [21, 35, 40, 42]. Multi-task reward learning approaches pretrain a reward function on multiple human intents and then fine-tune the reward function at test time [18, 30]. This has been shown to be more stable and scalable than meta-learning approaches [26], but still needs curating a large set of training environments. By contrast, we do not assume any knowledge of the test-time task distribution *a priori* and do not require access to a population of different reward functions during training. Rather, we focus on learning a method of task-agnostic representation learning model that can be utilized for multiple down-stream reward learning tasks and introduce representation learning in the next section.

2.3 Representation Learning

In supervised learning, machine learning models are designed to learn some mapping between input data and corresponding labels, such as classifying an image as containing a cat or a dog or discerning an email as spam or legitimate. Representation learning aims to uncover efficient and meaningful ways to represent raw data, capturing the underlying structure and patterns within. This approach entails generating a feature vector for each data point, which can subsequently be used as input for other downstream tasks. A well-trained representation model should be capable of excluding irrelevant information while learning high-level features of the data.

Examples of representation models encompass techniques such as variational autoencoders [24] (VAEs) and contrastive learning. In VAEs, an encoder generates a lower-dimensional feature vector, while a decoder attempts to reconstruct the original input from this representation. Contrastive learning, on the other hand, encourages the model to differentiate feature vectors from distinct classes, while simultaneously drawing together feature vectors from the same class. An example of contrastive learning is Contrastive Predictive Coding [31] (CPC) (fig 2.3), which learns to encode sequential data in a way that facilitates the prediction of future observations, relying on a contrastive loss function to distinguish between positive and negative samples. The framework comprises an encoder network, which generates latent representations of input data, and an autoregressive model that leverages the encoded context to predict future observations. The training process involves providing the model with a set of positive and negative samples: positive samples consist of actual future observations within the sequence, while negative samples are randomly sampled from other parts of the dataset. The model's objective is to maximize the similarity between the encoded context and the positive sample while minimizing the similarity with the negative samples.

In another instance of contrastive learning, Contrastive Unsupervised Representations

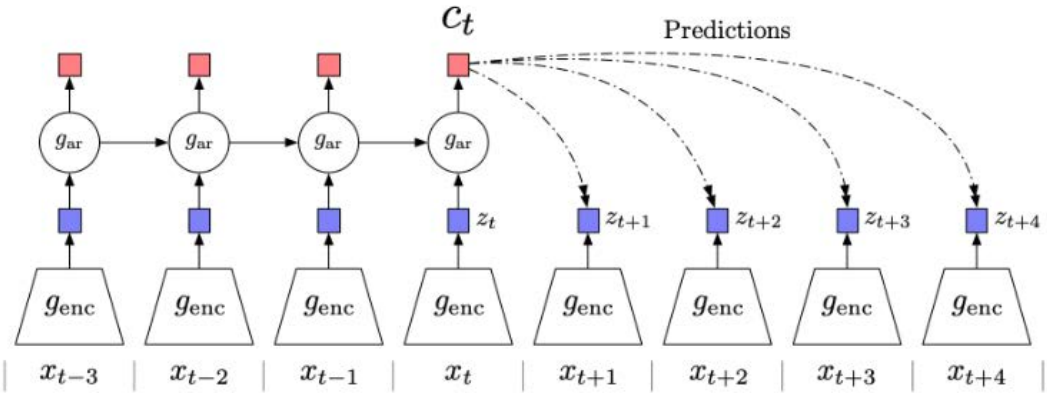


Figure 2.3: Contrastive Predictive Coding learns an encoding using contrastive learning

for Reinforcement Learning [36] (CURL) employs an encoder network to transform high-dimensional observations into a more manageable, lower-dimensional latent space. The technique incorporates a contrastive loss, similar to CPC, albeit with a distinct objective. Rather than converging representations of temporally proximate observations, as is the case with CPC, CURL seeks to bring together the representation of the image and the respective image with different image augmentations. As such, CURL learns a robust representation that is invariant to differences in the observations that, while visually dissimilar, to a human, doesn't change the inherent content.

However, assessing the quality of learned representations in these approaches can be challenging, as it is difficult to determine whether they have effectively captured the underlying structure of the data and not irrelevant signal within the data. To address this concern, human involvement can be integrated into the representation learning process, enabling the alignment of these models with attributes that are important to the human.

Two recent methods look at incorporating human input directly when learning representations. Reddy et al. [32] (Fig. 2.4) introduces Pragmatic Image Compression for Human-in-the-Loop Decision-Making (PLATO), a method that emphasizes representation learning for efficient compression. In PLATO, the compression model, consisting of an encoder and decoder, learns to extract and retain the most pertinent visual features that influence user behavior, while discarding extraneous details. This is achieved by training the encoder with an adversarial discriminator that attempts to distinguish between user actions taken in the original image state and those taken in the compressed image state. The learned representations capture essential elements of the image that align with the user's decision-making process, leading to more efficient and task-specific image compression. This novel representation learning approach allows the compression model to ensure that the compressed images maintain the critical information needed for effective decision-making while minimizing the required bit rate. By considering human input, PLATO refines its representation of the state space to focus on features that is important to humans.

Bobu et al. [6] presents a method for robots to learn missing features in their state representation by utilizing a new type of human input called feature traces. In this ap-

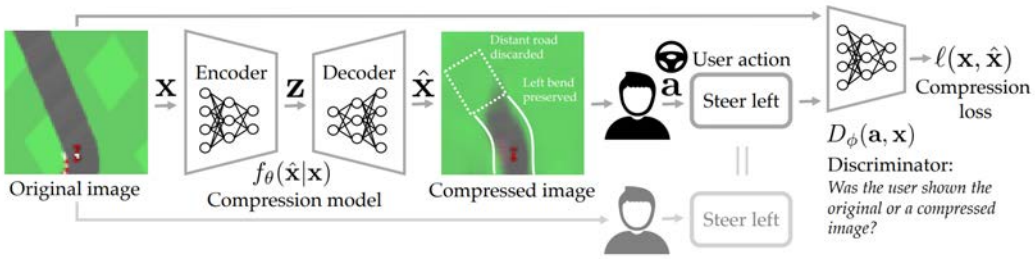


Figure 2.4: An overview of PLATO for efficient compression

proach, the person guides the robot between states where the missing feature is highly expressed and states where it is not, enabling the robot to learn the feature from the raw state space data. By explicitly focusing human input on the missing feature, the method improves the robot's understanding of the task, leading to better generalization and reduced sample complexity. In contrast, we learn a lower-dimensional feature representation all-at-once, rather than one at a time. Furthermore, rather than relying on the human to provide physical demonstrations for learning a good feature space [6, 7], we instead propose a more accessible and general form of human feedback: showing the user triplets of trajectories and simply asking them to label which two trajectories are the most similar, in a method similar to unsupervised contrastive learning. Triplet losses have been widely used to learn similarity models that capture how humans perceive objects [1, 2, 14, 27, 39]; however, to the best of our knowledge, we are the first to use a triplet loss to learn a general, task-agnostic similarity model of how humans perceive trajectories. We next introduce our method, similarity-based representation learning.

Chapter 3

Similarity-Based Representation Learning

3.1 Preliminaries

In this thesis, we consider a robot R operating in an environment with access to a human H. The goal of the robot is to maximize its performance on tasks by querying the human and learning from the human's responses. For each state $s \in \mathcal{S}$ in the environment, the robot executes some action $a \in \mathcal{A}$, inducing a next state s' following an unknown probabilistic transition function $P(s'|s,a)$ and receiving a reward R. The human knows the reward function r^* for each task in the environment and the optimal policy $\pi^*(a_H|s)$ that maximizes the cumulative reward from the reward function for that task. The robot does not know this reward function and must learn from the human's responses.

We introduce policy learning (imitation learning) and reward learning as two approaches to this scenario. Policy learning explicitly learns the actions that the human would choose for the states in the environment by learning a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ purely from human demonstrations that maximizes the probability of taking the human's actions at the respective states. The robot can then use its learned policy at test time with the assumption that the human provided near optimal demonstrations that maximized the total reward possible from the starting state.

Reward learning, on the other hand, learns a scalar reward value for behaviors. The robot's goal is to learn the human's preference over trajectories given by r^* that is unobserved by the robot and must be learned from human interaction. The robot reasons over a parameterized approximation of the reward function R_θ, where θ represents the parameters of a neural network. To learn θ, the robot collects human preference labels over trajectories and seeks to find parameters θ that maximize the likelihood of the human input. The robot can then use the learned reward function to score behaviors during motion planning.

