

Integrating AIRL-Based Reward Estimation into a Control Method Using Fuzzy Inference and Policy Gradient

Seiji Ishihara^{1*}

^{1*}School of Science and Engineering, Tokyo Denki University, Saitama, Japan.

Corresponding author(s). E-mail(s): ishihara_s@mail.dendai.ac.jp;

Abstract

A method integrating policy gradient with fuzzy inference has been proposed. This method, hereafter referred to as the base model, has high flexibility with respect to the configuration of fuzzy inference rules and fuzzy representation. It can also automatically adjust the rule weights and the membership functions in the antecedent and consequent parts using the same framework for learning. In methods that integrate reinforcement learning with fuzzy inference, however, the reward function must be designed in advance. It is therefore useful to add learning-based reward estimation to the base model. This paper proposes a fuzzy control method that integrates a reward estimation mechanism based on the adversarial inverse reinforcement learning (AIRL) framework into the base model. The generative adversarial nets (GAN) framework adopted in AIRL, however, tends to suffer from unstable learning. To address this, the proposed method treats the policy of the base model as the generator and also uses the set of membership functions in the policy as the cost function of the discriminator. This configuration allows the discriminator update for reward estimation and the generator update for policy estimation to be performed simultaneously on the parameters shared by both, without estimating a value function. In simulations of an illustrative vehicle speed control task, the proposed method acquired policies with high generalization performance through stable and efficient learning.

Keywords: Fuzzy control, Policy gradient, Adversarial inverse reinforcement learning, Reward estimation, Membership function

Article Highlights

- Fuzzy rule parameters that both decide an action and evaluate it are acquired automatically through learning.
- Sharing the parameters lets both kinds of learning proceed at once, without another function that evaluates an action.
- Desirable speed control was learned stably with far less training than by a conventional alternating learning method.

1 Introduction

Control based on fuzzy inference has high explainability in its process because the knowledge representation is readily understandable by humans. Methods for integrating reinforcement learning into such fuzzy control systems have been studied [3, 7, 10, 11, 13, 14, 18, 19]. These methods automatically adjust the parameters constituting if-then rules over continuous state and action spaces, using rewards designed according to the control task.

Among these is a method [11, 13], hereafter referred to as the base model, using the policy gradient method [2, 12, 20] to learn the weights representing the reliability of each fuzzy inference rule (rule weights) and the parameters that specify the shape of each membership function in the antecedent and consequent parts. This method has high flexibility with respect to rule configuration and fuzzy representation. It can also use the same learning framework for the automatic adjustment of the rule weights and that of the membership functions. In methods that integrate reinforcement learning with fuzzy inference, however, the reward function must be designed in advance.

Meanwhile, studies on imitation learning and inverse reinforcement learning (IRL) have been reported in recent years [1, 21]. These methods acquire a policy equivalent to that of an expert, or an effective reward function, from the state-action trajectories generated by the expert’s strategy for a given task. The approach of formulating the task as an optimization problem based on the maximum entropy principle and solving it by a gradient method [22] can be readily combined with neural networks. Among methods based on this approach, adversarial IRL (AIRL) [6] is efficient in that it updates the policy and the reward alternately and partially, and in that it takes as input the state-action pair at each time step. It also has the advantage of extracting a reward function that does not include environment dynamics. The generative adversarial nets (GAN) framework [9] adopted in AIRL, however, tends to suffer from unstable learning.

In such an integration of reinforcement learning and fuzzy inference, it is useful to add a mechanism that estimates the reward function through learning. This paper therefore proposes a fuzzy control method that integrates a reward estimation mechanism based on the AIRL framework into the base model.

The proposed method does not adopt the AIRL discriminator as it stands. Instead, to make the learning of both the reward function and the policy stable and efficient, it treats the policy of the base model as the generator and also uses the set of membership functions in the policy as the internal function of the discriminator, namely its

cost function¹. This cost function approximates a quantity corresponding to the advantage of an action with respect to the optimal policy, using the set of membership functions rather than a value function. In the learning stage, this allows the discriminator update for reward estimation and the generator update for policy estimation to be performed simultaneously on the parameters shared by both. Since the discriminator and the generator share the parameters of the membership functions and no value function has to be estimated, learning becomes stable and efficient. The proposed method thus forms an integrated structure, in which the membership functions of the policy-gradient-based generator are also applied to the configuration of the discriminator.

Simulations of an illustrative vehicle speed control task show that the proposed method achieves stable and efficient learning and acquires policies with high generalization performance.

The main contributions of this paper are as follows.

1. A fuzzy control method is proposed that integrates a reward estimation mechanism based on the AIRL framework into the base model. The reward function no longer has to be designed in advance.
2. A cost function of the discriminator is defined using the set of membership functions in the policy of the generator. The parameters of the membership functions are thus shared between the generator and the discriminator. Under the optimal policy, the cost function approximates a quantity corresponding to the advantage of an action without a value function.
3. The discriminator update for reward estimation and the generator update for policy estimation are performed simultaneously on the shared parameters, and no value function has to be estimated. Learning thus becomes stable and efficient.

The paper is organized as follows. Section 2 reviews related work on methods integrating reinforcement learning with fuzzy inference and on inverse reinforcement learning. Section 3 describes the base model. Section 4 outlines AIRL. Section 5

¹Since the internal function is not explicitly named in AIRL, this paper borrows the term from GCL [5] and defines the cost function so that a larger value is more desirable.

presents the configuration of the generator and the discriminator in the proposed method, together with their learning rules. Section 6 explains the control task and its application. Section 7 details the computer experiments and their results, and discusses the findings. Section 8 concludes the paper.

2 Related Work

2.1 Methods Integrating Reinforcement Learning with Fuzzy Inference

These integration methods can be broadly divided into two types according to the reinforcement learning they adopt.

Under the assumption of a Markov decision process (MDP) for the environment and the agent’s policy, the first type uses value-based reinforcement learning. Glorennec’s fuzzy Q-learning [7], which combines Q-learning with fuzzy inference, and its extensions [10, 14] belong to this type. Methods of this type generally apply fuzzy representation to the state input variables of the antecedent part, while the parameter values or functions corresponding to the Q-value of each action are given as the output of the consequent part without fuzzy representation. They also have no mechanism for adaptively strengthening or suppressing the rules themselves. One method adds rules as needed according to the matching degree of the antecedent part or the temporal difference (TD) error during reinforcement learning [3], but it does not weight the rules directly.

The second type uses policy-based reinforcement learning, specifically the policy gradient method [2, 12, 20], which learns the policy directly by a gradient method without using a value function. As in the first type, in one method [19] that uses GPOMDP [2], each rule consists of an antecedent part with fuzzy representation and a consequent part that outputs the parameters corresponding to the policy without fuzzy representation. Another method [18] has a consequent part that outputs the matching degree of a fuzzy representation defined by the parameters corresponding to the policy. This method integrates policy gradient methods such as GPOMDP with fuzzy inference. In these methods, however, the membership functions are fixed and the rule weights are not taken into account. The base model [11, 13],

by contrast, has high flexibility with respect to the configuration of fuzzy inference rules: it supports fuzzy representation not only in the antecedent part but also in the consequent part, and it allows rule weights corresponding to the reliability of each rule to be set. It can also be applied as a general-purpose method that uses the same framework for learning to automatically acquire the reliability of each rule and the concept of fuzzy representation based on the shape of each membership function [13].

2.2 Inverse Reinforcement Learning

Inverse reinforcement learning is a type of imitation learning that automatically acquires an expert’s policy from the expert’s trajectories, and it differs from other methods of imitation learning in that it can estimate the reward function explicitly. Multiple types of IRL methods have been studied [1]. Among them, methods combining a gradient method with maximum entropy IRL (ME-IRL) [22], a later approach, have been developed. Finn et al. proposed guided cost learning (GCL) [5], which alternately updates the policy and the cost function and estimates the partition function through sampling. GCL extends the applicability of ME-IRL and improves efficiency. They subsequently showed that GCL can be reformulated within the GAN framework (GAN-GCL) [4]. Fu et al. proposed AIRL [6], which takes as input the state-action pair at each time step rather than an entire trajectory and thereby learns more efficiently. By incorporating a value function into the discriminator, AIRL also estimates a reward function that does not include environment dynamics. In AIRL, the cost function of the discriminator has been shown to be equivalent to the advantage function under the optimal policy. The GAN framework adopted in these methods, however, tends to suffer from unstable learning. Approaches to stabilizing the training of GANs have therefore been explored [8, 15, 17]. For adversarial learning in imitation learning and IRL, a method has also been proposed that constrains the information flow in the discriminator by an information bottleneck to maintain informative gradients [16]. In the context of fuzzy inference, no method has been reported that uses membership functions as the cost function to make learning in the AIRL framework stable and efficient.

3 The Base Model

3.1 Fuzzy Inference Rules

The following fuzzy inference rules are used to control the agent's behavior.

