Estimation of Reward Function Maximizing Learning Efficiency in Inverse Reinforcement Learning

Yuki Kitazato and Sachiyo Arai
Graduate School & Faculty of Engineering, Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba, 263-8522, Japan

Keywords: Inverse Reinforcement Learning, Genetic Algorithm.

Abstract: Inverse Reinforcement Learning (IRL) is a promising framework for estimating a reward function given the behavior of an expert.However, the IRL problem is ill-posed because infinitely many reward functions can be consistent with the expert's observed behavior. To resolve this issue, IRL algorithms have been proposed to determine alternative choices of the reward function that reproduce the behavior of the expert, but these algorithms do not consider the learning efficiency. In this paper, we propose a new formulation and algorithm for IRL to estimate the reward function that maximizes the learning efficiency. This new formulation is an extension of an existing IRL algorithm, and we introduce a genetic algorithm approach to solve the new reward function. We show the effectiveness of our approach by comparing the performance of our proposed method against existing algorithms.

1 INTRODUCTION

Reinforcement Learning (Richard S. Sutton, 2000) is a framework for determining a function that maximizes a numerical reward signal through trial and error. Reinforcement learning is widely used because it only requires a scalar reward to be set for the target state, without requiring a teacher signal to be specified. On the other hand, in practice, the path leading to the target state, that is, the "desired behavior sequence", is often available. Applications that available the full path, such as the guidance loci in evacuation guidance and desired trajectories during road journeys and parking, are increasing, and reward designs that require these present a problem when using reinforcement learning.

One reward design method is inverse reinforcement learning. Inverse reinforcement learning was proposed by Russell (Andrew Y. Ng, 1999) as a framework for estimating the reward function given the desired behavior sequence to be learned by the agent, or an available environmental model (state transition probability).

Inverse reinforcement learning algorithms provide multiple reward functions, and thus the user needs to select from among them (Ng and Russell, 2000). Previous research has focused on the reproducibility of the desired behavior sequence and evaluated the obtained reward functions (Ng and Russell, 2000; Pieter Abbeel, 2004; U. Syed and Schapire, 2008; Neu and Szepesvari, 2007; B. D. Ziebart and Dey, 2008; M. Babes-Vroman and Littman, 2011). However, the evaluation criteria for the reward function are not explicitly incorporated into the framework of inverse reinforcement learning. Therefore, in this paper, we propose an inverse reinforcement learning method which introduces the learning efficiency and selects the reward function that maximizes this efficiency.

Specifically, we extend the method of Ng et al. (Ng and Russell, 2000) which poses inverse reinforcement learning as a linear programming problem and introduce an objective function that maximizes the learning efficiency. From the two viewpoints of reproducibility of optimal behavior and learning efficiency, we show the usefulness of the proposed method by comparison with existing methods.

In Section 2, we outline the concepts and definitions related to inverse reinforcement learning that are necessary to read this paper, and we discuss the problem with existing methods in Section 3. In Section 4, we propose an algorithm and solution for the inverse reinforcement learning problem to improve the learning efficiency, and perform computer experiments in Section 5. Section 6 considers the results of the experiment, and Section 7 concludes with a summary and areas for future work.

Kitazato, Y. and Arai, S.
Estimation of Reward Function Maximizing Learning Efficiency in Inverse Reinforcement Learning.
DOI: 10.5220/0006729502760283
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 2*, pages 276-283
ISBN: 978-989-758-275-2

2 PRELIMINARIES

In this section, we explain the Markov decision process model, the basic theory of reinforcement learning, and the methods of Ng (Ng and Russell, 2000) and Abbeel(Pieter Abbeel, 2004), which are representative of the current state of inverse reinforcement learning methods.

2.1 Markov Decision Process

The Markov decision process (MDP) model can be used to represent an agent's behavior as a series of state transitions.

