Inverse Reinforcement Learning with Leveraged Gaussian Processes

Kyungjae Lee, Sungjoon Choi, and Songhwai Oh

Abstract— **In this paper, we propose a novel inverse reinforcement learning algorithm with leveraged Gaussian processes that can learn from both positive and negative demonstrations. While most existing inverse reinforcement learning (IRL) methods suffer from the lack of information near low reward regions, the proposed method alleviates this issue by incorporating (negative) demonstrations of *what not to do*. To mathematically formulate negative demonstrations, we introduce a novel generative model which can generate both positive and negative demonstrations using a parameter, called *proficiency*. Moreover, since we represent a reward function using a leveraged Gaussian process which can model a nonlinear function, the proposed method can effectively estimate the structure of a nonlinear reward function.**

I. Introduction

Reinforcement learning (RL) has been widely used to learn behaviors to perform complex tasks in robotics [1]. RL aims to find the optimal behavior which maximizes the expected sum of rewards during the execution phase. A reward function indicates the one step performance measure about each control at each situation. In order to successfully learn a desirable behavior, the reward function must be elaborately designed to express the given task.

However, in some tasks such as driving a car [2], inverted helicopter flight [3], and socially adaptive path planning [4], it is difficult to design a proper reward function that accurately generates the desired behaviors. It is more natural to learn the desirable behaviors performing such tasks by imitating expert's demonstrations. This problem is often formulated as inverse reinforcement learning (IRL) [5]. IRL aims to find the reward function which best explains demonstrations by experts. A key assumption of IRL is that experts follow the optimal policy induced by the underlying reward function, hence, the main idea of solving IRL is to find a reward function that makes experts' behaviors (near) optimal. The reward function learned by IRL is further used to obtain the desired behaviors by solving the usual reinforcement learning problem.

Since demonstrations of experts are often distributed near high reward regions, the resulting reward function learned by IRL cannot approximate low reward regions accurately. This phenomenon was intensively investigated in [6], [7]. The authors argued that the lack of demonstrations of *what to do* in the critical situations will lead to unsatisfactory performance or fatal failure. For example, when learning how to drive, an autonomous vehicle occasionally encounters a risky situation, e.g., heading towards the side of the road. In order to avoid a catastrophic situation, the autonomous vehicle should recover back to the center of the road. However, such recovery behavior rarely appears in demonstrations from a good driver. In [6], Ross and Bargnell tackled this problem via continuous interaction with experts. However, it is not practical to rely on experts frequently.

To handle lack of demonstrations near low reward regions, we incorporate demonstrations about both *what to do* and *what not to do*. As demonstrations about *what not to do* will be often distributed near low reward regions, we can obtain information to avoid catastrophic failures from such demonstrations.

In this paper, we propose a novel inverse reinforcement learning algorithm with leveraged Gaussian processes that can incorporate examples of both *what to do* (positive demonstrations) and *what not to do* (negative demonstrations). We model a reward function using a leveraged Gaussian process (LGP) [8], which is capable of modeling a complex nonlinear function. To mathematically define positive and negative demonstrations, we introduce a novel generative model of a demonstrator, which can generate both positive and negative demonstrations using the same reward function. Our generative model incorporates an additional parameter, called *proficiency*, which can vary continuously from -1 to $+1$, such that a positive (or, respectively, negative) proficiency indicates a positive (or, respectively, negative) demonstration. Hence, in our problem, a demonstration consists of its proficiency value as well as a sequence of state-action pairs. The proposed IRL method finds a reward function such that positive demonstrations have higher values and negative demonstrations have lower values. We extensively validate the performance of the proposed method in terms of accuracy and sample efficiencies. In simulations, the proposed method is more efficient and can approximate the underlying reward function more accurately using a fewer number of demonstrations than existing methods. Moreover, it shows that the use of negative demonstrations is better than simply using positive demonstrations alone, illustrating the benefits of using negative examples.

The remainder of this paper is structured as follows. In Section II, related work is discussed. In Section III, the Markov decision process (MDP), Gaussian process IRL, and LGP are introduced. In Section IV, the new expert model and the proposed learning algorithm are explained. A simulation study is discussed in Section V.

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-16-0307, Basic Software Research in Human-Level Lifelong Machine Learning).

K. Lee, S. Choi, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 151-744, Korea (e-mail: {kyungjae.lee, sungjoon.choi, songhwai.oh}@cpslab.snu.ac.kr).

