STA303 Report

Group 13

January 2024

1 Introduction

Reinforcement Learning (RL) is used to describe and solve the problem of agents using learning policy to maximize returns or achieve specific goals during their interaction with the environment. Some complex reinforcement learning algorithms have intelligence to solve complex problems to a certain extent, and can reach or even exceed human level in some aspects. However, RL involves extensive interaction with the environment to acquire an optimal strategy, and defining the environment's reward function is often a challenging task. Learning from expert demonstrations can help alleviate these shortcomings.

Inverse Reinforcement Learning (IRL), is an important research method in the fields of reinforcement learning and imitation learning. This method solves the reward function through expert samples and solves the optimal strategy based on the reward function obtained, in order to achieve the goal of imitating expert strategies.

Recent optimization efforts in IRL have primarily centered on fine-tuning parameters across a broader spectrum, encompassing techniques such as Maximum Entropy Inverse Reinforcement Learning (e.g., [5]), Deep Inverse Reinforcement Learning (e.g., [4]), and Stochastic Inverse Reinforcement Learning (e.g., [1]). Despite these advancements, exploration through regression methods struggles to achieve a holistic global perspective.

Bayesian Inverse Reinforcement Learning (BIRL[3]) impressively navigates the reward space through multiple learning iterations. Our research delves into the effectiveness of BIRL in scenarios with non-informative priors, questioning whether its success arises from centralization. Additionally, we investigate the potential implementation of pre-training in IRL, exploring new hypotheses within this domain.

2 Background

2.1 Markov Decision Processes

A Markov Decision Process(MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where:

- $\mathcal{S}$ is a finite set of states.
- $\mathcal{A}$ is a finite set of actions.
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor, $\gamma \in [0, 1]$

The reward received at time-step t is denoted as R_t. And the reward received at state s is denoted as $R(s)$. The return G_t is the total discounted reward from time-step t. This

is written as $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. A policy π is a distribution over actions given states, $\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$

The state-value function $V_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π. This is defined as

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid s)[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_\pi(s')], \end{aligned} \tag{1}$$

where $\mathbb{E}_\pi[\cdot]$ represent the expectation with probability π. Similarly, the action-value function $Q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π. This is defined as

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \\ &= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_\pi(s', a'). \end{aligned} \tag{2}$$

The optimal state-value function is defined as $V^*(s) \triangleq \max_\pi V_\pi(s)$, and the optimal action-value function is defined as $Q^*(s, a) \triangleq \max_\pi Q_\pi(s, a)$. The optimal policy π^* can be found by maximizing over $Q^*(s, a)$. The optimal state-value function and action-value function can be found using

$$V^*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V^*(s') \tag{3}$$

and

$$Q^*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} Q^*(s', a') \tag{4}$$

2.2 Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning, as the name states, does exactly the inverse of Reinforcement Learning. Let $\mathcal{M} \backslash \mathcal{R}$ represent an MDP with an unknown reward function. Given a set of demonstrations $\mathcal{D} = \{\tau_i\}_i$ of an expert, consisting of trajectories $\tau = ((s_1, a_1), (s_2, a_2), \cdots, s_{|\tau|})$ through the state-action space where $|\tau|$ denotes the length of the trajectories (visited states), IRL aims to recover the underlying reward function which explains the behavior of the expert. The expert policy is defined as π^E.

To allow for computation with states, some people use so-called features. Feature function is defined as $\phi : \mathcal{S} \to \mathbb{R}^d$, mapping the state set $\mathcal{S}$ to a d-dimensional real number space $\mathbb{R}^d$. To facilitate the calculation of rewards on the entire trajectory, the feature function on the entire trajectory is defined as $\phi(\tau) = \sum_{t=1}^{|\tau|} \phi(s_t)$.

2.3 IRL Algorithms

This section introduces three IRL algorithms. The first is the Maximum Entropy IRL algorithm, the second is Deep Maximum Entropy IRL, and the third is Bayesian IRL.

2.3.1 Maximum Entropy IRL

The trajectory of the policy π^L implied by the recovery reward should to be the same as the trajectory based on the expert policy π^E. In equations, we can express this as the expectation of features over trajectories derived from the reward we want to recover, and

$$\mathbb{E}_{\pi^E}[\phi(\tau)] = \mathbb{E}_{\pi^L}[\phi(\tau)] \tag{5}$$

We can also express the expected feature visitation frequency as sum over trajectories

$$\mathbb{E}_{\pi^L}[\phi(\tau)] = \sum_{\tau} p_{\pi^L}(\tau)\phi(\tau) \tag{6}$$

with the probability distribution $p_{\pi^L}(\tau)$ of trajectories of the learner. There may be more than one reward function found through this method. To find the most likely one, someone has proposed that the reward function can be selected by maximizing the entropy.