Rule i :

$$\begin{aligned} &\text{if } (x_1 \text{ is } A_1^i) \text{ and } \dots \text{ and } (x_M \text{ is } A_M^i) \\ &\text{then } (y_1 \text{ is } B_1^i) \text{ and } \dots \text{ and } (y_N \text{ is } B_N^i) \\ &\text{with } \eta_i \end{aligned} \quad (1)$$

Here, i ($= 1, 2, \dots, n_R$) is the rule number, x_j ($j = 1, 2, \dots, M$) is an input variable representing the state, and y_k ($k = 1, 2, \dots, N$) is an output variable representing the action. A_j^i and B_k^i are linguistic fuzzy expressions for the variables x_j and y_k in the j -th antecedent part and the k -th consequent part of rule i , respectively. $A_j^i(x_j; \alpha_j^i)$ and $B_k^i(y_k; \beta_k^i)$ are their membership functions, specified by the parameters α_j^i and β_k^i . η_i (≥ 0) is a parameter representing the rule weight.

3.2 Objective Function and Policy

The agent decides its action at each time step so as to minimize the objective function

$$E(y; x) = - \sum_{i=1}^{n_R} \eta_i A^i(x; \alpha^i) B^i(y; \beta^i) \quad (2)$$

Here, the matching degrees of the input value x in the antecedent part and of the output value y in the consequent part of rule i are defined as $A^i(x; \alpha^i) \equiv \prod_{j=1}^M A_j^i(x_j; \alpha_j^i)$ and $B^i(y; \beta^i) \equiv \prod_{k=1}^N B_k^i(y_k; \beta_k^i)$, respectively. The objective function in Eq. (2) represents the total degree of support that the rules give to the output value y for the input value x .

Deciding the output value $y(t)$ for the input value $x(t)$ at time t is the policy of the agent, or learning system. The policy is defined as a probability given by the Boltzmann distribution with temperature T , in which $E(y(t); x(t))$ is used as the energy function.

$$\pi(y(t)|x(t)) \equiv \frac{\exp(-E(y(t); x(t))/T)}{\sum_y \exp(-E(y; x(t))/T)} \quad (3)$$

The policy π is thus stochastic, and T represents the degree of randomness in deciding the action.

3.3 Learning Rule for the Parameters by the Policy Gradient Method

The same reinforcement learning framework applies to all the parameters $\mu \in \{\alpha, \beta, \eta\}$. The policy gradient method gives the following learning rule, which maximizes the expected reward $\mathbb{E}_{\tau \sim \pi}[r(\tau)]$ for a trajectory $\tau = (x(0), y(0), \dots, x(L-1), y(L-1))$ in an episode.

$$\Delta \mu = \varepsilon \mathbb{E}_{\tau \sim \pi} \left[r(\tau) \sum_{t=0}^{L-1} e_\mu(t) \right] \quad (4)$$

Here, L is the episode length and ε (> 0) is the learning coefficient. The characteristic eligibility [20] $e_\mu(t)$ is given by the following expressions [12].

$$e_\mu(t) \equiv \frac{\partial}{\partial \mu} \ln \pi(y(t)|x(t)) \quad (5)$$

$$= -\frac{1}{T} \left\{ \frac{\partial E(y(t); x(t))}{\partial \mu} - \mathbb{E}_{\pi(y|x(t))} \left[\frac{\partial E(y; x(t))}{\partial \mu} \right] \right\} \quad (6)$$

Here, $\mathbb{E}_{\pi(y|x)}[f(y)] \equiv \sum_y f(y) \pi(y|x)$.

4 Adversarial Inverse Reinforcement Learning

AIRL is an IRL method that adopts the GAN framework, which consists of a discriminator and a generator. To obtain the optimal policy, AIRL alternately updates the discriminator and the policy in the generator. Specifically, with the expert trajectories prepared in advance, the process repeats a cycle consisting of generating sample trajectories with the policy of the generator, updating the discriminator, estimating the reward from the updated discriminator, and updating the policy in the generator.

4.1 Discriminator

Let x' denote the state one step ahead, reached from state x by action y . The output of the discriminator defined using the cost function $f_{\theta, \phi}(x, y, x')$ is given by

$$D(x, y, x') = \frac{\exp(f_{\theta, \phi}(x, y, x'))}{\exp(f_{\theta, \phi}(x, y, x')) + \pi(y|x)} \quad (7)$$

The cost function is specified by the parameters θ and ϕ and is decomposed as follows:

$$f_{\theta, \phi}(x, y, x') = g(x, y; \theta) + \kappa h(x'; \phi) - h(x; \phi) \quad (8)$$

Here, g is an approximation of the reward function, h is an approximation of the state value function, and κ is the discount rate. Under the optimal policy, $f_{\theta, \phi}(x, y, x')$ has been shown to be equivalent to the advantage function.

The value of $D(x, y, x')$ corresponds to the probability that the input (x, y, x') is contained in the expert trajectory τ_D , the demonstration. It is also the probability that the input is not contained in the trajectory τ_S , the sample, generated by the actions that the generator outputs following the policy $\pi(y|x)$. Let $\mathcal{D}_t$ denote the set of element triples $(x(t), y(t), x(t+1))$ based at time t in the demonstration τ_D . For correct discrimination between the demonstration and the sample, the parameters of the discriminator are learned so as to minimize the following cross-entropy loss function.

$$\mathcal{L}(\theta, \phi) = \sum_{t=0}^{L-1} \{ -\mathbb{E}_{\mathcal{D}_t} [\ln(D(x, y, x'))] - \mathbb{E}_{\pi_t} [\ln(1 - D(x, y, x'))] \} \quad (9)$$

Here, π_t denotes the distribution of $(x(t), y(t), x(t+1))$ generated by rolling out the policy π . The gradient-based learning rule for the discriminator is given by

$$\Delta\lambda = \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\mathcal{D}_t} \left[(1 - D(x, y, x')) \frac{\partial}{\partial \lambda} f_{\theta, \phi}(x, y, x') \right] - \mathbb{E}_{\pi_t} \left[D(x, y, x') \frac{\partial}{\partial \lambda} f_{\theta, \phi}(x, y, x') \right] \right\} \quad (10)$$

Here, $\lambda \in \{\theta, \phi\}$ and δ is the learning coefficient.

4.2 Generator

The output y of the generator is the action decided following the policy $\pi(y|x)$. The reward $\hat{r}(x, y, x')$ is estimated from the output of the discriminator as follows:

$$\hat{r}(x, y, x') = \ln(D(x, y, x')) - \ln(1 - D(x, y, x')) \quad (11)$$

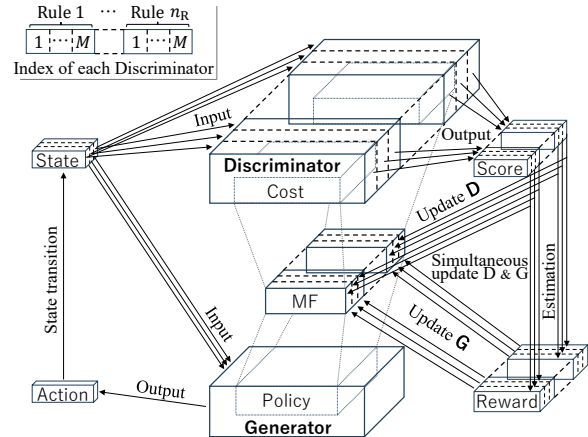


Fig. 1 Configuration of the proposed method

The policy $\pi(y \mid x)$ of the generator is learned with any reinforcement learning method using this estimated reward.

5 Proposed Method

The proposed method integrates the reward estimation mechanism of AIRL presented in Section 4 into the base model presented in Section 3 by sharing the membership functions. Fig. 1 shows the configuration of the proposed method. The proposed method treats the policy of the base model as the generator and also uses the set of membership functions (MF) in the policy as the cost function of the discriminator. Specifically, a cost function is defined using each membership function of the antecedent part, and this gives a discriminator for each fuzzy representation of each rule. From the viewpoint of the j -th fuzzy representation in the antecedent part of rule i , the j -th ($= 1, 2, \dots, M$) discriminator of rule i thus gives a score indicating how likely a trajectory of actions and states is to be contained in the demonstration. The outputs of the discriminators are used for two updates, one minimizing the loss function of the discriminator (Update D) and the other maximizing the expected reward of the generator (Update G), which are performed simultaneously on the parameters of the membership functions of the antecedent part shared by the discriminator and the generator. The following subsections present the proposed cost function and discriminator, together with the learning rules for the discriminator and the generator of the proposed method.

