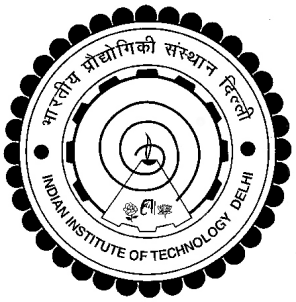


Dept of Electrical Engineering
Indian Institute of Technology Delhi

Course Code : ELD431
Academic Year : 2022 - 2023, Semester - I

Inverse Reinforcement Learning with Constraint Recovery

Nirjhar Das
2019EE30585

Advisor: Prof. Arpan Chattopadhyay

Abstract: In this work, we explore a novel Inverse Reinforcement Learning (IRL) method for Constrained Markov Decision Process (CMDP). Often, in modelling real world to train an agent, rewards are not enough. One needs to enforce certain constraints to be able to meaningfully model such situations. When constraints are present in an MDP, the policy of the agent has to respect it. In standard IRL, the objective is to recover the reward function of the MDP, given a set of trajectory demonstrations based on the optimal policy. Our work goes a step further to recover both the reward and the constraint functions, given the trajectory demonstrations from a CMDP's optimal policy. We formulate the learning problem as a convex optimization problem, deriving it based on the principle of maximum entropy. We illustrate a simple algorithm to solve the problem and demonstrate the effectiveness of our method on gridworld simulations.

1 Introduction

The goal of Artificial Intelligence is to produce agents that can learn from their surroundings and make their own decisions. Reinforcement Learning is a promising direction in which the goal of the agent is to learn an optimal policy by interacting with the environment. Reinforcement Learning is often applied in a standard Markov Decision Process setting. A Markov Decision Process (MDP) comprises of a state space and an action space. The agent gets a reward when it takes a particular action in a particular state. The rewards are usually stochastic in nature. Thereafter, the agent moves from the current state to the next state based on a state-transition probability distribution. The process can go on for finite as well as an infinite number of steps. The goal of the agent is to learn an optimal policy that helps the agent exploit the environment by collecting maximum return.

Reinforcement Learning is of great interest to researchers and several algorithms have been proposed to learn an optimal policy in various settings. Some of the popular algorithms are Policy Gradient[11], Actor-Critic Methods[3], Trust Region Policy Optimization[7], Deep Q-Networks[5], Dyna[10] and AlphaZero[9].

Inverse Reinforcement Learning (IRL) was first proposed and formulated in [6]. In IRL, the goal is to recover the reward function of the MDP when a collection of trajectories is given, assuming that the trajectories have been generated by an optimal policy. However, this problem is ill-posed as there may be several reward functions that give the same optimal policy. In fact, in the formulation given in [6], the reward function $R = 0$ for all states and actions is also a solution. Thus, several methods have been proposed to select one of the many possible reward functions. In [6], the authors introduce a condition that selects the reward function that maximizes the difference between the action-value of the expert action and the next best action. In recent years, the paradigm of Maximum Entropy Inverse Reinforcement Learning [12] has gained a lot of attention and popularity. In [12], the authors consider the set of probability distribution over trajectories that satisfy the constraint of feature distribution matching, and they propose to select that distribution that corresponds to the maximum entropy. The maximum entropy principle has been known to favour distributions which do not involve additional bias beyond the necessary.

Although several algorithms have been proposed for IRL in MDPs, there has been little attention to the Constrained Markov Decision Process (CMDP) in this context. In a CMDP, apart from the reward function, we also have a set of constraint functions. The optimal policy should be such that the return over the constraint function is less than or equal to the corresponding constraint budget. CMDPs are particularly relevant in modelling various real life scenarios. For example, while pouring water from a glass jar to a glass bottle, one has to respect the constraint that neither of the glass vessels break. In this case, the reward function is not enough as the agent should not only be able to achieve its goal of transferring the water, but also take care of the constraint. In this work, we consider the problem of recovering the reward and the constraint simultaneously when presented with a set of trajectories obtained by executing an optimal policy in a CMDP.

There are no prior works tackling this problem. However, some of the works of interest are [8] and [4]. In [8], the authors propose to recover an Maximum Likelihood Constraint set which when appended to the base MDP, forms a constrained MDP that explains the observed trajectories. However, they assume that the reward function of the MDP is known to the learner. In [4], the authors extend [8] to a parameterized policy setting (typically, a neural network) along with approximation of the indicator function with another parameterized binary classifier. However, the main objective of their work is to recover the expert's policy. Instead we focus on recovering both the reward function and the constraint function. This has potential application in teaching an agent from human demonstrations as the performance of RL algorithms depend heavily on the reward and constraint specification. In this work, we rely heavily on the principle of maximum entropy to favour a particular reward and constraint function over others. We demonstrate the performance of our algorithm over gridworld simulations.

