Ordinal Inverse Reinforcement Learning Applied to Robot Learning with Small Data

Adrià Colomé[1], and Carme Torras[1]

***Abstract*— Over the last decade, the ability to teach actions to robots in a user-friendly way has gained relevance, and a practical way of teaching robots a new task is to use Inverse Reinforcement Learning (IRL). In IRL, an expert teacher shows the robot a desired behaviour and an agent builds a model of the reward. The agent can also infer a policy that performs in an optimal way within the limitations of the knowledge provided to it. However, most IRL approaches assume an (almost) optimal performance of the teaching agent, which might become unpractical if the teacher is not actually an expert. In addition, most IRL focus on discrete state-action spaces that limit their applicability to certain real-world problems such as within the context of direct Policy Search (PS) reinforcement learning. Therefore, in this paper we introduce Ordinal Inverse Reinforcement Learning (OrdIRL) for continuous state variables, in which the teacher can qualitatively evaluate robot performance by selecting one among the predefined performance levels (e.g. $\{bad, medium, good\}$ for three tiers of performance). Once the OrdIRL has fit an ordinal distribution to the data, we propose to use Bayesian Optimization (BO) to either gain knowledge on the inferred model (exploration) or find a policy or action that maximizes the expected reward given the prior knowledge on the reward (exploitation). In the case of large-dimensional state-action spaces, we use Dimensionality Reduction (DR) techniques and perform the BO in the latent space. Experimental results on simulation and with a robot arm show how this approach allows for learning the reward function with small data.**

I. Introduction

Nowadays the scientific community is aiming at bringing robots to human daily environments. This rises several challenges. Among them, robots should be able to be taught to perform certain tasks, without the need of an expert. Ideally, a lay person would indicate the robot a task to perform and provide some demonstrations for the robot to have an initial understanding of the task, from where to improve with Reinforcement Learning (RL). This kind of approach would require the robot to use proper functional models to encode the task at hand, and also a way to evaluate performance after an execution in order to improve its behaviour through experience and repetition. In controlled environments, a roboticist can define such performance function (reward) to evaluate the outcome of each execution. But in a real scenario, the robot must learn such reward function from the demonstrator. This learning of the reward function is known as Inverse Reinforcement Learning (IRL).

This work was developed in the context of the project CLOTHILDE ("CLOTH manIpulation Learning from DEmonstrations"), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Advanced Grant agreement No 741930).

[1]Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, Spain [acolome, torras]@iri.upc.edu

In general IRL [1], a task is taught to the robot by an expert demonstator, and the demonstrations are assumed to be optimal (or close to optimal). Most applications of IRL consider a discrete action-space state in which, given a state (and possibly a transition model), each action (or sequence of actions) taken might have a different reward. When state spaces are continuous spaces, such as the case of policy search reinforcement learning [2], the parameters representing the action taken become the policy, and the IRL problem becomes that of finding the mapping from the parameter space to the reward value of each sample. An optimal policy or a manifold of optimal behaviour are learned thanks to the demonstrations provided. While many works in literature deal with IRL in discrete spaces, research on IRL in continuous spaces is limited. In [3], movement primitives were used to encode the state space from which to find an optimal policy given expert demonstrations, [4] also uses continuous state-action spaces and optimizes the relative entropy between the empirical distribution with optimal demonstrations vs a learned one. In [5] Gaussian Processes (GPs) were used to fit the reward function and maximize the likelihood between the GP and data. Other recent IRL methods focus on mathematical guarantees of the continuous IRL problem in smaller dimensions [6]. However, all these works assume the availability of (almost) optimal demonstrated data.

Still, human demonstrations are not always perfect. Or they can also be suboptimal. In this sense, approaches like preference-based RL [7], [8] rank the demonstrations and fit a reward function that preserves the ordinality between the demonstrations. Nevertheless, in the case of a human teaching a number of demonstrations to a robot, it can be hard to precisely order them all by performance. In this paper, we propose to categorize the performance of demonstrations and/or posterior executions of a task by ordered tiers of performance in an ordinal probability distribution. Such qualitative feedback is the kind of evaluation any human can provide, without needing neither to compare to every other execution, nor to provide a numerical evaluation. Some works in the human-cognitive field [9] already point to the existence of an ordinal encoding of data in the human brain, as it minimizes the synapse connections necessary to store data. Moreover, such feedback is much richer than classifying executions in just two categories, success or failure. The idea behind this proposed methodology is that an instructor teaching a robot might fail in some tries, and him/her should not repeat those demonstrations that are not (almost) optimal when learning a reward function, but rather being able to tell the robot its performance was *bad* and take great

advantage from it when inferring the reward function. And not restraining to either *good* or *bad* performance, but any number of ordered tiers of performance from which a human demonstrator can distinguish on eye-sight.

