Master in Intelligent Interactive Systems
Universitat Pompeu Fabra

Inverse reinforcement learning with linearly-solvable MDPs for multiple reward functions

Ahana Deb

Supervisor: Anders Jonsson, Vicenç Gómez, Mario Ceresa

July 2023

Contents

Acknowledgement

I would like to express my gratitude to my supervisors Prof. Anders Jonsson, Prof. Vicenç Gómez and Prof Mario Ceresa for providing me with constant guidance and invaluable feedback throughout the entire duration of this course. I am sincerely grateful to the authors whose research has inspired and supported this thesis. I would also like to thank my family, friends, and partner for their constant encouragement and unwavering support.

Abstract

A subclass of Markov Decision Processes (MDPs), the Linearly solvable Markov Decision Processes (LMDPs), which have discrete state space and continuous control space, allow for a significant simplification of the inverse reinforcement learning problem by eliminating the need to solve the forward problem, and requiring only the unconstrained optimization of a convex and easily computable log-likelihood. This however, has only been explored for the single-reward single-agent scenario, where a single agent is assumed to be imposing optimal control under the influence of a single fixed reward function. In this work, we aim to utilise the advantages in problem formulation and ease of computation for LMDPs, for a multiple-agent, multiple-reward scenario, using non-parametric Bayesian inverse reinforcement learning.

Keywords: Linearly solvable Markov Decision Process; Inverse Reinforcement Learning; Multiple Rewards; Non-parametric Bayesian Learning

Chapter 1

Introduction

Inverse reinforcement learning (IRL) is the problem of estimating or modeling the reward or cost function an agent is acting under, given the behavioral data from that agent and also utilizing the observed behavior of an expert to select optimal actions. The last few decades have seen a surge of research in IRL, and apprenticeship learning or imitation learning through IRL[1][2]. IRL has gained sufficient traction also in robotics, since estimating the reward function from the observable optimal behavior of experts can allow appropriate modeling of the task and also form a transferable definition of the task. Abbeel et al. (2004)[1] explores simulated highway driving through apprenticeship learning, whereas Ziebar et al. (2008)[3] uses imitation learning to model route preferences and infer destinations through imitation learning, based on only partial trajectories.

The two main motivations of IRL:

- IRL may seem as the natural answer to modeling human and animal behavior in nature [4][5]. When we examine behaviors of these agents in the environment, it can be much easier to approach the problem of estimating the reward function, than to a priori be aware of what reward functions the agents are acting under and estimate optimal action. For theoretical biology and econometrics, IRL can allow for a more natural formulation of the problem.

- Additionally in certain complex environments, it can be intractable to infer the optimal set of actions. For example, in driving a car from a starting point to a destination, the problem can be formulated as a traditional reinforcement learning problem, with the objective of learning an optimal policy. This might pose the enormous challenge of modeling the reward function appropriately, in an environment as complex as real-world driving. Posing this problem as an IRL problem, i.e., observing the actions of an expert agent (human), and inferring the reward function from that behavioral data might be a much easier problem formulation.

We consider the problem of inverse reinforcement learning for Linearly Solvable Markov Decision Processes (LMDPs)[**?**] when more than one reward function can be present. The inverse problem in the case of standard MDPs[6][7] involves estimating the underlying reward function distribution, while the optimally chosen actions and corresponding states, and the underlying transition dynamics, i.e., the model of the MDP environment is provided. In the case of LMDPs, although the aim remains the same, obtaining the reward function (for a single reward scenario) is computationally much more simple since the problem reduces to an unconstrained optimization of a simple, easily computable function.

Todorov el al. (2010)[8] explores the estimation of reward functions from state transition pairs and uncontrolled dynamics of the LMDP, however, performs under the assumption of a single reward function and the agent acting optimally based on it. In real-life, data is often recorded from multiple agents working sub-optimally under multiple distinct reward functions. Further, solving for the inverse reinforcement problem for each distinct trajectory poses an additional problem since it is often difficult to obtain enough data from one trajectory to effectively estimate the underlying reward function.

Babes et al. (2011)[6] explores an approach of combining inverse reinforcement learning with EM clustering, where inferred reward functions are used to cluster the given behavioral data. Choi, J. Kim, (2012)[7] further explores the non-parametric

Bayesian approach for reinforcement learning for multiple rewards utilizing Dirichlet mixture models [9] for clustering, allowing a countably infinite number of mixture components, unlike EM clustering which requires a predefined number of clusters for the behavioral data. However, this method is confined to only standard MDPs, and in this work, we explore the formulation and advantages of using LMDPs for the non-parametric Bayesian inverse reinforcement learning case.

1.1 Objectives

The main objectives of this work are as follows :

- Extending the inverse RL formulation for LMDPs to the multiple reward setting,
- Developing and implementing practical algorithms for this extension,
- Comparing experimentally the proposed methods with the existing ones proposed in the literature.

1.2 Structure of the Report

In the next chapter, we will discuss the theoretical background of inverse reinforcement learning, Bayesian learning of IRL, Expectation Maximisation (EM) approach for clustering multiple rewards, non-parametric Bayesian inverse reinforcement learning, and finally the LMDP formulation. In the third chapter, we will then elaborate on the LMDP formulation for both the EM approach as well as the non-Parametric Bayesian approach. In the 4th chapter, we will present our experimental setup and results obtained along with relevant discussions. In our 5th and final chapter, we will discuss the conclusions and future works that can be followed.

Chapter 2

Inverse Reinforcement Learning Paradigm

The goal of IRL, is to estimate the reward function an agent is acting under, given the trajectories from the agent and also the transition probabilities of the environment. To introduce the formulation of IRL, we need to first introduce the standard Markov Decision Procces notations and formulations that will be used throughout this work.

2.1 Markov Decision Process Notations and Formulation

A discounted discrete space MDP can be defined as $< S, A, P^a_{s,s'}, R, \gamma >$ where

- S is a finite set of states,
- A is a finite set of actions,
- $P^a_{s,s'}$ is the probability associated with ending in state s' from state s' after performing action a. Since we are defining a Markov process, the probability of transition to state s' only depends on the current state s and the action a taken by the agent, and is independent of any other part of the trajectory.

- $R : S \to \mathbb{R}$ is the reward distribution in range $[0, R_{max}]$ (which can be trivially extended to $R : S \times A \to \mathbb{R}$ if needed).

- γ is the discount factor.

We can define policy as a mapping of the state space to the action space as $\pi : S \to A$, the value function as $V : S \to \mathbb{R}$, quantifying how "good" a particular state is, and the state-action value as $Q : S \times A \to \mathbb{R}$ giving how desirable a state-action pair is. Let the vector $\mathbf{V}^\pi$ be the values of states under policy π and $\mathbf{V}^*$ be the optimal set of values, and our goal is to infer policy π under which $\mathbf{V}^\pi = \mathbf{V}^*$. It can be proved that there exists a deterministic policy π under which $\mathbf{V}^\pi$ is maximized for all $s \in S$.[10] [11]

Therefore the Bellman equations for such an MDP can be given as,

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P_{s,s'}^{\pi(s)} V^\pi(s') \tag{2.1}$$

and

$$Q^\pi(s, a) = R(s) + \gamma \sum_{a'} P_{s,s'}^{a} V^\pi(s') \tag{2.2}$$

where the optimal policy π satisfies the condition

$$\pi(s) \in \arg\max_{a \in A} Q^\pi(s, a) \tag{2.3}$$

In the problem of IRL, the main objective is to estimate the reward function $R(s)$ given the behavioral data of the agent assumed to be acting optimally. However, in real-life situations, there is barely enough trajectory data from the agent to approach this problem analytically. For example, in an MDP, more than one reward function can explain a sparse set of trajectories, and we need to take into account the uncertainty in our inferences. Therefore we resort to Bayesian inference, where we derive the posterior distribution of the reward by using the prior distribution of reward functions and the likelihood of observing a particular trajectory/transition from the agent given that reward. This is further elaborated in Section 2.2.

2.2 Bayesian IRL

Typically in inverse reinforcement learning problems, the observed trajectory data from the expert is often too sparse to accurately infer the underlying reward function. Also, different trajectories can be often explained by more than one distinct reward function. For example, as shown in [12], different reward functions can be used to explain the same behavioral data obtained from an MDP shown in Fig1. Modeling three reward functions with high positive values at S_1, S_2, S_3 respectively can explain similar trajectories. Therefore the uncertainty in the estimated reward function also needs to be represented.