The entropy with a probability distribution of p is defined as

$$H(p) = -\sum_{x} p(x) \log p(x) \tag{7}$$

Then the problem can be transformed into a constrained optimization problem

$$\begin{aligned} \arg\max_{p} \quad & H(p) \\ \text{subject to} \quad & \mathbb{E}_{\pi^L}[\phi(\tau)] = \mathbb{E}_{\pi^E}[\phi(\tau)], \\ & \sum_{\tau} p(\tau) = 1, \quad \forall \tau : p(\tau) > 0 \end{aligned} \tag{8}$$

This problem indicates that maximum entropy IRL method want to find the probability distribution p of the trajectory τ that satisfies the condition $\mathbb{E}_{\pi^L}[\phi(\tau)] = \mathbb{E}_{\pi^E}[\phi(\tau)]$ and has the maximum entropy.

To calculate the reward function on the state set $\mathcal{S}$ and trajectory τ, the feature function ϕ has been defined earlier. Now, it is assumed that there is a linear relationship between the reward function and the feature function. The reward function on the entire trajectory τ is defined as $R(\tau) \triangleq \omega^\top \phi(\tau)$ with $\omega \in \mathbb{R}^d$ being the reward parameter vector.

Solving the above problem using the method of Lagrange multipliers leads to $p(\tau) \propto \exp(R(\tau))$. If all actions in the MDP have predetermined outcomes, it will be represented as

$$p(\tau \mid \omega) = \frac{1}{Z(\omega)} \exp(\omega^\top \phi(\tau)) \tag{9}$$

with $Z(\omega) = \sum_\tau \exp(\omega^\top \phi(\tau))$. For MDPs with stochastic transitions, the transition probability is included and the probability of observing a set of trajectories is

$$p(\tau \mid \omega) \approx \frac{1}{Z(\omega)} \exp\left(\omega^\top \phi(\tau)\right) \prod_{s_{t+1}, a_t, s_t \in \tau} \mathcal{P}^{a_t}_{s_t s_{t+1}} \tag{10}$$

The reward weights are learned from the demonstrated behavior by maximizing the log likelihood $\mathcal{L}(\omega) = \sum_\tau \log(p(\tau \mid \omega))$ under the entropy distribution

$$\omega^* = \arg\max_{\omega} \mathcal{L}(\omega) = \arg\max_{\omega} \sum_{\tau} \log(p(\tau \mid \omega)) \tag{11}$$

Gradient descent is used to maximize the likelihood and the gradient is

$$\begin{aligned}\nabla\mathcal{L}(\omega) &= \mathbb{E}_{\pi^E}\left[\phi(\tau)\right] - \sum_{\tau} p(\tau \mid \omega)\ \phi(\tau) \\ &= \mathbb{E}_{\pi^E}\left[\phi(\tau)\right] - \sum_{s_i} D_{s_i}\phi(s_i)\end{aligned} \tag{12}$$

where D_{s_i} is the state-visitation frequency. Note that this gradient holds for both deterministic and stochastic transitions. Then the optimal reward at state s is

$$R^*(s) = {\omega^*}^\top \phi(s) \tag{13}$$

2.3.2 Deep Maximum Entropy IRL

This approach was put up and tested by Wulfmeier et al. [4] In previous research on Maximum Entropy method, we know that the Reward function can be expressed as linear combination of basis function $\phi(x)$, which is of the form $R(s) = \alpha\phi(x)$. However the linear combination can not accurately approximate any reward function with feature vectors in the given basis ϕ. In deep Maximum Entropy method, we can approximate the basis functions as non-linearly transformed linear functions as below:

$$R(s) = \alpha \cdot \sigma(\mathbf{W} \cdot \phi(s)) \tag{14}$$

where σ is a non-linear function(Sigmoid, as an instance).This concept can be extended to arbitrary depth n, formulating a neural network structure.

$$R(s) = \alpha \cdot \varphi_n(s) \tag{15}$$

where

$$\varphi_n(s) = \sigma\left(\mathbf{W}_n \cdot \varphi_{n-1}(s)\right) \tag{16}$$

With this structure and proper values for $\mathbf{W}_1, ..., \mathbf{W}_n$ any continuous feature map can be approximate and so are the Reward functions.

2.3.3 Bayesian IRL

There may be multiple reward functions that satisfy the conditions for a given expert strategy, so a probability distribution can be used to represent this uncertainty. Modeling the IRL problem from a Bayesian perspective. Use expert examples as evidence to infer the parameters of the reward function. The reward function obtained through this reasoning is a distribution, which is different from the previous point estimation, and there is no need to assume a certain structure (such as linear) for the reward function.