Furthermore, in this paper we propose to combine such ordinal IRL (OrdIRL) learning method with an end-to-end framework to learn tasks from scratch. We assume a number of demonstrations, tagged with their qualitative tier of performance. We fit those demonstrations with a parametric model and then perform, if necessary, DR to project it to a much smaller dimensional space, from where we can build the reward function. After fitting the framework with initial data, we can make the robot perform new executions at those points which we expect to be the most informative for improving our knowledge of the reward function, or at those places where we expect task performance to be the highest, by using Bayesian optimization (BO). Additionally, with a BO approach, we are capable of finding the underlying reward function or high-performing samples in very few data collection executions.

In particular, in this paper we will fit a parametric linear model for robot trajectories as Movement Primitive (MP) and then Gaussian Process Latent Variable Models (GPLVM) to project the MP parameteres onto a submanifold with a much smaller dimension. Then, with IRL, we will fit the reward function using ordinal regression and a Gaussian Process functional model. Once we have built the model, the robot will perform new executions at parameter points given by the Upper Confidence Bound (UCB) method for BO, which will allow the robot to improve both the knowledge of the reward function and the robot's performance at the task.

The paper is organized as follows: Sec. II introduces the concepts and techniques that provide the necessary background to then propose our methodology in Sec. III. Two simulated and one real execution experiments are presented in IV. The paper closes with a discussion in Sec. V.

II. Preliminaries

A. Gaussian Process Latent Variable Models

Gaussian Processes (GPs) [10] are the infinite-dimensional generalization of multivariate Gaussian distributions. They are defined as infinite-dimension stochastic processes such that, for any finite set of input locations $\mathbf{x}_1, ..., \mathbf{x}_n$, the random variables $f(\mathbf{x}_1), ..., f(\mathbf{x}_n)$ have joint Gaussian distributions. A GP is completely defined by its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$, often referred to as kernel, that must be symmetric and positive semi-definite. Usually GPs are expressed as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

GPs can be used for regression models of the form $\mathbf{y} = f(\mathbf{x}) + \varepsilon$, with ε an i.i.d. Gaussian noise, as they provide closed formulae to predict new values of the response variable $\mathbf{y}^*$, given new input location values $\mathbf{x}^*$.

Based on such formulation, Gaussian Process Latent Variable Models (GPLVMs) [11] and its variations [12] emerged as feature extraction methods that can be used as multiple-output GP regression models. In this way, these models, under a DR perspective, associate and learn low-dimensional representations of higher-dimensional observed data, assuming that observed variables are determined by the latent ones. GPLVMs provide, as a result of an optimization, a mapping from the latent space to the observation space, with a set of latent variables representing the observed values. While GPLVMs are capable of very efficient encoding of data in a smaller dimensional space compared to other DR methods, they do not provide a direct mapping from the original space to the latent space (see Fig. 1).

B. Ordinal Distributions and Regression

In classifying data, probability distributions such as the Bernouilli distribution or the Categorical distribution [13] are often used in reinforcement learning, either from a classical RL perspective [14] or in Deep RL [15]. The categorical distribution has many applications for classifying in machine learning. However, categories or labels of data can also have inherent rankings, such as the grade levels often used in exams ($A > B > C > ...$). The same kind of rankings exists on qualitative evaluations human commonly do (*very good*, *bad*, etc.). In order to fit data labelled with this kind of ranked information, the ordinal distribution is used.

Gaussian Processes can be used to fit an Ordinal distribution. To do so, the likelihood of a GP for an ordinal distribution can be defined as in [16], representing the probability of each modelled category given the GP as:

$$p(Z_i = 0 \mid f_i) = \phi\left(\frac{a_0 - f_i}{\sigma}\right) \tag{1}$$

$$p(Z_i = k \mid f_i) = \phi\left(\frac{a_k - f_i}{\sigma}\right) - \phi\left(\frac{a_{k-1} - f_i}{\sigma}\right) \tag{2}$$

$$p(Z_i = C \mid f_i) = 1 - \phi\left(\frac{a_{C-1} - f_i}{\sigma}\right), \tag{3}$$

where $p(Z_i)$ is the random variable of the class label, f_i is the GP prior evaluated on the input of the i-th data point, and a_k are the boundaries of each category. ϕ is the cumulative density function of a Gaussian. This likelihood function, a generalization of the *probit* likelihood function, can be differentiable and used to fit a GP to ordinal data. However, given the complexity of the ordinal likelihood prior for a GP, a variational GP is used instead [17].

C. Bayesian Optimization

Bayesian Optimization [18] is a very common approach to find the extrema of black-box functions, i.e., finding optimal values of functions we do not have an analytical expression of, but which can be evaluated at given inputs. In general, BO techniques present two elements: a stochastic surrogate model of the function to optimize, and an acquisition function that uses the surrogate model to assess which point in the search space is likely to have the best reward or will provide the most information gain on the surrogate model.