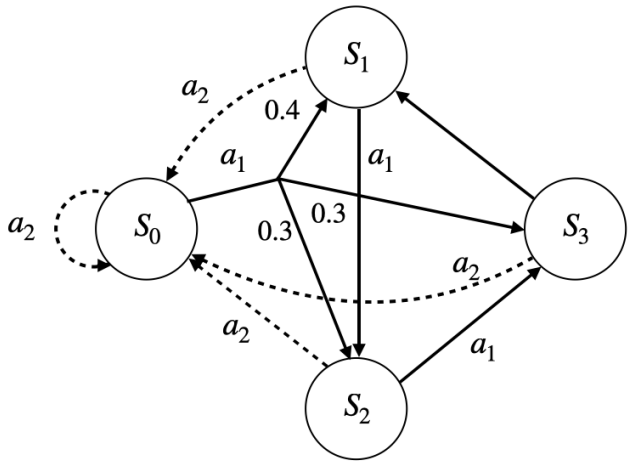


Figure 1: Figure from [12] shows an IRL example, where the action a_1 is the optimal action, and action a_1 from state S_0 has a probability of going to states S_1, S_2, S_3 with the probabilities $0.4, 0.3, 0.3$ respectively, and the other transitions are deterministic.

The main objective of Bayesian IRL is to estimate a posterior distribution of the reward function, given the prior distribution of the reward function and the probabilistic model of the observed agent, given the reward function. Using the MDP formulation defined in 2.1, we consider an MDP $< S, A, P^a_{s,s'}, R, \gamma >$, where an agent χ chooses actions A, and the reward R is drawn from a predefined distribution with prior P_R and the data obtained from agent χ is $X = \{(s_1, a_1), (s_2, a_2), (s_3, a_3) \dots (s_n, a_n)\}$.

Here we make two main assumptions about the agent χ:

- χ is executing a stationary or time-invariant policy.
- χ is not executing a ϵ-greedy policy, i.e., it is not trying to explore the environment, instead it already has a greedy policy and is trying to maximize its accumulated rewards.

Owing to our first assumption of a stationary policy, the following independence assumption can be made about a single trajectory of length n,

$$P(X|\mathbf{R}) = P((s_1, a_1)|\mathbf{R})P((s_2, a_2)|\mathbf{R}) \dots P((s_n, a_n)|\mathbf{R}) \tag{2.4}$$

Considering the state-action values $Q^*(s, a)$, the agent trying to maximize the accumulated reward is more likely to choose the state-action value with the highest value. This can be used to model the likelihood as,

$$P((s_i, a_i)|\mathbf{R}) = \frac{1}{Z_i} e^{\alpha_\chi Q^*(s_i, a_i, \mathbf{R})} \tag{2.5}$$

where $\alpha_\chi \in \mathbb{R}$ is a parameter used to denote the confidence in the agent χ's ability to choose the highest $Q^*(s_i, a_i)$, and Z_i is the appropriate normalizing constant. Therefore we can formulate the likelihood of the entire trajectory as

$$P(X|\mathbf{R}) = \frac{1}{Z} e^{\alpha_\chi E(X, \mathbf{R})} \tag{2.6}$$

where $E(X, \mathbf{R})$ denotes the expectation over the state-action values of the trajectory. Using Bayes theorem, we can now express the posterior probability distribution of the reward as

$$\begin{aligned} P(\mathbf{R}|X) &= \frac{P(X|\mathbf{R})P(\mathbf{R})}{P(X)} \\ &= \frac{1}{Z'} e^{\alpha_\chi E(X, \mathbf{R})} P(\mathbf{R}) \end{aligned} \tag{2.7}$$

where $Z' = Z \cdot P(X)$. While computing Z' is very difficult, we can utilize sampling

algorithms for inference.

2.3 Linearly Solvable Markov Decision Process

Linearly-Solvable MDPs (LMDPs) [13, 14] impose some restrictions on the dynamics and the reward function that make the Bellman optimality equations linear. This leads to a more efficient computation of the optimal policy [14] or the solution of the inverse problem [8].

Besides from leading to a linear Bellman equation, LMDPs offer other computational and theoretical benefits. For example, they are fundamental in the framework of RL as probabilistic inference [15, 16]. Thanks to the linearity in the Bellman equation, it is possible to compose a solution to a novel task from the solutions to previously solved task without learning [17, 18]. Such a property has been used in recent works combining compositionality and hierarchical reinforcement learning [19, 20, 21].

LMDPs have found applications across a wide array of domains, encompassing robotics [22, 23, 24], multi-agent systems [25, 26, 27], or for finding near optimal policies in other complex scenarios such as power grids [28], online forums [29], crowd-sourcing [30], or rankings [31].

In this chapter, we briefly review the theory of LMDPs following closely the derivations from [14] and [8].

2.3.1 State space

We define S as a set of finite states with $U(i)$ as a set of admissible controls for state $i \in S$, $l(i,u) \geq 0$ is the cost associated with being in state i and control "action" $u \in U(i)$. The stochastic matrix $P(u)$ with elements p_{ij} gives the probability of transitioning from state i to state j under control u. Initially considering a non-empty set of states absorbing states A in the state space, the unique solution to the

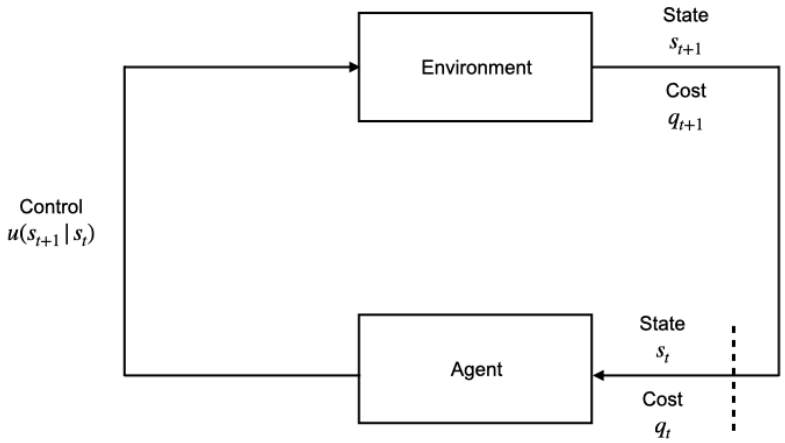


Figure 2: Schematic representation of a Linearly Solvable Markov Deciosion Process.

Bellman equation is given by

$$v(i) = \min_{u \in U(i)} \left\{ l(i,u) + \sum_j p_{ij}(u) v(j) \right\} \tag{2.8}$$

which is the analytical limit for traditional MDPs.

In our case of linearly-solvable MDPs, $\mathbf{u} \in \mathbb{R}^{|S|}$ is defined as a real-valued vector having cardinality equal to the state space, and the elements $u(j)$ have the effect of imposing control on the existing uncontrolled transition probabilities of the original MDP, allowing the controller to modify and rescale the underlying uncontrolled transition probabilities in any way. The controlled transition probabilities are defined as

$$p_{i,j}(\mathbf{u}) = \bar{p}_{i,j} \exp(u_j) \tag{2.9}$$

with $\bar{p}_{ij} = 0 \implies p_{i,j}(\mathbf{u}) = 0$ implying if the probability of a transitioning to a state under the uncontrolled condition is 0, it cannot be scaled to a non-zero value by the controller, and u_j has no effect, and also the row-sum of $P(\mathbf{u}) = 1$. The admissible controls under these conditions can be therefore given as,

$$U(i) = \left\{ \mathbf{u} \in \mathbb{R}^{|S|}; \sum_j \bar{p}_{i,j} \exp(u_j) = 1; \bar{p}_{ij} = 0 \implies u_j = 0 \right\} \tag{2.10}$$

The cost of imposing our control on the existing uncontrolled transition probabilities, which is equivalent to the price the controller is expected to pay before observing its defined transition, can be measured using KL-divergence between the two probability distributions, given as

$$q(i,u) = KL(\mathbf{p}_i(\mathbf{u})||\mathbf{p}_i(0)) = \sum_{j:\bar{p}_{i,j}\neq 0} p_{i,j}(\mathbf{u}) \log \frac{p_{i,j}(\mathbf{u})}{p_{i,j}(0)} \tag{2.11}$$