The finite Markov decision process consists of the tuple $\langle\, \mathcal{S},\, \mathcal{A},\, P^a_{ss'},\, \gamma,\, R\,\rangle$. Here, $\mathcal{S}$ is a finite state set, $\mathcal{A}$ is an action set, and $P^a_{ss'}$ is the state transition probability for transitioning to next state s' when performing action a in state s. Then, γ is the discount rate, and $R : \mathcal{S} \to \mathbb{R}$ represents a reward function that returns the reward r when transitioning to the state $s \in \mathcal{S}$.

2.2 Reinforcement Learning

Reinforcement learning (Richard S. Sutton, 2000) is a method for identifying the optimal policy without requiring an environmental model such as the state transition probability. The agent learns from a scalar reward without requiring a teacher signal to indicate the correct output. In general, agents learn to maximize the expected reward.

In this study, we use the tuple $\langle\, \mathcal{S},\, \mathcal{A},\, R,\, \pi\,\rangle$ to model the agent's behavior. Here the policy π is the probability of selecting action a in state s. The agent observes state $s_t \in \mathcal{S}$ at time t and selects action $a_t \in \mathcal{A}$ based on policy π_t. Next, at time $(t+1)$, the agent transitions to state s_{t+1} stochastically according to s_t, a_t and obtains the reward r_t. We generate the value function $V(s)$ or the action-value function $Q(s,a)$ from the earned reward, and evaluate and improve the policy π based on these values. The value function $V(s)$ and the action-value function $Q(s,a)$, given policy π, respectively satisfy the following:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^{\pi}(s'), \tag{1}$$

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s') V^{\pi}(s'). \tag{2}$$

2.3 Inverse reinforcement Learning Method of Ng

Ng et al. (Ng and Russell, 2000) presented a method for estimating the reward function from the optimal behavior from the MDP by posing the problem as a linear programming problem, as follows:

maximize

$$\sum_{i=1}^{M} \min_{a \in a_2,\ldots,a_k} \{(\boldsymbol{P}_{a_1}(i) - \boldsymbol{P}_a(i))\ \ (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R}\} - \lambda ||\boldsymbol{R}||_1 \tag{3}$$

subject to

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \geq 0 \quad \forall a \in A \backslash a_1. \tag{4}$$

Here, the state transition matrix $\boldsymbol{P}_a$ is an $M \times M$ matrix whose elements are (i,j) and the components of the state transition probability are $P_{ia}(j)$. The vector $P_a(i)$ represents the ith row vector of $\boldsymbol{P}_a$, M is the total number of states, and λ is a penalty factor, which is a parameter that adjusts the reward.

The constraint presented in Eq. (4) guarantees that the Q value derived from the optimal policy is larger than those for other policies. The derivation of this constraint is shown in the following equations (Eq. (5) to Eq. (9)). Since the optimal behavior a_1 maximizes Q value in each state,

$$a_1 \equiv \pi(s) \in \arg\max_{a \in A} R(s) + \sum_{s'} P_{sa}(s') V^{\pi}(s') \quad \forall s \in S. \tag{5}$$

In each state, the Q value of optimal action a_1 is greater than the Q value of other actions, implying (from Eq. (2))

$$\sum_{s'} P_{sa_1}(s') V^{\pi}(s') \geq \sum_{s'} P_{sa}(s') V^{\pi}(s') \quad \forall s \in S, a \in A. \tag{6}$$

By defining the state value function vector $\boldsymbol{V}^{\pi} = \{V^{\pi}(s_i)\}_{i=0}^{M}$ and the reward function vector $\boldsymbol{R} = \{R(s_i)\}_{i=0}^{M}$, we can rewrite Eq. (6) and Eq. (1) as respectively

$$\boldsymbol{P}_{a_1} \boldsymbol{V}^{\pi} \geq \boldsymbol{P}_a \boldsymbol{V}^{\pi}, \tag{7}$$

$$\boldsymbol{V}^{\pi} = \boldsymbol{R} + \gamma \boldsymbol{P}_{a_1} \boldsymbol{V}^{\pi}. \tag{8}$$

By solving equation Eq. (8) for $\boldsymbol{V}^{\$}$ and plugging the value into Eq. (7), we can derive the following constraint:

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \geq 0, \tag{9}$$

The first term of the objective function in Eq. (3) shows maximization of the differences in the Q values derived from the optimal policy and the second most optimal policy, which is equivalent to Eq. (10) below. The second term is based on the idea that the total reward is minimized and a simple reward function is preferable to a complicated reward function.

$$\sum_s (Q^\pi(s,a_1) - \max_{a \in A \backslash a_1} Q^\pi(s,a)) \tag{10}$$

2.4 Inverse Reinforcement Learning Method of Abbeel

Abbeel et al. (Pieter Abbeel, 2004) proposed an algorithm for estimating the reward function R in the feature space from the behavior of an expert. The state space $\mathcal{S}$ can be expressed by the feature vector $\phi : \mathcal{S} \to [0,1]^p$ with p features as elements. The reward function vector $\boldsymbol{R}(s)$ given state $s \in \mathcal{S}$ is

$$\boldsymbol{R}(s) = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(s),\ \boldsymbol{\theta} \in \mathbb{R}^p, \tag{11}$$

where $\boldsymbol{\theta}$ represents the weight of the feature and its domain is $||\boldsymbol{\theta}||_2 \leq 1$ to ensure the maximum value of the reward is less than or equal to 1. The expected value of the feature observed under policy π is called the feature expectation value $\boldsymbol{\mu}(\pi)$ and is defined as follows:

$$\boldsymbol{\mu}(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(s_t)|\pi] \in \mathbb{R}^p. \tag{12}$$

Given U and expert behavior $\{s_0^u, s_1^u, \ldots\}_{u=1}^U$, the expert feature expectation value $\boldsymbol{\mu}_E = \boldsymbol{\mu}(\pi_E)$ is

$$\boldsymbol{\mu}_E = \frac{1}{U} \sum_{u=1}^{U} \sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(s_t^u). \tag{13}$$

Abbeel's method computes the weight vector $\boldsymbol{\theta}$ that minimizes the error between the feature expected value of expert $\boldsymbol{\mu}_E$ and the estimated feature value $\boldsymbol{\mu}(\pi)$, using the algorithm of Fig. 1, called Projection Method $\boldsymbol{\mu}(\pi)$.

3 PROBLEM FORMULATION

In this section, we show that different reward functions can result in the same policy. First, we summarize the results of several studies on methods for determining an appropriate reward function and discuss problems with these methods. Then, we define the problem that is the subject of this paper.

```
Compute $\mu^0 = \pi^{(0)}$
Set $\theta^1 = \mu_E - \mu^0, i = 1$
Repeat (until $t^{(i)} \leq \varepsilon$)
Compute $\pi^{(i)}$ using the RL algorithm and
rewards $R = (w^{(i)})^T \phi$
Compute $\mu^{(i)} = \mu(\pi^{(i)})$
Set $i = i + 1$
Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} +$
$\frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$
Set $\theta^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$
Set $t^{(i)} = ||\mu_E - \bar{\mu}^{(i-1)}||_2$
```

Figure 1: Algorithm of projection method.

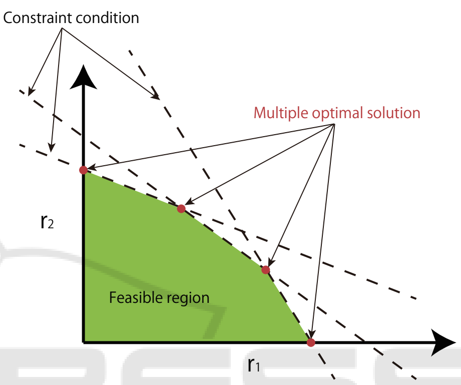


Figure 2: Existence of multiple reward functions.

3.1 Existence of Multiple Reward Functions

In 2.3, Ng et al. showed that the reward function leading to the optimal policy must satisfy constraint Eq. (9). An intuitive explanation of how multiple reward functions can satisfy this constraint is given in Fig. 2. Fig. 2 show a simple example for $|\mathcal{S}| = 2$. The vertical and horizontal axes show the rewards for states 1 and 2, respectively.