For expert trajectory $\tau = ((s_1, a_1), \cdots, (s_T, a_T))$, the probability of assuming that each state-action pair is independent of each other can be expressed as

$$\mathbb{P}(\tau \mid R) = \mathbb{P}((s_1, a_1) \mid R) \cdot \mathbb{P}((s_2, a_2) \mid R) \cdots \mathbb{P}((s_T, a_T) \mid R) \tag{17}$$

Maximizing cumulative rewards is equivalent to finding the action with the highest Q^* value in each state, then the likelihood of (s_t, a_t) conditioned on R can be expressed as an exponential distribution about Q^*

$$\mathbb{P}((s_t, a_t) \mid R) = \frac{1}{Z_t} e^{\alpha Q^*_R(s_t, a_t)} \tag{18}$$

where α is a parameter that represents the degree of confidence, and Z_t is the normalizing constant. The likelihood of the entire trajectory is written as

$$\mathbb{P}(\tau \mid R) = \frac{1}{Z} e^{\alpha \mathbb{E}_R(\tau)} \tag{19}$$

where $\mathbb{E}_R(\tau) = \sum_{t=1}^{T} Q_R^*(s_t, a_t)$. Using Bayes theorem, the posterior distribution of R can be written as

$$\mathbb{P}(R \mid \tau) = \frac{1}{Z'} e^{\alpha \mathbb{E}_R(\tau)} \mathbb{P}(R) \tag{20}$$

where Z' is the normalizing constant.

3 Existing Works

The trajectories in Inverse Reinforcement Learning (IRL) lack a singular solution. The suggested policy could be optimal for various reward functions, causing the regression to become ensnared in a local minimum. This injects uncertainty into the regression model, impeding its progress towards converging with the proposed reward function.

Contemporary implementations of Inverse Reinforcement Learning (IRL) predominantly revolve around fine-tuning parameters across a wider spectrum, as evidenced in [1]. Despite these efforts, regression-based exploration fails to achieve a holistic global perspective. On the contrary, Bayesian Inverse Reinforcement Learning (BIRL) demonstrates remarkable performance in reward space, leveraging multiple learning iterations. While highly effective, BIRL typically relies on a clear prior on the distribution of reward function, which is often unavailable in real-life scenarios. Our investigation seeks to evaluate the efficacy of BIRL in the absence of a well-defined prior for reward distribution and seeks to apply batch training on it.

Further, the current success of pretrained models like CLIP in image classification and ChatGPT in natural language processing has sparked our curiosity about extending pretraining methodologies to Inverse Reinforcement Learning (IRL). Our second investigation aims to explore the potential application of pretraining in accelerating IRL while enhancing its capacity for high-level generalization.

4 Hypotheses

1. Investigating the Efficacy of Bayesian Inverse Reinforcement Learning without a Clear Prior and Batch Training (Ensemble Inverse Reinforcement Learning)

2. Applying pretraining methodologies on Inverse Reinforcement Learning (IRL) to expedite its learning process.

We conducted two sets of experiments to validate our hypothesis.

5 Experiments 1

In this section, our goal is to test several hypotheses regarding Bayesian Inverse Reinforcement Learning without a clear prior of how reward function distributed. These include assessing

whether its performance relies on the centralization facilitated by Bayesian techniques and determining whether Bayesian methods remain effective when learners are provided with varying expert trajectories.

5.1 Objectworld and Trajectories

5.1.1 Objectworld

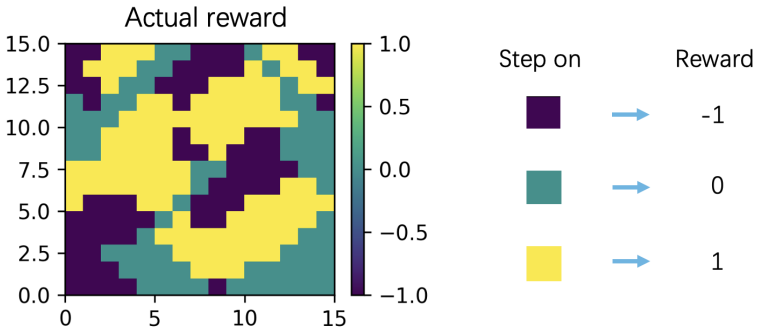


Figure 1: Objectworld map and reward

The objectworld is a structured environment tailored for addressing Inverse Reinforcement Learning (IRL) challenges. It represents an N × N grid layout, randomly housing colored objects within its cells. Each object comprises an inner and an outer color, both selected from a predetermined set of C colors. Movement within this grid is governed by actions such as stepping up, down, left, right, or remaining in place, with a 30% probability of moving randomly.