Since KL divergence ensures that $q(i,u) \geq 0$, substituting 2.9 in 2.11 we get,

$$q(i,u) = \sum_j p_{i,j}(\mathbf{u}) u_j \tag{2.12}$$

If a non-zero reward $r(i) \geq 0$ is associated with every state i, and $r(i) = 0$ for absorbing states A, the total cost associated with a state and control can be given as

$$l(i,u) = -r(i) + q(i,u) \tag{2.13}$$

Therefore the Bellman equation for the linearly solvable MDP can be given as,

$$v(i) = \min_{u \in U(i)} \left\{ -r(i) + \sum_j \bar{p}_{i,j} \exp(u_j)(u_j + v(j)) \right\} \tag{2.14}$$

2.3.2 Optimal control

To get the optimal control for this scenario, we can perform the constrained optimization in closed form of 2.14 using the Lagrange multiplier, subject to the constraint 2.10. The Lagrangian for each i is therefore defined as

$$\mathcal{L}(\mathbf{u}, \lambda_i) = \sum_j \bar{p}_{i,j} \exp(u_j)(u_j + v(j)) + \lambda_i \left(\sum_j \bar{p}_{i,j} \exp(u_j) - 1 \right) \tag{2.15}$$

Where $p_{ij} \neq 0$, the only solution of 2.15 is given as,

$$u_j^*(i) = -v(j) - \lambda_i - 1 \tag{2.16}$$

The second derivative of equation 2.15 can be shown to be greater than 0 since $\bar{p}_{i,j} \exp(u_j^*(i)) \geq 0$, confirming that 2.16 gives us the optimal control. Applying the constraint 2.10 to 2.16 to obtain the expression for λ_i, we can define the optimal control law as,

$$u_j^*(i) = -v(j) - \log \left(\sum_k \bar{p}_{ik} \exp(-v(k)) \right) \tag{2.17}$$

The optimal controller defined here is therefore a high-level controller, which tells the Markov chain to go to "good" states, which high $v(i)$ without defining how to get to those states. The optimally controlled transition probabilities imposed over the natural uncontrolled transition probabilities can be given as,

$$p_{ij}(\mathbf{u}^*(i)) = \frac{\bar{p}_{ij} \exp(-v(j))}{\sum_k \bar{p}_{ik} \exp(-v(k))} \tag{2.18}$$

Using the optimal control obtained in 2.17 in the Bellman equation 2.14 and taking *exponents* on both sides, we can derive

$$\exp(-v(i)) = \exp(r(i)) \sum_j \bar{p}_{ij} \exp(-v(j)) \tag{2.19}$$

Introducing the exponential transformation $z(i) = \exp(-v(i))$, we obtain the minimised linear Bellman equation

$$z(i) = \exp(r(i)) \sum_j \bar{p}_{ij} z(j) \tag{2.20}$$

which can be reduced to a linear eigenvalue problem as,

$$\mathbf{z} = G\bar{P}\mathbf{z} \tag{2.21}$$

where vector $\mathbf{z}$ has elements $z(i)$ and diagonal matrix G has values $r(i)$ along its main diagonal. In our case, for discounted-reward infinite-horizon problems, 2.21 is given as

$$\mathbf{z} = G\bar{P}\mathbf{z}^{\gamma} \tag{2.22}$$

where $\gamma < 1$ is the given discount factor. This is solved by iterating for $\mathbf{z}_k$ as

$$\mathbf{z}_{k+1} = G\bar{P}\mathbf{z}_k^{\gamma}, \qquad \mathbf{z}_0 = \mathbf{1} \tag{2.23}$$

which is equivalent to the power method, and can be shown to converge to the largest eigenvector. This formulation of traditional MDPs provides accurate approximations and is computed far more efficiently, for optimal control problems. We aim to utilize these properties of the LMDP for nonparametric Bayesian inverse reinforcement learning for the multiple rewards case.

2.3.3 OptV

Todorov et al.(2010)[8] introduces inverse optimal control for LMDPs, estimating the optimal control and thus the value function, and reward, since they are uniquely related to the control. Todorov further shows that the LMDP formulation allows for the problem to be reduced to an easily computable convex optimization problem.

Assuming we are given the uncontrolled dynamics p and a set of transitions $\{x_n, x'_n\}_{n=1,\dots,N}$, sampled from the optimally controlled dynamics, the optimal control can be given as $\frac{p(x'|x)z(x')}{G[z(\cdot)(x)]}$, where

$$G[z(\cdot)(x)] = \sum_{x'} p(x'|x)z(x'). \tag{2.24}$$

Therefore, the negative log-likelihood can be derived as

$$L[z(\cdot)] = -\sum_{n} \log z(x'_n) + \sum_{n} \log \sum_{x'} p(x'|x_n)z(x'). \tag{2.25}$$

Here, rewriting equation 2.25 in terms of v, the function $L[v(\cdot)]$ becomes convex, since it is a linear function containing log, summation, and exponents. An additional improvement can be made when the trajectory data is much higher than the number of states in the LMDP, by rewriting L as a function of visitation counts. The gradient and Hessian of equation 2.25 can be computed analytically, followed by Newton's method with backtracking line search.

Although the LMDP formulation here greatly simplifies the computation compared to standard MDPs, this algorithm still only allows the estimation of a single reward function. This approach can be used to estimate each reward per trajectory for the multiple rewards case followed by unsupervised clustering, but as mentioned before, there might not be enough recorded transitions from a single ground truth reward to allow for an efficient approximation.

2.4 Expectation Maximization MLIRL for Multiple Rewards

Babes et al.[6] proposes the estimation of multiple reward functions using an Expectation Maximization (EM) approach, and shows that a gradient ascent method modifying the reward function maximizing the likelihood of the behavioral data can be used to estimate multiple reward functions. They propose Maximum Likelihood Inverse Reinforcement Learning (MLIRL) similar to Bayesian IRL, a probabilistic model that aims to maximize likelihood.

Assuming the reward functions are parameterized by θ_A, the state-action values under reward θ_A are defined as $Q_{\theta_A}(s,a)$ for state and action s and a which can be given by the standard MDP formulation as

$$Q_{\theta_A}(s,a) = \theta_A^T \phi(s,a) + \gamma \sum_{s'} P^a_{s,s'} \otimes_{a'} Q_{\theta_A}(s',a') \tag{2.26}$$

where $\phi(s,a)$ is a feature vector for a state-action pair, and

$$\otimes_a Q(s,a) = \sum_a Q(s,a) e^{\beta Q(s,a)} / \sum_{a'} e^{\beta Q(s,a')} \tag{2.27}$$

which is the Boltzmann exploration [32] which makes the likelihood infinitely differentiable. The Boltzmann exploration policy can therefore be given by,

$$\pi_{\theta_A}(s,a) = \frac{e^{\beta Q_{\theta_A}(s,a)}}{\sum_{a'} e^{\beta Q_{\theta_A}(s,a')}}. \tag{2.28}$$

The likelihood of the given behavioral data can be expressed using this as

$$L(\mathcal{X}|\theta) = \log \prod_{i=1}^{N} \prod_{(s,a)\in\xi_i} \pi_\theta(s,a)^{w_i} = \sum_{i=1}^{N} \sum_{(s,a)\in\xi_i} w_i \log \pi_\theta(s,a), \tag{2.29}$$

where w_i gives the frequency of the trajectory i. Algorithm 1 from Babes et al.[6] shows how the maximum likelihood solution $\theta_A = \arg\max_\theta L(\mathcal{X}|\theta)$ can be estimated.

Algorithm 1 Maximum Likelihood IRL

1: **Input:** MDP, ϕ, trajectories $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$, trajectory weights $\{w_1, \ldots, w_n\}$, number of iterations M, step-size t.
2: **Initialize:** randomly reward weights θ_1.
3: **for** $t = 1$ **to** M **do**
4: Compute Q_{θ_t},π_{θ_t}.
5: $L = \sum_i w_i \sum_{(s,a)\in\xi} \log \pi_{\theta_t}(s,a)$
6: $\theta_{t+1} \leftarrow \theta_t + \alpha_t \nabla L$.
7: **Output:** return $\theta_A = \theta_M$.

The complete algorithm EM-MIRL proposed alternates between an expectation and a maximization step to estimate the reward functions. Defining z_{ij}^t as the probability of trajectory i belonging to cluster j and ρ_j^t as the prior probability of cluster j at iteration t, the expectation step of the EM algorithm can be given as,

$$z_{ij}^t = \prod_{(s,a)\in\xi_i} \frac{1}{Z} \pi_{\theta_j^t}(s,a) \rho_j^t, \tag{2.30}$$

where Z is the normalization factor. In the maximization step, the goal is to find reward weights which maximized the likelihood of the observed trajectory data, which is also the output of the MLIRL algorithm 1.