All of the reward functions in the shaded feasible region are consistent with the optimal policy. That is, there can be multiple reward functions satisfying constraint Eq. (9). In addition to satisfying the constraint, inverse reinforcement learning can be extended to consider the optimization of the reward function by introducing an additional objective function. Thus, finding the reward function that leads to the optimal policy and finding the optimal reward function are two separate problems. This paper deals with the latter problem.

3.2 Related Work: Objective Function in Reward Optimization

As described in the previous section, there are multiple reward functions that lead to the optimal policy. Therefore when considering an actual problem, it is necessary to unify on a single reward function. In previous research, the determination of the objective function is based solely on its ability to reproduce the optimal behavior. We summarize objective functions for inverse reinforcement learning that are representative of these studies below.

Ng et al. consider methods that determine the reward function which maximizes Q value. By using this objective function, actions other than the optimal action in each state have a lower expected reward than the optimal action. Therefore, those actions are less likely to be chosen.

The optimization of Abbeel et al. (Pieter Abbeel, 2004) is

$$\min ||\mu_\theta - \mu_E||, \tag{14}$$

which minimizes the difference between the feature expectation value μ_θ learned by the estimated reward function and the expert feature expectation value μ_E. The feature expectation value includes the elements of the transition state and time, and realizes the same behavior as the expert by selecting the weight whose feature expectation values coincide with those of the expert.

The objective function of Syed et al. (U. Syed and Schapire, 2008) was proposed to improve the performance of Abbeel 's method, as follows:

$$\min \theta(\mu_\theta - \mu_E), \tag{15}$$

Since the objective function $\theta \cdot \mu$, the product of the weight and the feature expected value, can be regarded as the average expected reward, we estimate the weight so that it becomes the average expected reward value equivalent to that of the expert. In particular, rather than taking the difference between the norm of the feature expectation values and using it as the norm of the expectation reward, we make a more accurate estimation.

Neu et al. (Neu and Szepesvari, 2007) considered the optimization

$$\min \sum_{s \in S, a \in A} \nu_E(s)(\pi(a|s) - \pi_E(a|s))^2, \tag{16}$$

where frequency of occurrence of each state $s \in S$ is $\pi(a|s)$, and $\nu_E(s) = 1/n \cdot \sum_{t=s)}^{N}$ represents the corresponding probability of taking action a. The optimization directly approximates an expert's policy.

Ziebart et al. (B. D. Ziebart and Dey, 2008) and Babes et al. (M. Babes-Vroman and Littman, 2011) considered the following optimization:

$$\max \sum_{\xi \in D} \log P(\xi|\theta), \tag{17}$$

where $\xi \in D$ represents one exercise action sequence, D represents a set of action sequences, and $P(\xi|\theta) = 1/Z(\theta)e^{\theta\mu}$ is the probability distribution of action sequence ξ given weight θ. Let $P_E(\xi)$ be the probability distribution of the expert's action sequence and $P(\xi|\theta)$ be the probability distribution of the action sequence for the estimated reward. Then the distribution of the KL information for the experts and estimated rewards can be expressed as

$$\sum_{\xi \in D} P_E(\xi) \log P_E(\xi)/P(\xi|\theta), \tag{18}$$

and so the likelihood function can be expressed as

$$\min \; -\sum_{\xi \in D} P_E(\xi) \log P(\xi|\theta) \tag{19}$$

When the amount of data is sufficient, the estimate of the objective function converges to Eq. (17) by the law of large numbers. Eq. (17) minimizes the difference between the probability distribution of the expert's action sequence and the probability distribution of the action sequence of the estimated reward. Thus, by including this comparison between these probability distributions, this method can be applied even if some inaccuracies exist in the expert's behavior sequence.

Although the above objective functions differ, they estimate the reward that increase reproducibility.

3.3 Definition of Reproducibility and Learning Efficiency

Reproducibility is not considered in previous studies, as they do not consider the determination of a unique objective function for the reward as shown in 3.2. In this paper, we define reproducibility and learning efficiency as objective functions for evaluating the reward function.