	Pos Demo	Pos and Neg Demo
Margin based model	[2], [5], [9]	[10]
Probabilistic model	[11]–[14]	[15], Ours

TABLE I: Classification of IRL algorithms

II. Related Work

Recently, a number of IRL methods have been proposed. They can be separated into four different categories based on two criteria. The first criterion is the formulation of problem: margin based or probabilistic model based. The second is the capability of considering both positive and negative demonstrations or not. Many existing algorithms consider only positive demonstrations and a handful of approaches utilize both types of demonstrations. The classification of state-of-the-art IRL algorithms, including ours, is summarized in Table I.

A margin based method maximizes the margin between the value of the expert's policy and all other policies [2], [5], [9]. The margin based algorithms generally assume that the reward function is a linear combination of features. In [2], Abbeel and Ng proposed an apprenticeship learning (AL) algorithm, which maximizes the margin between the expert's policy and randomly sampled policies. In [9], Ratliff et al. proposed the maximum margin planning (MMP) algorithm, where Bellman-flow constraints are utilized to consider the margin between the experts' policy and all other policies. MMP was mainly motivated from the structured support vector machine (SVM) [16].

A probabilistic model based method first defines a probability distribution of expert's demonstrations and optimize the parameter of the distribution [11]–[14]. To define the probability on a trajectory of state-action pairs, many probabilistic IRL algorithms utilize a stochastic policy. The stochastic policy model was first utilized in [11], [12] in order to handle the inconsistency of expert's policy. Ziebart et al. [11] proposed maximum entropy inverse reinforcement learning (MaxEnt) using the principle of maximum entropy to handle ambiguity issues of IRL, where the efficient way to compute the gradient of the likelihood of demonstrations is also proposed. Ramachandran et al. [12] proposed Bayesian inverse reinforcement learning (BIRL), where the Bayesian probabilistic model over demonstrations is defined and solved using a Metropolis-Hastings (MH) method. We also note that [13], [14] are variants based on [11], [12]. Gaussian process inverse reinforcement learning (GPIRL) was proposed in [13], where the reward function is represented as a sparse Gaussian process, which can express a nonlinear reward function in a feature space. Choi et al. [14] proposed a nonparametric Bayesian feature constructing method for IRL (BNP-FIRL) to identify useful composite features for learning a reward function.

Despite successful advances in IRL, the demonstration acquisition is still an open issue. Generating demonstrations by experts can be often an expensive process. To handle this problem, [10], [15] are proposed to utilize inexpert demonstrations. In [10], unlabeled demonstrations are considered, for which we do not know whether they are actually generated by an expert or not. [10] proposed semi-supervised apprenticeship learning (SSAL), which is an extension of [2]

While SSAL [10] mainly focused on classifying unlabeled demonstrations, [15] focused more on utilizing failed demonstrations. In [15], Shiarlis et al. proposed inverse reinforcement learning from failure (IRLfF). IRLfF reformulated MaxEnt [11] with new constraints that require the learned policy to maximally differ from failed demonstrations. However, failed demonstrations do not exactly provide the information about *what not to do*, since the failed demonstration may contain some level of desirable behavior while its outcome is failure. In our paper, we focus on representing negative demonstrations, i.e., *what not to do*, using an additional parameter called *proficiency*.

Recently, a score-based IRL framework has also been proposed [17], [18]. In this framework, the score is manually assigned by the expert and it is defined as the discounted sum of rewards or value of a demonstration. Consequently, the expert should come up with the rewards of every states the demonstration traversed, whereas our proposed method requires a single scalar indicating the proficiency of the demonstrator. Moreover, both approaches model the reward structure as a linear function of features while our model is nonlinear.