The complete algorithm for EM trajectory clustering can therefore be given as

Although this approach allows for the estimation of multiple reward functions, it still requires the prior knowledge of the number of appropriate clusters K.

Algorithm 2 EM Trajectory Clustering

1: **Input:** Trajectories $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ obtained from different ground truth rewards, number of clusters K, target number of iterations T.
2: **Initialize:** $\rho_1, \ldots, \rho_K$ $\theta_1, \ldots, \theta_K$ randomly.
3: **for** $t = 1$ **to** T **do**
4: **E step:** compute $z_{ij} = \prod_{(s,a)\in\xi_i} \frac{1}{Z}\pi_{\theta_j}(s,a)\rho_j$, where Z is the normalizing factor.
5: **M step:** For all l, $\rho_l = \sum_i z_{il}/N$. Compute θ_l via MLIRL given in algorithm 1 on $\mathcal{X}$ with weight z_{ij} on trajectory ξ_i.

2.5 Non-parametric Bayesian IRL for Multiple Rewards

In inverse reinforcement learning the main objective is to derive the underlying reward/cost function of an environment from the state-action pair and uncontrolled transitions or model of the environment. In Todorov et al. (2010)[8], and as mentioned in Section 2.3.3, the inverse optimal control for LMDPs has been reduced to an unconstrained convex optimization of an easily-computed function defined as **OptV**, which works on any dataset of consecutive pairs of state transitions $\{x_n, x'_n\}_{n=1\cdots N}$ sampled from the optimal control law π^* as

$$x_{n+1} \sim \pi^*(\cdot|x_n) \tag{2.31}$$

which assumes a single agent acting under optimally to maximize a single reward function. In practical scenarios, behavioral data is often collected from multiple agents maximizing potentially distinct underlying reward functions, and each trajectory may not contain enough data to infer the underlying reward function accurately. Choi, J. Kim (2012)[7], introduces a nonparametric Bayesian approach to address the IRL problem in the case of traditional MDPs with multiple reward functions, develops a Metropolis-Hastings sampler utilizing the gradient of the reward function posterior to infer reward functions from the given trajectory data, and extend the problem to estimating the reward associated with a new trajectory. In our study, we extend the problem to the case of LMDPs taking advantage of the

ease of computation.

An initial approach to Bayesian inverse reinforcement learning, as explored in Section 2.2 in traditional MDPs as explored in Ramachandran et al.(2007)[12] assumes the trajectory data is obtained from a single reward function, the prior encoded as the reward function preference, and the likelihood gives the measure of how likely the given trajectory is to be produced from the predicted reward function.

Assuming that the reward function entries are independently identically distributed (i.i.d.), the prior on the reward can be defined as $P(\mathbf{r}) = \prod_{d=1}^{D} P(r_d)$. Various distributions like uniform, Laplace, or beta distributions can be utilised for the reward prior.

The likelihood of observing a set of transitions under optimal control for the MDP, can be given as,

$$P(\mathcal{X}|\mathbf{r},\eta) = \prod_{m=1}^{M}\prod_{h=1}^{H} P(a_{m,h}|s_{m,h},\mathbf{r},\eta) = \prod_{m=1}^{M}\prod_{h=1}^{H} \frac{\exp(\eta Q^*(s_{m,h},a_{m,h};\mathbf{r}))}{\sum_{a'}\exp(\eta Q^*(s_{m,h},a;\mathbf{r}))} \tag{2.32}$$

for M trajectories with H transitions, where η is the confidence parameter of the selection action and $Q^*(\cdot,\cdot;\mathbf{r})$ denotes the optimal value of the Q function under reward $\mathbf{r}$. The posterior over the reward functions can be given by Bayes rule as

$$P(\mathbf{r}|\mathcal{X},\eta,\mu,\sigma) = P(\mathcal{X}|\mathbf{r},\eta)P(\mathbf{r}|\mu,\sigma) \tag{2.33}$$

which is still however limited to a single trajectory and single reward function.

In the multiple rewards scenario, we make one crucial assumption - each trajectory in the behavioral data is generated by an agent under only one reward function, i.e. within the transitions in a trajectory, the reward function remains unchanged. However, no assumptions are made about which agent generates which trajectory or about how many agents are present.

Unlike EM-MLIRL[6], Choi, J. Kim (2012)[7] combines the non-parametric Bayesian approach to inverse reinforcement learning with Dirichlet process mixture model[9],

allowing for clustering of trajectories under the estimated underlying reward functions, without having to specify the number of clusters in advance.

2.5.1 DPM-BIRL for Multiple Reward Functions

EMIRL as explained in Section 2.4 though extends this formulation to estimate multiple rewards, the number of clusters of rewards still needs to be predefined. Utilizing the Dirichlet Process Mixture (DPM) model with Bayesian Inverse Reinforcement Learning (BIRL), we can place a Dirichlet process prior on the reward function $\mathbf{r}_k$, and estimate multiple reward functions without having to predefine the number of clusters. We can further assume that the obtained behavioral data is being drawn from the following generative process -

1. The cluster assignment for a set of trajectories c_m is being drawn as -

   $$\mathbf{p}|\alpha \sim Dirichlet(\alpha/K, ..., \alpha/K)$$

   $$c_m|\mathbf{p} \sim Multinomial(p_1, ..., p_K)$$

2. The reward function $\mathbf{r}_k$ is being drawn from $P(\mathbf{r}) = \prod_{d=1}^{D} P(r_d)$.

3. The trajectory $\mathcal{X}$ is being drawn from $P(\mathcal{X}_m|\mathbf{r}_{c_m}, \eta)$ as given by equation 2.32.

The graphical model of DPM-BIRL is shown in 5.

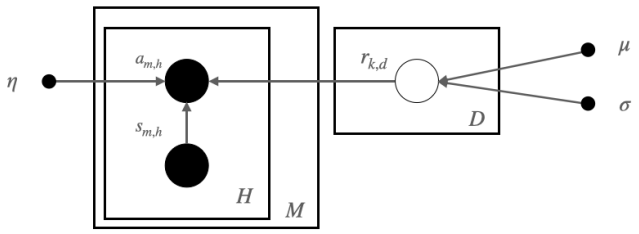


Figure 3: Graphical model of BIRL for MDP.

The joint posterior of the reward functions and the cluster assignment can be given as

$$P(\mathbf{c}, \mathbf{r_k}|\mathcal{X}, \eta, \mu, \sigma, \alpha) = P(\mathbf{c}|\alpha) \prod_{k=1}^{K} P(\mathbf{r}_k|\mathcal{X}_{c(k)}, \eta, \mu, \sigma) \tag{2.34}$$

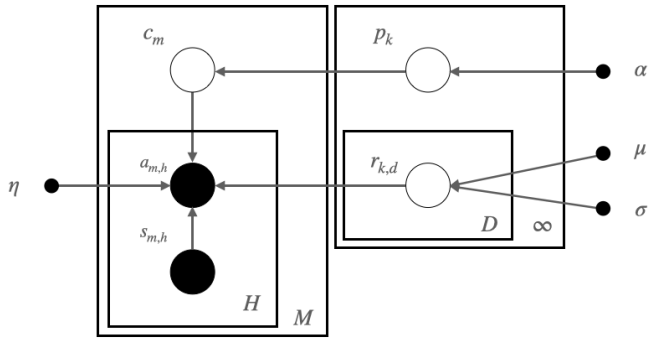


Figure 4: Graphical model of DPM-BIRL for MDP.

where the posterior $P(\mathbf{r}_k|\mathcal{X}, \eta, \mu, \sigma)$ is given by equation 2.33 and $\mathbf{c} = \{c_m\}_{m=1}^{M}$, $\mathbf{r_k} = \{\mathbf{r}_k\}_{k=1}^{K}$. Metropolis-Hastings (MH) algorithm can be used for inference, sampling each variable in turn. To sample c_m for the MH update, the conditional distribution can be defined as

$$P(c_m|\mathbf{c}_{-m}, \mathbf{r_k}, \mathcal{X}, \eta, \alpha) \propto P(\mathcal{X}_m|\mathbf{r}_{c_m}, \eta)P(c_m|\mathbf{c}_{-m}, \alpha)$$
$$P(c_m|\mathbf{c}_{-m}, \alpha) \propto \begin{cases} n_{-m,c_j}, \text{if } c_m = c_j \text{ for some } j \\ \alpha \text{ if } c_m \neq c_j \text{ for all } j \end{cases} \tag{2.35}$$

where $\mathbf{c}_{-m}$ is the set of all c_i such that $i \neq m$, and n_{-m,c_j} is the total number of trajectories (including $\mathcal{X}_m$) assigned to the cluster c_j.