We focus on explicitly using human input to first learn a good representation and then use that representation for either downstream policy or reward learning. We introduce the similarity-based representation learning framework (SIRL) that can be used in multiple

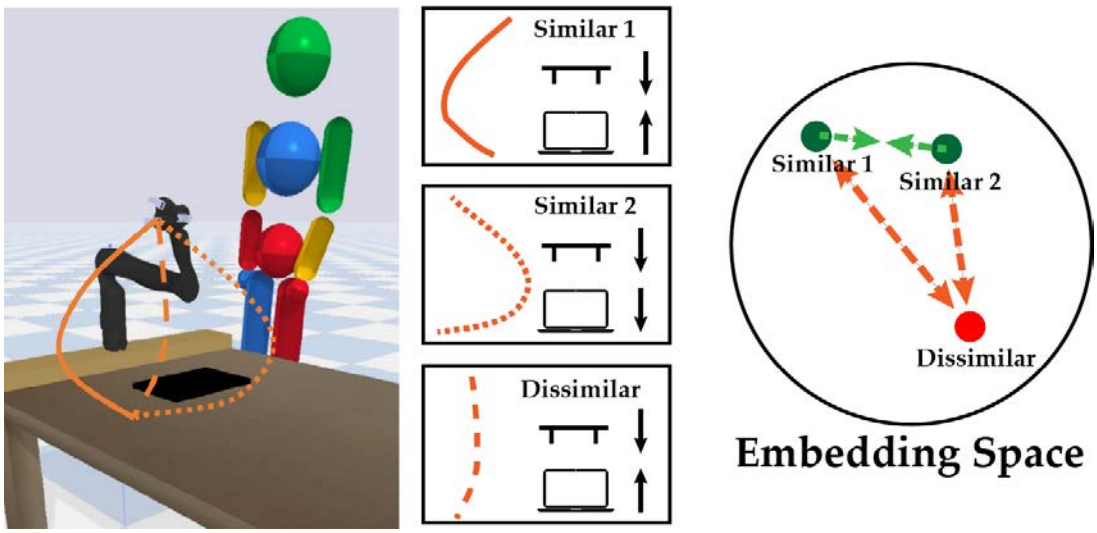


Figure 3.1: For a triplet of trajectories, the human selects the two most similar trajectories based on distance to the laptop and distance to the table. During training, the representation model pushes together the embedding for those two similar trajectories while pushing apart the embedding for the dissimilar trajectory.

tasks to improve model generalization and performance.

3.2 SIRL Framework

In order to learn a generalizable representation for both states and trajectories, we treat states as length 1 trajectories and operate on learning representations for trajectories of arbitary length.

SIRL learns a latent space that is useful for multiple downstream tasks such that similar trajectories have representations that are close in Euclidean space. One way to learn a model of similarity would be to ask users to judge whether two trajectories are similar or not; however, humans are better at giving relative rather than binary or quantitative assessments of similarity [23, 38]. Another idea for learning this representation is treating it as a regression problem and asking the human for feature values directly. Unfortunately, to learn anything useful, the robot would need a very large set of labels from the person and furthermore, it is difficult for humans to directly provide feature values by simply looking at the state or trajectory. Thus, we instead focus on qualitative similarity queries.

For learning representations, we present the user with a visualization of three trajectories and ask them to pick the two most similar ones (equivalently the most dissimilar one). In Fig 3.1, human's queries form a data set $\mathcal{D}_{sim} = \{\xi_{P_1}, \xi_{P_2}, \xi_N\}$, where ξ_{P_1} and ξ_{P_2} are the trajectories that are most similar and ξ_N is the trajectory most dissimilar to the other two.

Given a dataset of similarity queries, $\mathcal{D}_{sim}$, we make use of the triplet loss [5]:

$$\mathcal{L}_{trip}(\xi_A, \xi_P, \xi_N) = \max\left(\|\phi(\xi_A) - \phi(\xi_P)\|_2^2 - \|\phi(\xi_A) - \phi(\xi_N)\|_2^2 + \alpha, 0\right) \quad , \tag{3.1}$$

a form of contrastive learning where ξ_A is the anchor, ξ_P is the positive example, ξ_N is the negative example, and $\alpha \geq 0$ is a margin between positive and negative pairs. However, because our queries do not contain an explicit anchor, our final loss is as follows:

$$\mathcal{L}_{sim}(\phi) = \sum_{\{\xi_{P_1}, \xi_{P_2}, \xi_N\} \in \mathcal{D}_{sim}} \mathcal{L}_{trip}(\xi_{P_1}, \xi_{P_2}, \xi_N) + \mathcal{L}_{trip}(\xi_{P_2}, \xi_{P_1}, \xi_N) \ . \tag{3.2}$$

We train a similarity embedding function $\phi : \xi \mapsto \mathbb{R}^d$, where d is the dimensionality of the representation, that minimizes the above similarity loss. The intuition is that optimizing this loss should push together the latent embeddings of similar trajectories and push apart the latent embeddings of dissimilar trajectories. Before training the representation with the loss in Eq. (3.2), we may also pre-train it using unsupervised learning [24].

Lastly, in order to evaluate the efficacy of the learned representation, we introduce the policy learning (imitation learning) task for state representations and preference learning for trajectory representations.

Algorithm 1 SIRL: Similarity-based Representation Learning

Require: N: number of similarity queries, f: human similarity query function

Initialize pretrained representation learning model Φ
Initialize dataset of triplets $\mathcal{D} \leftarrow \emptyset$
Collect a large set of trajectories $\mathcal{T}$
for $i \in \{1, \ldots, N\}$ **do**
    Sample 3 trajectories randomly: $\tau_1, \tau_2, \tau_3 \sim \mathcal{T}$
    Query human for similarity: $k \leftarrow f(\tau_1, \tau_2, \tau_3)$
    Obtain the most similar pair of trajectories: (τ_{k_1}, τ_{k_2})
    Add triplet to dataset: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tau_{k_1}, \tau_{k_2}, \tau_{k_3})\}$, where $k_3 \neq k_1, k_2$
end for
while not converged **do**
    $(\tau_{a_1}, \tau_{a_2}, \tau_{a_3}) \sim \mathcal{D}$
    $\mathbf{e}_i \leftarrow \Phi(\tau_{a_i})$ for $i \in \{1, 2, 3\}$
    $\mathcal{L} = \max(\|\mathbf{e}_{a_1} - \mathbf{e}_{a_2}\|^2 - \|\mathbf{e}_{a_1} - \mathbf{e}_{a_3}\|^2 + \alpha, 0) + \max(\|\mathbf{e}_{a_1} - \mathbf{e}_{a_2}\|^2 - \|\mathbf{e}_{a_2} - \mathbf{e}_{a_3}\|^2 + \alpha, 0)$
    $\Phi \leftarrow \Phi - \eta \nabla_\Phi \mathcal{L}$
end while

Ensure: Φ: learned representation

3.2.1 Policy Learning

Given a learned embedding ϕ, we can use it to mimic human demonstrations. We collect a dataset of human demonstrations $\mathcal{D}_{\text{demo}}$ by concatenating human rollouts in the environment. We then train an imitation learning policy $\pi_{BC} : \mathcal{S} \to \mathcal{A}$ on each state-action pair in $\mathcal{D}_{\text{demo}}$. We learn a policy by incorporating the cross entropy loss if the environment actions are discrete.

$$\mathcal{L}_{BC}(\theta) = -\sum_{(s,a)\in\mathcal{D}_{\text{demo}}} \log \pi(\phi(s);\theta) \ . \tag{3.3}$$

We similarly incorporate the mean squared error loss if the environment actions are continuous.

$$\mathcal{L}_{BC}(\theta) = -\sum_{(s,a)\in\mathcal{D}_{\text{demo}}} (\pi(\phi(s);\theta) - a)^2 \ . \tag{3.4}$$

3.2.2 Reward Learning

Given a learned embedding ϕ, we use it for learning models of specific user preferences. While we focus on learning from pairwise preferences, we note that ϕ can in principle be used in downstream tasks that learn from many types of human feedback [22]. When learning a reward function from human preferences, we show the human two trajectories, ξ_A and ξ_B, and then ask which of these two the human prefers. We collect a data set of such preferences $\mathcal{D}_{pref} = \{\xi_A, \xi_B, \ell\}$ where $\ell = 1$ if $\xi_A \succ \xi_B$ and $\ell = 0$ otherwise and use the Bradley-Terry preference model [8]:

$$P(\xi_A \succ \xi_B;\theta) = \frac{e^{R_\theta(\phi(\xi_A))}}{e^{R_\theta(\phi(\xi_A))} + e^{R_\theta(\phi(\xi_B))}} \ . \tag{3.5}$$

We learn the reward function by incorporating Eq. (3.5) in a simple cross-entropy loss:

$$\mathcal{L}_{pref}(\theta) = -\sum_{(\xi_A,\xi_B,\ell)\in\mathcal{D}_{pref}} \ell \cdot \log P(\xi_A \succ \xi_B;\theta) + (1-\ell)\cdot \log P(\xi_B \succ \xi_A;\theta) \ . \tag{3.6}$$

Chapter 4

Experiments in Simulation

Initially, we conduct experiments within simulated environments to compare the performance of SIRL against established baselines in two separate tasks: reward learning and imitation learning. We then validate our method with a user study.

4.1 Experimental Setup for Reward Learning

We first investigate the quality of SIRL-trained representations and their benefits for preference learning using simulated human input in two environments with ground truth rewards and features.

4.1.1 Environments

GridRobot (Fig. 4.1) is a 5-by-5 gridworld with two obstacles and a laptop (denoted by the blue, green, and black boxes). Trajectories are sequences of 9 states with the start and end in opposite corners. The 19-dimensional input space consists of the x and y coordinates of each state and a discretized angle in $\{-90°, -60°, -30°, 0°, 30°, 60°, 90°\}$ at the end state. The simulated human answers queries based on 4 features ϕ^* in this world: Euclidean distances to each object, and the absolute value of the angle orientation.