Reproducibility. We define the reproducibility as the concordance rate between the optimal action a_1 of a given policy and the action of the estimated policy $\pi_{\mathrm{IRL}}(s)$ by inverse reinforcement learning. Let $f(s)$ be a function over $\mathcal{S}$ that returns 1 when the action in the estimated policy is the same as the action for the given policy, and 0 otherwise:

$$f(s) = \begin{cases} 1 & if \max_a \pi_{\mathrm{IRL}}(s) = a_1, \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

If M is the total number of states, then the reproducibility is expressed using Eq. (20) as

$$\frac{1}{M}\sum_{s\in S} f(s). \tag{21}$$

Learning Efficiency. The learning efficiency is defined as the sum of the number of steps for all episodes. Here we define one step as the minimum time required to make a decision and one episode as the total time required to reach the target state. Specifically, the learning efficiency is

$$\sum_{i=1}^{N} H_i(R), \tag{22}$$

where R is the the reward function, the number of steps required for each episode i is $H_i(R)$, and the final episode is N.

4 PROPOSED METHOD: REWARD FUNCTION FOR MAXIMIZING LEARNING EFFICIENCY

In this section, we propose a method for estimating the reward function for maximizing the learning efficiency as defined in Section 3.3. First, we reformulate the inverse reinforcement learning method proposed by Ng et al. to maximize the learning efficiency. Then, we introduce a genetic algorithm (GA) in order to solve the formulated problem efficiently.

4.1 Formulation of Inverse Reinforcement Learning to Maximize Learning Efficiency

The formulation of Ng et al. (Eq. (3)) consists of objective functions and constraints. The objective functions increase the reproducibility and the constraints guarantee that a given policy is feasible. Since the purpose of this paper is to maximize learning efficiency, we introduce an objective function for this. In particular, we introduce Eq. (23) below, which has an objective function chosen in order to minimize the the learning efficiency given in Eq. (22) up to N_{early} episodes.

$$\min \sum_{n=1}^{N_{\mathrm{early}}} H_n(R) \tag{23}$$

Only the initial episode is used to calculate the objective function to suppress unnecessary search steps at the early learning stage and to prevent a decrease in the learning efficiency differences for each reward function due to an increase in the number of episodes.

4.2 GA Approach

In solving the proposed formulation, we have two issues. The first issue is that the derivative of the objective function can not be analytically determined. The second issue is that the objective function of the proposed method is not a convex function but a multimodal function. In the gradient method which is a well-known nonlinear optimization method is not appropriate for these issues. For the first issue, since the differential value can not be analytically obtained, numerical differentiation is required and the calculation cost becomes enormous. For the second issue, since it is a multimodal function, the gradient method converges to a local solution that depends on the initial value with high probability.

Therefore, we introduce GA to solve these issues. GAs are metaheuristics that do not require assumptions such as differentiability and multimodality of their objective functions. Therefore, GAs can be applied to various problems due to their flexibility.

The algorithm for the proposed method is shown as Algorithm 1. We set the l th individual of I^k in the kth generation population to $ind_l^k \in I$ $(l = 0, ..., L)$, where each individual consists of two elements: $ind = \{\boldsymbol{ch}, fitness\}$. $\boldsymbol{ch} = \{R(s_i)\}_{i=0}^{M}$ is a gene vector that consists of the reward value of all states, and $fitness \in \mathbb{R}$ represents the adaptive degrees.

Since the GA cannot directly handle constraints, we derive the following method to generate an individual that satisfies the constraints. Each individual evolves over the following two phases.