The reward function employed in our environment is straightforward, associating the colors black, green, and yellow with rewards of -1, 0, and 1, respectively.

A rough estimate of accuracy of the regression can be estimated by the image similarity.

5.1.2 Expert Trajectories

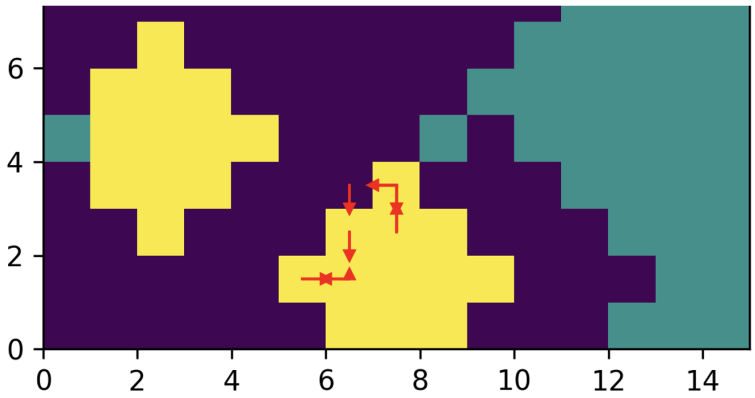


Figure 2: An expert trajectory

Experts are randomly positioned on the map and follow the optimal strategy, traversing the terrain multiple times. However, to diversify exploration, there's a 30% chance of random movement during their traversal. Each expert trajectory is generated once their movement length meets the predefined setting. Generating additional expert trajectories not only reveals more insights into the policy but also divulges additional information about the reward structure. Inverse reinforcement learning aims to deduce the reward function by analyzing the behaviors observed in these expert trajectories.

In the centralization test, all learners receive identical expert trajectories. Conversely, in the distinct expert trajectories test, each learner receives independently generated trajectories while being trained with the same policy.

5.2 Evaluation Procedure

We initiated experiments to validate the efficacy of non-informative priors of BIRL using a Uniform distribution in the objectworld setting. This was followed by an exploration of centralization's potential influence on their effectiveness.

The environment featured a 15×15 objectworld with 25 colored blocks in 2 colors, coupled with a 0.9 discount factor. We generated 40 expert trajectories, each spanning 8 grid-size lengths, to establish the optimal policy. Our approach relied on Deep Maxent Inverse Reinforcement Learning (DeepMaxent) as the foundational BIRL algorithm, structured with 2 layers of 3×3 matrices. All experiments were capped at 250 epochs.

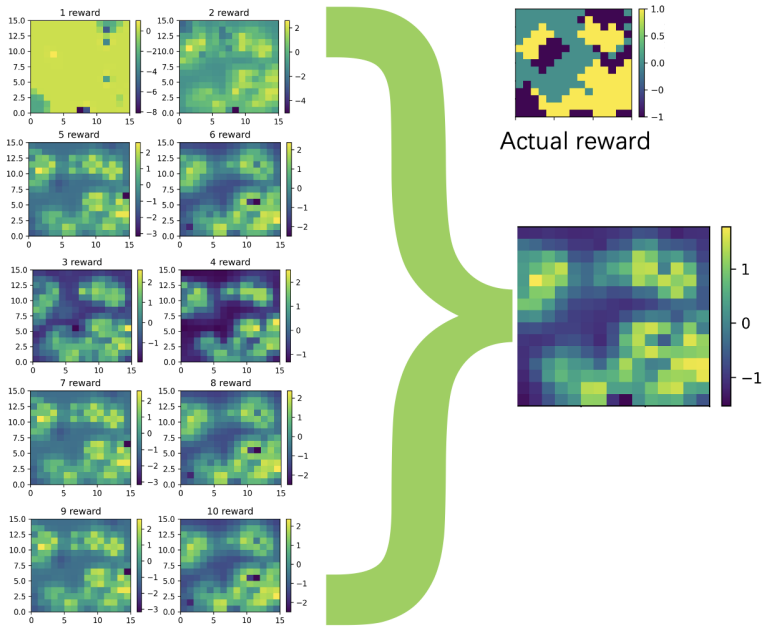


Figure 3: AR-IRL demo

5.2.1 AR-IRL

We trained 10 independent learners, labeled as M_i for i = 1 to 10, using DeepMaxent Inverse Reinforcement Learning within the objectworld environment. The resulting reward matrices R_i, i = 1 to 10, showcased diverse outcomes influenced by distinct probability distributions, leading to varied regression directions. Aggregating and averaging these outcomes yielded the BIRL outcome R^* (see Figure 3), representing an estimated value in the absence of a clear prior. This entire process, termed **AR-IRL**, signifies a non-informative application of BIRL on DeepMaxent IRL.

$$R^* = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{21}$$

After evaluating AR-IRL, we delved into testing our hypothesis about its effectiveness. Specifically, we investigated whether its efficacy is rooted in centralization. Additionally, we explored the utility of AR-IRL using learners formed with batched different expert trajectories. A detailed account of these experiments will be provided in subsequent sections.