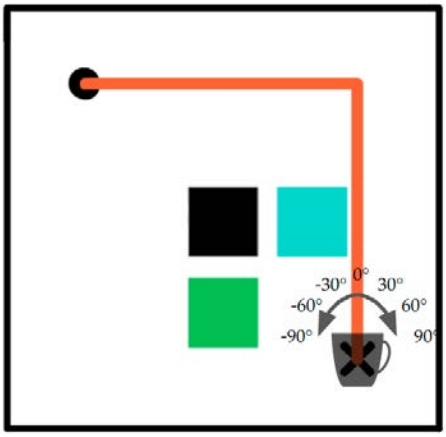


Figure 4.1: GridRobot.

JacoRobot (Fig. 4.2) is a pybullet [13] simulated environment with a 7-DoF Jaco robot arm on a tabletop, with a human and laptop in the environment. Trajectories are length 21, and each state consists of 97 dimensions: the xyz positions of all robot joints and objects, and their rotation matrices. This results in a 2037-dimensional input space, much larger than for GridRobot. The 4 features of interest ϕ^* for the simulated human are: a) *table* — distance of the robot's End-Effector (EE) to the table; b) *upright* — EE orientation relative to upright, to consider whether objects are carried upright; c) *laptop* — xy-plane distance of the EE to a laptop, to consider whether the EE passes over the laptop at any height; d) *proxemics* [29] — proxemic xy-plane distance of the EE to the human, where the EE is considered closer to the human when moving in front of the human that to their side.

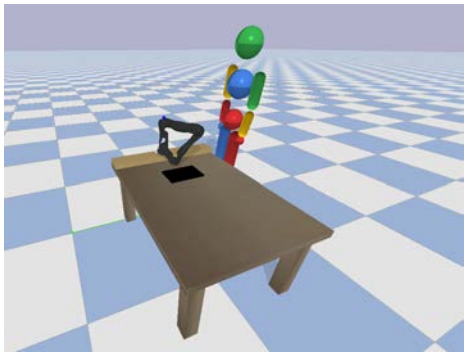


Figure 4.2: Jacorobot.

In GridRobot the state space is discretized, so the trajectory space Ξ can be enumerated; however, the JacoRobot state space is continuous, so we construct Ξ by smoothly perturbing the shortest path trajectories from 10,000 randomly sampled start-goal pairs (see App. A.1). We generate similarity and preference queries by randomly sampling from Ξ. The simulated human answers similarity queries by computing the 4 feature values for each of the three trajectories and choosing the two that were closest in the feature space. For preference queries, the simulated human computes the ground truth reward and samples the trajectory with the higher reward. The space of true reward functions (used to simulate preference labels) is defined as linear combinations of the 4 features described above. The robot is not given access to the ground-truth features nor the ground-truth reward function but must learn them from similarity and preference labels over raw trajectory observations.

4.1.2 Experimental Setup

Manipulated Variables. We test the importance of user input that is designed to teach the representation by comparing SIRL with multi-task learning techniques from generic preference queries, and unsupervised representation learning. We have 4 baselines: a) ***VAE***, which learns a representation with a variational reconstruction loss [24]; b) ***MultiPref***, a multi-task baseline [18, 26, 30], where we learn the representation ϕ implicitly by training multiple reward functions (each with shared initial layers) via preference learning; c) ***SinglePref***, a hypothetical method that learns from an ideal user who weighs all features equally; d) ***Random***, a randomly initialized embedding, which does not benefit from human data but is also immune from any spurious correlations that might be learned from biased data. For MultiPref, we trained versions with 10 and 50 simulated human preference rewards for good coverage of the reward space. All embeddings have the same network size: for GridRobot we used MLPs with 2 layers, 128 units each, mapping to 6 output neurons, while for JacoRobot we used 1024 units to handle the larger input space (see App. A.2). For a fair comparison, we gave SIRL, SinglePref, and MultiPref equal

amounts of human data for pre-training: N similarity queries for SIRL, and N preference queries (used for a single human for SinglePref or equally distributed amongst humans for MultiPref). We also performed ablations with and without VAE pre-training and found that SinglePref and MultiPref are better without the VAE objective (see App. A.3).

Dependent Measures. To test the quality of the learned representations, we use two metrics: *Feature Prediction Error (FPE)* and *Test Preference Accuracy (TPA)*. The ***FPE*** metric is inspired by prior work that argues that good representations are linearly separable [12, 25, 33]. Our goal is to measure whether the embeddings contain the necessary information to recover the 4 ground-truth features in each environment. We generate data sets of sampled trajectories labeled with their ground truth (normalized) feature vector $\mathcal{D}_{FPE} = \{\xi, \phi^*\}$. We freeze each embedding and add a linear regression layer on top to predict the feature vector for a given trajectory. We split $\mathcal{D}_{FPE}$ into 80% training and 20% test pairs, and *FPE* is the mean squared error (MSE) on the test set between the predicted feature vector and the ground truth feature vector. For the human query methods, we report *FPE* with increasing number of representation training queries N.

For ***TPA***, we test whether good representations necessarily lead to good learning of general preferences. We use the trained embeddings as the base for 20 randomly selected test preference rewards. For each R_{θ_i}, we generate a set of labeled preference queries $\mathcal{D}^{\theta_i}_{pref} = \{\xi_A, \xi_B, l\}$, which we split into 80% for training and 20% for test. We train each reward model with M preference queries per test reward, and we vary M. All preference networks have the same architecture: we take the embedding ϕ pre-trained with the respective method, and add new fully connected layers to learn a reward function from trajectory preference labels. For GridRobot we used MLPs with 2 layers of 128 units, and for JacoRobot we used 1024 units. We found that all methods apart from SIRL worked better with unfrozen embeddings (App. A.3). We report TPA as the preference accuracy for the learned reward models on the test preference set, averaged across the test human preferences.

Hypotheses: H1. Using similarity queries specifically designed to teach the representation (SIRL) leads to better learned representations than unsupervised (VAE), implicit (MultiPref, SinglePref), or random representations. **H2.** The SIRL representations result in more generalizable imitation learning.

4.2 Experimental Setup for Policy Learning

We then investigate the quality of SIRL-trained representations for behavioral cloning using simulated human demonstrations and similarity queries.

4.2.1 Environments

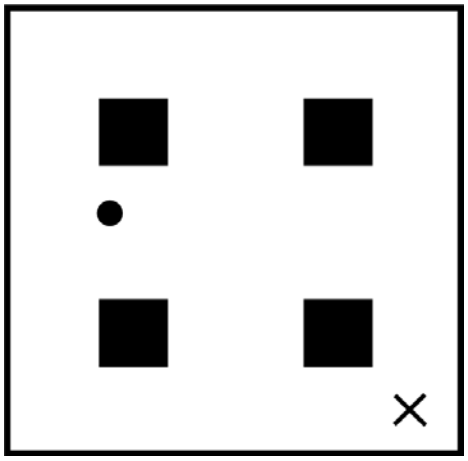


Figure 4.3: Gridworld.

Gridworld (Fig. 4.3) is a larger 39-by-39 gridworld with four 5-by-5 obstacles placed equidistant from the center. The agent is placed at a random position, denoted with a black circle, within the gridworld, and a goal is placed with a minimum distance of 15 units away, denoted with a black x. The 4-dimensional input consists of the x and y coordinates of the current position of the agent and the coordinates of the goal. The discrete action space comprises of the four cardinal directions in which the agent could move next. The simulated human answers queries based on the direction of movement that would navigate the agent around the obstacles towards the goal state with minimal distance, selecting the two trajectories with the least Euclidean distance between vectors of optimal movement. We accomplish this using the A* search algorithm [20] and average the vectors from the start location to the next three states in the path to determine the optimal direction of movement. For determining the optimal action for imitation learning, the agent utilizes the A* search algorithm to generate an optimal path and selects the cardinal direction corresponding to the next state within the identified path.

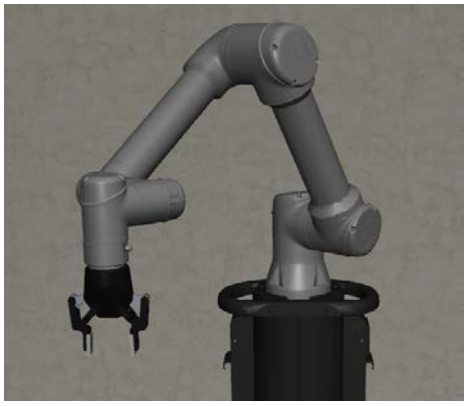


Figure 4.4: Robosuite.

Robosuite (Fig. 4.4) is a simulated robosuite [43] environment featuring a 6-DoF UR5e arm on a tabletop, along with a laptop placed centrally on the tabletop. The robot arm has an initial configuration that situates the end effector to the left of the laptop. The objective for the robot is to reach a target position to the right of the laptop. The 9-dimensional state space encompasses the xyz coordinates of the end effector, the target xyz position, and the end effector's velocity vector. The robot arm moves using an operational space controller, with the 3-dimensional action space interpreted as the delta values from the current state. At each time step, the agent obtains a reward proportional to the negative distance to the goal position, and receives a penalty if it enters a specific proximity to the laptop, with the penalty magnitude increasing as a linear function of the distance to the laptop. For answering similarity queries, the simulated human featurizes the state based on 2 features: the 2-dimensional direction to the laptop and the 3-dimensional direction to the goal coordinates. For determining the optimal action for imitation learning, the human calculates the optimal direction using TrajOpt [34], a path planning algorithm that optimizes a path that accounts for the penalty of being too close to the laptop. In order to avoid degenerate trajectories, TrajOpt is initialized with

a spline that avoids the laptop.