2 Preliminaries

An MDP is characterized by a five tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is

the reward function, p is the state-transition probability set and γ is the discount factor. When an agent in state $s \in \mathcal{S}$ takes an action $a \in \mathcal{A}$, then the agent receives a reward $r(s,a)$ and moves to the next state according to the probability distribution $p(s'|s,a),\ s' \in \mathcal{S}$. The *return* of a sequence of rewards $\{r_t\}_{t=1}^{T}$ obtained by the agent is defined as

$$G_T = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{T-1} r_T$$

A *policy* in an MDP is the function $\pi : S \rightarrow p(a),\ a \in \mathcal{A}$, that is, a mapping from the state space to a probability distribution over the action space. Thus, if an agent is following a policy π, it takes an action in state $s \in \mathcal{S}$ based on the probability distribution $\pi(a|s),\ a \in \mathcal{A}$. The return of an agent under a policy π, starting from state s is called the *value* of the state. Thus, the *value function* is defined as:

$$V^{\pi}(s) = \mathbb{E}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t, a_t)}} \Big[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \Big| s_1 = s \Big]$$

The optimal policy is thus given by

$$\pi^* = \arg\max_{\pi} V^{\pi}(s)\ \forall s \in \mathcal{S}$$

Moving to the CMDP setting, we have, apart from the reward function, one or more constraint functions and their corresponding constraint budgets. In this work, we will consider only one constraint function although it can be easily extended to multiple constraints. We denote the constraint function as c and its corresponding *budget* as α. Let us define the constraint value of the state as

$$C^{\pi}(s) = \mathbb{E}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t, a_t)}} \Big[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \Big| s_1 = s \Big]$$

Thus, the formulation of the optimal policy is now given by

$$\begin{aligned} \pi^* = \arg\max_{\pi} \quad & \sum_{s \in \mathcal{S}} p_0(s) V^{\pi}(s) \\ \text{s.t.} \quad & \sum_{s \in \mathcal{S}} p_0(s) C^{\pi}(s) \leq \alpha \end{aligned} \tag{1}$$

where $p_0(\cdot)$ is the initial state distribution.

In IRL, the objective is to learn the reward function from a set of trajectories $D = \Big\{ \tau = \{(s_t, a_t)\}_{t=1}^{T} \Big\}$ generated by executing the optimal policy π^* in an MDP. We are also given the MDP\R, that is, the tuple $(\mathcal{S}, \mathcal{A}, p, \gamma)$. The problem is often formulated as a Maximum Likelihood problem given as

$$r^* = \arg\max_{r} \quad \prod_{\tau \in D} p(\tau|r)$$

In the next section, we formulate the case for the CMDP using ideas developed in this section.

3 Formulation

In a CMDP setting, our goal is to recover both the reward and the constraint function simultaneously. Thus, we want to solve the following problem:

$$r^*, c^* = \arg\max_{r,c} \quad \prod_{\tau \in D} p(\tau|r, c, \alpha)$$

First we define a new the constraint function c' as $c'(s,a) = c(s,a)/\alpha\ \forall s \in \mathcal{S}, a \in \mathcal{A}$. Although both c and α are unknown to us, we can restate the above formulation in terms c' since from eq.(1), we can see that the optimal policy is unchanged if we divide both sides by α in the constraint. This implies that the constraint specification (c, α) is the same as $(c', 1)$. Thus, our objective is to solve

$$r^*, c^* = \arg\max_{r,c'} \quad \prod_{\tau \in D} p(\tau|r, c', 1) \tag{2}$$

Now, we make some assumptions about the reward and constraint structure of the CMDP. (1) We assume that the reward and constraint are independent of the actions, that is: $r(s,a) = r(s) \forall\ s \in \mathcal{S},\ a \in \mathcal{A}$ and similarly for $c(\cdot,\cdot)$. This is a standard assumption in several papers in the domain of IRL, as it simplifies the computation. However, the method developed here can be easily generalized to the case where this assumption does not hold. (2) We assume that the reward or constraint are linear with respect to the features of the states. That is,

$$r(s) = \mathbf{w}_r^T \Phi_r(s)$$
$$c'(s) = \mathbf{w}_c^T \Phi_c(s)$$

for some unknown parameters $\mathbf{w}_r \in \mathbb{R}^{d_r}$ and $\mathbf{w}_c \in \mathbb{R}^{d_c}$. Thus, $p(\tau|r, c', 1) = p(\tau|\mathbf{w}_r, \mathbf{w}_c)$. This is a reasonable assumption since the user may be able to specify a set of useful features to the learner in real settings. Moreover, in extreme case, the features can be taken to be one-hot encoded vectors corresponding to the states.