For the MH update, conditional distribution for sampling $\mathbf{r}_k$ is given as,

$$P(\mathbf{r}_k|\mathbf{c}, \mathbf{r}_{-k}, \mathcal{X}, \eta, \mu, \sigma) \propto P(\mathcal{X}_{\mathbf{c}(k)}|\mathbf{r}_k, \eta)P(\mathbf{r}_k|\mu, \sigma) \tag{2.36}$$

The resulting MH algorithm consists of two steps-

1. **Updating cluster assignment c** - From equation 2.35, we sample a new cluster c'_m, and if c'_m is not present in $\mathbf{c}_{-m}$, a new reward function $\mathbf{r}_{c'_m}$ is drawn, and we set $c_m = c'_m$ with the acceptance probability of $min\{1, \frac{P(\mathcal{X}_m|\mathbf{r}_{c'_m}, \eta)}{P(\mathcal{X}_m|\mathbf{r}_{c_m}, \eta)}\}$.

2. **Updating reward function** $\{\mathbf{r}_k\}_{k=1}^{K}$ - A new reward function $\mathbf{r}'_k$ is sampled

as

$$\mathbf{r}'_k = \mathbf{r}_k + \frac{\tau^2}{2}\nabla \log f(\mathbf{r}_k) + \tau\epsilon \tag{2.37}$$

and the reward function is set as $\mathbf{r}_k = \mathbf{r}'_k$ with the acceptance probability $min\{1, \frac{f(\mathbf{r}'_k)g(\mathbf{r}'_k,\mathbf{r}_k)}{f(\mathbf{r}_k)g(\mathbf{r}_k,\mathbf{r}'_k)}\}$ where ϵ is sampled from a normal distribution $\mathcal{N}(0,1)$, τ is a non-negative real number for scaling, $f(\mathbf{r}_k)$ is the target distribution for the MH update and the function $g(x,y)$ is given (from [33]) by the equation,

$$g(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi\tau^2}\left\|\mathbf{x}-\mathbf{y}-\frac{1}{2}\tau^2\nabla \log f(\mathbf{x})\right\|_2^2 \tag{2.38}$$

Here, the target distribution $f(\mathbf{r}_k)$ is equal to $P(\mathcal{X}_{c(k)}|\mathbf{r}_k,\eta)P(\mathbf{r}_k|\mu,\sigma)$.

Therefore, the complete algorithm for estimating multiple rewards for MDPs is shown in 3, and the graphical model of DPM-BIRL is shown in 5.

Algorithm 3 Metropolis-Hastings algorithm for MDP

Initialise $\mathbf{c}$ and $\{\mathbf{r}_k\}_{k=1}^K$
for $t = 1$ **to** *maxiter* **do**
  for $m = 1$ **to** M **do**
    $c'_m \sim P(c_m|\mathbf{c}_{-m},\alpha)$
    if $c'_m \notin \mathbf{c}_{-m}$ **then** $\mathbf{r}_{c'_m} \sim P(\mathbf{r}|\mu,\sigma)$
      $< c_m, \mathbf{r}_m > \leftarrow < c'_m, \mathbf{r}'_m >$ with probability $min\{1, \frac{P(\mathcal{X}_m|\mathbf{r}_{c'_m},\eta)}{P(\mathcal{X}_m|\mathbf{r}_{c_m},\eta)}\}$
  for $k = 1$ **to** K **do**
    $\epsilon \sim \mathcal{N}(0,1)$
    $\mathbf{r}'_k \leftarrow \mathbf{r}_k + \frac{\tau^2}{2}\nabla \log f(\mathbf{r}_k) + \tau\epsilon$
    $\mathbf{r} \leftarrow \mathbf{r}'_k$ with probability $min\{1, \frac{f(\mathbf{r}'_k)g(\mathbf{r}'_k,\mathbf{r}_k)}{f(\mathbf{r}_k)g(\mathbf{r}_k,\mathbf{r}'_k)}\}$

Chapter 3

Bayesian IRL for Multiple Reward setting in LMDPs

As discussed in Section 2.3, LMDPs make computations and inferences far more tractable as compared to MDPs, and therefore can be considered an efficient approximation of the traditional MDP setting. Also, as mentioned in Section 2.5,Choi, J. Kim, (2012)[7] have established the formulation of the non-parametric Bayesian inverse reinforcement learning framework for estimating reward functions for only the traditional MDP case. Therefore in this work, our primary objective is to extend the framework to the LMDP setting which can allow both faster computations and reasonable approximation of the estimated reward functions.

In the LMDP setting, under optimal control, the trajectories obtained from the agent are of the form, $X = \{(s_0, s_1), (s_1, s_2), (s_2, s_3) \dots (s_{n-1}, s_n)\}$, i.e., state pairs, in absence of explicit "actions" from the agent. Therefore the probability distribution of a state-state transition (s_i, s_{i+1}) under this context can be reformulated from equation 2.5, as

$$P((s_i, s_{i+1})|\mathbf{R}) = \frac{1}{Z_i} e^{\alpha_\chi z^*(s_i, s_{i+1}, \mathbf{R})}, \tag{3.1}$$

where the normalizing value Z_i is dependent on the possible neighbors of state s_i.

Therefore in this context, the likelihood of the M trajectories under a reward func-

tion $\mathbf{r}$, given as equation 2.32 for the MDP setting, can be modified as,

$$P(\mathcal{X}|\mathbf{r},\eta) = \prod_{m=1}^{M}\prod_{h=1}^{H} P(s'_{m,h}|s_{m,h},\mathbf{r},\eta) = \prod_{m=1}^{M}\prod_{h=1}^{H} \frac{\exp(\eta \log(z^*(s'_{m,h};\mathbf{r})))}{\sum_{s''\in ne(s_{m,h})} \exp(\eta \log(z^*(s''_{m,h};\mathbf{r})))} \tag{3.2}$$

where $z^*(s;\mathbf{r})$ is the desirability value of a state s under reward function $\mathbf{r}$, H is the number of transitions, and η is the confidence parameter of the optimal control.

Following the DPM-BIRL algorithm steps as given in Section 2.5.1, we keep the cluster assignment step unchanged, however, the reward update step requires the gradient $\nabla \log f(\mathbf{r}_k)$ where $f(\mathbf{r}_k)$ is the target distribution of the Metropolis-Hastings update, or the unnormalized posterior of the $\mathbf{r}_k$, i.e., $P(\mathcal{X}_{c(k)}|\mathbf{r}_k,\eta)P(\mathbf{r}_k|\mu,\sigma)$ as given in equation 2.37.

In the LMDP setting, the gradient $\nabla \log f(\mathbf{r}_k)$ can be obtained by differentiating the log-likelihood and the log-prior given in equation 3.2, as

$$\nabla \log f(\mathbf{r}_k) = \nabla \log(P(\mathcal{X}_{c(k)}|\mathbf{r}_k,\eta)) + \nabla \log(P(\mathbf{r}_k|\mu,\sigma)). \tag{3.3}$$

Considering a known prior on the reward $\mathbf{r}_k$, the derivative of the likelihood of one particular transition, with respect to all features of the reward $\mathbf{r}_k$ can be obtained by differentiating equation 3.2 for a single transition from state s to s', as,

$$\begin{aligned}\frac{d}{dr(i)}\left[\frac{\exp(\eta(z(s')))}{\sum_{s''\in ne(s)}\exp(\eta(z^*(s'')))}\right] &= \frac{\eta\exp(\eta(z(s')))}{\sum_{s''\in ne(s)}\exp(\eta\log(z(s'')))}\cdot\frac{dz(s')}{dr(i)} \\ &- \frac{\exp(\eta(z(s')))}{\left(\sum_{s''\in ne(s)}\exp(\eta\log(z(s'')))\right)^2}\sum_{s''\in ne(s)}\eta\exp(\eta(z(s')))\frac{dz(s'')}{dr(i)}.\end{aligned} \tag{3.4}$$

The derivative of z-values with respect to $r(i)$ can be derived from equation 2.20 as,

$$\frac{dz(i)}{dr(i)} = \frac{1}{\gamma}z(i) + \frac{exp(r(i))}{\gamma}\cdot\sum_j p_{ij}\frac{dz(j)}{dr(i)} \tag{3.5}$$

which can be empirically shown to converge for $r(i) \in [-1, 0]$.