In our case, we will use a BO method called Upper Confidence Bound and the obtained surrogate model will be our estimation of the reward function based on the ordinal

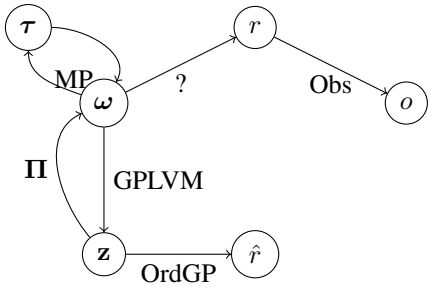

Fig. 1: *Schematic view of the variables and their interrelations. A trajectory $\boldsymbol{\tau}$ can be executed and its qualitative performance (ordinal variable o) can be observed. This observation is a projection from an unknown true reward r. The trajectory $\boldsymbol{\tau}$ can be converted to a weight vector $\boldsymbol{\omega}$. The weight vectors are used as input to find a latent space representation of data points, $\mathbf{z}$, from which the weight vectors $\boldsymbol{\omega} = \boldsymbol{\Pi}(\mathbf{z})$ can be inferred. Also, our IRL inference model is built to find an estimation of the reward, $\hat{r}(\mathbf{z})$. A prediction function can also estimate $\boldsymbol{\omega}$ from $\mathbf{z}$.*

data feedback. The acquisition function is defined as in [19], with the term κ weighting the tradeoff between the expected function value $\mu(\mathbf{x})$ and the uncertainty at such point, represented by its standard deviation $\sigma(\mathbf{x})$.

$$\text{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}). \tag{4}$$

The acquisition function in Eq.(4) therefore leverages the sum of the surrogate mean value, plus κ times its standard deviation. This function is then maximized in order to find the best point for which to next evaluate the function and obtain a new sample:

$$\mathbf{x}_{\text{sample}} = \text{argmax}_{\mathbf{x}\in\boldsymbol{\Omega}_X}\text{UCB}(\mathbf{x}) \tag{5}$$

III. Methodology

Let us assume that we have a set of demonstrated data in the shape of motion parameters $\boldsymbol{\omega}_k$, for each demonstration $k = \{1, ..., K\}$, with K being the number of provided samples with their qualitative ordinal feedback $o = \{o_1, ..., o_k\}$, that is a projection of the true unknown reward function:

$$\boldsymbol{\Omega} \xrightarrow{r} \mathbb{R} \xrightarrow{Obs} \mathbb{Z} \tag{6}$$

$$\boldsymbol{\omega}_k \overset{r}{\longmapsto} r_k \overset{Obs}{\longmapsto} o_k \tag{7}$$

where Obs is the observed qualitative reward.

In this work, our aim is three-fold: 1) find a stochastic approximation $\hat{r}$ of the true reward function r given the Data $= \{(\boldsymbol{\omega}_1, o_1), ..., (\boldsymbol{\omega}_K, o_K)\}$, 2) find samples the robot can execute autonomously with the supervision of the teacher so that they bring the best knowledge possible to $\hat{r}$ when qualitatively evaluated, and 3) within the same architecture, generate samples that will likely yield the best $\hat{r}$.

Given the data, we can fit a variational GP with an ordinal likelihood function as in Eq.(1). Then, the GP can be used to improve the model as explained in the following section.

A. Bayesian Optimization for improving the model

After an initial guess of the reward function has been defined, the system can ask for more samples. The idea behind this is that, either if we want the robot to improve its performance on the task or we want it to better learn the reward function, randomly-generated samples might not provide enough information to our system. Therefore, we use BO in order to find the points in the latent space that will provide the most information. Among the BO methods [20], we use UCB (see Sec. II-C). In the acquisition function of UCB, there is a parameter κ that sets the relative importance between the mean and the variance (or how imprecise is the prediction of the GP at that point). In this paper, we propose to tune this κ factor depending on whether we want the new samples to *explore* the IRL model (improve it) or *exploit* it (use the IRL model to improve the behaviour at the task).

In a purely-exploiting UCB method, the agent would take $\kappa = 0$ and find the point with the best expected outcome. However, this would only be useful in a one-shot sample, as few information would be provided to the model for future sampling. On the other hand, setting $\kappa = \infty$ would be equivalent to removing μ from Eq.(4), thus resulting in sampling on the maximum variance, which is the maximum Shannon entropy* sample for a Gaussian. In practice, we will use neither of these extreme values, as we want to allow for some exploration when exploiting and favour higher reward regions when exploring:

Exploiting the model means we will use the standard UCB method and set $\kappa = 2$, meaning we will add two standard deviations to the mean in order to find the next sampling point. This value of κ showed to be a good tradeoff in our previous works [21]. **Exploring the model** results in a new κ that modulates the relation between the effects of exploration (find best-performing sample) and exploitation (find sample that provides the most information to the IRL model). In order to find a middle-ground between $\kappa = 0$ and $\kappa = \infty$, we propose to create a function $\kappa(z, \text{Data})$ that favours regions with fewer samples. From the data, we evaluate the span S of the searching box in the latent space used for BO, that includes the difference of upper and lower bounds for each dimension. Then, we use a factor $\lambda \in (0, 0.5)$ indicating a width of the interval around a given point z, which will be $I_\lambda(z) = (z - \lambda \cdot S/2, z + \lambda \cdot S/2)$. Now, the expected amount of samples in this interval if the sample distribution was uniform, is $\mathbb{E}_{samples}[I_\lambda(z)] = \lambda N_s$, N_s being the total number of samples. Therefore, if $n(I_\lambda(z))$ is the amount of actual samples in such interval, we define:

$$\kappa = \min\left(2 \cdot \frac{\mathbb{E}_{samples}[I_\lambda(z)]}{n(I_\lambda(z))}, 10\right). \tag{8}$$