Further, we define the quantities *empirical feature expectation* (EFE) and *policy feature expectation* (PFE) as follows:

$$\begin{aligned} \text{EFE:} \quad & \tilde{\Phi}_x = \frac{1}{|D|} \sum_{\tau \in D} \Phi_x(\tau) \\ \text{PFE:} \quad & \hat{\Phi}_x = \sum_{\text{all } \tau} p(\tau) \Phi_x(\tau) \end{aligned} \tag{3}$$

where $\Phi_x(\tau) = \sum_{s_t \in \tau} \gamma^{t-1} \Phi_x(s_t)$ and $x \in \{r, c\}$, with a slight abuse of notation. Note that in eqn.(3), $p(\tau) = p(\tau|\mathbf{w}_r, \mathbf{w}_c)$ and the probability distribution over the trajectories is generated by executing an optimal policy π^* computed according to eqn.(1).

In [1], the authors show that it is necessary and sufficient to match EFE and PFE to guarantee the same performance as the optimal policy under assumption (2). Although this condition has been stated for reward features, we extend this to constraint features. We also introduce a new constraint corresponding to the requirement that the distribution be such that the constraint value is less than or equal to the constraint budget. Now, we use the principle of maximum entropy to determine the distribution $p(\tau|\mathbf{w}_r, \mathbf{w}_r)$ out of the many possible distributions that satisfy these constraints. The problem can be formulated as:

$$\begin{aligned} \min_{p} \quad & \sum_{\text{all } \tau} p(\tau) \log(p(\tau)) \\ \text{s.t.} \quad & \sum_{\text{all } \tau} p(\tau) \Phi_r(\tau) = \tilde{\Phi}_r \\ & \sum_{\text{all } \tau} p(\tau) \Phi_c(\tau) = \tilde{\Phi}_c \\ & \sum_{\text{all } \tau} p(\tau) \mathbf{w}_c^T \Phi_c(\tau) \leq 1 \\ & \sum_{\text{all } \tau} p(\tau) = 1 \\ & p(\tau) \geq 0 \ \forall \ \tau \end{aligned} \tag{4}$$

This is a convex optimization problem. Thus, the Lagrangian of the problem is:

$$\begin{aligned} \mathcal{L}(p, \theta, \beta, \nu, \zeta, \delta) = & \sum_{\text{all } \tau} p(\tau) \log(p(\tau)) \\ & + \theta^T \Big(\sum_{\text{all } \tau} p(\tau) \Phi_r(\tau) - \tilde{\Phi}_r \Big) \\ & + \beta^T \Big(\sum_{\text{all } \tau} p(\tau) \Phi_c(\tau) - \tilde{\Phi}_c \Big) \\ & + \nu \Big(\sum_{\text{all } \tau} p(\tau) \mathbf{w}_c^T \Phi_c(\tau) - 1 \Big) \\ & + \zeta \Big(\sum_{\text{all } \tau} p(\tau) - 1 \Big) \\ & - \Big(\sum_{\text{all } \tau} \delta(\tau) p(\tau) \Big) \end{aligned} \tag{5}$$

where $\theta, \beta, \nu, \zeta, \delta$ are the dual variables of appropriate dimensions. Also note that $\nu \geq 0$ and $\delta(\tau) \geq 0 \ \forall \tau$.

From the KKT condition, we get:

$$\frac{\partial \mathcal{L}}{\partial p(\tau)} = 0 \tag{6}$$

for the optimal variables $(p^*, \theta^*, \beta^*, \nu^*, \zeta^*, \delta^*)$ Thus, we have

$$\begin{aligned} 1 + \log(p^*(\tau)) + \theta^{*T} \Phi_r(\tau) + \beta^{*T} \Phi(\tau) \\ + \nu^* \mathbf{w}_c^T \Phi_c(\tau) + \zeta^* - \delta^*(\tau) = 0 \end{aligned}$$

Solving for $p^*(\tau)$, we get

$$\begin{aligned} p^*(\tau) = & \exp(-1 - \zeta^* + \delta^*(\tau)) \\ & \times \exp\big(- \theta^{*T} \Phi_r(\tau) - [\beta^* + \nu^* \mathbf{w}_c]^T \cdot \Phi_c(\tau) \big) \end{aligned} \tag{7}$$

Now, we can interpret $-\theta^*$ as $\mathbf{w}_r$ since the higher the value of the term $-\theta^{*T} \Phi_r(\tau)$, higher is the probability of that trajectory. Similarly, we can interpret $\beta^* = \mathbf{w}_c$ as the choice of $\mathbf{w}_c$ is with us. Thus, the expression simplifies to

$$p^*(\tau|\mathbf{w}_r, \mathbf{w}_c) \propto \exp(\mathbf{w}_r^T \Phi_r(\tau) - (1 + \nu^*) \mathbf{w}_c^T \Phi_c(\tau))$$

Since $\nu^* \geq 0$, we can set $\lambda = 1 + \nu^* \geq 1$ as a new variable to finally obtain

$$p^*(\tau|\mathbf{w}_r, \mathbf{w}_c) \propto \exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau)) \tag{8}$$

Thus, we arrive at the hypothesis that the probability distribution over a trajectory is the Boltzman distribution. This form is intuitive as it states that higher

is the reward and lower is the constraint value of a trajectory, the more it is preferred. Further, in the Linear Program formulation of eq.(1), it is possible to convert[1] a CMDP to an unconstrained MDP by modifying the reward function. To modify the reward function, the constraint function multiplied with the optimal Lagrange multiplier is subtracted from the original reward function. This is exactly the form we recover using the Maximum Entropy Principle.