5.1 Cost Function and Discriminator Sharing the Membership Functions

This paper proposes the following cost function defined using each membership function in the antecedent part.

$$f_{\alpha_j^i}(x(t), y(t), x(t+1)) = \ln \left\{ 1 - |v_{\alpha_j^i}(x(t+1); x(t), y(t))| \right\} \quad (12)$$

where

$$v_{\alpha_j^i}(x(t+1); x(t), y(t)) \equiv \mathbb{E}_{\mathcal{D}_t} [A_j^i(x'_j; \alpha_j^i)] - A_j^i(x_j(t+1); \alpha_j^i) \quad (13)$$

Under the assumption that $0 < A_j^i(x_j) < 1$, it follows that $0 \leq |v_{\alpha_j^i}| < 1$ and $f_{\alpha_j^i} \leq 0$. Eqs. (12) and (13) show that the value of $f_{\alpha_j^i}$ approaches zero as the matching degree $A_j^i(x_j(t+1))$ for the input approaches the expected matching degree $\mathbb{E}_{\mathcal{D}_t}[A_j^i(x'_j)]$ for the demonstration. The cost function $f_{\alpha_j^i}$ therefore evaluates the value of $(x(t), y(t))$ by the matching degree for $x_j(t+1)$. Under the optimal policy, $\mathbb{E}_{\pi_t}[A_j^i(x'_j)]$ equals $\mathbb{E}_{\mathcal{D}_t}[A_j^i(x'_j)]$. In this case, the difference $f_{\alpha_j^i}(x(t), y(t), x(t+1)) - \mathbb{E}_{\pi_t}[f_{\alpha_j^i}(x, y, x')]$ between the value of a given $(x(t), y(t))$ and the expected value over all samples generated by π represents the relative value of the action $y(t)$ with respect to the optimal policy π . This difference can therefore be regarded as a form of approximation of the advantage without using a value function. Using the cost function in Eq. (12), the output of the j -th discriminator of rule i is defined as

$$D_j^i(x(t), y(t), x(t+1)) = \frac{\exp(f_{\alpha_j^i}(x(t), y(t), x(t+1)))}{\exp(f_{\alpha_j^i}(x(t), y(t), x(t+1))) + \pi(y(t) | x(t))} \quad (14)$$

5.2 Learning Rule for the Discriminator

The loss function of the discriminator in the proposed method is

$$\mathcal{L}(\alpha_j^i) = \sum_{t=0}^{L-1} \left\{ -\mathbb{E}_{\mathcal{D}_t} [\ln(D_j^i(x, y, x'))] - \mathbb{E}_{\pi_t} [\ln(1 - D_j^i(x, y, x'))] \right\} \quad (15)$$

In updating the discriminator, the parameters of the policy π of the generator at time t are fixed, and $\pi(y(t) | x(t))$ in Eq. (14) is treated as a constant. The learning rule that minimizes this loss function is then given by

$$\Delta \alpha_j^i = \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\mathcal{D}_t} \left[(1 - D_j^i(x, y, x')) \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] - \mathbb{E}_{\pi_t} \left[D_j^i(x, y, x') \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right\} \quad (16)$$

The gradient used for this minimization is thus only that of the cost function $f_{\alpha_j^i}$. Even though the policy π of the generator shares the parameters with the discriminator, it is treated as a constant in the learning rule for the discriminator. The derivation of Eq. (16) is given in Appendix A.

5.3 Learning Rule for the Generator

The evaluation function of the generator in the proposed method, which the policy gradient method maximizes, is defined as

$$J(\mu) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \gamma^t \hat{r}(x(t), y(t), x(t+1)) \right] \quad (17)$$

where

$$\hat{r}(x, y, x') = \begin{cases} \hat{r}_j^i(x, y, x') & \text{if } \mu = \alpha_j^i \\ \sum_{j=1}^M \hat{r}_j^i(x, y, x') & \text{if } \mu \in \{\beta_k^i, \eta_i\} \end{cases} \quad (18)$$

$$\hat{r}_j^i(x, y, x') = \ln(D_j^i(x, y, x')) - \ln(1 - D_j^i(x, y, x')) \quad (19)$$

Here, γ is the discount rate. In updating the generator, the parameters of the discriminators D_j^i at time t are fixed, and $\hat{r}(x(t), y(t), x(t+1))$ in Eq. (17) is treated as a constant. The learning rule that maximizes this evaluation function is then given by

$$\Delta \mu = \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=q}^{L-1} \gamma^{u-q} \hat{r}(x(u), y(u), x(u+1)) \right\} e_{\mu}(t) \right] \quad (20)$$

Here, $q = 0$ corresponds to Eq. (4) and $q = t$ corresponds to the REINFORCE learning rule. The gradient used for this maximization is thus only the characteristic eligibility e_μ . Even though the reward $\hat{r}$ is given by the outputs of the discriminators D_j^i , which share the parameters with the generator, it is treated as a constant in the learning rule for the generator.

With the learning rule in Eq. (20), the parameters of the membership functions in the consequent part and of the rule weights can also be learned, in addition to those in the antecedent part. In the following discussion, however, only the parameters α_j^i of the membership functions in the antecedent part shared with the discriminator are considered as the learning target in the generator. The characteristic eligibility $e_{\alpha_j^i}(t)$ for the update $\Delta\alpha_j^i$ is given by

$$\begin{aligned} e_{\alpha_j^i}(t) &= -\frac{1}{T} \eta_i \{ \mathbb{E}_{\pi(y|x(t))} [B^i(y)] - B^i(y(t)) \} \frac{\partial}{\partial \alpha_j^i} A^i(x(t)) \end{aligned} \quad (21)$$

The derivation of Eq. (21) is given in Appendix B.

5.4 Simultaneous Learning

In the proposed method, the discriminator and the generator share the parameters, and α_j^i is updated by Eqs. (16) and (20) simultaneously. A single update with one set of samples therefore covers both the discriminator and the reward estimation for the generator, just as one alternating cycle does in AIRL. In this case, the amount of a single update of α_j^i is given by

$$\begin{aligned} \Delta\alpha_j^i &= \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\mathcal{D}_t} \left[(1 - D_j^i(x, y, x')) \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right. \\ &\quad \left. - \mathbb{E}_{\pi_t} \left[D_j^i(x, y, x') \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right\} \\ &+ \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=q}^{L-1} \gamma^{u-q} \hat{r}(x(u), y(u), x(u+1)) \right\} e_{\alpha_j^i}(t) \right] \end{aligned} \quad (22)$$

In Eq. (22), the first term on the right-hand side corresponds to an update that increases the

difference in the output of the membership function A_j^i between the demonstration and the sample, while the second term corresponds to an update that decreases it. If the update of the generator can bring the shapes of the membership functions close to a constant function, the output of the cost function easily becomes large regardless of the state of the sample, and a policy that generates trajectories similar to the demonstration does not necessarily have to be learned in the generator. Because the first term keeps the membership function away from such a constant-like shape, the second term can update the membership function so that the policy generates samples close to the states of the demonstration.

6 Control Task and Its Application

This section describes a simulation problem [13] that assumes an illustrative vehicle speed control task, and the application of the proposed method to this problem.

6.1 Vehicle Speed Control Problem

Two vehicles are assumed to travel in the same direction along a one-dimensional straight line representing a road, with the speed of the leading vehicle v_{front} (km/h) held constant. The task of this problem is to control the speed of the following vehicle x_v (km/h) by accelerating and braking, so that the inter-vehicle distance x_d (m) enters the target range $[l_{\text{near}}, l_{\text{far}}]$ (m) within 80 seconds from the initial state at $t = 0$ (sec) and then remains within that range until $t = 110$ (sec).

With the time discretized at $\Delta t = 1$ (sec), the speed control of the following vehicle starts at $t = 0$ and ends when $t = 110$ is reached. The length of one episode is therefore $L = 110$. However, if the inter-vehicle distance becomes less than 0 m, which means a rear-end collision, or reaches 400 m or more, the control is terminated at that point and the episode is regarded as ended.

6.2 Fuzzy Inference Rules

The fuzzy inference rules for controlling the speed of the following vehicle are given by Eq. (1) with $M = 2$ and $N = 1$. The input variables in the antecedent part are $\mathbf{x}_1 = (x_d, l_{\text{near}}, l_{\text{far}})$ and $\mathbf{x}_2 = (x_v, v_{\text{front}})$. The output variable y_1 in the consequent part corresponds to the accelerator input

($0 \leq y_1 \leq 5$) and the brake input ($-5 \leq y_1 \leq 0$), and the resulting acceleration of the following vehicle is $2y_1$ (km/h/sec).

The fuzzy sets A_1^{long} and A_1^{short} correspond to fuzzy representations of the inter-vehicle distance, long and short, respectively. The fuzzy sets A_2^{fast} and A_2^{slow} correspond to fuzzy representations of the speed of the following vehicle, fast and slow, respectively. The fuzzy sets $B_1^{\text{high-ac.}}$, $B_1^{\text{low-ac.}}$, $B_1^{\text{high-br.}}$, $B_1^{\text{low-br.}}$, and $B_1^{\text{no-op.}}$ correspond to fuzzy representations of high accelerator input (ac.), low accelerator input, high brake input (br.), low brake input, and no operation (no-op.), respectively. With these combinations ($2 \times 2 \times 5 = 20$), there are 20 fuzzy inference rules and rule weights η_i for $i = 1$ to $i = 20$.