5.3 Experiment Design and Findings

5.3.1 The performance evaluation of AR-IRL on the DeepMaxent test.

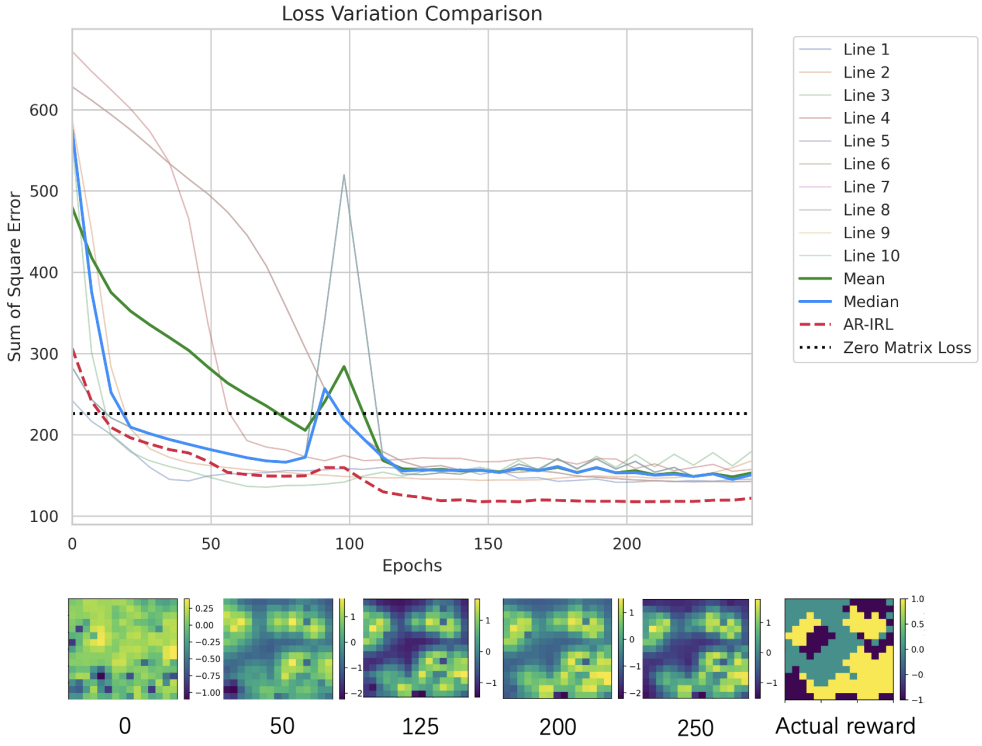


Figure 4: The outcome obtained by AR-IRL

The 10 independent Deep Maxent IRL learners were designated as 'Line 1' to 'Line 10', with their respective mean and median loss denoted as 'Mean' and 'Median'. AR-IRL, an implementation of BIRL using a non-informative prior with a Uniform distribution on Deep Maxent IRL, was evaluated. The 'Zero Matrix Loss' represents the loss derived from a matrix entirely composed of zeros, serving as a baseline comparison.

The comparison of loss variations between AR-IRL and a basic DeepMaxent IRL, using a Uniform distribution as non-informative, revealed significant improvements. AR-IRL demonstrated substantial enhancements over DeepMaxent. Across over 30 experiments, AR-IRL outperformed DeepMaxent IRL by 23.89% in comparison to its mean and 17.27% in comparison to its median. These findings highlight a noteworthy improvement in the results achieved by AR-IRL.

Next, our exploration aims to ascertain whether the efficacy of AR-IRL is primarily derived from its centralization aspect.

5.3.2 Assessment of Centralization

The standard process of IRL often induces a shift in reward values towards an imbalanced range due to the trajectory selection focusing solely on the differences between blocks. For example, while the actual reward may range from -1 to 1, the regression might span a broader range, such as 0 to 4. This shift impacts both the scale and centralization, leading to an increased Sum of Square error compared to the actual reward.