This function will provide a higher value of $\kappa(z)$ when the sample density is much smaller than expected, as well

*The Shannon entropy of a probability distribution $p(x)$ is $H(x) = -\int p(x)\log p(x)dx$ and, for a Gaussian distribution, is $H(x) = \frac{1}{2}(\log(2\pi\sigma^2) + 1)$. In the case of a multivariate normal distribution, it is also $H(x) = A + \frac{1}{2}\log(\det(\boldsymbol{\Sigma}))$, with A a constant. In both cases, the maximum entropy point is that with the highest variance/covariance.

Algorithm 1 Active sampling of the model

Input: Initial reward model $\hat{r}_0 \sim \mathcal{GP}(\text{Data})$
Number of new samples to generate K^{new}.
Latent Space projection $\mathbf{\Pi}$.
Output: Updated model $\hat{r} \sim \mathcal{GP}(\text{Data})$

```
Initialize r̂ = r̂_0
for s = 1 : K^new do
    κ = FindKfactor(r, Data)
    Find z_s = argmax_z UCB(r̂, z)
    Execute policy Π(z_s), obtain category feedback o_s
    Append(Data, (z_s, o_s))
    Update model with new data r̂ ~ GP(Data)
end for
```

as a smaller value when there are already many samples in that region. In the case when the samples in $I_\lambda(z)$ are the amount expected, $\kappa = 2$ which is the value for exploitation.

Also note that, for certain values of λ and samples close to the sampling bounds, $I_\lambda(z)$ will become the intersection between itself and the sampling bounds, and the values in Eq.(8) will change.

B. Higher Dimensional Problems

In order to improve the fitting of the reward model, we will use BO. In particular, UCB. However, BO may struggle when searching for optimal solutions in high-dimensional spaces. In the case of parametrized robot motion policies, it is often common to use parametrizations of a large dimensionality (larger than one hundred for several degrees of freedom). Therefore, given the impossibility to perform optimization in such large-dimensional spaces, we perform DR on the parameter space in order to have a more tractable dimension.

While linear DR techniques such as PCA can be useful, their linearity may often require a larger latent space dimension in order not to lose too much information on the latent space projection. Therefore, we used GPLVM (see Sec.II-A) to project the policy parameters to a much smaller dimensional space. Algorithm 1 shows the procedure to update an initial ordinal GP model of the reward, given that the policy parameters $\boldsymbol{\omega}$ are assumed to be dependent on a latent space variable $\mathbf{z}$, from which we can reconstruct $\boldsymbol{\omega}$.

IV. Experimental Results

In this section, we will perform three experiments. The first experiment is a simple, two-dimensional toy example. As such, we will not use DR techniques nor motion primitives. The second experiment will consist in processing a dataset of time-dependent synthetic data with a designed reward function, and in the third experiment we will test the method on a real robot folding a cloth garment on a table. For the second and third experiments, we will use a linear model:

$$\mathbf{y}_t = \mathbf{\Psi}_t^T \boldsymbol{\omega} + \boldsymbol{\epsilon}_y, \tag{9}$$

as a motion characterization, where $\mathbf{\Psi}_t^T$ is a set of equally-spaced in time Gaussian kernels, $\boldsymbol{\epsilon}_y$ is the error fitting the linear model, and $\boldsymbol{\omega}$ is a trajectory weight obtained by least squares, similarly as in a Probabilistic Movement Primitive [22], and GPLVM as a DR technique. This will result in a 3-layer motion representation. Each trajectory maps to a trajectory weight vector, that maps (through GPLVM) to a latent space. From such latent space, the estimated reward can be evaluated, and compared to the reward value obtained with the true function evaluated on the reprojection of the latent space to the MP parameters and the whole trajectory.

In all experiments, we will use a Variational GP model [10] with an ordinal likelihood as in Sec.II-B to fit the data to a model $\hat{r}$. We used a *Matern32* kernel and set the Sigma (noise) value to $\sigma = 0.25$, which is a fourth of the span of each category. We used Automatic Relevance Determination (ARD) for the lengthscales of the kernel, i.e.: a different lengthscale for each dimension, and initialized at one tenth of the span of each dimension. Moreover, we define the tiers of performance as integer values $(0, 1, 2, ...)$, and therefore set the class boundary parameters a_k in Eq. (1) to be the mid values on them $(0.5, 1.5, ...)$. The scale of the ordinal distribution tiers is not relevant for the problem, as the reward function for a reinforcement learning problem is usually a relative measure between samples.