6.3 Membership Functions and Parameters

The membership functions for the nine fuzzy sets are defined as

$$A_1^i(\mathbf{x}_1; \alpha_1^i) \equiv \begin{cases} A_1^{\text{long}}(\mathbf{x}_1) & \text{if } i = 1, \dots, 10 \\ A_1^{\text{short}}(\mathbf{x}_1) & \text{if } i = 11, \dots, 20 \end{cases} \quad (23)$$

$$A_2^i(\mathbf{x}_2; \alpha_2^i) \equiv \begin{cases} A_2^{\text{fast}}(\mathbf{x}_2) & \text{if } i = 1, \dots, 5, 11, \dots, 15 \\ A_2^{\text{slow}}(\mathbf{x}_2) & \text{if } i = 6, \dots, 10, 16, \dots, 20 \end{cases} \quad (24)$$

$$B_1^i(y_1; \beta_1^i) \equiv \begin{cases} B_1^{\text{high-ac.}}(y_1) & \text{if } i = 1, 6, 11, 16 \\ B_1^{\text{low-ac.}}(y_1) & \text{if } i = 2, 7, 12, 17 \\ B_1^{\text{high-br.}}(y_1) & \text{if } i = 3, 8, 13, 18 \\ B_1^{\text{low-br.}}(y_1) & \text{if } i = 4, 9, 14, 19 \\ B_1^{\text{no-op.}}(y_1) & \text{if } i = 5, 10, 15, 20 \end{cases} \quad (25)$$

A common membership function is thus used for the same fuzzy set even in different rules. For example, the membership functions for the fuzzy sets A_1^1 through A_1^{10} are all $A_1^{\text{long}}(\mathbf{x}_1)$, which corresponds to the fuzzy representation “long.”

6.4 Deciding the Action

In the training phase, the output value for the action at time t is selected probabilistically from

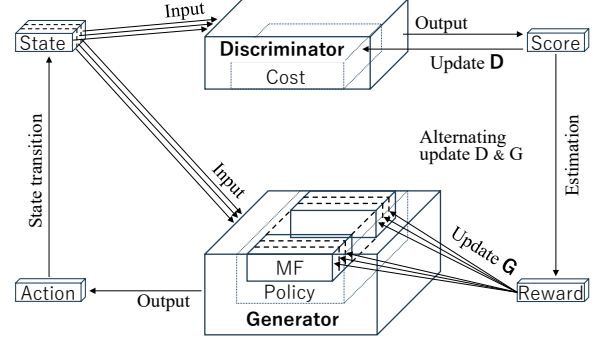


Fig. 2 Configuration of AIRL-BMP

the discrete values of y_1 according to the policy $\pi(y_1 | \mathbf{x}(t))$. In the evaluation phase, the output value is decided deterministically as the expected value $\sum_{y_1} y_1 \pi(y_1 | \mathbf{x}(t))$ over the discrete values of y_1 .

7 Computer Experiments

In the computer experiments, the parameters were learned by applying the proposed method to the control task described in Section 6, and the results were evaluated. For comparison, experiments were also conducted with AIRL-BMP (AIRL with Base-Model Policy), an IRL method that uses the discriminator of AIRL described in Section 4 as it stands and takes the policy of the base model as the generator. Fig. 2 shows the configuration of AIRL-BMP. In AIRL-BMP, the discriminator and the generator share no parameters, and they are updated alternately rather than simultaneously. The learning rule for the discriminator in AIRL-BMP is Eq. (10), and that for the generator is given by substituting Eq. (11) into $\hat{r}$ in Eq. (20), the learning rule of the proposed method.

In these experiments, each membership function in the antecedent part, namely $A_1^{\text{long}}(\mathbf{x}_1)$, $A_1^{\text{short}}(\mathbf{x}_1)$, $A_2^{\text{fast}}(\mathbf{x}_2)$, and $A_2^{\text{slow}}(\mathbf{x}_2)$, was represented by a neural network. Among the parameters $\{\alpha, \beta, \eta\}$ of the policy, the weight parameters $\alpha_j^i \in \{\alpha_1^{\text{long}}, \alpha_1^{\text{short}}, \alpha_2^{\text{fast}}, \alpha_2^{\text{slow}}\}$ of these neural networks were the learning target. The membership functions $B_1^i(y_1)$ in the consequent part were fixed to those adopted in the previous experiments on the base model [13]. The definitions of $B_1^i(y_1)$ and the discretization of y_1 are given in Appendix D.

The rule weights η_i were fixed as

$$\eta_i = \begin{cases} 1 & \text{if } i = 5, 6, 13, 20 \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

In Eq. (26), for each of the four combinations of antecedent fuzzy representations, a weight of 1 was assigned to the fuzzy inference rule whose consequent fuzzy representation was considered the most appropriate, and 0 to the others.

For both methods, the temperature T in Eq. (3) for the policy was set to 0.02. In Eq. (20) for the learning rule of the generator, q and γ were set to 0 and 1, respectively. For the proposed method, the learning coefficients δ in Eq. (16) and ε in Eq. (20) were set to 0.00002 and 0.000001, respectively. For AIRL-BMP, the learning coefficients δ in Eq. (10) and ε in Eq. (20) were set to 0.0002 and 0.00003, respectively. The discount rate κ in Eq. (8) for the cost function of AIRL-BMP was set to 0.9. These values were chosen empirically based on the results of preliminary experiments.

7.1 Configuration of the Neural Networks

7.1.1 Membership Functions in the Antecedent Part

Each of the neural networks representing the four membership functions in the antecedent part consisted of an input layer with three or two units, a hidden layer with 120 units, and an output layer with one unit that outputs the matching degree. The activation functions were PReLU in the hidden layer and the sigmoid function in the output layer. The initial negative slope of PReLU was 0.25. The weight parameters corresponding to the biases were initialized with a uniform distribution over $[0, 0.01]$, and the other weight parameters were initialized with a normal distribution of standard deviation $1/\sqrt{n}$, where n is the number of input units of each layer. The input values to the neural networks were scaled to one hundredth of the original values, namely $(x_v, v_{\text{front}})/100$ or $(x_d, l_{\text{near}}, l_{\text{far}})/100$.

7.1.2 Discriminator of AIRL-BMP

In the discriminator of AIRL-BMP, the input of the function $g(\mathbf{x}_1, \mathbf{x}_2, y_1; \theta)$ was six-dimensional and that of the function $h(\mathbf{x}_1, \mathbf{x}_2; \phi)$ was five-dimensional. Each of these functions was represented by a neural network consisting of an input

layer with six or five units, a hidden layer with 120 units, and an output layer with one unit. The activation functions were PReLU in the hidden layer and the identity function in the output layer. The initial negative slope of PReLU, the initial values of the weight parameters, and the scaling of $\mathbf{x}_1$ and $\mathbf{x}_2$ were the same as in the neural networks representing the membership functions in the antecedent part. In addition, y_1 was scaled to one fifth of the original value when it was input to the network.

7.2 Experimental Procedure in the Training Phase

7.2.1 Training Problems

A total of $5 \times 5 \times 5 \times 5 = 625$ speed control problems were prepared by exhaustively combining the speed of the leading vehicle (km/h) from $\{10, 30, 50, 70, 90\}$, the initial speed of the following vehicle (km/h) from $\{0, 20, 40, 60, 80\}$, the initial inter-vehicle distance (m) from $\{10, 30, 50, 70, 90\}$, and the target range of the inter-vehicle distance (m) from $\{[10, 15], [30, 35], [50, 55], [70, 75], [90, 95]\}$. Problems in which a rear-end collision was theoretically unavoidable or hard to prevent were then excluded: those in which the initial speed of the following vehicle exceeded the speed of the leading vehicle by 50 km/h or more and the initial inter-vehicle distance was 30 m or less, and those in which the initial speed of the following vehicle exceeded the speed of the leading vehicle by 70 km/h or more. The remaining 580 problems were used for training. For each of these 580 problems, one successful episode of the demonstration τ_D was created by manual control and used as the expert trajectory.

7.2.2 Training Procedure

After the parameters to be learned were initialized, the following procedure was used.

Step 1. Generate a trajectory sample τ_S for a training problem from the output of the generator, using the action decision in the training phase.

Step 2. For each training problem, repeat the sample generation of Step 1 and the update below in sequence. Proceed to Step 3 when the parameters have been updated once for each training problem.

Table 1 Control results for the 580 training problems (task success rate, number of successful tasks, and breakdown of failed tasks) over the 20 sets of learned parameters

	Avg. task success rate	Successful tasks				Avg. no. of failed tasks		
		min.	max.	avg.	SD	collision	out of range	delay
Proposed	0.981	565	572	569.25	2.34	10.50	0.00	0.25
AIRL-BMP	0.721	285	539	418.45	74.59	19.45	44.15	97.95
Base model ^a	0.983	519	580	570.05	12.16	6.90	3.05 ^b	

^a Values quoted from [13]

^b Combined count for “out of range” and “delay” failures, which are not distinguished in the base model

Table 2 Number of training epochs over the 20 training runs

	Training epochs			
	min.	max.	avg.	SD
Proposed	35	78	48.80	10.51
AIRL-BMP	327	1350	746.65	257.83

Proposed method: Update the parameters α_j^i shared by the discriminator and the generator using the learning rule in Eq. (22).

AIRL-BMP: Update the parameters $\{\theta, \phi\}$ of the discriminator using the learning rule in Eq. (10). Then update the parameters α_j^i of the generator using the learning rule in Eq. (20), into which the reward in Eq. (11) from the updated discriminator is substituted.