Now we are set to formulate the learning problem stated in eq.(2) using the $p(\tau|\mathbf{w}_r, \mathbf{w}_c)$ we found in eq.(8). To normalize the probability distribution, the partition function $Z(\mathbf{w}_r, \mathbf{w}_c)$ is the constant of proportionality. Thus,

$$p^*(\tau|\mathbf{w}_r, \mathbf{w}_c) = \frac{1}{Z(\mathbf{w}_r, \mathbf{w}_c)} \exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau)) \tag{9}$$

Hence, eq.(2) now becomes

$$\begin{aligned} \mathbf{w}_r^*, \mathbf{w}_c^* = \arg\max_{\mathbf{w}_r, \mathbf{w}_c} \quad & \prod_{\tau \in D} p^*(\tau|\mathbf{w}_r, \mathbf{w}_c) \\ \text{s.t.} \quad & \mathbf{w}_c^T \hat{\Phi}_c \leq 1 \end{aligned} \tag{10}$$

Here, we once again enforce the constraint so as to not allow arbitrary choices of $\mathbf{w}_r$ and $\mathbf{w}_c$ that would still produce the same $\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau)$ as the original $\mathbf{w}_r$, $\mathbf{w}_c$. Simplifying eq.(10) and taking the log-likelihood, we get

$$\begin{aligned} \min_{\mathbf{w}_r, \mathbf{w}_c} \quad & \frac{1}{|D|} \sum_{\tau \in D} -\mathbf{w}_r^T \Phi_r(\tau) + \lambda \mathbf{w}_c^T \Phi_c(\tau) \\ & + \log\Big(\sum_{\text{all } \tau} \exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau)) \Big) \\ & \textbf{s.t.} \quad \mathbf{w}_c^T \hat{\Phi}_c \leq 1 \end{aligned} \tag{11}$$

since $Z(\mathbf{w}_r, \mathbf{w}_c) = \sum_{\text{all } \tau} \exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau))$.

The above form in eq.(11) is a convex optimization problem as the first term is linear and the second term is $\log(\sum \exp(\cdot))$ of a linear function. Also, the constraint is linear. Thus we can employ standard techniques to solve this problem. However, a difficulty is that evaluating the partition function is computationally difficult as the number of trajectories grows at an exponential rate with the length of the trajectory.

[1]See [2] for a detailed treatment of CMDPs

To avoid these difficulties, we use gradient based approaches which allows us to avoid the pitfall. The gradient of the objective function $\mathcal{O}$ is:

$$\begin{aligned} \nabla_{\mathbf{w}_r} \mathcal{O} &= -\frac{1}{|D|} \sum_{\tau \in D} \Phi_r(\tau) \\ &\quad + \sum_{\text{all } \tau} \frac{\exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau))}{\sum_{\text{all } \tau} \exp(\mathbf{w}_r^T \Phi_r(\tau) - \lambda \mathbf{w}_c^T \Phi_c(\tau))} \cdot \Phi_r(\tau) \\ &= -\tilde{\Phi}_r + \hat{\Phi}_r \\ \nabla_{\mathbf{w}_c} \mathcal{O} &= \lambda(\tilde{\Phi}_c - \hat{\Phi}_c) \end{aligned} \tag{12}$$

Now we can use any gradient based method to solve the problem. In algorithm 1 we use Exponentiated Gradient Descent (EGD) as the intermediate gradient step. Other algorithms like Online Gradient Descent will also work.

However, finding the true $\mathbf{w}_r$ and $\mathbf{w}_c$ requires one more step—computing the *PFE*. Although it is not possible to sum over all the trajectories, PFE can still be calculated if we know the policy π. This is done using the *state visitation frequency* defined as

$$\rho_\pi(s) = \sum_{t=1}^{\infty} \gamma^{t-1} p(s_t = s|\pi)$$

. For a finite time horizon T, it can be computed using the recursive equations

$$\begin{aligned} d_1(s) &= p_0(s) \quad \forall s \in \mathcal{S} \\ d_{t+1}(s') &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \gamma \cdot d_t(s) \cdot \pi(a|s) \cdot p(s'|s, a) \\ & \qquad \forall s' \in \mathcal{S},\ t = 1, 2 \ldots, T-1 \end{aligned} \tag{13}$$

With the help of the state visitation frequency, we can write PFE as

$$\hat{\Phi}_x = \sum_{t=1}^{T} \sum_{s \in S} d_t(s) \Phi_x(s) \tag{14}$$

where $x \in \{r, c\}$.