III. Background

A. Markov Decision Processes and a Stochastic Policy

A common method to formulate a skill learning problem is a Markov decision process (MDP). An MDP can be characterized by a tuple $\mathbf{M} = \{\mathbf{S}, \mathbf{F}, \mathbf{A}, \mathbf{T}, \gamma, \mathbf{r}\}$, where $\mathbf{S}$ is the state space, $\mathbf{F}$ is the corresponding feature space, $\mathbf{A}$ is the action space, $\mathbf{T}(s'|s,a)$ is the transition probability from $s \in \mathbf{S}$ to $s' \in \mathbf{S}$ by taking an action $a \in \mathbf{A}$, γ is a discount factor, and $\mathbf{r}$ is the reward function. For inverse reinforcement learning (IRL), the problem is expressed as $\mathbf{M}/\mathbf{r}$ with experts' demonstrations $\mathcal{D} = \{\zeta_1, \ldots, \zeta_N\}$, where ζ_i is a sequence of state-action pairs, i.e., $\zeta_i = \{(s_{i,0}, a_{i,0}), \ldots, (s_{i,T}, a_{i,T})\}$. In IRL, it is a general assumption that the expert always obeys the optimal policy. In practice, however, demonstrations from experts can be inconsistent with each other. To handle this issue, a stochastic policy model has been widely used in IRL problems [11]–[13]. In this model, the probability of choosing action a at state s is exponentially proportional to the state-action value function $Q(s,a)$ and the resulting stochastic policy function is defined as follows:

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{a'} e^{Q(s,a')}}.$$

The stochastic policy is computed by the soft Bellman equation, which was proposed in [19].

Under this policy, the log likelihood of demonstration ζ under $\mathbf{r}$ can be written as

$$\log P(\zeta|\mathbf{r}) \propto \sum_{t=0}^{T} \left[Q(s_t, a_t) - V(s_t) \right], \tag{1}$$

where $V(\cdot)$ is a soft value function, i.e., $V(s) = \log \sum_{a'} e^{Q(s,a')}$.

B. Gaussian Process Inverse Reinforcement Learning

Gaussian process inverse reinforcement learning (GPIRL) was proposed in [13]. GPIRL uses the stochastic policy model and represents the reward function as a Gaussian process, where its structure is determined by its kernel function and hyperparameters θ. In order to apply Gaussian process regression to estimate a reward function, training outputs $\mathbf{u} \subset \mathbb{R}$ and corresponding feature inputs $\mathbf{X_u} \subset \mathbf{F}$ are required. But, for IRL, training outputs do not exist since we only observe actions, not the reward outputs. Due to this reason, the true training outputs $\mathbf{u}$ are also estimated during the learning phase.

The most likely values of $\mathbf{u}$ and θ can be found by maximizing the following likelihood given demonstrations:

$$\begin{aligned} &P(\mathbf{u}, \theta | \mathbf{X_u}, \mathcal{D}) \propto P(\mathcal{D}, \mathbf{u}, \theta | \mathbf{X_u}) \\ &= \left[\int_{\mathbf{r}} P(\mathcal{D}|\mathbf{r}) P(\mathbf{r}|\mathbf{u}, \mathbf{X_u}, \theta) d\mathbf{r} \right] P(\mathbf{u}|\mathbf{X_u}, \theta) P(\theta|\mathbf{X_u}), \end{aligned} \tag{2}$$

where $\mathcal{D}$ is a set of demonstrations, i.e., $\mathcal{D} = \{\zeta\}$, $\mathbf{r}$ is a reward function of entire feature space $\mathbf{F}$, $P(\mathcal{D}|\mathbf{r})$ is the likelihood of demonstrations which can be computed using (1), $P(\mathbf{r}|\mathbf{u}, \mathbf{X_u}, \theta)$ is the GP posterior of the reward function, $P(\mathbf{u}|\mathbf{X_u}, \theta)$ is the prior probability of GP, and $P(\theta|\mathbf{X_u})$ is the predefined prior for hyperparameters. $P(\mathbf{u}|\mathbf{X_u}, \theta)$ has the Gaussian distribution with a covariance matrix, whose entries are given by the following squared exponential (SE) kernel function[1]:

$$k_{se}(x_i, x_j; \theta) = \beta \exp\left(-\frac{1}{2}(x_i - x_j)^T \Lambda (x_i - x_j) \right),$$

where $\theta = \{\beta, \Lambda\}$, β is the gain of the SE kernel, and Λ is a diagonal matrix of length parameters. $P(\mathbf{r}|\mathbf{u}, \mathbf{X_u}, \theta)$ also has the Gaussian distribution. However, the complexity of $P(\mathcal{D}|\mathbf{r})$ makes the integral intractable. In order to handle the integral in (2), the approximation method [20] has been used. One approximation is to assume that $\mathbf{r}$ is deterministic, which means the predictive variance is zero. Under this assumption, the integral disappears and the reward function $\mathbf{r}(x_*)$ at an unseen input x_* becomes $\mathbf{k_{*u}} \mathbf{K_{uu}^{-1}} \mathbf{u}$, where $[\mathbf{k_{*u}}]_i = k_{se}(x_*, \mathbf{X_{u}}_i)$, $\mathbf{X_u}_i$ is the ith element of $\mathbf{X_u}$, and $[\mathbf{K_{uu}}]_{ij} = k_{se}(\mathbf{X_u}_i, \mathbf{X_u}_j)$. The resulting likelihood can be written as:

$$P(\mathcal{D}, \mathbf{u}, \theta | \mathbf{X_u}) = P(\mathcal{D}|\mathbf{r} = \mathbf{K_{Fu}} \mathbf{K_{uu}^{-1}} \mathbf{u}) P(\mathbf{u}|\mathbf{X_u}, \theta) P(\theta|\mathbf{X_u}),$$

where $[\mathbf{K_{Fu}}]_{ij} = k_{se}(\mathbf{F}_i, \mathbf{X_u}_j)$ and $\mathbf{F}_i$ is the ith element of the feature space $\mathbf{F}$. Once the likelihood is optimized, the approximated reward can be used to recover the expert's policy on the entire state space.

[1]Note that other kernel functions can be used as well.

C. Leveraged Gaussian Process

In [8], leveraged Gaussian processes (LGP) are proposed to use both positive and negative training samples for Gaussian process regression (GPR). A leveraged kernel function makes the prediction result of GPR close to positive samples and drift away from negative samples. Each training sample has its leverage value varying from -1 to $+1$, where -1 indicates a fully negative sample and $+1$ indicates a fully positive sample. A smooth leveraged kernel function proposed in [21] is defined as follows.

$$k(x_i, l_i, x_j, l_j; \theta) = \cos\left(\frac{\pi}{2} |l_i - l_j| \right) k_{se}(x_i, x_j; \theta), \tag{3}$$

where x_i and x_j are inputs, l_i and l_j are leverage values of the ith and jth inputs, respectively.

A leveraged Gaussian process (LGP) can be used to express multiple correlated Gaussian processes with the same covariance structure by defining cross-covariance function of two Gaussian processes f and g as (3). We note that $\cos\left(\frac{\pi}{2}|l_i - l_j|\right)$ controls the correlation between two Gaussian processes. For learning hyperparameters in LGP regression, derivatives of a leveraged kernel function with respect to hyperparameters are required and they can be computed as follows:

$$\begin{aligned} \frac{\partial k(x_i, x_j, l_i, l_j)}{\partial \beta} &= \frac{k(x_i, x_j, l_i, l_j)}{\beta} \\ \frac{\partial k(x_i, x_j, l_i, l_j)}{\partial \lambda_k} &= -\frac{1}{2}(x_{i,k} - x_{j,k})^2 k(x_i, x_j, l_i, l_j), \end{aligned} \tag{4}$$

where λ_k indicates the kth diagonal element of Λ and $x_{i,k}$ is the kth element of x_i.

IV. Inverse Reinforcement Learning with Leveraged Gaussian Processes

A. Benefits of Negative Demonstrations

Many existing IRL algorithms focus on using demonstrations of *what to do*. However, as mentioned in [6], the fact that experts rarely encounter fatal situations leads to the lack of information about how to overcome in a fatal situation. To handle this problem, we provide the information about *what not to do* using a negative demonstrator. Demonstrations from experts (positive demonstration) are mostly distributed near the high reward regions. However, we model a negative demonstrator having an inverted reward function compared to an expert. Hence, negative demonstrations from a negative demonstrator is more likely to be generated near low reward regions.

For example, consider the objectworld experiment [13]. Figure 1 shows examples of positive and negative demonstrations and results of IRL algorithms. In an $N \times N$ objectworld, colored objects are randomly populated where two outer colors (red and blue) exist and the state is the cell in the map. Possible actions are moving towards four adjacent grid cells or staying at the current cell. The reward function is defined such that the cell near both red and blue colored objects has a reward of $+1$, the cell near only blue colored objects has a reward of -1, and other cells