Step 3. Generate a trajectory sample τ_S for a training problem from the output of the generator, using the action decision in the evaluation phase, then check and record whether the task succeeds. Proceed to Step 4 when this has been done for all training problems.

Step 4. If the task succeeds for all training problems, adopt the current parameters as the result and end the training. If any problem fails, return to Step 1. However, the number of epochs, each consisting of Steps 1 to 4, is limited to 500 for the proposed method and 2500 for AIRL-BMP. When this maximum is reached, adopt as the result the parameters that gave the largest number of successful problems, and end the training. If more than one set of parameters qualifies, adopt the one obtained at the smallest number of epochs.

For both methods, stochastic gradient descent (SGD) was used as the optimizer. The update

equations derived from each learning rule are given in Appendix C.

The procedure described above, from the initialization of the parameters to the end of the training, is counted as one training run. Each training run therefore gives one set of learned parameters. In this study, 20 training runs were conducted, giving 20 sets of learned parameters.

7.3 Experimental Results and Discussion in the Training Phase

The control results for the training problems with the weight parameters from each training run are shown in Table 1, and the number of epochs for each run is shown in Table 2. For comparison, Table 1 also includes the control results reported in [13] for the same training problems with the base model, which uses a reward function set manually in advance. Table 1 shows that the proposed method achieved an average task success rate of 98.1%, close to the 98.3% of the base model, with small variation across the 20 sets of results. The average task success rate of AIRL-BMP, in contrast, was 72.1%, about 26 points lower than that of the proposed method. Moreover, AIRL-BMP gave a smaller minimum and a smaller maximum number of successfully controlled problems than the proposed method, and its variation across the sets was also greater, with a standard deviation about 32 times as large. Table 2 shows that the number of epochs required for training by the proposed method was about one fifteenth of that for AIRL-BMP, and that its variation across training runs also tended to be small, with a standard deviation about one twenty-fifth as large. A Mann-Whitney U test between the proposed method and AIRL-BMP showed that the proposed method achieved a significantly larger number of successful

tasks (median 570.00, IQR [567.75, 571.25] versus median 407.50, IQR [370.50, 478.75]; $U = 400$, $p < 0.001$, $r = 0.855$). It also required significantly fewer epochs (median 47.00, IQR [40.75, 55.00] versus median 662.50, IQR [584.25, 887.25]; $U = 0$, $p < 0.001$, $r = 0.855$), with $n = 20$ for each method².

Task failures can be classified into three types: collision, being out of the target range at $t = 110$ (out of range), and remaining within the target range until $t = 110$ with the start of that period later than $t = 80$ (delay). The breakdown of the average number of failed tasks in Table 1 shows that the proposed method had no cases of “out of range” and only 0.25 cases of “delay,” and the base model also had only 3.05 cases of “out of range” and “delay” combined. AIRL-BMP, in contrast, had a large total average number of failed tasks, with “delay” the most frequent at 97.95 cases, followed by “out of range” at 44.15 cases. These results suggest that during training AIRL-BMP could not improve many trajectories of “delay” and “out of range,” which are relatively close to those of successful tasks, whereas the proposed method was able to learn to succeed in these tasks. Most of the 10.50 cases of “collision” in the proposed method occurred in relatively demanding problem settings in which the initial speed of the following vehicle was 50 km/h higher than the speed of the leading vehicle, as was also the case for “collision” in the base model.

The average reward per time step for all fuzzy representations at each training epoch is shown in Fig. 3. The value at each epoch is averaged over all the training problems in all the training runs, and the error bars show the minimum and maximum of the averages over the 580 problems in each training run. Fig. 3 shows that the average reward for the proposed method decreased almost monotonically toward zero, the theoretical value of the reward under the optimal policy, within a small number of epochs. This suggests that the proposed method efficiently learned parameters achieving a high average task success rate. The average reward for AIRL-BMP, in contrast, started from a negative value, dropped sharply, and then rose sharply once. It never reached zero in any of the training runs,

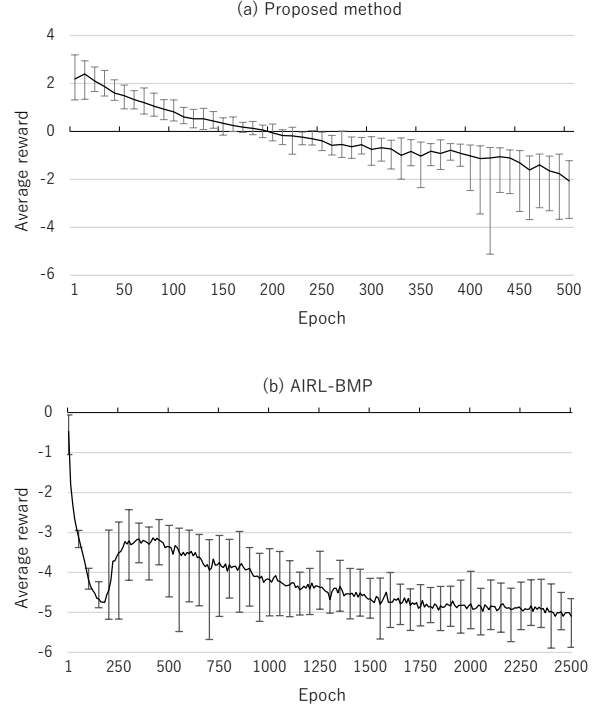


Fig. 3 Average reward per time step for all fuzzy representations at each training epoch. **a** Proposed method, **b** AIRL-BMP

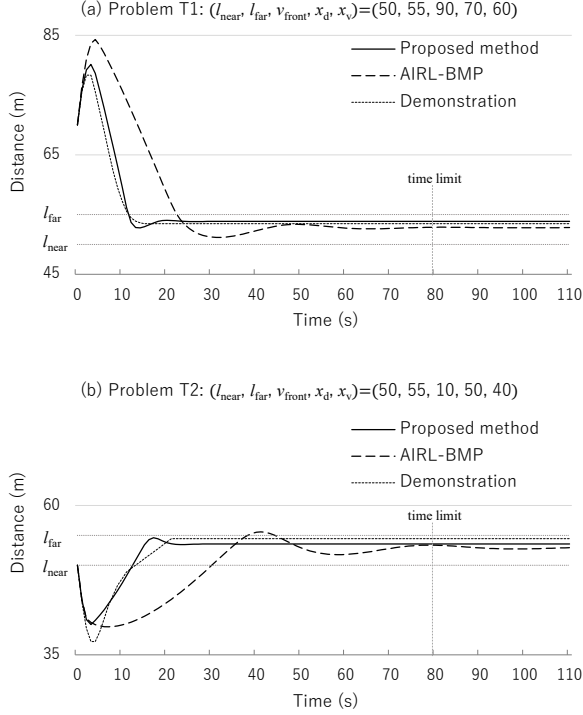
however, and thereafter decreased almost monotonically. When the trajectories of the samples from the generator differ from those of the demonstration, with large deviations in the inter-vehicle distance or the speed, the output of the discriminator generally becomes small and the reward tends to take a large negative value. Since the average reward for AIRL-BMP never became zero or higher throughout training with five times as many epochs as for the proposed method, AIRL-BMP was less effective than the proposed method in learning to bring the trajectories of the samples for each problem closer to those of the demonstration on average.

As successful examples among the control results for the training problems, the trajectories of the inter-vehicle distance after training with the proposed method and with AIRL-BMP are shown in Fig. 4, together with the trajectory of the demonstration, for problem T1, which required acceleration from the initial state, and problem T2, which required deceleration. In problems T1 and T2, $(l_{\text{near}}, l_{\text{far}}, v_{\text{front}})$ were (50, 55, 90) and (50, 55, 10), and the initial values of (x_d, x_v) were (70, 60) and (50, 40), respectively. For both

²The statistical test is applied only to the results obtained in the experiments of this paper.

Table 3 Trajectory characteristics for the successful training problems

	Time to settle		Settling delay		Exits from range		SD after settling
	avg.	SD	avg.	SD	avg.	SD	pooled value
Proposed	22.83	1.84	5.71	1.28	0.48	0.13	0.28
AIRL-BMP	47.91	6.65	29.51	10.22	1.35	0.41	0.96
Demonstration	14.60	—	2.68	—	0.21	—	0.35

**Fig. 4** Trajectories of the inter-vehicle distance with the proposed method, AIRL-BMP, and the demonstration. **a** Problem T1, **b** Problem T2

methods, the trajectories in Fig. 4 were generated by the policies using the parameters from the training run that gave the largest number of successful problems. Fig. 4 shows that the proposed method, as in the demonstration, tended to settle the inter-vehicle distance within the target range earlier than AIRL-BMP and to give trajectories with smaller oscillations. The trajectories from AIRL-BMP, in contrast, tended to settle within the target range later and to require more time for the oscillations to converge. Table 3 quantifies these characteristics for the successful training problems. The time to settle is the start of the period in which the inter-vehicle distance remained within the target range until $t = 110$, the settling delay is the interval between the first time the distance was within the

range and the time to settle, the exits from range is the number of times the distance moved from inside to outside the target range, and the SD after settling is the standard deviation of the distance from the time to settle to $t = 110$. For the first three measures, the values were averaged over the successful problems for each set of learned parameters, and Table 3 shows the mean and the standard deviation of these averages over the 20 sets. The last measure is pooled over all the successful problems in all the sets. Table 3 also includes the values for the demonstration for reference. Since only one demonstration was created for each problem, the values were averaged over the 580 problems and no standard deviation over sets is available. A Mann-Whitney U test between the proposed method and AIRL-BMP showed that the differences in all four measures were significant ($p < 0.001$, $n = 20$ for each method), where the pooled standard deviation for each set was used for the last measure. Continuous oscillations of the trajectory do not occur in the demonstration and represent unnatural control. They not only increase the possibility of task failure but can also reduce usefulness even when the task succeeds, and they are generally undesirable. For the proposed method, in contrast, such oscillations requiring time to converge hardly occurred. These results indicate that the proposed method was less prone to continuous oscillations of the trajectory than AIRL-BMP.