A. Low-dimensional toy task

As a first test to evaluate the OrdIRL method without DR, we used a simple one-dimensional problem with a continuous action variable. The problem consists in setting the pitch parameter of a projectile motion, given a fixed initial velocity, in order to land at a certain point considering a constant initial speed. Each sample x has an associated reward depending on the distance of the landing point to a target, of which we extract the observed ordinal feedback o as

$$o(\boldsymbol{\tau}) = \left\lfloor \frac{r(\boldsymbol{\tau}) - r_{\text{min}}}{r_{\text{max}} - r_{\text{min}}} \cdot C \right\rfloor, \tag{10}$$

where r_{min} and r_{max} were predefined bounds on the error values.

This problem is of interest because it has two optimal values: for a certain angle α, the landing point is the same as for $\beta = \frac{\pi}{2} - \alpha$. In Fig. 2 we observe a set of 20 initial samples and the ordinal GP trained with them (upper left) with $C = 3$ categories. However, no samples of the top category were provided. The value function used to generate samples can be seen in the upper right plot, where, after a common sample at 0.36 the $\kappa = 2$ exploitation policy focuses on that region of the sampling space as its expected reward is already higher than the UCB where the other optimal value is. Meanwhile, the variable κ method distributes the samples more evenly and is capable of finding both optimal regions in two samples. Note that another issue encountered when fitting GPs with an ordinal likelihood function is that, as the changes from one class to another are rather abrupt, the variance in the bounds separating one class from another grows, forcing the UCB method to focus sampling on the edges of the classes. This causes a premature convergence on the exploitation policy that concentrates the samples on the same point, but it is mitigated by the variable κ policy.

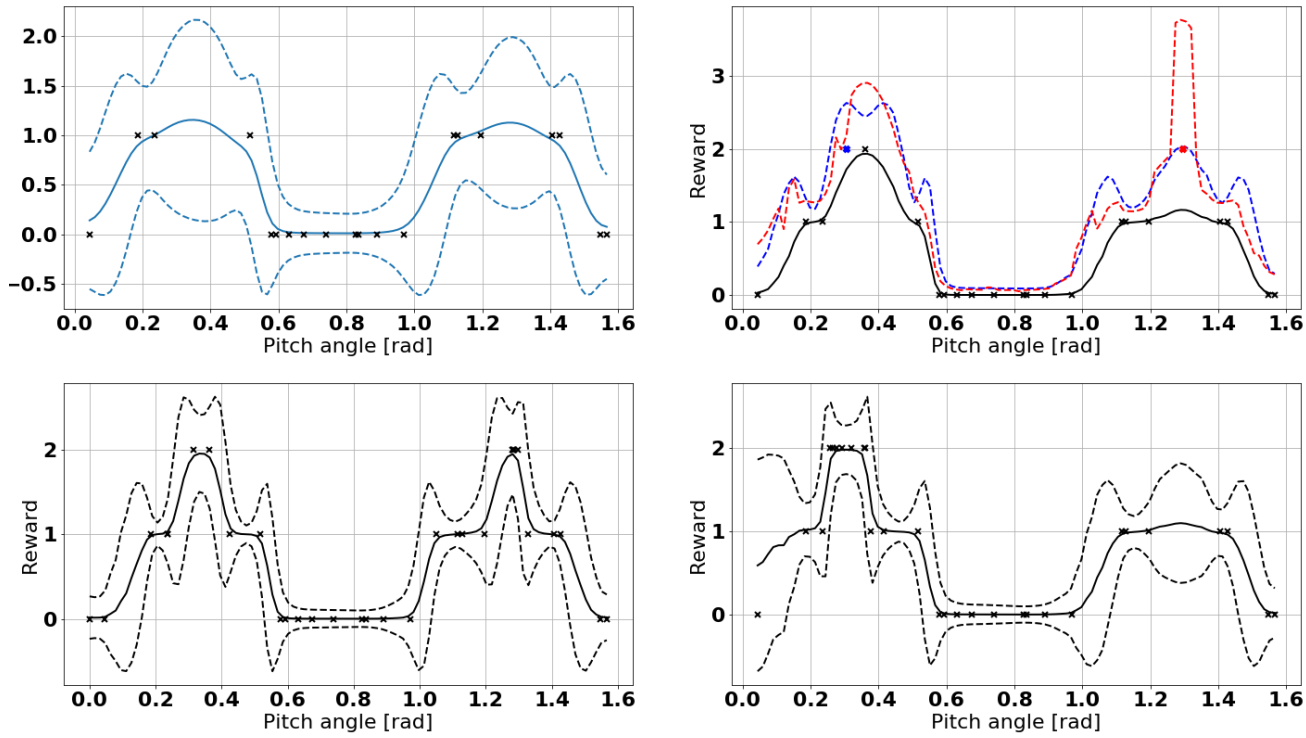

Fig. 2: *Samples used for testing the method in one dimension with $N = 20$ initial samples. Upper left, we see the initial model and its samples (black markers), on the upper right plot, we see a later iteration of the method with UCB for exploitation ($\kappa = 2$, in blue) and exploration (with variable κ in red), where we can distinguish how the red sample favours unexplored areas, while the blue sample focuses on places with more expected reward, despite having other samples close. The lower plots show the final reward model by using the exploration κ (left) and exploitation κ (right), both after 10 added samples.*