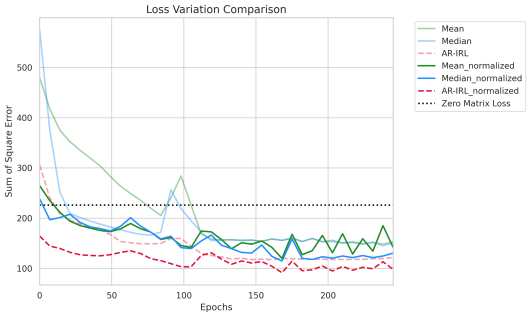
The AR-IRL methodology involves an averaging process that naturally moderates extreme values, gradually shifting overall values towards a more centralized space, thus shaping a consolidated final reward distribution. As the volume of reward data increases, there's a tendency for the probability distribution to further centralize these rewards. It's important to note that this process might not significantly alter the function itself with individual rewards.

Investigating the potential influence of centralization on its enhanced performance, we conducted a normalization procedure. This procedure entailed mapping the reward functions represented by R onto a standardized scale denoted as R', within the range of -1 to 1, which matches the range of the actual rewards. The objective was to transition the reward values from lower to higher ranges, aligning them with the actual reward distributions. Notably, in our approach, the AR-IRL methodology normalized the computed rewards after processing to ensure fairness in the comparisons.

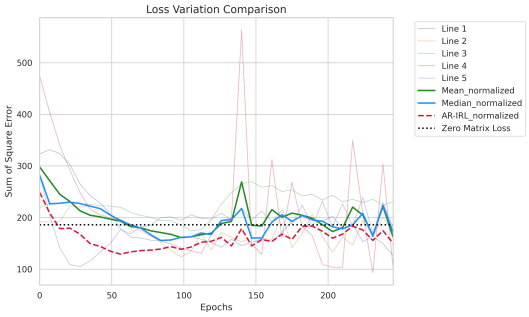
$$x'_{i,j} \in R'$$

$$x'_{i,j} = \frac{2(x_{i,j} - \min(R))}{\max(R) - \min(R')} - 1, \quad x_{i,j} \in R$$

Figure 5(a) illustrates both the normalized and unnormalized outcomes of this process.



(a) Comparative Analysis of AR-IRL Results: Normalized vs. Unnormalized



(b) Comparison of Normalized Results in AR-IRL with 5 Learners

Figure 5: Centralization test

The normalized line indicates outcomes derived from separate independent processes carried out in the same objectworld environment. The original non-normalized values appear blurred or less defined in comparison.

Upon normalization, both AR-IRL and DeepMaxent exhibited improved performance, as illustrated in Figure 5(a). Remarkably, AR-IRL showcased approximately 19.22% improvement post-normalization, while DeepMaxent displayed mean and median improvements of 7.23% and 11.81%, respectively. Surprisingly, the improvement gap between AR-IRL and DeepMaxent widened from 20.44% before normalization to 30.72% after, indicating that the efficacy of AR-IRL might not be contingent on centralization.

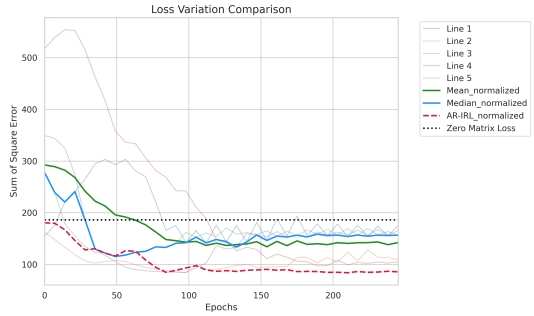
To delve deeper into this feature, a compelling question arises: Can AR-IRL uphold its robust performance even with a decreased number of trajectories, potentially diminishing the centralization feature?

We decreased the number of learners to 5, consequently impacting the quality of centralization in BIRL.(see Figure 5(b)) Despite this alteration, following normalization, AR-IRL continued to exhibit improvement although the gap is narrowing.

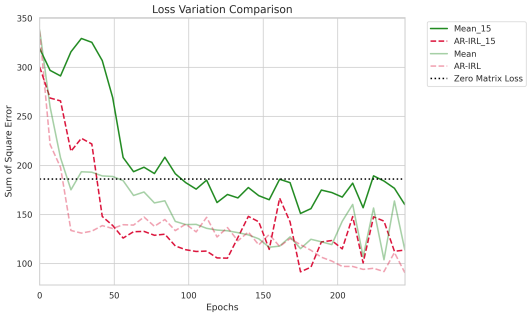
Observing AR-IRL's success in both the normalization and reduced learner tests, we are inclined to conclude that BIRL's optimization might not be contingent on centralization.