The modified graphical model of DPM-BIRL for LMDP setting is given in Fig 5.

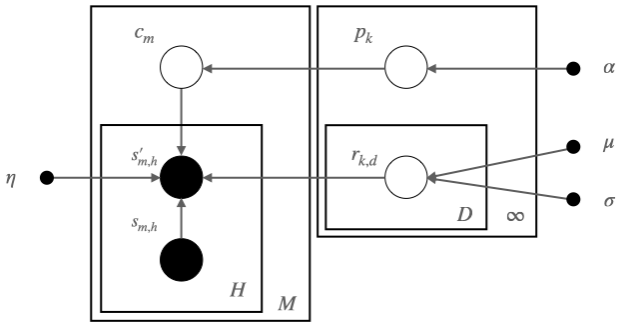


Figure 5: Graphical model of DPM-BIRL for LMDP.

Similarly for the EM-MLIRL approach, the likelihood from equation 2.29 can be rewritten as,

$$L(\mathcal{X}|\theta) = \sum_{i=1}^{N} \sum_{(s,s')\in\xi_i} w_i \log \frac{e^{\beta z(s,s';\theta)}}{\sum_{s''} e^{\beta z(s,s'';\theta)}}, \tag{3.6}$$

and ∇L can be computed as shown in equation 3.4 for the MLIRL update.

Chapter 4

Experimental Results

We evaluate the performance of DPM-BIRL in the both the MDP and LMDP scenario for a multiple rewards gridworld. We estimate trajectory clusters and associated rewards from the behavioral data, and compare the estimated rewards to the ground truth, by evaluating the optimal policies obtained in both the cases.

4.1 Gridworld Problem

Here, we take a sample grid world (Fig 6), of size 8×8, where each of the 64 positions of the agent on the grid is considered as a state, with no absorbing or terminal states. The agent is allowed to move in all 4 directions, East, West, North, South, and the uncontrolled transition probability is considered .25 in all directions, except for the boundary positions. The grid is also sub-partitioned into 2×2 blocks and the feature function is defined as unity over each subgrid. The reward function was linearly parameterised as $R(s) = \sum_{d=1}^{D} r_d \phi_d(s)$ where $\phi_d : S \to \mathbb{R}$ and $r = [r_1, ...r_D]$.

The performance of the algorithm is evaluated by the expected value difference (EVD) given by $|V^*(\mathbf{r}_g) - V^{\mathbf{u}^*(\mathbf{r}_{est})}(\mathbf{r}_g)|$ where $\mathbf{r}_g$ is the underlying ground truth reward function responsible for the behavioral data, and $\mathbf{r}_{est}$ is our estimated reward function associated with the trajectory, and $V^{\mathbf{u}^*(\mathbf{r}_{est})}(\mathbf{r}_g)$ is the value of the states of the optimally controlled LMDP under $\mathbf{r}_{est}$, evaluated over $\mathbf{r}_g$. The EVD here thus

gives a measure of the performance error in estimating the ground truth reward function accurately.

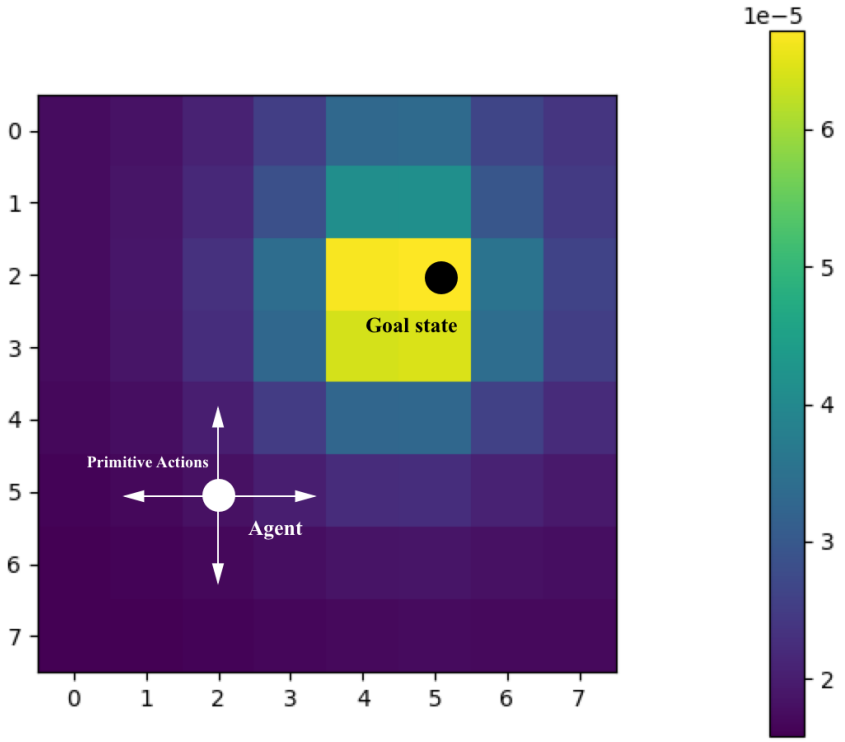


Figure 6: Gridworld with 8×8 grids, the colormap indicates the z or desirability values of the LMDP.

We first calculate the EVD between the ground truth and estimated rewards, for a confidence value of $\eta = 10$ and compute the average over the total number of trajectories.

For this task, we assume 3 ground truth reward functions and plot the average EVD obtained as we increase the number of trajectories generated for each reward function/agent. As observed in Fig 7, the graph shows an overall downward trend as we increase the number of trajectories per agent, implying our estimated reward functions are closer and closer to the ground truth. Comparing the LMDP, MDP, and EM-MLIRL[6] for the LMDP case with the number of predefined clusters set to 3, it can be observed that in both the LMDP and MDP scenario, the overall average EVD is much lower, along with noise due to random initialization of the ground

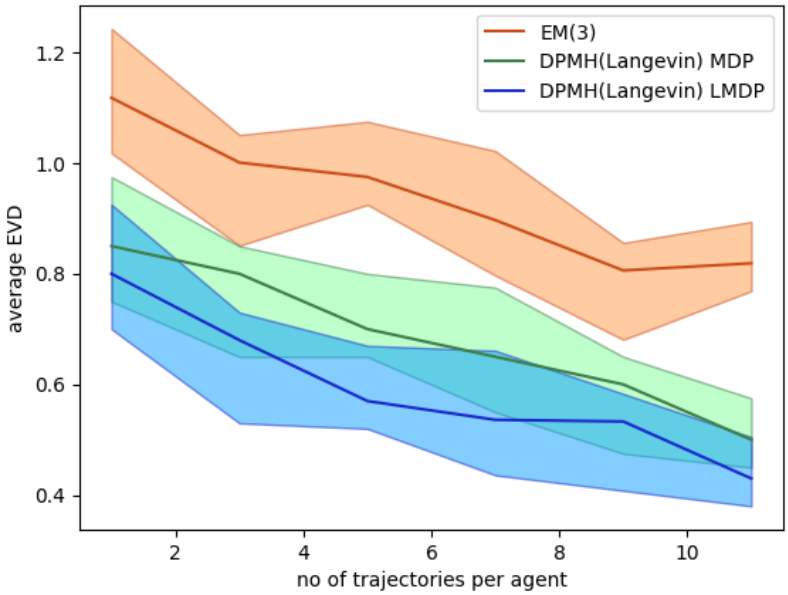


Figure 7: Average EVD with increasing number of trajectories per agent.

truth rewards and the initial estimated rewards. The average EVDs are obtained by taking an average over 20 runs with random initializations for each case.