7.4 Experimental Procedure in the Evaluation Phase

As speed control problems different from those used for training, $5 \times 5 \times 5 \times 5 = 625$ problems were prepared by exhaustively combining the speed of the leading vehicle (km/h) from $\{45, 55, 65, 75, 85\}$, the initial speed of the following vehicle (km/h) from $\{0, 10, 30, 50, 70\}$, the initial inter-vehicle distance (m) from $\{30, 45, 65, 70, 80\}$, and the target range of the inter-vehicle distance

Table 4 Control results for the 722 evaluation problems (task success rate, number of successful tasks, and breakdown of failed tasks) over the 20 sets of learned parameters

	Avg. task	Successful tasks				Avg. no. of failed tasks		
	success rate	min.	max.	avg.	SD	collision	out of range	delay
Proposed	1.000	722	722	722.00	0.00	0.00	0.00	0.00
AIRL-BMP	0.924	586	722	667.15	47.05	0.00	12.65	42.20
Base model ^a	0.994	637	722	717.65	18.99	0.00	4.35 ^b	

^a Values quoted from [13]

^b Combined count for “out of range” and “delay” failures, which are not distinguished in the base model

(m) from $\{[10, 30], [20, 40], [40, 60], [50, 60], [60, 70]\}$. In addition, $3 \times 3 \times 3 \times 3 = 81$ problems were prepared by exhaustively combining the speed of the leading vehicle (km/h) from $\{40, 50, 60\}$, the initial speed of the following vehicle (km/h) from $\{20, 40, 60\}$, the initial inter-vehicle distance (m) from $\{20, 40, 60\}$, and the target range of the inter-vehicle distance (m) from $\{[10, 20], [40, 50], [45, 60]\}$. A further $4 \times 1 \times 2 \times 2 = 16$ problems were prepared by exhaustively combining the speed of the leading vehicle (km/h) from $\{20, 30, 50, 60\}$, the initial speed of the following vehicle (km/h) from $\{30\}$, the initial inter-vehicle distance (m) from $\{10, 50\}$, and the target range of the inter-vehicle distance (m) from $\{[10, 15], [30, 45]\}$. For these 722 evaluation problems, one set of control results was obtained with each of the 20 sets of learned weight parameters and under the action decision in the evaluation phase.

7.5 Experimental Results and Discussion in the Evaluation Phase

The control results for the evaluation problems with the weight parameters from each training run are shown in Table 4. For comparison, as in Table 1, Table 4 also includes the results reported in [13] for the same evaluation problems with the base model. Table 4 shows that the proposed method achieved an average task success rate of 100%, higher than the 99.4% of the base model, with no variation across the sets. AIRL-BMP, in contrast, had an average task success rate of 92.4%, lower than that of the base model, and its variation across the sets was relatively large, as was also the case for the training problems. A Mann-Whitney U test between the proposed method and AIRL-BMP showed that the former achieved a significantly

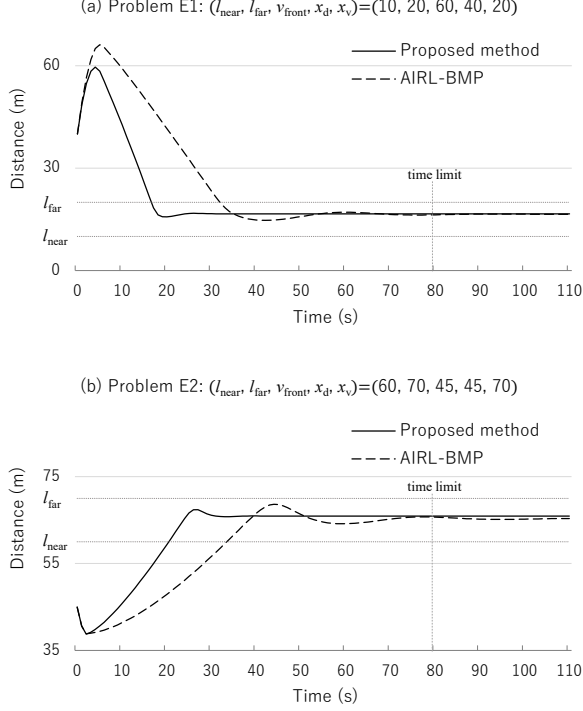
larger number of successful tasks (median 722.00, IQR $[722.00, 722.00]$ versus median 671.50, IQR $[638.50, 715.25]$; $U = 360$, $p < 0.001$, $r = 0.684$, $n = 20$ for each method).

In the breakdown of the average number of failed tasks for AIRL-BMP, “delay” was the most frequent at 42.20 cases, followed by “out of range” at 12.65 cases. As was also the case for the training problems, a tendency toward insufficient learning performance was suggested. For all three methods, no case of “collision” occurred in any of the results. This is thought to be partly because there was no evaluation problem in which the initial speed of the following vehicle exceeded the speed of the leading vehicle by 50 km/h or more.

As successful examples among the control results for the evaluation problems, the trajectories of the inter-vehicle distance after training with the proposed method and with AIRL-BMP are shown in Fig. 5, for problem E1, which required acceleration from the initial state, and problem E2, which required deceleration. In problems E1 and E2, $(l_{\text{near}}, l_{\text{far}}, v_{\text{front}})$ were $(10, 20, 60)$ and $(60, 70, 45)$, and the initial values of (x_d, x_v) were $(40, 20)$ and $(45, 70)$, respectively. For both methods, the trajectories in Fig. 5 were generated by the policies using the parameters from the training run that gave the largest number of successful problems. Fig. 5 shows that the proposed method tended to settle the inter-vehicle distance within the target range earlier than AIRL-BMP and to give trajectories with smaller oscillations, as was the case for the training problems. Table 5 quantifies these characteristics for the successful evaluation problems. For the first three measures, the values were averaged over the successful problems for each set of learned parameters, and Table 5 shows the mean and the standard deviation of these averages over the 20 sets. The last measure is pooled over all the successful problems

Table 5 Trajectory characteristics for the successful evaluation problems

	Time to settle		Settling delay		Exits from range		SD after settling
	avg.	SD	avg.	SD	avg.	SD	pooled value
Proposed	17.51	2.06	4.56	0.47	0.28	0.02	0.81
AIRL-BMP	31.26	3.22	11.61	3.66	0.45	0.19	2.20

**Fig. 5** Trajectories of the inter-vehicle distance with the proposed method and AIRL-BMP. **a** Problem E1, **b** Problem E2

in all the sets. A Mann-Whitney U test between the proposed method and AIRL-BMP showed that the differences in all four measures were significant ($p < 0.001$, $n = 20$ for each method), where the pooled standard deviation for each set was used for the last measure. These results suggest that the proposed method acquired higher generalization performance than AIRL-BMP, capable of generating trajectories for successful tasks that settled earlier and with smaller oscillations.

8 Conclusion

This paper proposed a fuzzy control method that integrates a reward estimation mechanism based on the AIRL framework into the base model, in which reinforcement learning is integrated with

fuzzy inference. In the proposed method, the policy of the base model is treated as the generator, and the set of membership functions in the policy is also used as the cost function of the discriminator. This cost function approximates a quantity corresponding to the advantage of an action with respect to the optimal policy, using the set of membership functions rather than a value function. With this configuration, the proposed method makes stable and efficient learning possible by performing the discriminator update for reward estimation and the generator update for policy estimation simultaneously on the parameters shared by both, without estimating a value function.

Simulations of an illustrative vehicle speed control task were conducted to compare the performance of the proposed method with that of AIRL-BMP, which uses the discriminator of AIRL as it stands and takes the policy of the base model as the generator. The results were also compared with those reported on the same problems in previous work for the base model, which uses a reward function set manually in advance. For the training problems, the proposed method achieved an average task success rate almost equal to that of the base model and much higher than that of AIRL-BMP, with far fewer epochs, and the variation in the success rate across training runs was smaller for the proposed method than for the other methods. For the evaluation problems, the proposed method achieved the highest average task success rate among the three methods, 100%, with no variation across the 20 sets of results. These results show that the proposed method achieves stable and efficient learning and can acquire policies with high generalization performance.