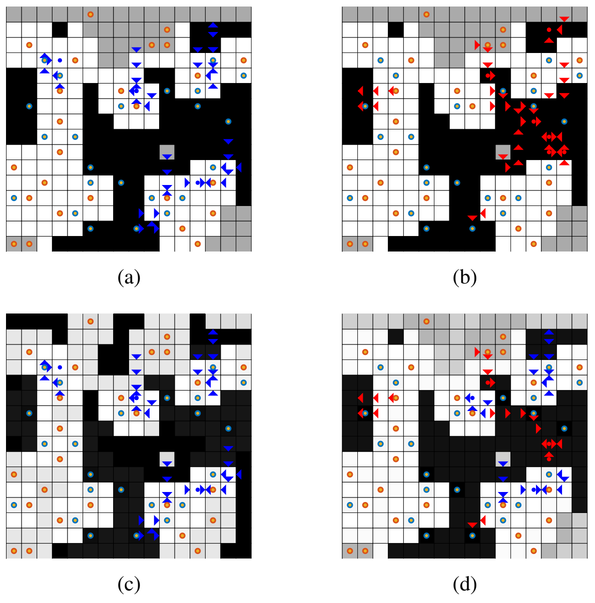


Fig. 1: A 16×16 objectworld. Colored circles are objects and the brighter the grid color is, the higher the reward is. Blue arrows are positive demonstrations and red arrows are negative demonstrations. (a) Examples of positive demonstrations. (b) Examples of negative examples. (c) The reward function reconstructed by GPIPL [13]. (d) The reward function reconstructed by the proposed method.

have a reward of 0. (More details about the objectworld are discussed in Section V.) In Figure 1(a), most of positive demonstrations are rarely distributed near low reward regions. Conversely, in Figure 1(b), negative demonstrations are more likely to be distributed near low reward regions and can provide more information about *what not to do*. In Figure 1(c) and 1(d), reward reconstruction results of GPIRL [13] and the proposed method are shown, respectively. The result from the proposed method, which uses both positive and negative demonstrations, is more accurate than GPIRL, which uses the same number of demonstrations (but only positive demonstrations). We can draw the conclusion that negative demonstrations can provide information about low reward regions and we can estimate the reward function more precisely using both positive and negative demonstrations.

B. *Demonstrator Modeling*

Before presenting the problem formulation used in this paper, we describe the model of a demonstrator with multiple levels of proficiencies. The main contribution of the proposed model is that it allows the use of negative and positive demonstrations in a single framework. A graphical representation of the proposed demonstrator model is shown in Figure 2. The proficiency of a demonstrator is represented as the leverage parameter in an LGP and we will refer to the leverage parameter as the *proficiency*. The proficiency of an expert is $+1$. On the other hand, a fully negative demonstrator has the proficiency of -1, i.e., she optimizes a reward

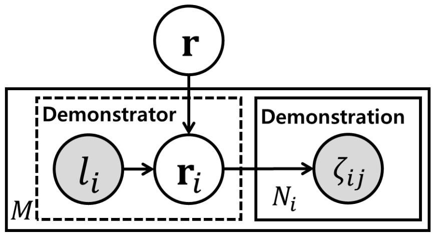


Fig. 2: A graphical representation of the proposed demonstrator model with multiple proficiencies, where $\mathbf{r}$ is the true reward function (expert's reward) with proficiency $+1$, M is the number of demonstrators, N_i is the number of demonstrations from the ith demonstrator, l_i is the proficiency of the ith demonstrator, $\mathbf{r}_i$ is the reward function of the ith demonstrator, and ζ_{ij} is the jth demonstration from the ith demonstrator.

function which is inverted from the expert's reward function. A demonstrator with the positive or negative proficiency will be referred to as a positive or negative demonstrator, respectively. Each demonstrator has a different version of the reward function depending on its proficiency, but it is correlated with the original reward function of the expert.

C. *Problem Formulation*

We consider the problem of finding the reward function from given demonstrations and proficiencies and it can be formulated as follows:

$$\underset{\mathbf{u},\theta}{\text{maximize}} \quad \log P(\mathbf{u},\theta|\mathbf{X}_\mathbf{u},\bar{\mathcal{D}}), \tag{5}$$

where the reward function is parameterized by $\mathbf{X}_\mathbf{u}$ and $\mathbf{u}$ indicating a subset of features and corresponding reward values, respectively, and $\bar{\mathcal{D}} = \{l_i, \{\zeta_{ij}\}_j^{N_i}\}_i^M$.