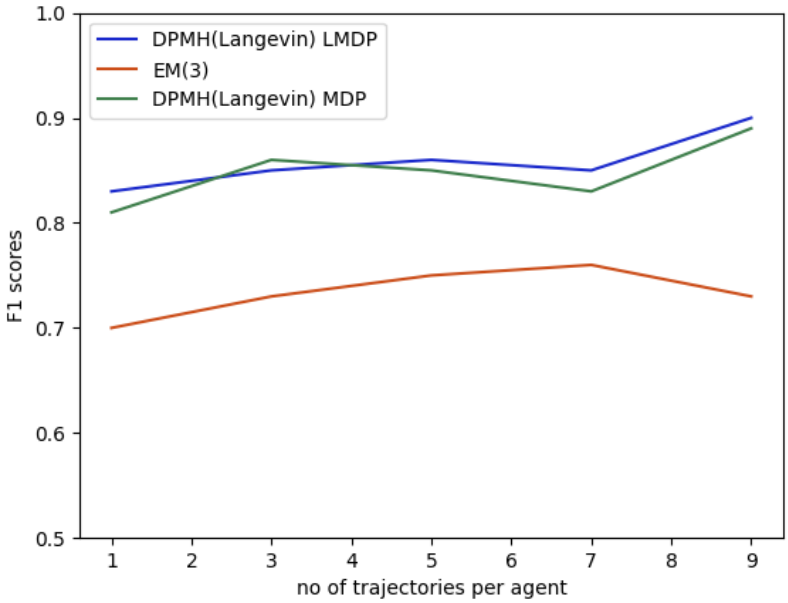
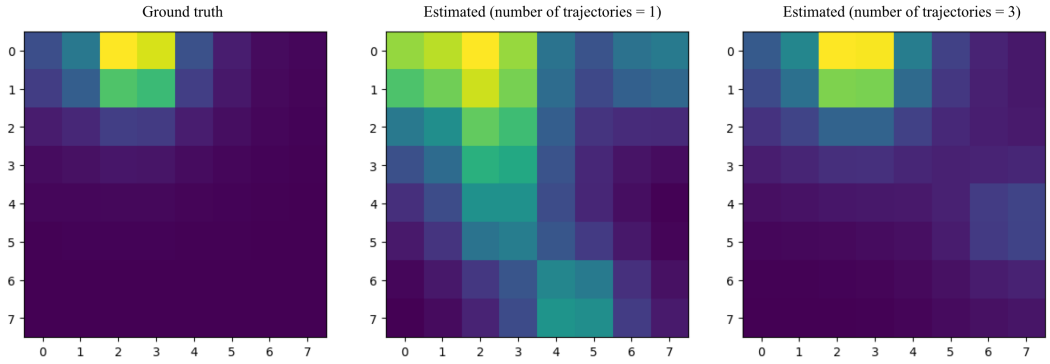
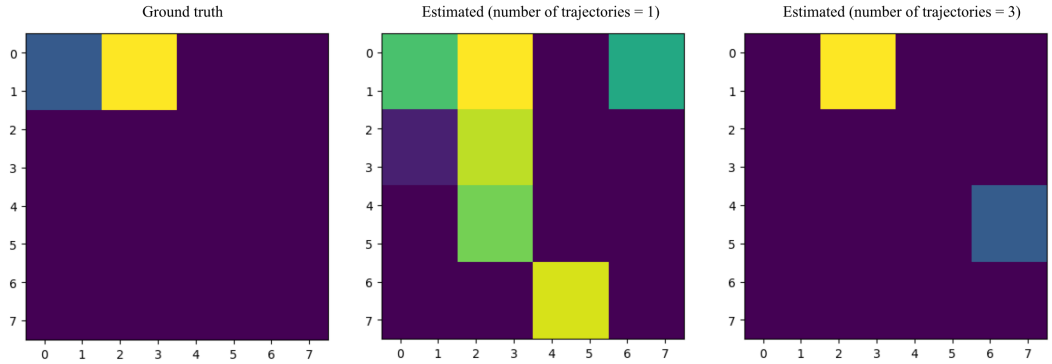


Figure 8: F1 scores with an increasing number of trajectories per agent.

Fig 8 shows the F1 scores obtained from the clustering for the three cases for increasing number of ground truth trajectories per agent. The EM-MLIRL(3) performs much worse compared to the DPMH (Langevin) cases, whereas both the MDP and LMDP scenarios have comparable and higher F1 scores.



(a) z function



(b) reward function

Figure 9: Comparison of ground truth z values vs estimated z values (Fig 9a) and estimated rewards (Fig 9a) with increasing number of ground truth trajectories generated.

We further observe the ground truth rewards and z functions estimated for different numbers of ground truth trajectories used for inference in the DPM-BIRL (LMDP) setting. As seen in Fig 9a, the z functions estimated for a higher number of trajectories per agent (=3) resembles the ground truth z function much more closely as compared to the 1 trajectory per ground truth reward case, which enforces the decrease in EVD values as shown in Fig 7. In Fig 9b, the corresponding estimated reward functions are shown.

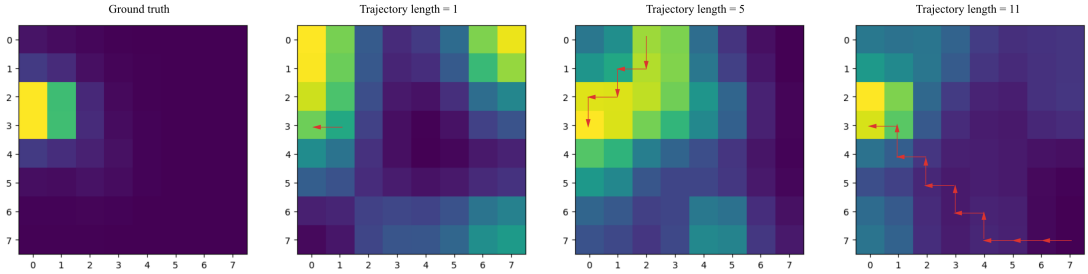


Figure 10: Trajectories and z values obtained for different values of s_0 for a deterministic case.

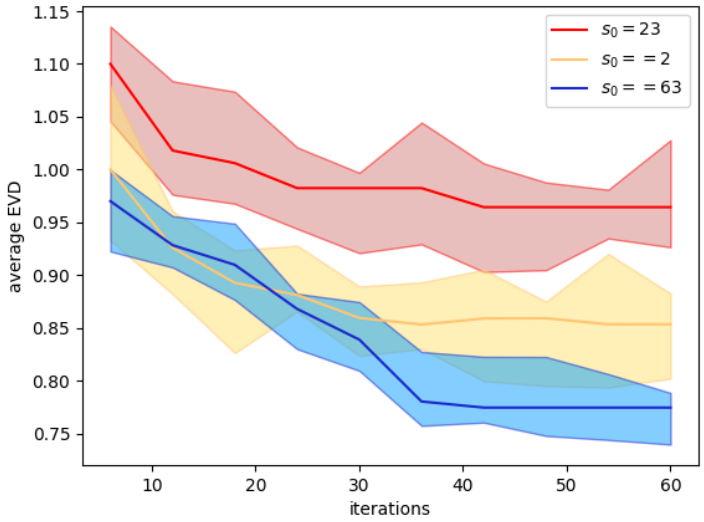


Figure 11: Comparison between average EVD values obtained over iterations for different values of s_0.

The similarity of both the estimated reward functions and the z functions however are dependent on the the initial state s_0 of the trajectories. Fig 10 shows three sampled trajectories of different lengths. It can be observed that with the increase in length of the trajectory, i.e., s_0 is situated at a much larger distance from the reward concentration, we get a lesser value of EVD as shown in Fig 11. This can be due to the decrease in "exploration" of the gridworld if the initial state s_0 is close to the reward concentration. Choosing an initial point at a higher distance allows more discovery of the gridworld, and thus the trajectory has more information that can be used to accurately estimate the underlying reward function.

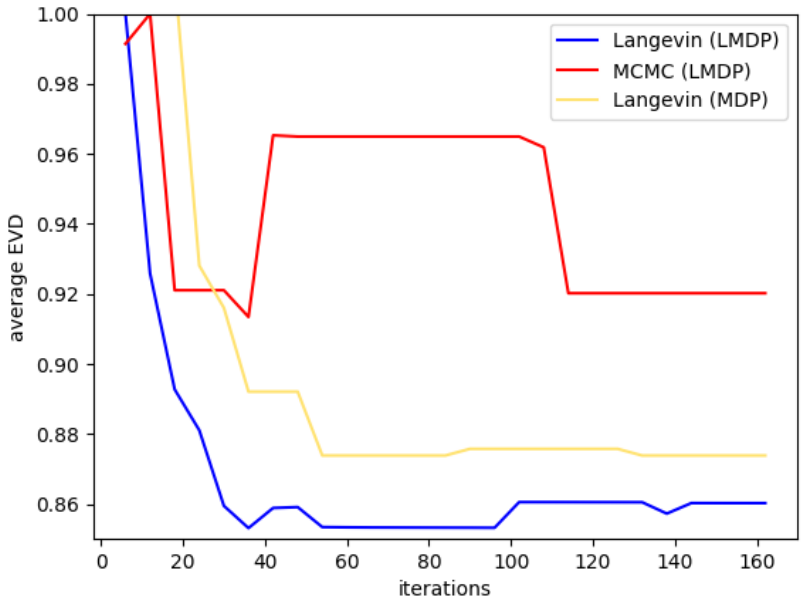


Figure 12: Average EVD with number of iterations, for DPM-BIRL Langevin (LMDP), MCMC (LMDP) and Langevin (MDP).