In the Gridworld environment, we collect 50,000 states by randomly sampling start-goal pairs. To collect diverse data for learning a similarity-based representation in the Robosuite environment, we leverage random network distillation (RND) [9], a powerful approach for exploration in deep reinforcement learning that provides reward bonuses based on the error of a neural network predicting the output of a randomly-initialized network. We gather a dataset of state-action transitions observed while training a Soft Actor Critic [19] policy using the RND bonus as the sole reward signal. We collect 1,000 trajectories for a total of 50,000 states. We generate similarity queries by randomly sampling triplets of states from this set of collected states.

4.2.2 Experimental Setup

Manipulated Variables. In order to assess the performance of similarity-based representation learning in the context of imitation learning, we designed an experimental comparison with three baseline methods: a) **Random**, a randomly initialized embedding that operates similar to the random baseline for SIRL in trajectory input scenarios but with state input instead of trajectory input, b) **BC**, a neural network trained with imitation learning data, and c) **VAE**, a Variational Autoencoder which learns to generate latent representations from the state input. Each of the embedding networks are of the same size with 2 hidden layers of 512 units each, mapping to 6 output neurons. Since the magnitude of the input space is similar between robosuite and gridworld, we keep these hyperparameters the same across the two environments.

Dependent Measures. To compare the effectiveness of the representations, we use three metrics: *Average Rollout Reward (ARR)*, *Action Prediction Error (APE)*, *Feature Prediction Error (FPE)*. Behavioral cloning has poor generalizability and minor inaccuracies in predicting human actions can lead to the agent encountering unfamiliar states within the state space, causing trajectories to deviate from the intended target location. An effective representation should enable an imitation learning agent to establish a mapping from states to human actions and reach good performance on the task. As such, we utilize *Average Rollout Reward (ARR)* as a metric to assess the agent's ability to mimic the human effectively to reach the goal state. We generate rollouts in the environment with the simulated human to create a dataset of human demonstrations, $\mathcal{D}_{Demos} = \{\mathcal{S}, \mathcal{A}\}$, to train the behavioral cloning policy. At test time, we rollout the imitation learning policy 100 times in the environment and calculate the average reward to determine the model's ARR.

The **APE** metric instead tests whether representations can generalize across the entire state space instead of only the states seen from human demonstrations. The goal of APE is to measure the generalization of feature learning to unseen states and whether the learned embedding captures information to recover the optimal action. We gather a set of states $\mathcal{D}_{APE} = \{\mathcal{S}\}$ and split the dataset into 80% training and 20% test states. We freeze each embedding and add a linear regression layer to predict the optimal action for that state. For Gridworld, *APE* is the cross entropy loss between the predicted action and

ground truth action whereas for Robosuite, we use the mean squared error instead. To fairly compare the embeddings learned by BC and SIRL, we train both with an equivalent amount of human data: BC receives an equal amount of human imitation data of state and action pairs as SIRL receives in similarity queries.

We reuse the **FPE** metric from SIRL with reward learning to measure the effectiveness of the learned representations to capture the ground-truth features within each environment. We once again freeze each embedding and add a linear regression layer, generate a set of states $\mathcal{D}_{FPE} = \{\xi, \phi^*\}$, split into $80 - 20$ training and test set, and measure the MSE between the predicted feature vector and ground truth feature vector. To fairly compare the embeddings learned by BC and SIRL, similar to APE metric, BC receives an equal amount of human imitation data of state and action pairs as SIRL receives in similarity queries.

Hypothesis. H3. Using similarity queries for teaching the representation (SIRL) leads to better learned representations than unsupervised (VAE), implicit (BC), or random representations. **H4.** The SIRL representations result in more generalizable reward learning without compromising the loss of necessary information about the state space.

Chapter 5

Simulated Results

5.1 SIRL for Reward Learning

We first show in the qualitative results which trajectories SIRL learned to be similar and which it learned was least similar. In the quantitative results, we analyze the performance of SIRL on the metrics defined in chapter 4.

5.1.1 Qualitative Results

In Fig. 5.1 we show similar and dissimilar trajectories learned by SIRL in a simplified GridRobot environment with only the laptop and the joint angle. *Top*: the given trajectory stays far from the laptop and holds the cup on its side; SIRL learns that trajectories that share those features are similar, despite being dissimilar in the state-space. *Bottom*: the trajectory stays close to the laptop and holds the cup at an angle; SIRL learns that trajectories that hold the cup on its side and stay far from the laptop are dissimilar, despite being similar in the state-space (going through the top left corner).

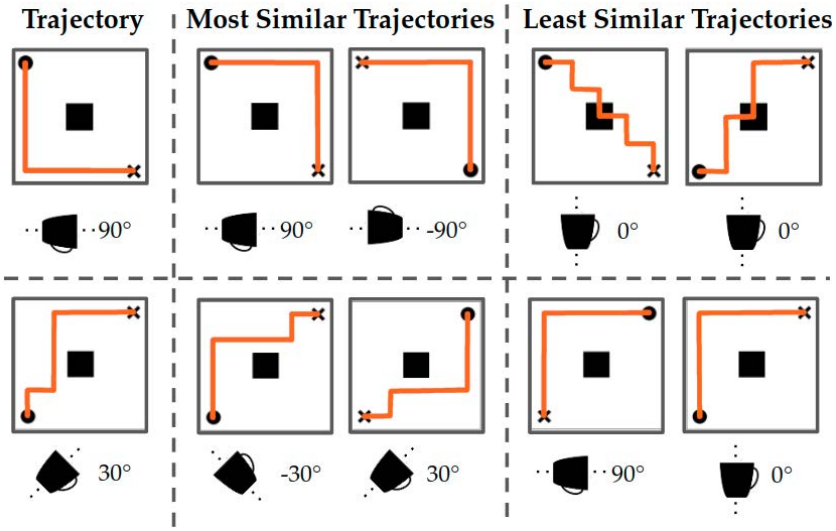


Figure 5.1: For a given trajectory, SIRL selects the two most and least similar trajectories.

5.1.2 Quantitative Results

In Fig. 5.7 we show the *FPE* score for both environments with varying representation queries N from 100 to 1000. For GridRobot, both versions of SIRL (with or without VAE pre-training) perform similarly and outperform all baselines. When pre-training with preference queries, MultiPref with 10 humans performs better than SinglePref or MultiPref with 50 humans: SinglePref may be overfitting to the one human preference it has seen, while when MultiPref has to split its data budget among 50 humans it ends up learning a worse representation than Random. There is a balance to be struck between the diversity in human preferences covered and the amount of data each preference gets, a problem which SIRL avoids by being preference-agnostic. For the more complex JacoRobot, both versions of SIRL outperform all baselines, although SIRL without VAE scores better than with it.

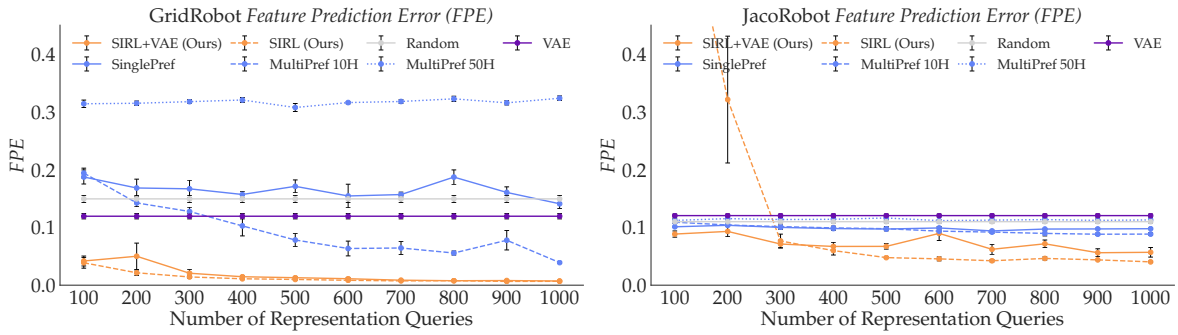


Figure 5.2: *FPE* for the GridRobot (left) and JacoRobot (right) environments with simulated human data. With enough data, SIRL learns representations more predictive of the true features ϕ^*.

In Fig. 5.3 we present the *TPA* score for both environments with a varying amount of test preference queries M from 10 to 190, and $N = 100$, 500, and 1000. For GridRobot, each respective method performs comparably with different Ns, suggesting that this is a simple enough environment that low amounts of representation data are sufficient. For JacoRobot, this is not the case: with just 100 queries, SIRL with VAE pre-training performs like VAE, SIRL without pre-training has random performance (since it's frozen), and the preference baselines all perform close to Random, as if they weren't trained with queries at all. For larger N, both versions of SIRL start performing better than the baselines, suggesting that with enough data a good representation can be learned.