Here, we maximize the probability of reward outputs and hyperparameters given inputs, demonstrations, and proficiencies. We can decompose the objective function into four parts as follows.

$$\begin{aligned} &P(\mathbf{u},\theta|\mathbf{X}_\mathbf{u},\bar{\mathcal{D}}) \propto P(\bar{\mathcal{D}},\mathbf{u},\theta|\mathbf{X}_\mathbf{u}) \\ &\propto \prod_{i=1}^{M}\prod_{j=1}^{N_i}\int_{\mathbf{r}_i} P(\zeta_{ij}|\mathbf{r}_i)P(\mathbf{r}_i|l_i,\mathbf{u},\mathbf{X}_\mathbf{u},\theta)P(\mathbf{u}|\mathbf{X}_\mathbf{u},\theta)P(\theta|\mathbf{X}_\mathbf{u}), \end{aligned}$$

where l_i and $\mathbf{r}_i$ are the proficiency and reward of the ith demonstrator, respectively, and $P(\mathbf{r}_i|l_i,\mathbf{u},\mathbf{X}_\mathbf{u},\theta)$ is the LGP posterior. Since the integral cannot be analytically computed, we utilize the deterministic sparse Gaussian process approximation, similar to [13], [20]. Then, the integration can be avoided and the resulting probability can be written as below.

$$\prod_{i=1}^{M}\prod_{j=1}^{N_i} P(\zeta_{ij}|\mathbf{r}_i = \mathbf{K}_{\mathbf{F}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{u})P(\mathbf{u}|\mathbf{X}_\mathbf{u},\theta)P(\theta|\mathbf{X}_\mathbf{u}),$$

where the kernel matrices $\mathbf{K}_{\mathbf{u}\mathbf{u}}$ and $\mathbf{K}_{\mathbf{F}\mathbf{u}}$ are computed by leveraged kernel function (3) using the proficiency l_i and the expert's proficiency $+1$. The equation consists of three parts: the likelihood of demonstrations, the LGP marginal likelihood of outputs $\mathbf{u}$, and the prior on hyperparameters θ.

Finally, (5) becomes

$$\max_{u,\theta} \sum_{i=1}^{M}\sum_{j=1}^{N_i} \underbrace{\log P(\zeta_{ij}|\mathbf{r}_i)}_{\text{IRL likelihood}} + \underbrace{\log P(\mathbf{u}|\mathbf{X_u},\theta)}_{\text{LGP marginal likelihood}} + \underbrace{\log P(\theta|\mathbf{X_u})}_{\text{prior}}, \tag{6}$$

where $\log P(\zeta_{ij}|\mathbf{r}_i)$ is given in (1). The other two terms can be computed as

$$\log P(\mathbf{u}|\mathbf{X_u},\theta) = -\frac{1}{2}\mathbf{u}^T\mathbf{K_{uu}}^{-1}\mathbf{u} - \frac{1}{2}\log|\mathbf{K_{uu}}| - \frac{n}{2}\log 2\pi$$

$$\log P(\theta|\mathbf{X_u}) = -\frac{1}{2}\mathbf{tr}(\mathbf{K_{uu}}^{-2}) - \sum_k \log(\lambda_k + 1),$$

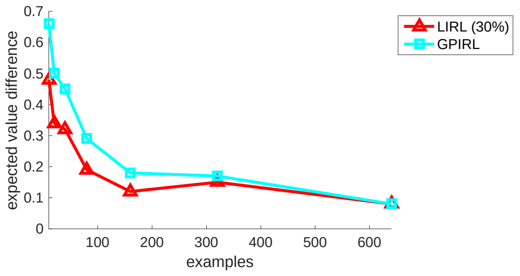
where λ_k is the kth diagonal entry of Λ. The LGP marginal likelihood and the prior on hyperparameters have an effect of regularization [13]. The IRL likelihood in (6) allows us to incorporate multiple proficiency information for learning an expert's reward function. We optimize the objective function using a gradient ascent method. The derivatives of each part in (6) can be computed by applying the chain rule and using the kernel derivatives in (4).

V. Simulations and Experiments

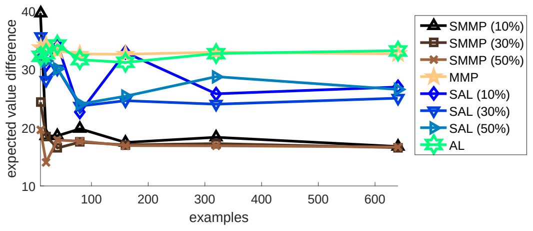
In this section, we evaluate the performance of the proposed inverse reinforcement learning algorithm by comparing against existing methods. The proposed method (LIRL) is compared with IRL algorithms using only positive demonstrations: AL [2], MMP [9], BIRL [12], MaxEnt [11], GPIRL [13], and BNP-FIRL [14]. We also compare with SSAL [10], a relatively recent algorithm, which uses both positive and negative demonstrations. The original SSAL is a semi-supervised IRL method which utilizes both labeled and unlabeled demonstrations simultaneously by clustering unlabeled demonstrations. However, in our simulation, we treat SSAL as supervised AL (SAL), which maximizes the margin between positive and negative demonstrations by providing fully labeled demonstrations. We also implement a supervised version of MMP (SMMP) with a new constraint, which enforces the resulting value function to be bigger than that of negative demonstrations in a max-margin framework.