Now, we can formulate an algorithm to recover both $\mathbf{w}_r$ and $\mathbf{w}_c$. The essential idea is that, starting from a random value of $\mathbf{w}_r$ and $\mathbf{w}_c$, we compute the optimal

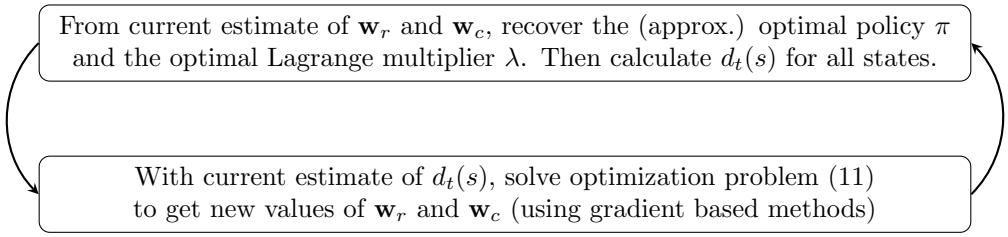


Figure 1: Scheme for recovery of the reward and constraint

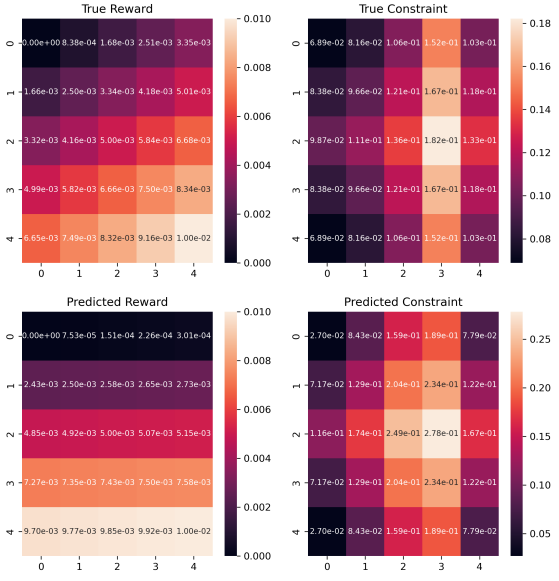
policy corresponding to it. Using the optimal policy, we calculate the state visitation frequency and hence, the PFE. After that, we apply a gradient step on $\mathbf{w}_r$ and $\mathbf{w}_c$. Thereafter, we go back to computing the optimal policy. Thus, we alternate between the two steps till the change in $\mathbf{w}_r$ and $\mathbf{w}_c$ are within the tolerance level. This has been illustrated in fig.1. The full description is given algorithm 1 which solves the problem of reward and constraint recovery in a CMDP.

Algorithm 1 Iterative Algorithm for Reward and Constraint Recovery

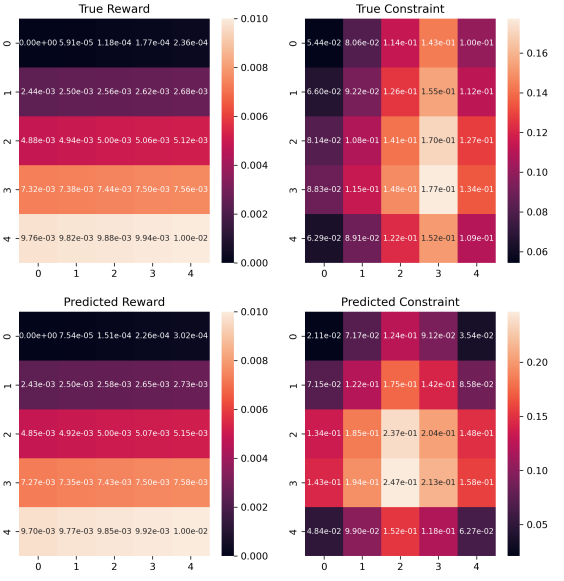
Require: $p_0,\ p(s'|s,a),\ D,$ learning rate $= \kappa$
$\mathbf{w}_r \leftarrow \mathbb{R}^{d_r} \sim$ Unit Sphere
$\mathbf{w}_c \leftarrow \mathbb{R}^{d_c} \sim$ Unit Sphere
Compute $\tilde{\Phi}_r$ and $\tilde{\Phi}_c$ using eq.(3)
repeat
  $\pi, \lambda \leftarrow$ Optimal Policy and Optimal Lagrange Multiplier for CMDP$(\mathbf{w}_r, \mathbf{w}_c)$
  Calculate $d_t(s)\ \forall\ t = 1 \ldots, T,\ \forall\ s \in S$ under π using eq.(13)
  Compute $\hat{\Phi}_r$ and $\hat{\Phi}_c$ using eq.(14)
  Compute the gradients $\nabla_{\mathbf{w}_r}\mathcal{O}$ and $\nabla_{\mathbf{w}_c}\mathcal{O}$ using eq.(12)
  $\mathbf{w}_r \leftarrow \mathbf{w}_r \cdot \exp(-\kappa\nabla_{\mathbf{w}_r}\mathcal{O})$
  $\mathbf{w}_c \leftarrow \mathbf{w}_c \cdot \exp(-\kappa\nabla_{\mathbf{w}_c}\mathcal{O})$
  $\mathbf{w}_r \leftarrow \arg\max_{\mathbf{w}}\ D_{KL}(\mathbf{w}||\mathbf{w}_r)$ s.t $\sum_i[\mathbf{w}]_i = 1, \mathbf{w} \geq 0$
  $\mathbf{w}_c \leftarrow \arg\max_{\mathbf{w}}\ D_{KL}(\mathbf{w}||\mathbf{w}_c)$ s.t $\sum_i[\mathbf{w}]_i = 1,\ \mathbf{w} \geq 0,\ \mathbf{w}^T\hat{\Phi}_c \leq 1$
until Convergence
Return $\mathbf{w}_r\ \ \mathbf{w}_c,\ \pi$