Focusing on $N = 1000$, for GridRobot both SIRLs outperform all baselines, while for JacoRobot SIRL without VAE is the best, and SIRL with VAE performs only marginally better. Also note that while VAE performs comparably to other baselines in GridRobot, it severely underperforms in JacoRobot. This suggests that the reconstruction loss struggles to recover a helpful starting representation when the input space is more high-dimensional and correlated. As a result, using the VAE pre-training to warmstart SIRL hinders performance when compared to starting from a blank slate. Meanwhile, in the GridRobot environment, VAE pre-training helps. When comparing the preference-based methods, in GridRobot they all perform similarly apart from MultiPref with 50 humans, while

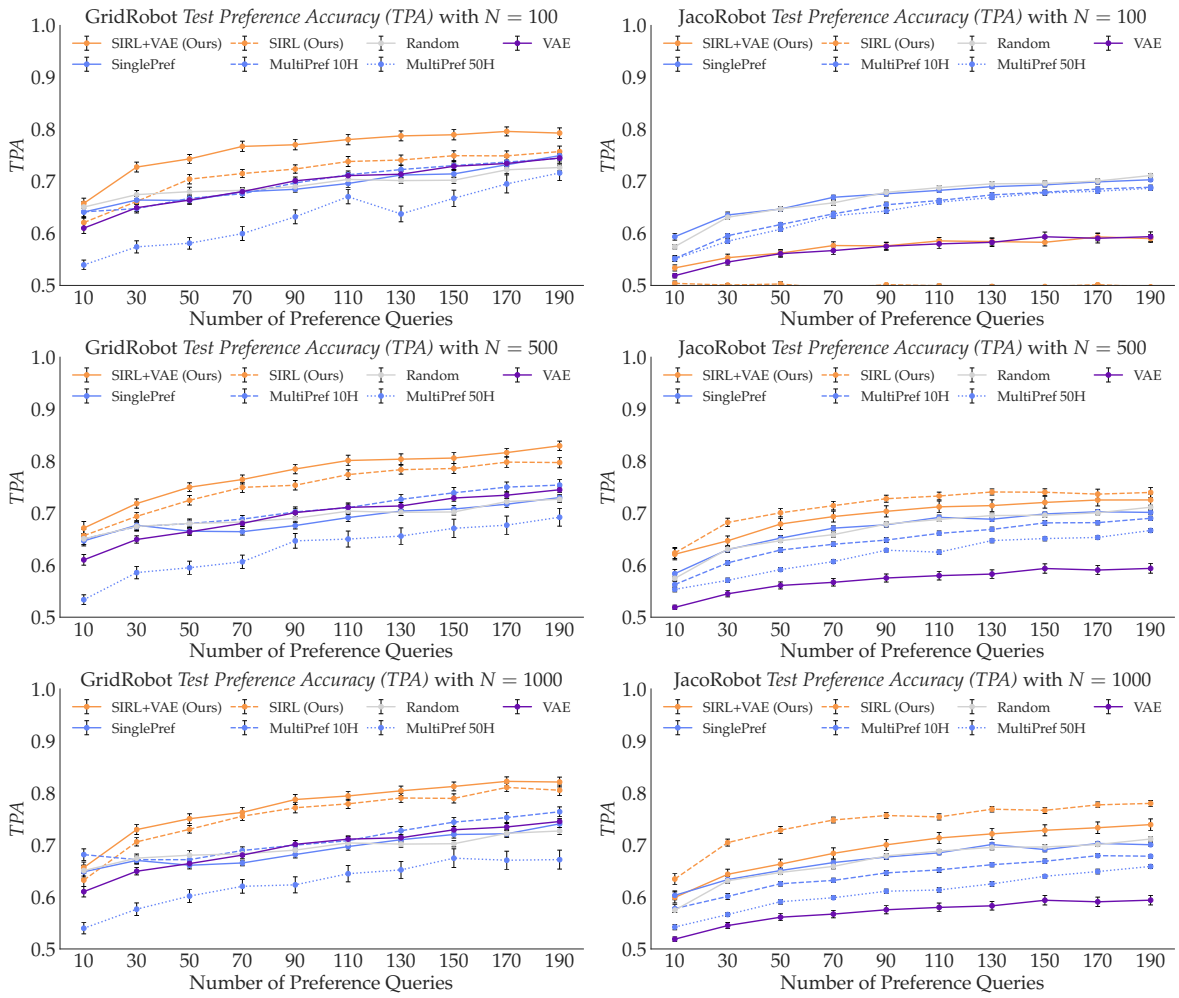


Figure 5.3: *TPA* for GridRobot (left) and JacoRobot (right) with simulated human data. With enough data, SIRL recovers more generalizable rewards than unsupervised, preference-trained, or random representations.

in JacoRobot we see a trend that more preference humans does not necessarily result in better performance. This confirms our observation from Fig. 5.7 that deciding on a number of preference humans that works across environments is challenging, a problem that SIRL bypasses.

Summary. With enough representation data SIRL can outperform baselines by at least 10%, learning more generalizable rewards (H1 and H2). When VAE pre-training is suitable, it can further reduce the human queries SIRL needs; however, when the reconstruction loss fails to recover sensible representations, it can hurt performance. Surprisingly, Random is often better than pre-training with preference queries: more correlated information can be more harmful than starting from scratch.

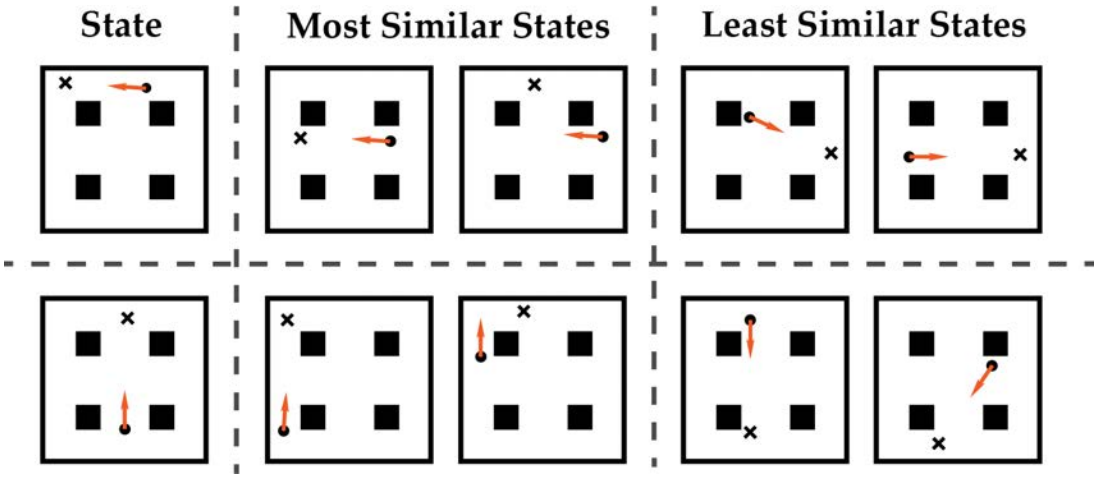


Figure 5.4: For a given state, SIRL selects the two most and least similar states based on the direction of optimal movement.

5.2 SIRL for Policy Learning

Similar to SIRL for reward learning, We first show in the qualitative results which states SIRL learned to be similar and dissimilar. In the quantitative results, we analyze the performance of SIRL on the metrics.

5.2.1 Qualitative Results

In Fig 5.4, we present an example of the most and least similar states for the Gridworld environment learned by SIRL. The orange arrow denotes the direction of optimal movement. *Top*: in this state, the optimal direction is to the left towards the goal state. SIRL learns that other states where the direction of movement to the left are similar. Notice for the second of the most similar states, while the goal is to the top middle, the optimal direction is to the left to avoid the obstacle. SIRL learns/ a featurization more complex than a simple featurization that only considers direction to the goal. *Bottom*: in this case, the optimal direction is towards the top of the gridworld. SIRL learns that other starting locations and goal locations are similar to the state despite being visually different due to the same direction of optimal movement.

5.2.2 Quantitative Results

In Figure 5.5, we present the ARR score as a function of varying quantities of demonstrated state-action pairs, while maintaining a constant number of SIRL similarity queries at 1000. In the case of Gridworld, we observe that SIRL combined with BC surpasses the performance of BC alone when provided with a limited number of demonstrations. This implies that the SIRL representation possesses a degree of generalizability that benefits imitation learning when integrated with BC, particularly when faced with fewer human demonstrations. However, the advantage of the SIRL representation appears to diminish

as the number of demonstrations increases. When 800 demonstrated state-action pairs are provided, BC outperforms SIRL with BC. This could be attributed to the fixed nature of the SIRL representation, which may limit its adaptability when presented with an abundance of human demonstrations covering a significant portion of the state space. In the Robosuite environment, SIRL with BC outperforms the baselines, proving to be more consistent than BC and VAE in rollout rewards.