To demonstrate the benefit of using negative demonstrations, we have prepared two types of demonstrations: positive and negative. A positive demonstration is sampled from the optimal policy with the proficiency of $+1$. A negative demonstration is sampled from the policy, which optimizes the inverted reward function of the original reward function and its proficiency is -1. The performance of each algorithm is evaluated using the expected value difference (EVD), which is the difference between the optimal value and the value obtained by following the policy learned by an IRL algorithm.

We validated the performance of IRL methods using the *objectworld* experiment [13], where the state and action space consist of an $N \times N$ grid map and five actions (up, down, left, right, or staying), respectively. Given an action, an agent successfully performs the action with probability of 0.7 or, otherwise, makes a random movement. Inside the grid map, objects with random colors are randomly deployed where each object has an inner and outer colors. Both inner and outer colors are selected from $C \geq 2$ distinct colors.



(a)



(b)

Fig. 3: Average expected value differences of different IRL algorithms from the 32×32 objectworld experiment with two colors. (a) Results of LIRL with 30% mixed demonstrations and GPIRL. (b) Results of SMMP, SAL, MMP and AL.

The reward function is defined as follows:

$$r(s) = \begin{cases} 1, & \text{if } d_1(s) < 3 \wedge d_2(s) < 2 \\ -2, & \text{if } d_1(s) < 3 \wedge d_2(s) \geq 2 \\ 0, & \text{otherwise,} \end{cases}$$

where $d_1(s)$ and $d_2(s)$ are the Euclidean distances from state s to the nearest object whose outer color is c_1 and c_2, respectively. Intuitively speaking, if the state is located near both c_1 and c_2 outer colored objects, the agent gets a positive reward. But if the agent is near only c_1 outer colored objects, it gets a negative reward. The state is represented using a binary feature $\phi(s)$ [13], where

$$\phi_i^k(s)_j = \begin{cases} 1, & \text{if } d_i^k(s) \leq j \\ 0, & \text{if } d_i^k(s) > j, \end{cases}$$

for $i = 1,\ldots,C$, $j = 1,\ldots,N$, and $k = 1,2$ where $d_i^k(s)$ indicates the Euclidean distance from state s to the nearest object whose inner ($k=1$) or outer ($k=2$) color is c_i. Hence, by combining inner and outer colors with C colors and N distance thresholds, the dimension of a feature becomes $2CN$. The reason why binary feature is utilized is that some algorithms [14], [22] only work with binary features. In our simulations, we set $N = 32$ and $C = 2$.

We have prepared several sets of demonstrations under three different ratios of the number of negative demonstrations to the number of all demonstrations: 10%, 30% and 50%. Algorithms which can handle both positive and negative demonstrations are provided with three different

Algorithms	Sample Size						
	10	20	40	80	160	320	640
LIRL (10%)	**0.42**	**0.3**	0.20	0.24	**0.12**	0.16	0.11
LIRL (30%)	0.48	0.34	0.32	**0.19**	**0.12**	0.15	**0.08**
LIRL (50%)	0.79	0.36	**0.17**	0.43	0.23	**0.08**	0.13
SMMP (10%)	39.62	18.53	18.60	19.79	17.45	18.34	16.76
SMMP (30%)	24.35	18.51	16.50	17.51	17.04	17.25	16.57
SMMP (50%)	19.59	14.09	17.86	17.68	16.91	16.88	16.68
SAL (10%)	33.00	30.69	34.28	22.62	32.86	25.79	26.95
SAL (30%)	35.61	28.17	30.10	23.70	24.61	24.01	25.04
SAL (50%)	32.49	31.40	30.08	24.04	25.38	28.73	26.54
BIRL	14.72	15.37	15.25	13.26	12.52	12.48	12.91
GPIRL	0.66	0.50	0.45	0.29	0.18	0.17	**0.08**
BNP-FIRL	10.76	10.43	9.99	9.74	10.19	10.17	10.27
MaxEnt	18.33	15.76	15.68	14.29	13.79	13.50	13.70
MMP	33.72	34.20	33.15	32.61	32.57	32.92	32.63
AL	32.20	32.69	34.16	31.63	31.17	32.70	33.18