Our subsequent investigation seeks to determine whether offering distinct expert trajectories to individual learners continues to yield improved performance.

5.3.3 The assessment of BIRL with batched expert trajectories.



(a) Batched Learner with independent expert trajectories AR-IRL Result



(b) Comparative Analysis: Batched vs. Full-Dataset AR-IRL Results

Figure 6: Centralization test

In panel (a), each learner was provided with distinct expert trajectories, while in panel (b), Mean_15 and AR-IRL_15 represent the outcomes from the batched expert trajectories for learners.

The conventional method in BIRL often involves iteratively learning the same expert trajectories multiple times, leveraging the randomness within the IRL algorithm to reach an expected global optimal value. However, in real-world applications, expert trajectories might be collected in batches. Resetting the previous results and using the entire updated dataset for multiple training iterations can be resource-intensive. We aim to investigate whether independently training learners on individual batches and subsequently employing AR-IRL to ensemble them can yield considerable results.

To achieve a relative balanced overall quantity of expert trajectories, we reduced both the number of learners to 5 and the number of expert trajectories for each learner to 20 in Figure 6(a). This adjustment aimed to balance the overall quantity of expert trajectories. The AR-IRL approach, leveraging optimization from multiple directions, still yielded a significant improvement in performance compare with each batch's DeepMaxent learner. What about comparing with the Full-Data set learner AR-IRL?

We employed a batch size of 15, encompassing a total of 75 distinct expert trajectories. The choice of batch size is pivotal; if it's too small, regression may suffer, rendering the result unreliable. In Figure 6(b), a degree of instability is evident in the regression of batched AR-IRL compared to Full-Dataset AR-IRL. However, despite this instability, the reward loss remains significant within the observed 250 epochs. The resource-saving advantages offered by batched AR-IRL make it a practical solution for real-world applications dealing with batched data in IRL scenarios.

6 Experiments 2

We aim to train a neural network model using randomly generated reward functions and their corresponding expert trajectories. This approach is geared towards obtaining a pre-trained Inverse Reinforcement Learning model.

6.1 Setup

- **Data** The training and testing data are derived through value iteration approach. The feature is policy and label is reward function. Value iteration algorithm relies on knowledge of the transition probabilities. It estimates the value function for each state; the policy in any given state is then whichever action is expected to lead to the highest-valued state. Details of value iteration algorithm are shown in the appendix. Firstly, reward function is randomly generated using `Random.random()` in Python in the form of flattened 1-D array. Length of the array is the number of states s, and each value stand for its respective reward. Then policy is a 1-D flattened array derived from value iteration method with length of 100. Each value of the array align with a probability, which indicate how likely agent would move to the other state.

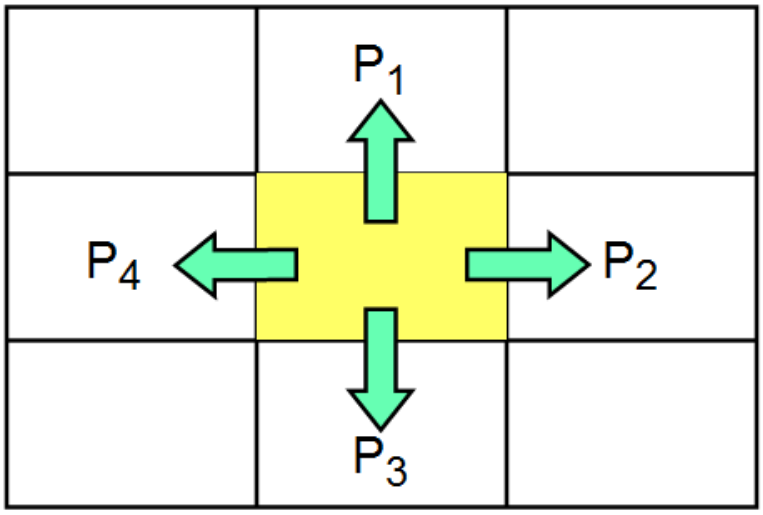


Figure 7: Each grid stands for a state. Its respective policy includes the four probabilities P_i that agent toward four directions

- **Model** The model is a neural network consist of three linear layers. Activation function is ReLu. It was trained using the data generated in the prior phase.
- **Environment** The experiment is conducted on the gridworld environment[2]. Gridworld is a $n \times n$ grid environment that each grid represents a state. It is a powerful tool to clearly demonstrate reinforcement learning process.by four walls There are four primitive actions: up, down, right, and left. In each motion, there is 70% of chance that move towards the named direction while of 30% of chance the agent will move randomly.
- **Evaluation Metric** The reward function is illustrated in the form of 5×5 gridworld. In each grid there is a respective reward value $R(s_i)$. Our evaluation metric is mean square error between labels and inferences:

$$\sum_{s \in \mathcal{S}} [R(s) - R(s)_{inference}]^2$$

6.2 Results

From the experiment we formulate numerical evaluation of our model. It is clear that Max Entropy method outperformed our model. While the model consumed far less time (less than 1e-4 of the max entropy approach)to gave an inference.