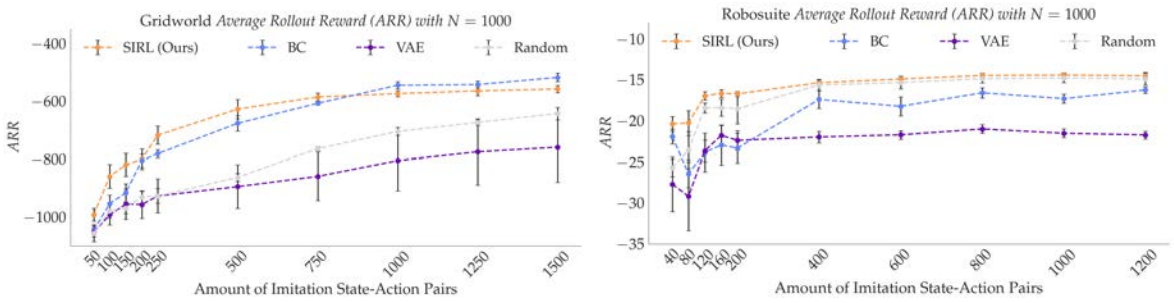


Figure 5.5: *ARR* for the Gridworld (left) and Robosuite (right) environments

In Fig. 5.6, we show the scores of SIRL and the baselines on the APE metric. In both environments, for a fair comparison, SIRL receives an equivalent number of similarity queries as BC does for imitation learning data. For Gridworld, VAE and random featurizations perform much better than BC and SIRL with SIRL performing the worst. This is likely due to there being multiple optimal actions in the environment since if the goal state was diagonally top-right of the current state, going up and going to the right are both optimal. In the Robosuite environment where such an issue doesn't exist, SIRL has the lowest action prediction error amongst the baselines.

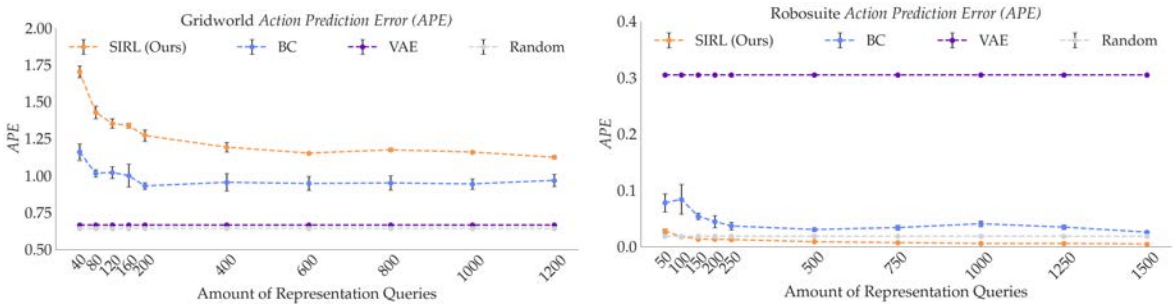


Figure 5.6: *APE* for the Gridworld (left) and Robosuite (right) environments

In Fig. 5.7, we evaluate SIRL and the baselines on the FPE metric. In both environments, for a fair comparison, SIRL receives an equivalent number of similarity queries as BC does for imitation learning data. In Gridworld, with enough similarity queries, SIRL outperforms the baselines and reaches a much longer FPE score. In Robosuite, SIRL is able to learn a representation that matches better with the ground truth features than the baselines even with a low number of similarity queries.

Summary. With enough representation data SIRL can outperform baselines in the FPE metric, proving to learn representations that are more aligned with the human's

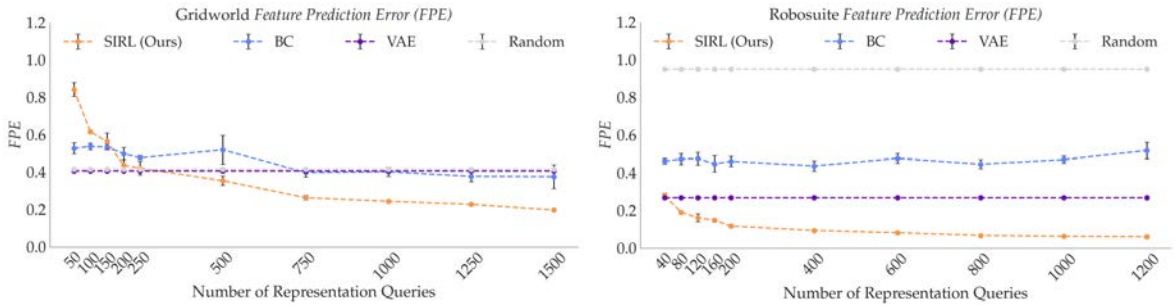


Figure 5.7: *FPE* for the Gridworld (left) and Robosuite (right) environments

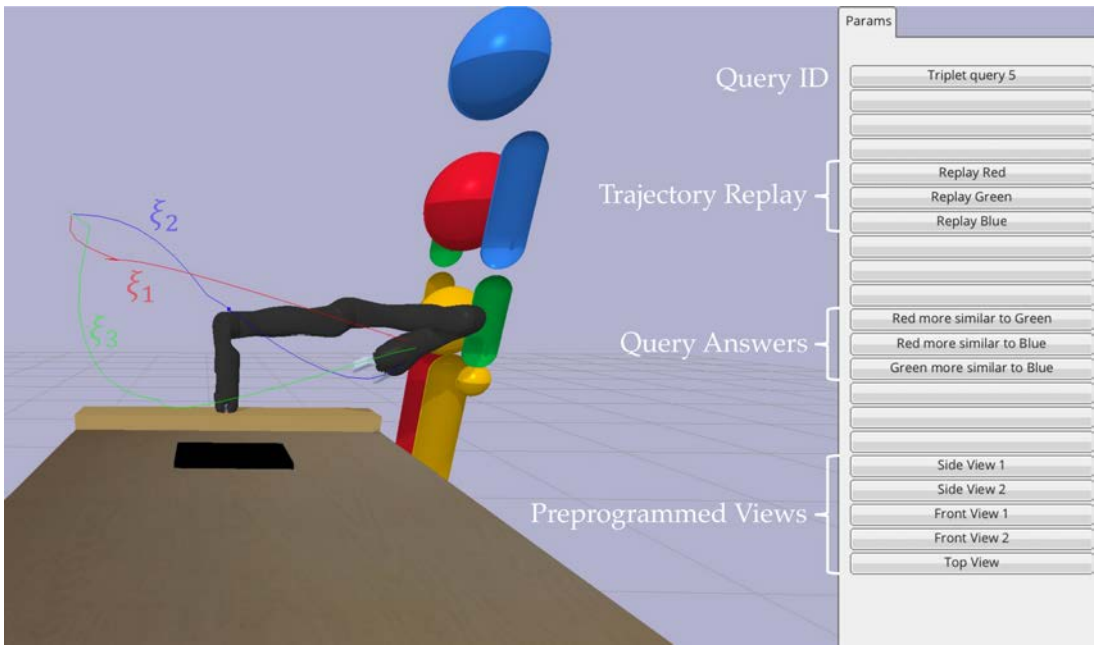
featurization of states than the baselines (H3 and H4). Furthermore, when learning a policy using the learned representation, SIRL performs better in imitating the human and is more generalizable in Robosuite for states that are out of distribution of the imitation learning data (H4).

Chapter 6

User Study

6.1 Experiment Design

We ran a user study in the JacoRobot environment, modified for only two features: *table* and *laptop* (we removed the humanoid in the environment). We designed an interface where people can click and drag to change the view, and press buttons to replay trajectories and record their query answer (Fig ??). We chose to display the Euclidean path of each trajectory in the query traces, as we found that to help users more easily compare trajectories to one another.



The study is split into two phases: collecting similarity queries and collecting preference queries. In the first phase, we introduce the user to the interface and we describe the two features of interest. Because similarity queries are preference-agnostic, we describe examples of possible preferences akin to the ones in Sec. **??**, but we do not bias the participant towards any specific preference yet. We have each participant practice

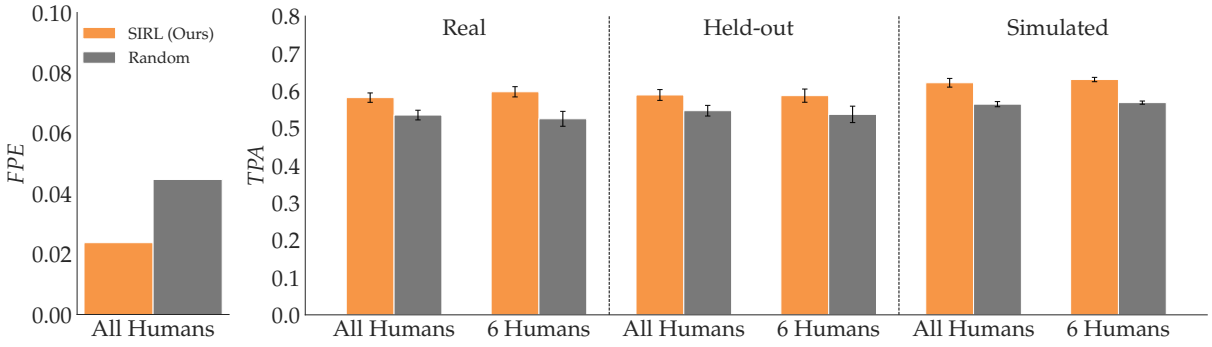


Figure 6.1: Study values for *FPE*, and *TPA* with real and simulated preferences. Even with novice similarity queries, SIRL outperforms the baseline.

answering a set of pre-selected, unrecorded similarity queries, and then ask them to answer 100 recorded similarity queries. In the second phase, we describe a scenario in the environment that has a specific preference associated with it (e.g. "There's smoke in the kitchen, so the robot should stay high from the table" or "There is smoke in the kitchen and the robot's mug is empty, so you want to stay far from the table and close to the laptop.") and assign different preference scenarios to each participant. Each participant practices giving unrecorded preference queries, then answers 100 preference queries.