TABLE II: Results from the 32×32 objectworld experiment. The average EVDs from eight independent runs are shown. The best performance is marked in bold. The percentage value inside parentheses is the mixing ratio of the number of negative demonstrations to the number of total demonstrations.

sets of mixed demonstrations. Algorithms using only positive demonstrations are provided with the same number of positive demonstrations.

The average EVDs of different algorithms from eight independent runs are shown in Table II and Figure 3. Since the proposed method (LIRL) and GPIRL constantly outperforms other methods, results from two algorithms are shown in Figure 3(a). LIRL shows better performance than GPIRL given the same number of demonstrations. Moreover, the average EVD of LIRL with 160 mixed demonstrations is better than that of GPIRL with 320 positive demonstrations. Figure 3(b) shows the benefits of using negative examples when the technique is applied to other methods. SMMP and SAL, which are extensions of MMP and AL with both positive and negative demonstrations, perform better than MMP and AL. This result empirically shows that the use of negative demonstrations can enhance performance of inverse reinforcement learning. The overall results are shown in Table II.

VI. Conclusion

In this paper, a new inverse reinforcement learning algorithm is proposed. The proposed algorithm uses a leveraged Gaussian process to model a nonlinear reward function and can learn from both positive and negative demonstrations. We have also introduced a novel demonstrator model for modeling demonstrations with different proficiencies. While many existing IRL methods assume that demonstrations are generated from experts, negative demonstrations are utilized by using our demonstrator model to learn expert's reward function. To the best of our knowledge, the proposed method is the first algorithm which can learn a nonlinear reward function using both positive and negative demonstrations. In simulation, the proposed method outperforms existing IRL algorithms. Our experimental results also demonstrate the benefit of using negative demonstrations in inverse reinforcement learning.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the 21st International Conference on Machine learning*, July 2004.

[3] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[4] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, January 2015.

[5] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning." in *Proc. of the 17th International Conference on Machine Learning*, June 2000.

[6] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proc. of the 13rd International Conference on Artificial Intelligence and Statistics*. JMLR.org, may 2010.

[7] S. Ross, "Interactive learning for sequential decisions and predictions," Ph.D. dissertation, Carnegie Mellon University, 2013.

[8] S. Choi, E. Kim, K. Lee, and S. Oh, "Leveraged non-stationary gaussian process regression for autonomous robot navigation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015.

[9] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *Proc. of the 23rd International Conference on Machine learning*, June 2006.

[10] M. Valko, M. Ghavamzadeh, and A. Lazaric, "Semi-supervised apprenticeship learning," in *Proc. of the Tenth European Workshop on Reinforcement Learning*. JMLR.org, June 2012.

[11] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. of the 23rd AAAI Conference on Artificial Intelligence*. AAAI Press, July 2008.

[12] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, January 2007.

[13] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., December 2011.

[14] J. Choi and K. Kim, "Bayesian nonparametric feature construction for inverse reinforcement learning," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, August 2013.

[15] K. Shiarlis, J. Messias, M. van Someren, and S. Whiteson, "Inverse reinforcement learning from failure," in *RSS 2015: Proc. of the 2015 Robotics: Science and Systems Conference, Workshop on Learning from Demonstration*, July 2015.

[16] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proc. of the 22nd International Conference on Machine learning*, August 2005.

[17] L. E. Asri, B. Piot, M. Geist, R. Laroche, and O. Pietquin, "Score-based inverse reinforcement learning," in *Proc. of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. AMMAS, 2016, pp. 457–465.

[18] B. Burchfiel, C. Tomasi, and R. Parr, "Distance minimization for reward learning from scored trajectories," in *Proc. of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, February 2016.

[19] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon University, 2010.

[20] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. 13, pp. 1939–1959, 2005.

[21] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstration using leveraged gaussian processes and sparse constrained opimization," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016.

[22] S. Levine, Z. Popovic, and V. Koltun, "Feature construction for inverse reinforcement learning," in *Advances in Neural Information Processing Systems 23*, December 2010.