In this paper, the proposed method was verified on an illustrative vehicle speed control task. Applying the method to more realistic problems, in which the task and the shapes of the membership functions differ from those in this example, is one of the next steps. The simultaneous learning of the parameters of the membership functions in

the consequent part and of the rule weights was not addressed in this paper, and will be examined in future work.

Funding. No funding was received to assist with the preparation of this manuscript.

Data availability. The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest. The author has no conflicts of interest to declare that are relevant to the content of this article.

Appendix A Derivation of Eq. (16)

Substituting Eq. (14) into the right-hand side of Eq. (15) gives

$$\begin{aligned} \mathcal{L}(\alpha_j^i) = & \sum_{t=0}^{L-1} \left\{ -\mathbb{E}_{\mathcal{D}_t} \left[f_{\alpha_j^i}(x, y, x') \right. \right. \\ & \left. \left. - \ln \left(\exp(f_{\alpha_j^i}(x, y, x')) + \pi(y|x) \right) \right] \right. \\ & \left. - \mathbb{E}_{\pi_t} \left[\ln(\pi(y|x)) \right. \right. \\ & \left. \left. - \ln \left(\exp(f_{\alpha_j^i}(x, y, x')) + \pi(y|x) \right) \right] \right\} \end{aligned} \quad (\text{A1})$$

Treating π as a constant and differentiating Eq. (A1) partially with respect to α_j^i gives

$$\begin{aligned} & \frac{\partial}{\partial \alpha_j^i} \mathcal{L}(\alpha_j^i) \\ &= \sum_{t=0}^{L-1} \left\{ -\mathbb{E}_{\mathcal{D}_t} \left[\frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right. \right. \\ & \quad \left. \left. - \frac{\exp(f_{\alpha_j^i}(x, y, x')) \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x')}{\exp(f_{\alpha_j^i}(x, y, x')) + \pi(y|x)} \right] \right. \\ & \quad \left. + \mathbb{E}_{\pi_t} \left[\frac{\exp(f_{\alpha_j^i}(x, y, x')) \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x')}{\exp(f_{\alpha_j^i}(x, y, x')) + \pi(y|x)} \right] \right\} \end{aligned} \quad (\text{A2})$$

$$\begin{aligned} &= \sum_{t=0}^{L-1} \left\{ -\mathbb{E}_{\mathcal{D}_t} \left[(1 - D_j^i(x, y, x')) \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right. \\ & \quad \left. + \mathbb{E}_{\pi_t} \left[D_j^i(x, y, x') \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right\} \end{aligned} \quad (\text{A3})$$

Multiplying the right-hand side of Eq. (A3) by $-\delta$ gives Eq. (16).

Appendix B Derivation of Eq. (21)

From Eq. (6), the characteristic eligibility $e_{\alpha_j^i}(t)$ is expressed as

$$\begin{aligned} e_{\alpha_j^i}(t) &= -\frac{1}{T} \left\{ \frac{\partial E(y(t); x(t))}{\partial \alpha_j^i} - \mathbb{E}_{\pi(y|x(t))} \left[\frac{\partial E(y; x(t))}{\partial \alpha_j^i} \right] \right\} \end{aligned} \quad (\text{B1})$$

From Eq. (2), the first term in the braces on the right-hand side of Eq. (B1) is

$$\frac{\partial E(y(t); x(t))}{\partial \alpha_j^i} = -\eta_i B^i(y(t)) \frac{\partial}{\partial \alpha_j^i} A^i(x(t)) \quad (\text{B2})$$

By replacing $y(t)$ with y in Eq. (B2) and taking the expectation of both sides, the second term in the braces on the right-hand side of Eq. (B1) is expressed as

$$\begin{aligned} & \mathbb{E}_{\pi(y|x(t))} \left[\frac{\partial E(y; x(t))}{\partial \alpha_j^i} \right] \\ &= -\eta_i \mathbb{E}_{\pi(y|x(t))} [B^i(y)] \frac{\partial}{\partial \alpha_j^i} A^i(x(t)) \end{aligned} \quad (\text{B3})$$

Substituting Eqs. (B2) and (B3) into the right-hand side of Eq. (B1) gives Eq. (21).

Appendix C Update Equations Used in the Computer Experiments

C.1 Discriminator in the Proposed Method

From Eqs. (12) and (13), the cost function for $(x(t), y(t), x(t+1))$ contained in the trajectory of

Table C1 Gradient of $|v_{\alpha_j^i}|$ in each case

demonstration $\geq$ sample		demonstration $<$ sample	
x_d	x_v	x_d	x_v
$\frac{\partial v_{\alpha_1^{\text{long}}}}{\partial \alpha_1^{\text{long}}}$	$\frac{\partial v_{\alpha_2^{\text{fast}}}}{\partial \alpha_2^{\text{fast}}}$	$-\frac{\partial v_{\alpha_1^{\text{long}}}}{\partial \alpha_1^{\text{long}}}$	$-\frac{\partial v_{\alpha_2^{\text{fast}}}}{\partial \alpha_2^{\text{fast}}}$
$-\frac{\partial v_{\alpha_1^{\text{short}}}}{\partial \alpha_1^{\text{short}}}$	$-\frac{\partial v_{\alpha_2^{\text{slow}}}}{\partial \alpha_2^{\text{slow}}}$	$\frac{\partial v_{\alpha_1^{\text{short}}}}{\partial \alpha_1^{\text{short}}}$	$\frac{\partial v_{\alpha_2^{\text{slow}}}}{\partial \alpha_2^{\text{slow}}}$

the demonstration is

$$f_{\alpha_j^i}(x(t), y(t), x(t+1)) = \ln\{1 - |\mathbb{E}_{\mathcal{D}_t}[A_j^i(x'_j; \alpha_j^i)] - A_j^i(x_j(t+1); \alpha_j^i)|\} \quad (\text{C1})$$

When the trajectory of the demonstration for each problem consists of one episode, $\mathbb{E}_{\mathcal{D}_t}[A_j^i(x'_j; \alpha_j^i)] = A_j^i(x_j(t+1); \alpha_j^i)$ holds, and the right-hand side of Eq. (C1) is therefore zero. The gradient of Eq. (C1) with respect to the parameter α_j^i is then also zero, and the first term in the braces on the right-hand side of Eq. (16) is zero. Consequently, in the computer experiments with the learning rule of the proposed method, the update equation for the parameters $\alpha_j^i \in \{\alpha_1^{\text{long}}, \alpha_1^{\text{short}}, \alpha_2^{\text{fast}}, \alpha_2^{\text{slow}}\}$ in the cost function $f_{\alpha_j^i}$ of the discriminator D_j^i is

$$\begin{aligned} \Delta \alpha_j^i &= \delta \sum_{t=0}^{L-1} \left\{ -\mathbb{E}_{\pi_t} \left[D_j^i(x, y, x') \frac{\partial}{\partial \alpha_j^i} f_{\alpha_j^i}(x, y, x') \right] \right\} \\ &= \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\pi_t} \left[\frac{\frac{\partial}{\partial \alpha_j^i} |v_{\alpha_j^i}(x'; x, y)|}{1 - |v_{\alpha_j^i}(x'; x, y)| + \pi(y|x)} \right] \right\} \end{aligned} \quad (\text{C2})$$

In these experiments, $A_1^{\text{long}}(\mathbf{x}_1)$ and $A_2^{\text{fast}}(\mathbf{x}_2)$ were assumed to be monotonically increasing, and $A_1^{\text{short}}(\mathbf{x}_1)$ and $A_2^{\text{slow}}(\mathbf{x}_2)$ were assumed to be monotonically decreasing. Under these assumptions, the sign of $v_{\alpha_j^i}$ is determined by the magnitude relation between x_d or x_v of the demonstration and that of the sample. The gradient of $|v_{\alpha_j^i}|$ was therefore computed as shown in Table C1.

C.2 Generator in the Proposed Method

From Eq. (21), in the computer experiments with the learning rule of the proposed method, the characteristic eligibility for the policy of the generator is expressed as

$$e_{\alpha_j^i}(t) = -\frac{1}{T} \eta_i \{ \mathbb{E}_{\pi(y_1|x(t))} [B_1^i(y_1)] - B_1^i(y_1(t)) \} \cdot \frac{\partial}{\partial \alpha_j^i} \{ A_1^i(x_1(t)) A_2^i(x_2(t)) \} \quad (\text{C3})$$

Eqs. (18) and (19) for the case $\mu = \alpha_j^i$ and Eq. (C3) are substituted into Eq. (20). For $q = 0$ and $\gamma = 1$, the update equation for the parameters $\alpha_j^i \in \{\alpha_1^{\text{long}}, \alpha_1^{\text{short}}, \alpha_2^{\text{fast}}, \alpha_2^{\text{slow}}\}$ in the membership functions A_j^i that constitute the policy π of the generator is then expressed as

$$\begin{aligned} \Delta \alpha_j^i &= \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=0}^{L-1} (\ln(D_j^i(x(u), y(u), x(u+1))) - \ln(1 - D_j^i(x(u), y(u), x(u+1)))) \right\} e_{\alpha_j^i}(t) \right] \\ &= \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=0}^{L-1} (f_{\alpha_j^i}(x(u), y(u), x(u+1)) - \ln(\pi(y(u)|x(u)))) \right\} e_{\alpha_j^i}(t) \right] \end{aligned} \quad (\text{C4})$$

For example, in the case of rule $i = 5$, from Eqs. (23), (24), and (25), the characteristic eligibility $e_{\alpha_1^5}(t) = e_{\alpha_1^{\text{long}}}(t)$ is

$$\begin{aligned} e_{\alpha_1^{\text{long}}}(t) &= -\frac{1}{T} \eta_5 \{ \mathbb{E}_{\pi(y_1|x(t))} [B_1^{\text{no-op.}}(y_1)] - B_1^{\text{no-op.}}(y_1(t)) \} \\ &\quad \cdot A_2^{\text{fast}}(x_2(t)) \frac{\partial A_1^{\text{long}}(x_1(t))}{\partial \alpha_1^{\text{long}}} \end{aligned} \quad (\text{C5})$$

Substituting Eq. (C5) into Eq. (C4) gives the update equation for the parameter $\alpha_1^5 = \alpha_1^{\text{long}}$ in the generator.