Participants. We recruited 10 users (3 female, 6 male, 1 non-binary, aged 20-28) from the campus community to provide queries. Most users had technical background, so we caution that our results will speak to SIRL's usability with this population rather than the general population.

Manipulated Variables. Guided by the results in Fig. 5.3, we compare our best performing method, SIRL without VAE, to Random, the best performing realistic baseline. For SIRL we collect 100 similarity queries from each participant and train a shared representation using all of their data.

Dependent Measures. We present the same two metrics from Sec. **??**, *FPE* and *TPA*. For *TPA*, we collect 100 preference queries for each user's unique preference, we use 70% for training individual reward networks which we evaluate on the remaining 30% queries (Real). We compute *TPA* with cross-validation on 50 splits. To demonstrate how well SIRL works for new people who don't contribute to learning the similarity embedding, we also train SIRL on the similarity queries of 9 of the users and compute *TPA* on the held-out user's preference data (Held-out), for each user, respectively. Lastly, because real data tends to be noisy, we also compute *TPA* with 70 simulated preference queries for 10 different rewards, which we also evaluate on a simulated test set (Simulated).

Hypotheses: H5. Using similarity queries (SIRL) recovers more salient features than a random representation, even with novice user data. **H6**. The SIRL representation results in more generalizable reward learning, even with novice similarity queries.

6.2 Analysis

Fig. 6.1 summarizes the results. On the left, SIRL recovers a representation twice as predictive of the true features, supporting H3. A 2-sided t-test ($p < .0001$) confirms this.

This suggests SIRL can recover aspects of people's feature representation even with noisy similarity queries from novice users. On the right (Real), SIRL recovers more generalizable rewards on average than Random, providing evidence for H4. Furthermore, using the SIRL representation on a novel user (Held-out) also performs better than Random, and the result appears almost indistinguishable from Real. This suggests that similarity queries can be effectively crowd-sourced and the resulting representation works well for novel user preferences. Lastly, training with simulated preference queries slightly improves performance for both methods, suggesting that noise in the human preference data can be substantial. Three ANOVAs with method as a factor find a significant main effect ($F(1, 18) = 6.0175$, $p = .0246$, $F(1, 18) = 4.7547$, $p = .0427$, and $F(1, 18) = 16.1068$, $p < .001$, respectively). For each of the 3 cases, we also separated the 6 humans that were assigned preferences pertaining to both features (e.g. "There is smoke in the kitchen and the robot's mug is empty, so stay far from the table and close to the laptop."). SIRL performance is slightly better than in Real, hinting that perhaps the learned representation entangled the two features.

Chapter 7

Conclusion

7.1 Discussion

In this paper, we introduced a new type of human input useful for learning feature representations more aligned with humans, which can be used efficiently for learning generalizable downstream reward functions and policies. The qualitative and quantitative results obtained from our experiments clearly demonstrate the robustness and generalizability of SIRL. However, we also identified several areas for future research and improvement. We are particularly interested in exploring the applicability of SIRL to reinforcement learning tasks or tasks with high-dimensional image state spaces. These more complex environments present unique challenges and opportunities for refining our approach and further pushing the boundaries of what SIRL can achieve. High-dimensional image state spaces, for instance, require significantly more data to train, which may require more efficient training techniques or the incorporation of image-based unsupervised learning methods. Reinforcement learning, on the other hand, need to be adaptable during the training process, potentially requiring the integration of online learning or continual learning strategies for tuning the learned representation.

7.2 Limitations

Our method only analyzed environments with a maximum of two features and it becomes increasingly difficult to answer similarity queries while trading off many features. In such cases, the task of learning representations aligned with human understanding becomes more intricate, as it necessitates capturing intricate relationships between various features while maintaining their interpretability.

While the user study results do show a significant effect, the effect size is much lower than in simulation. This is attributable in part to the interface difficulty of analyzing the robot trajectories, which means more work on the best interfaces that enable users to accurately answer similarity queries is needed. Moreover, some users reported struggling to trade off the different features that are important, which means that similarity queries might not be entirely preference-agnostic. Nonetheless, our results underscore that there

are gains by explicitly aligning robot and human representations, rather than hoping it will happen as a byproduct of learning rewards from standard queries.

Bibliography

[1] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie, "Generalized non-metric multidimensional scaling," in *Artificial Intelligence and Statistics*, PMLR, 2007, pp. 11–18.

[2] E. Amid, A. Gionis, and A. Ukkonen, "A kernel-learning approach to semi-supervised clustering with relative distance comparisons," vol. 9284, Sep. 2015.

[3] M. Babes, V. N. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *ICML*, 2011.

[4] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning robot objectives from physical human interaction," in *Proceedings of the 1st Annual Conference on Robot Learning*, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., ser. Proceedings of Machine Learning Research, vol. 78, PMLR, 2017, pp. 217–226.

[5] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," Jan. 2016, pp. 119.1–119.11.

[6] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Inducing structure in reward learning by learning features," *The International Journal of Robotics Research*, vol. 0, no. 0, p. 02 783 649 221 078 031, 0. eprint: https://doi.org/10.1177/02783649221078031.

[7] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Feature expansive reward learning: Rethinking human input," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '21, Boulder, CO, USA: Association for Computing Machinery, 2021, 216–224.

[8] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[9] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *Seventh International Conference on Learning Representations*, 2019, pp. 1–17.

[10] J. Choi and K.-E. Kim, "Nonparametric bayesian inverse reinforcement learning for multiple reward functions," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[12] A. Coates and A. Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*, 2012.

[13] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, `http://pybullet.org`, 2016–2019.

[14] C. Demiralp, M. Bernstein, and J. Heer, "Learning perceptual kernels for visualization design," vol. 20, Nov. 2014.

[15] C. Dimitrakakis and C. A. Rothkopf, "Bayesian multitask inverse reinforcement learning," in *European workshop on reinforcement learning*, Springer, 2011, pp. 273–284.

[16] A. D. Dragan, K. Muelling, J. A. Bagnell, and S. S. Srinivasa, "Movement primitives via optimization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2339–2346.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, 1126–1135.

[18] A. Gleave and O. Habryka, "Multi-task maximum entropy inverse reinforcement learning," *arXiv preprint arXiv:1805.08882*, 2018.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[21] C. Huang, W. Luo, and R. Liu, "Meta preference learning for fast user adaptation in human-supervisory multi-robot deployments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 5851–5856.

[22] H. J. Jeon, S. Milli, and A. Dragan, "Reward-rational (implicit) choice: A unifying formalism for reward learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4415–4426, 2020.

[23] M. G. Kendall, "Rank correlation methods.," 1948.

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[25] C.-I. Lai, "Contrastive predictive coding based feature for automatic speaker verification," *arXiv preprint arXiv:1904.01575*, 2019.

[26] Z. Mandi, P. Abbeel, and S. James, "On the effectiveness of fine-tuning versus meta-reinforcement learning," *arXiv preprint arXiv:2206.03271*, 2022.

[27] B. McFee, G. Lanckriet, and T. Jebara, "Learning multi-modal similarity.," *Journal of machine learning research*, vol. 12, no. 2, 2011.

[28] V. Mnih *et al.*, *Playing atari with deep reinforcement learning*, 2013.

[29] J. Mumm and B. Mutlu, "Human-robot proxemics: Physical and psychological distancing in human-robot interaction," in *Proceedings of the 6th international conference on Human-robot interaction*, 2011, pp. 331–338.

[30] K. Nishi and M. Shimosaka, "Fine-grained driving behavior prediction via context-aware multi-task inverse reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2281–2287.

[31] A. van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, 2018.

[32] S. Reddy, A. D. Dragan, and S. Levine, *Pragmatic image compression for human-in-the-loop decision-making*, 2021.

[33] C. J. Reed *et al.*, "Self-supervised pretraining improves self-supervised pretraining," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 1050–1060.

[34] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: science and systems*, Citeseer, vol. 9, 2013, pp. 1–10.

[35] S. K. Seyed Ghasemipour, S. S. Gu, and R. Zemel, "Smile: Scalable meta inverse reinforcement learning through context-conditional policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[36] A. Srinivas, M. Laskin, and P. Abbeel, *Curl: Contrastive unsupervised representations for reinforcement learning*, 2018.

[37] A. Sripathy, A. Bobu, Z. Li, K. Sreenath, D. S. Brown, and A. D. Dragan, *Teaching robots to span the space of functional expressive motion*, 2022.

[38] N. Stewart, G. D. Brown, and N. Chater, "Absolute identification by relative judgment.," *Psychological review*, vol. 112, no. 4, p. 881, 2005.