B. High-dimensional illustrative task

We used a set of 3-D trajectories synthetically generated that present a high variability in some areas and very low variability in others, as seen in Fig 3. As *ground truth* reward, we used the sum of the squared distances to two via-points (0.65 at $t = 30\%$ and -0.5 at the end for all DoFs). We generated 15 random trajectories as initialization. For each trajectory $\boldsymbol{\tau}_k$, we evaluated the ground truth reward r and obtained its observed category as in Eq. (10). Then trajectories were mapped into weight vectors $\boldsymbol{\omega}_k$ as in Eq. (9). Once having these weight vectors, we used a standard GPLVM (with a *Matern32* kernel) to map these onto a variable $\mathbf{z}$ in the latent manifold. Starting with this initial model, we performed both exploration and exploitation following our method. For exploitation, we used $\kappa = 2$ and measured the average value of the new samples generated, whereas for measuring the performance of exploration, we generated 10000 samples $\mathbf{z}_s$ in the latent space and evaluated their reward in two fashions: First, the IRL mapping from the latent space to the estimated reward, and second, projecting z_s upwards (see Fig. 1) and evaluating the known true reward. We use their difference as the Mean Squared Error (MSE) for a given model.

The results in Table I show that the greedier κ policy generates better samples on average, as one could expect. Also, if we take a look at the IRL model MSE in Fig. 4, we

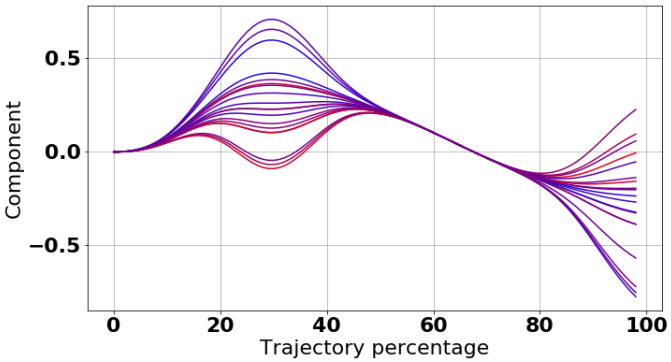

Fig. 3: *Sample trajectories (for one of the DoFs) used for testing the method.*

can see that the eploration policy with variable κ is capable of reducing the error on the reward model faster.

C. Robot execution

In order to test the method in a real scenario, we set up a proof of concept experiment in which a robotic arm (a Barrett's WAM robot) would partially fold a hand towel on a table (see Fig. 5). The aim of the experiment is then to have the robot learn the reward function while it finds a good policy. In order to assess how good the folding was, we

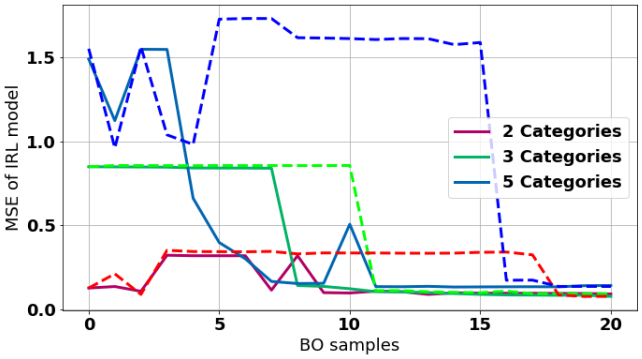

Fig. 4: *MSE of the fitting of the reward. The continuous lines represent the variable κ policy, while dashed lines represent exploitation policy ($\kappa = 2$)*

Categories	2	3	5
Explore	-1.079	-1.208	-0.832
Exploit	-0.931	-1.081	-0.664

TABLE I: *Mean reward of the samples used*

provided a qualitative feedback of each execution to the robot, with three tiers of performance: The top tier (2) corresponds to the fold of the cloth garment being in a delimited region marked with a number 2, as in Fig.5. The intermediate level of performance (1) would represent the fold being in a region adjacent to the previous one, in either sides, marked with a 1. The lower tier of performance would correspond to the robot placing the cloth fold too far from the preferred area.