Phase 1. Search for solutions that satisfy all constraints

Phase 2. Search for solutions to improve the learning efficiency

For **Phase 1**, we apply a penalty that is proportional to the number of individuals that violated the constraints. The individuals that satisfy all constraints

Algorithm 1: Proposed algorithm.

```
 1: ▷Initialization
 2: $k \leftarrow 0$                                   ▷ generation
 3: for $l = 0$ to $L$ do
 4:     for $i = 0$ to $M$ do
 5:         $ind_l^k.ch_i \leftarrow$ Random$(1, -1)$
 6:     end for
 7:     $ind_l^k.fitness \leftarrow$ maxvalue
 8: end for
 9: ▷Main Loop
10: for $k = 1$ to $K$ do
11:     for $l = 1$ to $L$ do
12:         ▷ Phase 1 : Satisfy constraint
13:         if $Eq.$ (9) $< 0$ then            ▷ Violating constraints
14:             $ind_l^k.fitness \leftarrow$ number violating constraints penalty
15:         ▷ Phase 2 : Optimize learning efficiency
16:         else if $Eq.$ (9) $> 0$ then       ▷ Satisfying all constraint
17:             $ind_l^k.fitness \leftarrow$ Eq. (23)
18:         end if
19:     end for
20:     ▷Genetic operation
21:     Tournament Selection$(I^k)$
22:     Crossover$(I^k)$
23:     Mutation$(I^k)$
24: end for
```

go to **Phase 2**, learning efficiency is calculated using the reward function of the individual using Q learning, and the fitness is determined according to the learning efficiency.

An advantage of the two-phase approach is the reduction in the calculation time. That is, functions that do not satisfy the constraints are reward functions that cannot guarantee the acquisition of the optimal behavior. Therefore, by excluding these functions from the search space through **Phase 2**, we can reduce the computation time.

5 EXPERIMENTS

We evaluate the usefulness of the proposed method by comparison with Ng's and Abbeel's methods, which are typical inverse reinforcement learning methods. We use the reproducibility and learning efficiency defined in Section 3.3 to evaluate the performances of the methods. The GridWorld problem is used, as it is widely used to benchmark reinforcement learning methods, and we vary the length of the (square) grid from 5 to 8. This problem consists of finding the shortest path from the start to the goal. As an example, let Fig. 3 denote the GridWorld environment for a 5×5 grid, with start coordinates (0, 0) in the lower left corner, and the goal in the upper right corner (4, 4).

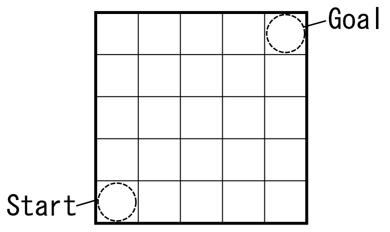


Figure 3: Experimental environment.

5.1 Experiment Setting

The parameters are the range of the reward function, $-1 \leq R \leq 1$, and the penalty coefficient $\lambda = 0$. The genes in the GA are represented by 0's and 1's, and the reward of one state is discretized into 4 bits. The genetic operations are tournament selection, uniform crossover, mutation, and elite selection. The number of generations for the proposed method is 10000, the number of individuals is 100, and the parameter value $N_{\text{early}} = 100$ is used in Eq. (23) to calculate the learning efficiency. The termination condition for the Abbeel inverse reinforcement learning method is set as $\tau = 0.2$. We set each parameter so that the learning efficiency is maximized.

The reproducibility and learning efficiency are evaluated by Q learning with the reward function obtained by each method from Eq. (21) and Eq. (22), respectively. The parameters for the Q learning are as follows: the learning rate is 0.03, the discount rate is 0.9, and the number of episodes is 10000. The action selection for the Q learning is ε-greedy, and ε becomes zero after 9900 episodes.

Here, in Ng's method, the same reward function is obtained even if the trial is repeated due to the use of the linear programming method. However, in Abbeel's method and the proposed method, different reward functions are estimated in each trial. Abbeel's method estimates the feature expectation values by reinforcement learning, and the proposed method estimates the reward function by the GA and calculates the fitness by reinforcement learning. In these methods, randomness is inherent in the estimation of the reward function. Therefore, to fairly evaluate the performance of each algorithm, each method is applied 10 times to each environment, and 10 reward functions are thus obtained. We calculate the average and standard deviation of the reproducibility and learning efficiency for the 100 obtained reward functions, and use the best reward functions among the 10 reward functions for the comparison of the methods. The total number of states M necessary to calculate the reproducibility is the same as the number of squares in GridWorld.