[39] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. T. Kalai, "Adaptively learning the crowd kernel," *arXiv preprint arXiv:1105.1033*, 2011.

[40] K. Xu, E. Ratner, A. Dragan, S. Levine, and C. Finn, "Learning a prior over intent via meta-inverse reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6952–6962.

[41] K. Xu, E. Ratner, A. Dragan, S. Levine, and C. Finn, "Learning a prior over intent via meta-inverse reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 6952–6962.

[42] L. Yu, T. Yu, C. Finn, and S. Ermon, "Meta-inverse reinforcement learning with probabilistic context variables," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[43] Y. Zhu *et al.*, *Robosuite: A modular simulation framework and benchmark for robot learning*, 2020.

Appendix A

Appendix

A.1 Trajectory generation

In GridRobot the state space is discretized, so the trajectory space Ξ can be enumerated; however, the JacoRobot state space is continuous, so we need to construct Ξ by sampling the infinite-dimensional trajectory space. We randomly sample 10,000 start-goal pairs and compute the shortest path in the robot's configuration space for each of them, ξ^{SG}. Each trajectory has a horizon length H and consists of n-dimensional states. We then apply random torque deformations u to each trajectory to obtain a deformed trajectory ξ_D^{SG}. In particular, we randomly select up to 3 states along the trajectory, and then deform each of the selected states with a different random torque u. To deform a trajectory in the direction of u we follow:

$$\xi_D^{SG} = \xi^{SG} + \mu A^{-1}\tilde{u} \ , \tag{A.1}$$

where $\mu > 0$ scales the magnitude of the deformation, $A \in \mathbb{R}^{n(H+1)\times n(H+1)}$ defines a norm on the Hilbert space of trajectories and dictates the deformation shape [16], and $\tilde{u} \in \mathbb{R}^{n(H+1)}$ is u at indices nt through $n(t+1)$ and 0 otherwise ($\tilde{u}$ is 0 outside of the chosen deformation state index). For each deformation, we randomly generated μ and the index of the state the deformation is applied to. For smooth deformations, we used a norm A based on acceleration, but other norm choices are possible as well (see Dragan *et al.* [16] for more details). We took inspiration for this deformation strategy from Bajcsy *et al.* [4].

A.2 Training details

We present architecture and optimization details that can assist in reproducing our training setup.

A.2.1 Feature networks

All embeddings have the same network size: for GridRobot we used MLPs with 2 hidden layers, 128 units each, mapping to 6 output neurons, while for JacoRobot we used 1024

units to handle the larger input space. For both environments, we used ReLU non-linearities after every linear layer.

We trained the VAE network with a standard variational reconstruction loss [24] also including a KL-divergence-based regularization term (to make the latent space regular). The regularization part of the loss had a weight of $\lambda = 0.01$. For both environments, we optimized the loss function using Adam for 2000 epochs with an exponentially decaying learning rate of 0.01 (decay rate 0.99999) and a batch-size of 32.

SinglePref and MultiPref with 10 and 50 humans are trained using the standard preference loss in Eq. (3.6). Christiano *et al.* [11] ensured that the rewards predicted by the preference network remain small by normalizing them on the fly. We instead add an l_2 regulatization term on the predicted reward to the preference loss, with a weight of 10 for GridRobot and 1 for JacoRobot. All three methods optimize this final loss in the same way: for GridRobot, we use Adam for 5000 epochs with a learning rate of 0.01 and batch-size 32, while for JacoRobot we found a lower learning rate of 0.001 to result in more stable training.

Lastly, for SIRL we had the option to first pre-train with the above VAE loss. Training with the similarity objective in Eq. (3.2) happens disjointly, after pre-training. For both GridRobot and JacoRobot, we optimized this loss function using Adam for 3000 epochs with an exponentially decaying learning rate of 0.004 (decay rate 0.99999) and batch-size 64.

We note that our current architectures assume fixed-length trajectories but one could adopt an LSTM-based architecture for trajectories of varying length [37].

A.2.2 Preference networks

For evaluating *TPA*, we used preference networks on top of the embeddings for the respective methods we evaluate. For GridRobot we used MLPs with 2 hidden layers of 128 units, and for JacoRobot we used 1024 units for larger capacity. For both environments, we used ReLU non-linearities after every linear layer. We added the same l_2 regularization to the loss in Eq. (3.6) as before, with weight 10 for GridRobot and 1 for JacoRobot. For GridRobot, we optimized our final loss function using Adam for 500 epochs with a learning rate of 0.001 and a batch-size of 64. For JacoRobot, we increased the number of epochs to 1000.

A.3 Ablations

Fig. 5.3 illustrates results with frozen SIRL, and unfrozen baselines without VAE pre-training, as these were the best configurations we found for each method. In this section, we show the complete ablation we performed to decide which methods benefit from frozen or unfrozen embeddings, or VAE pre-training. Fig. A.1 showcases the result of this ablation on both GridRobot and JacoRobot. Overall, we see that SIRL does better when the learned representation is frozen, while all the other methods do better when the representations is unfrozen. SinglePref and the MultiPref baselines perform better

without VAE pre-training, while SIRL sometimes benefits from pre-training in simple environments like GridRobot.

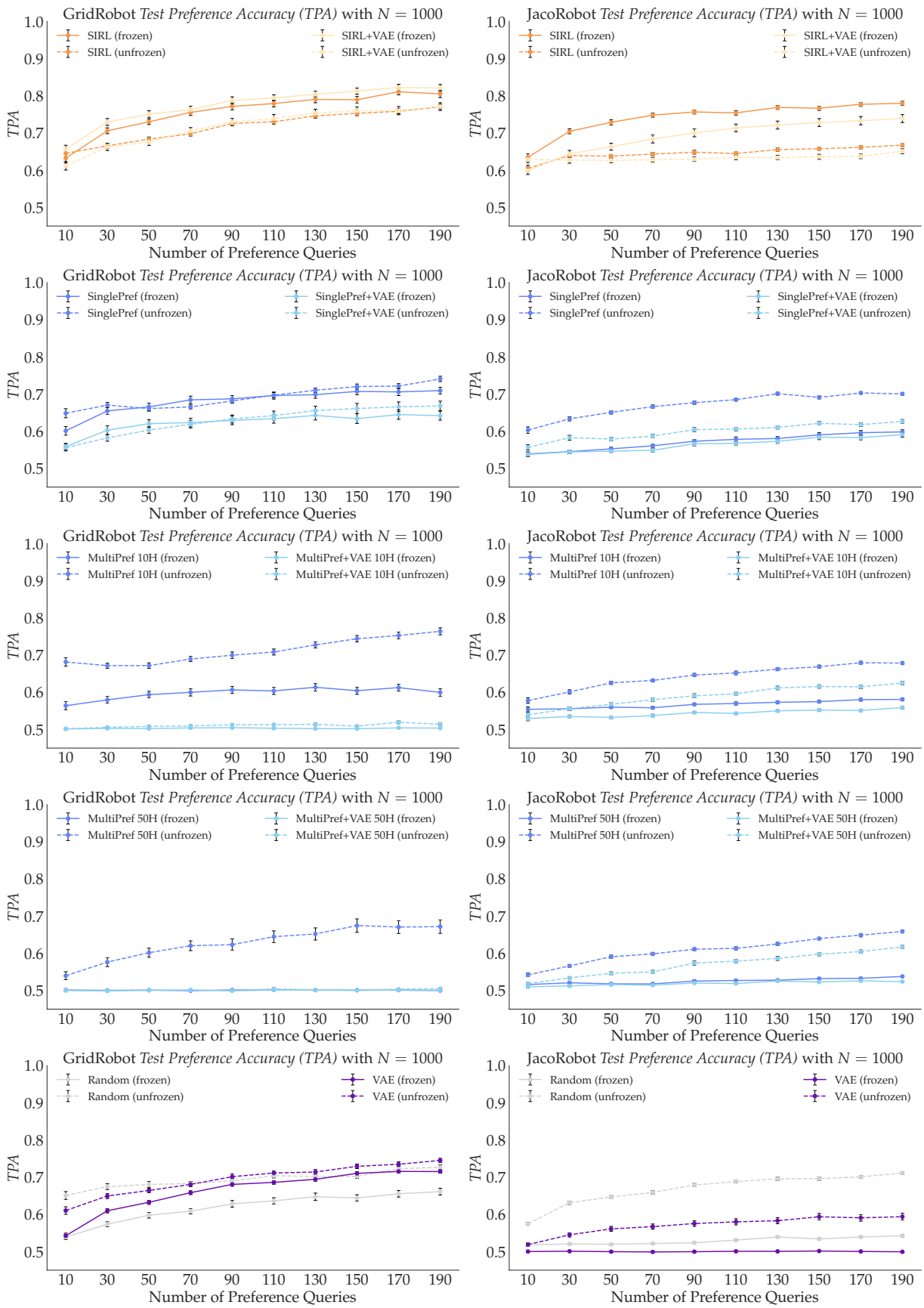


Figure A.1: Ablation Results for GridRobot (left) and JacoRobot (right). Overall, SIRL does better when the learned representation is frozen, while all the other method do better when the representations is unfrozen. SinglePref and the MultiPref baselines perform better without VAE pre-training, while SIRL sometimes benefits from pre-training in simple environments like