We kinesthetically taught the robot 20x 6-dimensional cartesian trajectories of around $5s$ duration. We recorded the data and aligned trajectories in time. Then we used a linear model as in Eq. (9). We used 12 kernels per degree of freedom, adding up to a 72-dimensional parametric policy. The policy was then projected to a 3-dimensional latent space with GPLVM (see Sec. II-A). We then fit an OrdGP model to the unknown reward function using a *Matern32* kernel with ARD and, as in the previous experiments, we fixed $\sigma = 0.25$. Then, using UCB we calculated batches of 5 different new samples to execute, evaluated their level of performance, and added to the model. In order to evaluate the experiment, given that the true reward function was unknown, we used an exploitation $\kappa = 2$ and measured the mean observed reward on the executions of each batch. In Table II, we see that the demonstrations had an average of $\hat{r} = 0.85$ (out of the maximum of 2), and this value increased in each batch of samples until, at the fourth iteration, the 5 samples generated performed the task successfully.

Iter. 0	Iter. 1	Iter. 2	Iter. 3	Iter. 4
0.85 ± 0.73	1.20 ± 0.40	1.40 ± 0.80	1.60 ± 0.49	2

TABLE II: *Mean reward of the samples at each iteration* $\pm$ *their standard deviation. Note that Iteration* 0 *consists of* 20 *training samples, while the rest of the iterations consist of* 5 *new samples obtained through UCB.*

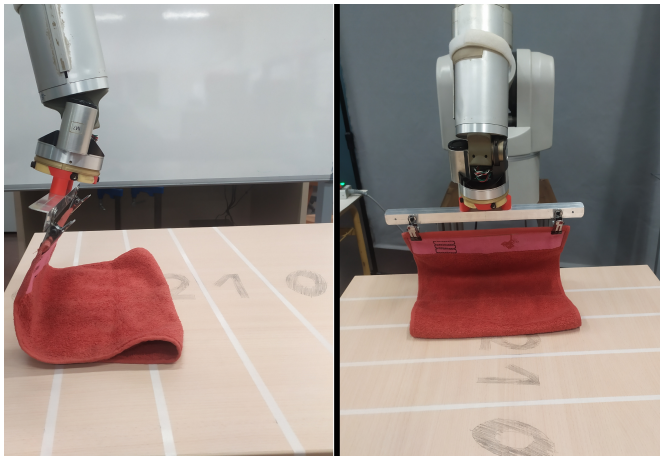

Fig. 5: *Setup for the robotic experiment.*

D. Robustness wrt. noise

In order to assess the robustness of the method, we will use the same kind of data as in experiment IV-A and check the model fitting error for a different amount of samples provided, as well as by adding noise to the samples. Moreover, we will compare the results of the OrdIRL with the Trajectory-ranked Reward EXtrapolation (T-REX) method in [8]. In order to implement T-REX in a small data framework, we used Gaussian processes instead of neural networks and the loss function defined in [8], adapted for single-shot evaluations:

$$\mathcal{L}(\boldsymbol{\omega}) = -\sum_{o_i < o_j} \log \frac{\exp(r_j(x))}{\exp(r_j(x)) + \exp(r_i(x))} + \lambda \boldsymbol{\omega}^T \boldsymbol{\omega}, \tag{11}$$

where r_i is the GP-predicted mean reward, given its parameters $\boldsymbol{\omega}$ at input point x. The term $\lambda \boldsymbol{\omega}^T \boldsymbol{\omega}$ is a regularization term for which we used $\lambda = 10^{-3}$. We then optimized the GP's parameters $\boldsymbol{\omega}$ using the same kernel function as for OrdIRL, but minimizing the loss defined in (11). Note that those samples $\{i, j\}$ so that $\{o_i = o_j\}$ are not considered in the loss function. In Fig. 6, we see an example of how the two methods fit a certain dataset consisting of 20 samples. In it, we see how the OrdIRL presents a smoother profile in both the mean and variance. The T-REX adapted to GPs with small data preserves the preferences in-between the parameters but overfits a bit on the samples, generating undesired oscillations on the mean and a high variance in-between samples.

To test the robustness of the method, we added noise to the measurements. When evaluating the true reward of each sample and converting it to a category, we added noise with different variance to the true reward before converting it to an ordinal category. In Fig. 7 we see the fitting results for a different set of input samples by adding noise with a variance of 0.25 to the reward, also yielding a better fitting with the OrdIRL method. Table III shows the Root Mean Squared Error (RMSE) of an uniform sampling on the input span and 95% confidence interval (with 5 randomized samplings of the input samples) in fitting the true reward function with both methods, depending on the variance of the added noise. Note that the RMSE of OrdIRL is overall smaller.

noise var:	0	0.25	0.5
OrdIRL	0.226 ± 0.091	0.336 ± 0.113	0.359 ± 0.069
T-REX	0.302 ± 0.108	0.409 ± 0.100	0.450 ± 0.085

TABLE III: *Mean fitting error wrt. noise variance*

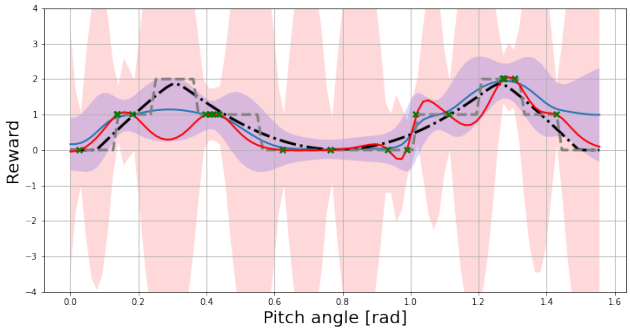

Fig. 6: *Comparison of the fitting of ordinal data with OrdIRL (blue) and T-REX (red). The ground truth reward is shown in black, with its associated categories in gray.*