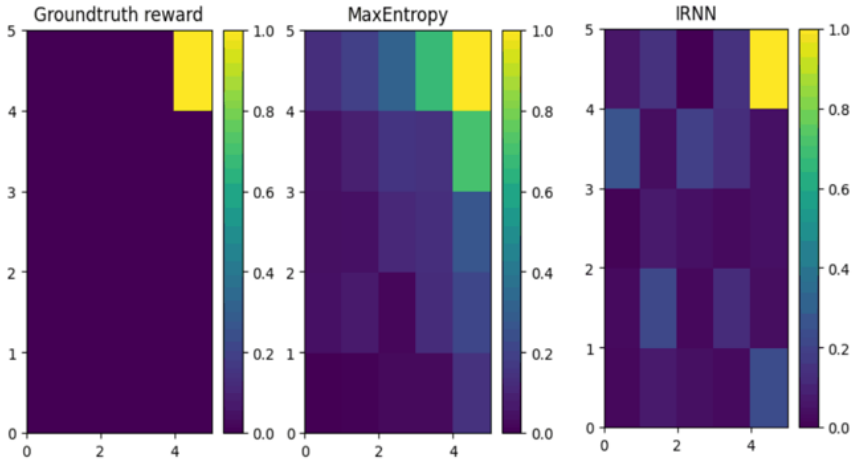


Figure 8: Visualized outcome of three reward functions

Method	Mean square error	time (second)
Max Entropy	3.2	17.0789
Neural Network(3 layers)	4.8	0.0005
Neural Network(5 layers)	4.6	0.0014

Table 1: Comparison between Max Entropy approach and neural networks

7 Conclusion

In our primary hypothesis, we applied Bayesian Inverse Reinforcement Learning (BIRL) without a distinct prior in DeepMaxent IRL, referred to as AR-IRL, resulting in a significant global enhancement in outcomes. Remarkably, our analysis suggests that this improvement cannot be solely attributed to centralization. Furthermore, AR-IRL showcased its ability to synthesize information from diverse learners exposed to varying expert trajectories within an objectworld scenario. This feature holds promise in addressing real-world challenges encountered in batched Inverse Reinforcement Learning, potentially leading to substantial advancements towards achieving optimal global solutions.

In our second hypothesis, we assume that neural network can accelerate the learning progress and enhance generalization ability. Through our results we find that applying neural network thoroughly on the progress can greatly reduce the time cost of inference. However, the precision that our neural network can perform is not as high as the precision of original approach. Our model is trained over 2000 observations of policy-reward pairs and so it is more capable of handling the general problems in this environment.

8 Team contribution

- **Guo Cheng 31%**: Execution, analysis, and presentation of Experiment 1. The organization of the overall project structure.
- **Ye Yanshun 22%**: Inverse Reinforcement Learning literature review.
- **Ma Junhao 22%**: Conduction and analysis of experiment 2.
- **Li Yibo 25%**: Conduction and analysis of experiment 2.

9 Appendix

We release the code we used for experiment 1 at https://github.com/Iwan000/SUSTECH-AI-B-Final.

The code of experiment 2 is available in https://github.com/hasqdwuduw/IRL-NN

Algorithm 1: Value iteration

Input: $\mathcal{P}^a_{ss'}, R(s), \mathcal{S}, \mathcal{A}, \gamma$
foreach $s \in \mathcal{S}$ **do**
  $V(s) \leftarrow 0$
end
$\Delta \leftarrow \infty$;
while $\Delta > \epsilon$ **do**
  $\Delta \leftarrow 0$;
  foreach $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$;
    $V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}(R(s') + \gamma V(s'))$;
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
  end
end
foreach $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}(R(s') + \gamma V(s'))$;
end
return π

Figure 9: value iteration algorithm

References

[1] Ce Ju and Dong Eui Chang. Stochastic inverse reinforcement learning. *CoRR*, abs/1905.08513, 2019.

[2] Amy McGovern and Richard S Sutton. Macro-actions in reinforcement learning: An empirical analysis. *Computer Science Department Faculty Publication Series*, page 15, 1998.

[3] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. University of Illinois at Urbana-Champaign, Computer Science Department, Urbana, IL 61801, Year, if available.

[4] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

[5] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.