Table 1: Comparison of the reproducibility[%].

	Ng	Abbeel	Proposed method
5×5	65.4 (6.11)	62.2 (3.32)	**92.3** (2.91)
6×6	67.0 (6.37)	51.5 (2.89)	**86.3** (4.08)
7×7	63.3 (6.19)	51.7 (3.68)	**84.4** (3.04)
8×8	64.9 (7.73)	56.9 (2.70)	**87.3** (2.72)

Table 2: Comparison of the efficiency[10^3].

	Ng	Abbeel	Proposed method
5×5	173(63.4)	92.7 (0.125)	**91.2**(0.118)
6×6	1470(23.6)	**115** (46.3)	116 (0.147)
7×7	1410(94.2)	1463 (45.2)	**149** (0.175)
8×8	295(111)	**174** (0.294)	179 (0.185)

5.2 Results

The reproducibility and learning efficiency of each method are shown in Table 1 and Table 2, respectively. Bold in Table 1 and Table 2 indicates the best results.

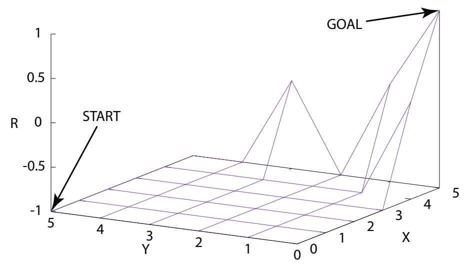
In terms of the reproducibility, the proposed method performs better than the other methods (Table 1). For the learning efficiency in the cases of the 5×5 and 7×7 grids, the proposed method again performs best (see Table 2). However, Abbeel's method is slightly better than the proposed method for the 6×6 and 8×8 grids. The significance of the differences in the reproducibility and the learning efficiency between the proposed method and the standard methods is compared with a t-test at the 5% significance level. The null hypothesis states that the mean and standard deviation of the two methods are equal to those of the proposed method. Since the null hypothesis was rejected, it is confirmed that there is a significant difference between our proposed method and the standard methods.

6 DISCUSSION

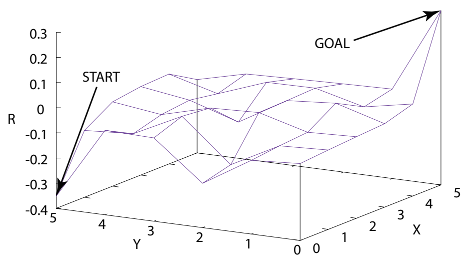
In this section, the reproducibilities and learning efficiencies of the existing and proposed methods are considered based on the results of the experiments.

6.1 Evaluation of Reproducibility

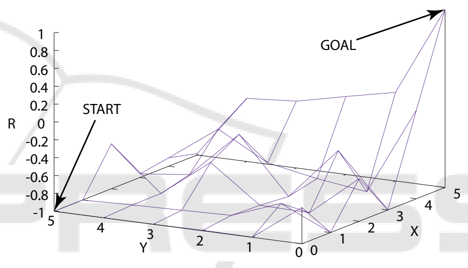
From Table 1, the reproducibility of the proposed method is about 22% and 32% higher than those of Ng's and Abbeel's methods, respectively. We consider the reason for this in terms of the shape of the reward function for the 6×6 GridWorld of each method as shown in Fig. 4.



(a) Ng



(b) Abbeel



(c) Proposed method

Figure 4: Reward function of 6 6 GridWorld.