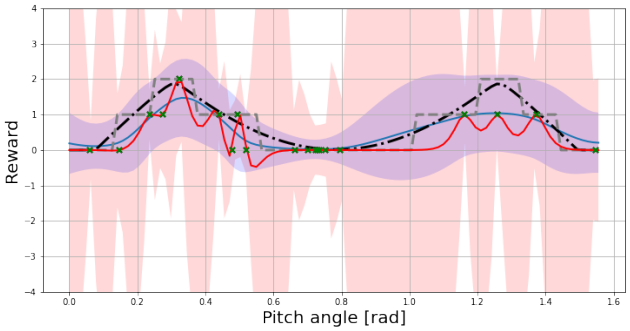

Fig. 7: *Comparison of the fitting of ordinal data with OrdIRL (blue) and T-REX (red) with added* 0.25 *noise.*

V. Conclusions

In this paper, we have presented an IRL method that can find the underlying reward function of a task, given an ordinal assessment of the quality of the demonstrated motions. The method is capable of finding such IRL function in a continuous state-action space, and with a limited amount of samples. We tested our method in three experimental scenarios of increasing complexity, and the method was able to effectively build the reward function and/or find a good policy in them. We also compared the proposed method to a preference-based one and ours showed to work better in a small data scenario. This kind of methods are user-friendly and can open the door for a wider population being able to teach new tasks to robots, thanks to the use of an ordinal distribution that matches the qualitative performance tiers in demonstrations/executions. Moreover, the method takes benefit of wrong demonstrations without the need of discarding data, thus being sample-efficient, and thanks to DR techniques, we can still perform efficient optimization in large-dimensional spaces. However, using DR for projecting the data onto a much smaller dimensional space also limits the exploration capabilities, as the exploration in the latent space is still within a sub-manifold of the policy parameters ($\boldsymbol{\omega}$ in this paper). A further analysis of this limitation is to be done in a future work by, for example, perturbing $\boldsymbol{\Pi}(\mathbf{z})$ in the higher-dimensional space, and then recalculating the GPLVM. Other further developments on the topic include exploring feature-dependencies in the policies.

References

[1] P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, 2021.

[2] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and trends in Robotics*, vol. 297, pp. 1–142, 2013.

[3] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 1561–1566.

[4] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 182–189.

[5] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Int. Conf. on Machine Learning (ICML)*, 2012, p. 475–482.

[6] G. Dexter, K. Bello, and J. Honorio, "Inverse reinforcement learning in the continuous setting with formal guarantees," *ArXiv*, vol. abs/2102.07937, 2021.

[7] L. El Asri, M. Geist, R. Laroche, and O. Pietquin, "Score-based inverse reinforcement learning," in *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2016, p. 457–465.

[8] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *Int. Conf. on Machine Learning*, 2019, pp. 783–792.

[9] A. Pitti and et al, "In search of a neural model for serial order: A brain theory for memory development and higher level cognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 279–291, 2022.

[10] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," in *Summer School on Machine Learning*. The MIT Press, 2006.

[11] N. Lawrence and A. Hyvärinen, "Probabilistic non-linear principal component analysis with gaussian process latent variable models." *Journal of Machine Learning Research*, vol. 6, no. 11, 2005.

[12] P. Li and S. Chen, "A review on gaussian process latent variable models," in *CAAI Transactions on Intelligence Technology,*, vol. 1, 2016, pp. 366–376.

[13] Y. M. Bishop, S. E. Fienberg, and P. W. Holland, *Discrete Multivariate Analysis Theory and Practice*. Springer Science Business, 1975.

[14] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Int. Conf. on Machine Learning (ICML)*, vol. 70, 2017, p. 449–458.

[15] G. Barth-Maron and et al, "Distributional policy gradients," in *Int. Conf. on Learning Representations*, 2018.

[16] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *Machine Learning Research*, vol. 6, no. 5, pp. 1019–1041, 2005.

[17] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian processs classification," in *Artificial Intelligence and Statistics*, 2015, pp. 351–360.

[18] N. Koganti, T. Tamei, K. Ikeda, and T. Shibata, "Bayesian nonparametric learning of cloth models for real-time state estimation," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 916–931, 2017.

[19] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *Journal of Machine Learning Research*, vol. 13, no. 57, pp. 1809–1837, 2012.

[20] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[21] J. A. Delgado-Guerrero, A. Colomé, and C. Torras, "Contextual policy search for micro-data robot motion learning through covariate gaussian process latent variable models," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 5511–5517.

[22] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Int. Conf. on Neural Information Processing Systems*, 2013, p. 2616–2624.