We also compare the effect of using the gradients of the estimated likelihood with respect to the reward, with only MCMC sampling. In Fig 12, for clustering of 3 ground truth trajectories, we can observe that the DPMH(LMDP) with the use of likelihood gradients performs significantly better and converges to a much lower average EVD value as compared to both random sampling and DPMH (MDP), where, the latter also takes significantly more number of iterations to converge.

Fig 13 shows the comparison of average EVD values, for different values of η for DPM (LMDP). We compare 3 values of η, and as the value of η decreases, since the trajectories now produced from the ground truth reward functions contain much more noise, our ground truth algorithm performs much worse. Increasing the value of η ensures much less noise in the ground truth trajectories, therefore we can observe smaller values for the calculated average EVD.

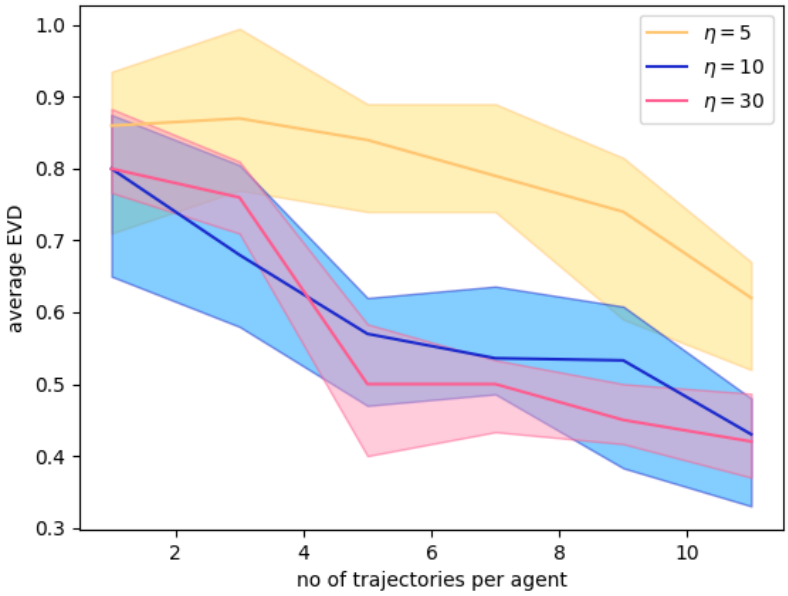


Figure 13: Average EVD of trajectories obtained from DPM (Langevin) LMDP, with an increasing number of trajectories per agent, for different values of η.

Chapter 5

Discussion

In this work, we introduce the formulation of LMDPs in nonparametric Bayesian approach to IRL for multiple reward functions using the Dirichlet process mixture model, and extend the existing work in inverse optimal control beyond a single fixed reward function. Our method allows us to learn an unspecified number of reward functions for LMDPs and also permits incorporation of any domain knowledge of the reward function as the reward prior in the Bayes equation. As observed from the results, the LMDP formulation converges to a lower error value for the multiple reward scenario, as compared to the standard MDP formulation, as well as taking much less time for computation, since policy iteration in MDP is far more computationally expensive.

5.1 Future Work

1. The ease of computation allows us to explore and experiment with varying real life behavioral data, which is often generated by multiple agents, acting under distinct, unknown reward functions.

2. Current ongoing work is on estimating the reward function associated with a new trajectory, without training on it.

3. Exploring performance on different distribution of ground truth reward func-

tions as well as different starting states for the generated trajectories.

4. We want to further explore function approximations for LMDPs for this multiple rewards scenario, to allow us to estimate reward functions for larger state spaces, where tabular LMDPs might be too expensive.

List of Figures

List of Algorithms

Bibliography

[1] Abbeel, P. & Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning* (2004).

[2] Neu, G. & Szepesvári, C. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 295–302 (2007).

[3] Ziebart, B., Maas, A., Bagnell, J. & Dey, A. Maximum entropy inverse reinforcement learning. 1433–1438 (2008).

[4] Watkins, C. J. C. H. *Learning from Delayed Rewards*. Ph.D. thesis, King's College, Oxford (1989).

[5] Touretzky, D. & Saksida, L. Operant conditioning in skinnerbots. *Adaptive Behaviour* **5** (1997).

[6] Babes-Vroman, M., Marivate, V., Subramanian, K. & Littman, M. Apprenticeship learning about multiple intentions. 897–904 (2011).

[7] Choi, J. & Kim, K.-e. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, vol. 25 (2012).

[8] Dvijotham, K. & Todorov, E. Inverse optimal control with linearly-solvable mdps. ICML'10, 335–342 (Omnipress, Madison, WI, USA, 2010).

[9] Neal, R. M. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics* **9**, 249–265 (2000).

[10] Sutton, R. & Barto, A. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks* **9**, 1054–1054 (1998).

[11] Bertsekas, D. & Tsitsiklis, J. *Neuro-Dynamic Programming*, vol. 27 (1996).

[12] Ramachandran, D. & Amir, E. Bayesian inverse reinforcement learning. IJCAI'07, 2586–2591 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007).

[13] Todorov, E. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, vol. 19 (MIT Press, 2006).

[14] Todorov, E. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences* **106**, 11478–11483 (2009).

[15] Kappen, H. J., Gómez, V. & Opper, M. Optimal control as a graphical model inference problem. *Machine Learning* **87**, 159–182 (2012).

[16] Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* (2018).

[17] Todorov, E. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, 1856–1864 (2009).

[18] Jonsson, A. & Gómez, V. Hierarchical linearly-solvable markov decision problems. In *26th International Conference on Automated Planning and Scheduling*, ICAPS'16, 193–201 (AAAI Press, 2016).

[19] Saxe, A. M., Earle, A. C. & Rosman, B. Hierarchy through composition with multitask LMDPs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3017–3026 (2017).

[20] Jonsson, A. & Gómez, V. Hierarchical Linearly-Solvable Markov Decision Problems. *Proceedings of the 26th International Conference on Automated Planning and Scheduling* 193–201 (2016).

[21] Infante, G., Jonsson, A. & Gómez, V. Globally Optimal Hierarchical Reinforcement Learning for Linearly-Solvable Markov Decision Processes. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 6970–6977 (2022).

[22] Williams, G., Aldrich, A. & Theodorou, E. A. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics* **40**, 344–357 (2017).

[23] Gómez, V., Kappen, H. J., Peters, J. & Neumann, G. Policy search for path integral control. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 482–497 (Springer, 2014).

[24] Matsubara, T., Gómez, V. & Kappen, H. J. Latent Kullback Leibler control for continuous-state systems using probabilistic graphical models. *30th Conference on Uncertainty in Artificial Intelligence* (2014).

[25] Van Den Broek, B., Wiegerinck, W. & Kappen, B. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research* **32**, 95–122 (2008).

[26] Gómez, V., Thijssen, S., Symington, A. C., Hailes, S. & Kappen, H. J. Real-time stochastic optimal control for multi-agent quadrotor systems. In *26th International Conference on Automated Planning and Scheduling* (2016).

[27] Wan, N., Gahlawat, A., Hovakimyan, N., Theodorou, E. A. & Voulgaris, P. G. Cooperative path integral control for stochastic multi-agent systems. *arXiv preprint arXiv:2009.14775* (2020).

[28] Chertkov, M., Chernyak, V. Y. & Deka, D. Ensemble control of cycling energy loads: Markov decision approach. In *Energy Markets and Responsive Grids*, 363–382 (Springer, 2018).

[29] Thalmeier, D., Gómez, V. & Kappen, H. J. Action selection in growing state spaces: control of network structure growth. *Journal of Physics A: Mathematical and Theoretical* **50**, 034006 (2016).

[30] Abbasi-Yadkori, Y., Bartlett, P., Chen, X. & Malek, A. Large-scale Markov decision problems with KL control cost and its application to crowdsourcing. In *International Conference on Machine Learning*, 1053–1062 (2015).

[31] Tabibian, B., Gómez, V., De, A., Schölkopf, B. & Gomez Rodriguez, M. On the design of consequential ranking algorithms. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, vol. 124, 171–180 (2020).

[32] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley Sons, Inc., USA, 1994), 1st edn.

[33] Roberts, G. O. & Rosenthal, J. S. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **60**, 255–268 (1998).