In the reward function obtained by the method of Ng (Fig. 4 (a)), the reward value is given only to the states close to the target state and the reward value is thus 0 for most states. In states where there are two optimal actions, it is highly likely that actions other than the actions for which the reward is given will be taken, so the reproducibility is low. In the reward function obtained by Abbeel's method (Fig. 4 (b)), the reward value is given only to the state around the optimal route, so the learner takes only the optimal actions near the shortest path. In the reward function obtained by the proposed method (Fig. 4 (c)), a reward value is given for each state, so the reproducibility is relatively high. This is the result of explicitly including the learning efficiency in the objective function. Although the improvement in the learning efficiency is introduced explicitly as the objective function, the reproducibility is reflected only in the constraint. However, compared with the existing methods, it was confirmed that the reproducibility is increased.

Table 3: Comparison of the efficiency of 10 trials [10^3].

	Ng	Abbeel	Proposed method
5×5	173(0)	189 (12.6)	92.0 (0.132)
6×6	1470 (0)	257 (86.7)	120 (0.172)
7×7	1410 (0)	1900 (581)	154 (0.166)
8×8	295 (0)	3500 (2490)	193 (0.178)

6.2 Evaluation of Learning Efficiency

From Table 2, the learning efficiency of the proposed method was high for all GridWorld settings. The reason for this is that the methods of Ng and Abbeel maximize only the reproducibility, whereas the proposed method explicitly considers the learning efficiency. On the other hand, Abbeel's method had a slightly higher efficiency than the proposed method in the 6×6 and 8×8 grids. The reason for this is that, as described in 5.1, the same reward function is estimated every time when repeating trials with Ng's method, but in Abbeel's method and the proposed method, there is an inherent randomness, resulting in different estimates of the reward function with each repetition of the trial. In other words, although Abbeel's method does not explicitly consider learning efficiency, a reward function with a high learning efficiency can be determined by chance over multiple trials.

In order clarify this point, the learning efficiency and standard deviation are calculated for the ten reward functions used in the experiment, and the average value and standard deviations are shown in Table 3.

In Table 3, the standard deviation is 0 because Ng's method provides the same reward function for all 10 trials. On the other hand, for Abbeel's method, the standard deviation is very large compared with other methods, and thus the learning efficiency varies widely between trials. Since the proposed method estimates the reward function using a GA, there is no guarantee that it converges to the optimal solution. Nevertheless, the standard deviation for the proposed method is small, showing that the proposed method can provide stable estimates for the reward function with a high learning efficiency for each trial.

6.3 Limits of the Proposed Method

In GridWorld environments larger than 8×8, the proposed method could not obtain a solution that satisfies all of the restrictions presented in Eq. (9). It is reasonable to suppose that since the executable area is narrowed by increasing the number of constraints and states, it is difficult to search the executable area with a simple penalty method. In particular, a drawback of the GA method is that it cannot explicitly handle constraints.

7 CONCLUSION

In this paper, we proposed an inverse reinforcement learning method for finding the reward function that maximizes the learning efficiency. In our proposed method, an objective function for the learning efficiency is introduced using the framework of the inverse reinforcement learning of Ng et al. Moreover, since our proposed objective function is nonlinear and convex, we find the solution using a GA.

A limitation of the method proposed in this paper is that the GA cannot always converge to an optimal solution when the number of states is too high, so future work will consider the following: 1) relaxation of the constraints, 2) formulation of our method as a linear or quadratic programming problem, and 3) application of a strong GA for finding solutions satisfying the constraints.

REFERENCES

Andrew Y. Ng, Daishi Harada, S. R. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*.

B. D. Ziebart, A. Maas, J. A. B. and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *The AAAI Conference on Artificial Intelligence*.

M. Babes-Vroman, V. Marivate, K. S. and Littman, M. (2011). Apprenticeship learning about multiple intentions. In *International Conference on Machine Learning*.

Neu, G. and Szepesvari, C. (2007). Apprenticeship learning using inverse reinforcement learning and gradient methods. In *annual Conference on Uncertainty in Artificial Intelligence*.

Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*.

Pieter Abbeel, A. Y. N. (2004). Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*.

Richard S. Sutton, A. G. B. (2000). *Reinforcement Learning: An Introduction*. Xko.

U. Syed, M. B. and Schapire, R. E. (2008). Apprenticeship learning using linear programming. In *International Conference on Machine Learning*.