4 Results

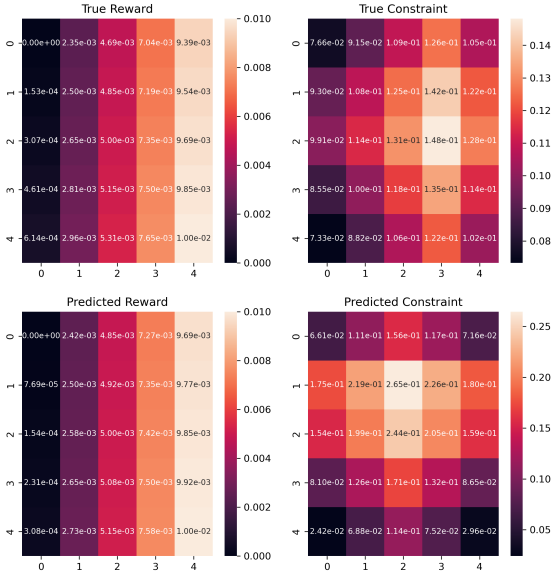
In this section, we present the results of our proposed method on several gridworld simulations. For the purpose of testing, we consider the 5×5 grid. The agent always starts from the top left corner of the grid. At any time step, the agent has four actions available: up, down, left, right. On taking an action, with 70% probability, the agent moves to the next grid in the direction of chosen action, while with 30% probability, it randomly moves into any of the neighbouring grids. The reward distribution is kept the same over all the experiments while the constraint function is changed. The trajectory length was allowed to be 2000. The comparison between the learned reward and constraint functions is shown in fig.2. We also compare the learned policy from the learned reward and constraint with the true policy computed from the true reward and constraint, as shown in fig.3.