C.3 Discriminator in AIRL-BMP

From Eqs. (8) and (10), the update equations for the parameters θ and ϕ in the cost function $f_{\theta, \phi}$

Table D1 Node values defining the consequent membership functions for the accelerator and brake inputs

y_1	-5	-4	-3	-2	-1	0	1	2	3	4	5
$B_1^{\text{high-ac.}}$	0	0	0	0	0	0	0.2	0.3	0.6	0.8	0.9
$B_1^{\text{low-ac.}}$	0	0	0	0	0	1	0.8	0.6	0.5	0.4	0.2
$B_1^{\text{high-br.}}$	0.9	0.7	0.5	0.3	0.2	0	0	0	0	0	0
$B_1^{\text{low-br.}}$	0.1	0.2	0.4	0.6	0.7	1	0	0	0	0	0

of the discriminator D in AIRL-BMP are

$$\Delta\theta = \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\mathcal{D}_t} \left[(1 - D(x, y, x')) \frac{\partial}{\partial \theta} g(x, y) \right] - \mathbb{E}_{\pi_t} \left[D(x, y, x') \frac{\partial}{\partial \theta} g(x, y) \right] \right\} \quad (\text{C6})$$

$$\Delta\phi = \delta \sum_{t=0}^{L-1} \left\{ \mathbb{E}_{\mathcal{D}_t} \left[(1 - D(x, y, x')) \frac{\partial}{\partial \phi} (\kappa h(x') - h(x)) \right] - \mathbb{E}_{\pi_t} \left[D(x, y, x') \frac{\partial}{\partial \phi} (\kappa h(x') - h(x)) \right] \right\} \quad (\text{C7})$$

C.4 Generator in AIRL-BMP

Eqs. (11) and (C3) are substituted into Eq. (20). For $q = 0$ and $\gamma = 1$, the update equation for the parameters $\alpha_j^i \in \{\alpha_1^{\text{long}}, \alpha_1^{\text{short}}, \alpha_2^{\text{fast}}, \alpha_2^{\text{slow}}\}$ in the membership functions A_j^i that constitute the policy π of the generator in AIRL-BMP is then expressed as

$$\begin{aligned} \Delta\alpha_j^i &= \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=0}^{L-1} (\ln(D(x(u), y(u), x(u+1))) - \ln(1 - D(x(u), y(u), x(u+1)))) \right\} e_{\alpha_j^i}(t) \right] \\ &= \varepsilon \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{L-1} \left\{ \sum_{u=0}^{L-1} (f_{\theta, \phi}(x(u), y(u), x(u+1)) - \ln(\pi(y(u) | x(u)))) \right\} e_{\alpha_j^i}(t) \right] \quad (\text{C8}) \end{aligned}$$

Appendix D Definition of the Consequent Membership Functions

The output variable y_1 was discretized with a step of 0.1 over $[-5, 5]$. The membership functions for the accelerator and brake inputs were defined as the piecewise linear interpolation of the node values listed in Table D1, with the functions for the accelerator input zero for $y_1 < 0$ and those for the brake input zero for $y_1 > 0$. $B_1^{\text{no-op.}}$ takes the value 1 only at $y_1 = 0$.

References

- [1] Arora S, Doshi P (2021) A survey of inverse reinforcement learning: challenges, methods and progress. *Artif Intell* 297:103500. <https://doi.org/10.1016/j.artint.2021.103500>
- [2] Baxter J, Bartlett PL (2001) Infinite-horizon policy-gradient estimation. *J Artif Intell Res* 15:319–350. <https://doi.org/10.1613/jair.806>
- [3] Er MJ, Deng C (2004) Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning. *IEEE Trans Syst Man Cybern B* 34(3):1478–1489. <https://doi.org/10.1109/TSMCB.2004.825938>
- [4] Finn C, Christiano P, Abbeel P, et al (2016) A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In: *NIPS 2016 Workshop on Adversarial Training*, arXiv:1611.03852
- [5] Finn C, Levine S, Abbeel P (2016) Guided cost learning: deep inverse optimal control via policy optimization. In: *Proc. of the 33rd International Conference on Machine Learning*, New York, USA, pp 49–58

- [6] Fu J, Luo K, Levine S (2018) Learning robust rewards with adversarial inverse reinforcement learning. In: Proc. of the 6th International Conference on Learning Representations, Vancouver, Canada
- [7] Glorennec PY (1994) Fuzzy Q-learning and dynamical fuzzy Q-learning. In: Proc. of the 3rd IEEE International Conference on Fuzzy Systems, Orlando, USA, pp 474–479, <https://doi.org/10.1109/FUZZY.1994.343739>
- [8] Goodfellow IJ (2015) On distinguishability criteria for estimating generative models. In: ICLR 2015 Workshop, arXiv:1412.6515
- [9] Goodfellow IJ, Pouget-Abadie J, Mirza M, et al (2014) Generative adversarial nets. In: Advances in Neural Information Processing Systems 27, pp 2672–2680
- [10] Horiuchi T, Fujino A, Katai O, et al (1996) Fuzzy interpolation-based Q-learning with continuous states and actions. In: Proc. of the 5th IEEE International Conference on Fuzzy Systems, New Orleans, USA, pp 594–600, <https://doi.org/10.1109/FUZZY.1996.551807>
- [11] Imanina N, Igarashi H (2014) Policy gradient method using fuzzy controller in policies and its application. In: Proc. of the International Conference on Artificial Intelligence and Pattern Recognition, Kuala Lumpur, Malaysia, pp 167–174
- [12] Ishihara S, Igarashi H (2006) Applying the policy gradient method to behavior learning in multiagent systems: the pursuit problem. Syst Comput Jpn 37(10):101–109. <https://doi.org/10.1002/scj.20248>
- [13] Ishihara S, Ichige S, Igarashi H (2024) Learning of membership functions in a control method integrating fuzzy inference and the policy gradient method. J Jpn Soc Fuzzy Theory Intell Inform 36(3):666–675. <https://doi.org/10.3156/jsoft.36.3.666>, (in Japanese)
- [14] Jouffe L (1998) Fuzzy inference system learning by reinforcement methods. IEEE Trans Syst Man Cybern C 28(3):338–355. <https://doi.org/10.1109/5326.704563>
- [15] Karras T, Aila T, Laine S, et al (2018) Progressive growing of GANs for improved quality, stability, and variation. In: Proc. of the 6th International Conference on Learning Representations, Vancouver, Canada
- [16] Peng XB, Kanazawa A, Toyer S, et al (2019) Variational discriminator bottleneck: improving imitation learning, inverse RL, and GANs by constraining information flow. In: Proc. of the 7th International Conference on Learning Representations, New Orleans, USA
- [17] Salimans T, Goodfellow I, Zaremba W, et al (2016) Improved techniques for training GANs. In: Advances in Neural Information Processing Systems 29, pp 2234–2242
- [18] Tavakol Aghaei V, Onat A (2017) Tuning scaling factors of fuzzy logic controllers via reinforcement learning policy gradient algorithms. In: Proc. of the 3rd International Conference on Mechatronics and Robotics Engineering, Paris, France, pp 146–151, <https://doi.org/10.1145/3068796.3068827>
- [19] Wang X, Xu X, He H (2004) Policy gradient fuzzy reinforcement learning. In: Proc. of the 3rd International Conference on Machine Learning and Cybernetics, Shanghai, China, pp 992–995, <https://doi.org/10.1109/ICMLC.2004.1382332>
- [20] Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8:229–256. <https://doi.org/10.1007/BF00992696>
- [21] Zare M, Kebria PM, Khosravi A, et al (2024) A survey of imitation learning: algorithms, recent developments, and challenges. IEEE Trans Cybern 54(12):7173–7186. <https://doi.org/10.1109/TCYB.2024.3395626>
- [22] Ziebart BD, Maas A, Bagnell JA, et al (2008) Maximum entropy inverse reinforcement learning. In: Proc. of the 23rd AAAI Conference on Artificial Intelligence, Chicago, USA, pp 1433–1438