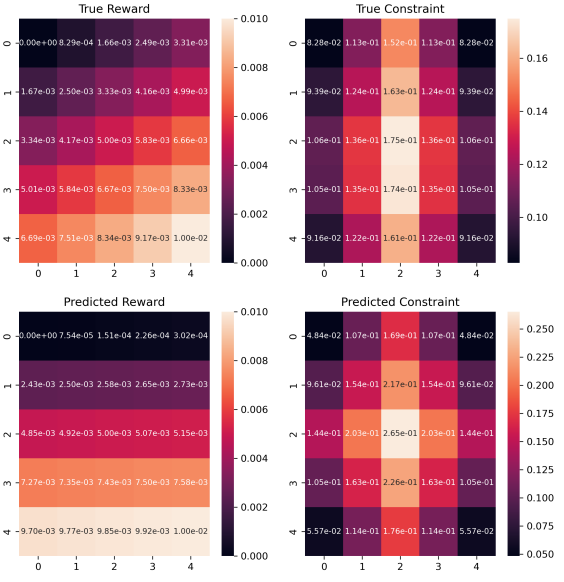
(a) Simulation 1



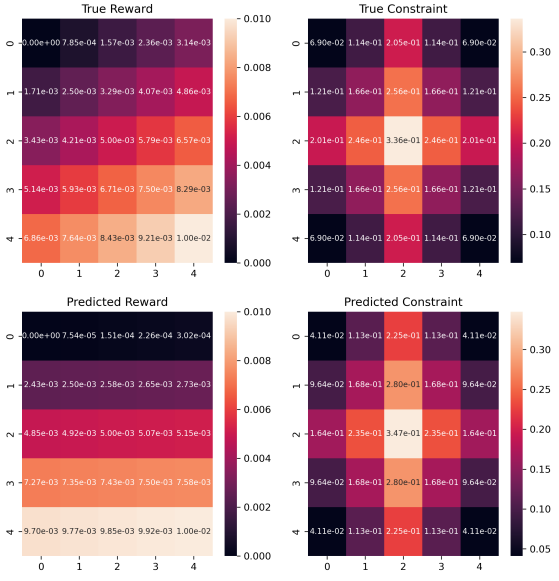
(b) Simulation 2



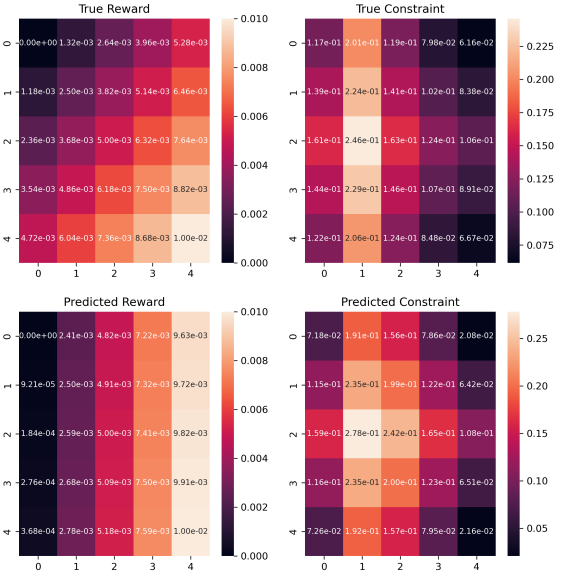
(c) Simulation 3



(d) Simulation 4

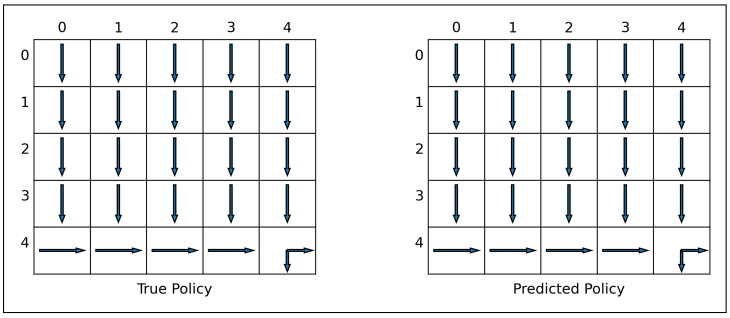


(e) Simulation 5

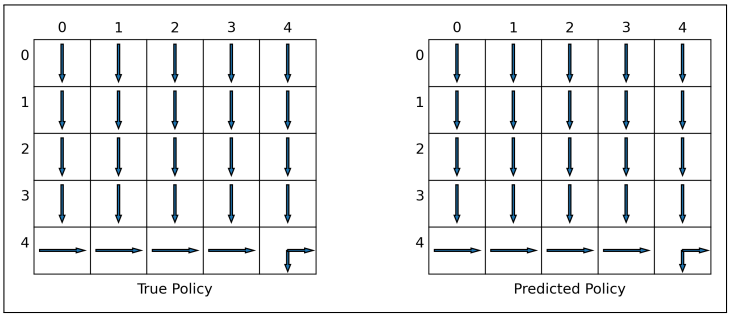


(f) Simulation 6

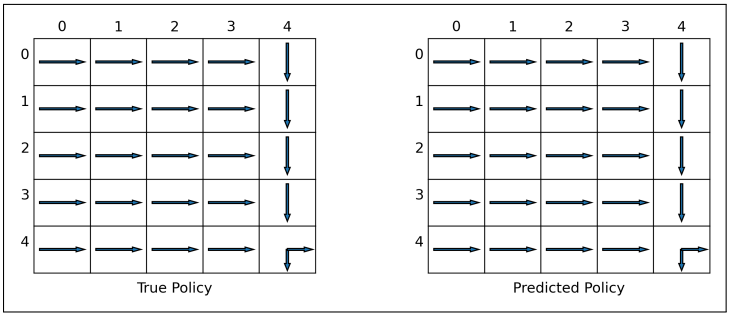
Figure 2: For every subfigure, the top-left square is the true reward, top-right is the true constraint, bottom-left is the predicted reward and the bottom-right is the predicted constraint



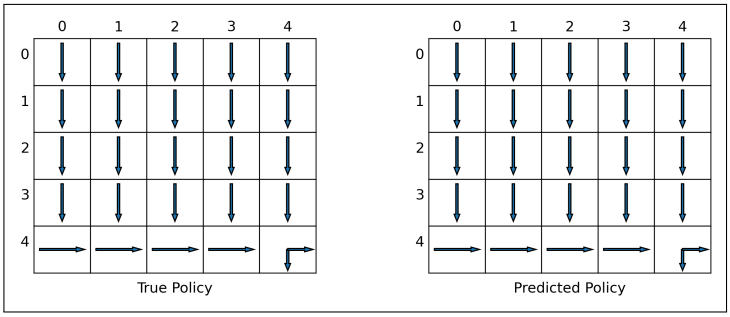
(a) Simulation 1



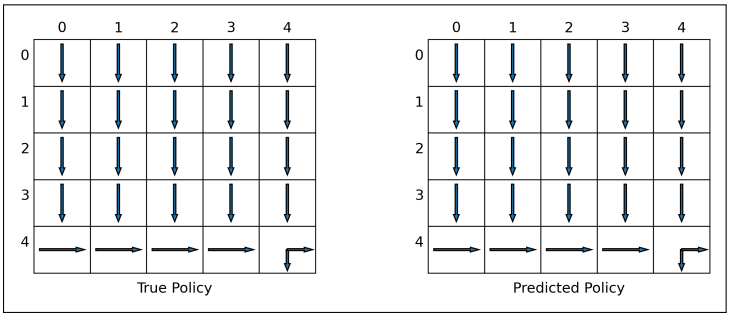
(b) Simulation 2



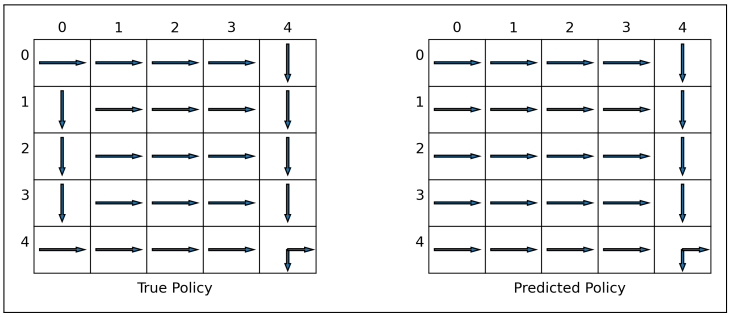
(c) Simulation 3



(d) Simulation 4



(e) Simulation 5



(f) Simulation 6

Figure 3: For every subfigure, left square is the true policy and the right is the predicted policy

5 Discussion

Through extensive simulation, we see that the recovered policy is the same as or very close to the true policy. It has been observed that as the length of the trajectories is increased, the accuracy of the recovered policy increases. This is probably because, in CMDP, the stationary optimal policy corresponds to an infinite length trajectory. Since we generate our data using the stationary policy, it is justified that the length of the trajectory should be high for the algorithm to perform well. Moreover, we see that the reward function, and especially the constraint is recovered well. In fig.2, the darker colour corresponds to smaller values and brighter colour corresponds to higher values. The constraint function has peak in all the simulations by design but the location of the peak and the slope of the decrease varies in the simulations. However, our algorithm is able to capture the varying location of the peak. Although the plot of the reward function looks a bit different, the values recovered for the reward function follows the same trend as the true reward function (increasing from left to right and from top to bottom). We speculate that the recovered reward function has the least bias while also matching the data (hence the policy), but the true reward function may have some extra bias. However, the constraint functions do not suffer from the same problem, making it possible to recover the constraint function with better accuracy.

6 Future Work

There are several possible future steps. This work focuses on the stationary offline setting, in which the reward and the constraint functions are not changing over time, and the trajectory data is available in a batch. One immediate future step is to extend the framework to the following setting:

1. Stationary Online
2. Non-stationary Offline
3. Non-stationary Online

Further, the applicability of this method needs to be tested on real-life data so as to ascertain the utility of our method. Moreover, theoretical guarantees need to be derived in terms of convergence and regret of the algorithm.

7 Conclusion

In this work, we explore a novel method of recovering the reward and the constraint functions simultaneously from the trajectory demonstrations of an optimal policy of a CMDP. There have been no prior work in this direction, thus there is a lack of a previous baseline to compare the performance of our work. However, we demonstrate strong results on a qualitative basis. We hope that expanding our present work to provide more robust and quantitative results will support the performance of our proposed method. This work can be of great relevance to the development of autonomous agents that can learn from demonstrations in a more robust and meaningful way.

8 Acknowledgement

I hereby acknowledge that this work would not have been possible without the guidance and support of my advisor Prof. Arpan Chattopadhyay. I also acknowledge the support in terms of computational needs received from HPC, IIT Delhi.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery.

[2] Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[3] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[4] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7390–7399. PMLR, 18–24 Jul 2021.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[6] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[7] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[8] Dexter R.R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations*, 2020.

[9] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[10] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991.

[11] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[12] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.