Inverse Reinforcement Learning from Failure

Kyriacos Shiarlis
Informatics Institute
University of Amsterdam
Science Park 904
Amsterdam, Netherlands
k.c.shiarlis@uva.nl

Joao Messias
Informatics Institute
University of Amsterdam
Science Park 904
Amsterdam, Netherlands
j.messias@uva.nl

Shimon Whiteson
Dept. of Computer Science
University of Oxford
Wolfson Building, Parks Rd
Oxford, United Kingdom
shimon.whiteson@cs.ox.ac.uk

ABSTRACT

Inverse reinforcement learning (IRL) allows autonomous agents to learn to solve complex tasks from successful demonstrations. However, in many settings, e.g., when a human learns the task by trial and error, *failed* demonstrations are also readily available. In addition, in some tasks, purposely generating failed demonstrations may be easier than generating successful ones. Since existing IRL methods cannot make use of failed demonstrations, in this paper we propose *inverse reinforcement learning from failure* (IRLF) which exploits both successful and failed demonstrations. Starting from the state-of-the-art *maximum causal entropy IRL* method, we propose a new constrained optimisation formulation that accommodates both types of demonstrations while remaining convex. We then derive update rules for learning reward functions and policies. Experiments on both simulated and real-robot data demonstrate that IRLF converges faster and generalises better than maximum causal entropy IRL, especially when few successful demonstrations are available.

Keywords

Inverse reinforcement learning, learning from demonstration, social navigation, robotics, machine learning.

1. INTRODUCTION

In *inverse reinforcement learning* (IRL) [10], an *apprentice* aims to learn a policy for acting in an environment, typically modelled as a *Markov decision process* (MDP), for which the reward function is not available, but successful demonstrations provided by an *expert* performing the task are given instead. An IRL algorithm tries to find a reward function that leads the apprentice to behave similarly to the expert while generalising well to situations for which expert data is not available.

Numerous IRL methods have been proposed. In [14] a structured prediction formulation using maximum margin is developed and applied to mobile robotics. In [12], a Bayesian formulation is proposed along with approximations necessary for computational tractability. Other work considers nonlinear representations of the reward function using Gaussian processes [8] or decision trees [13]. IRL has also been applied to a wide range of applications, from autonomous driving [1, 7] to socially appropriate navigation [6, 18] and training parsers for natural language processing [9].

Existing IRL algorithms learn only from successful demonstrations, i.e., from data gathered by an expert performing the task well. This is consistent with the main motivation of IRL since it allows learning in tasks where the reward cannot be trivially hard-coded. For example, the reward function that allows an agent to perform complicated manoeuvres while flying a helicopter cannot be trivially determined, but example demonstrations are easy to obtain from an expert.

In many realistic scenarios, failed demonstrations are also readily available. Consider for example tasks such as driving a car. Since humans also learn this task by trial and error, demonstrations of both successful and failed behaviour are available. Although hand-coding a reward function for this task would be infeasible, labelling each trial as successful or failed is straightforward. In addition, in many tasks, it may be easier to demonstrate failure than success. If expert demonstrations are scarce but a safe simulator is available, a non-expert can often easily demonstrate multiple failure modes, yielding data that complements the scarce successful demonstrations. This idea is explored in [16], where a robot learns to avoid people using simulations of failed avoidance. Finally, failed demonstrations may be used to explore the state-action space, an idea previously leveraged in learning from demonstration [5].

In existing IRL methods, failed demonstrations have been treated as noise [19] and filtered out in order to improve robustness. However, such methods do not actually use such demonstrations for learning.

In this paper, we introduce *inverse reinforcement learning from failure* (IRLF), which is to our knowledge the first IRL algorithm that can learn from both successful and failed demonstrations. In doing so, we address a key difficulty in IRL: the problem is typically under-constrained since many reward functions are consistent with the expert's behaviour. By exploiting failed demonstrations, our method reduces this ambiguity, resulting in faster and better learning.

To derive IRLF, we start from the state-of-the-art *maximum causal entropy IRL* [22, 20] method, which is also related to [2]. We propose a new constrained optimisation formulation that accommodates both successful and failed demonstrations while remaining convex. We then derive update rules for learning policies and reward functions.

We evaluate our algorithm on the task of social navigation for a mobile robot, using both simulated and real-robot

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

data, as well as the *Factory* problem, a well known decision making benchmark. On the simulated scenarios, our results demonstrate that IRLF generalises better than maximum causal entropy IRL when successful demonstrations are scarce, with little additional computational cost. On the real-robot data, IRLF also outperforms this baseline.

2. BACKGROUND

Most IRL methods formalise the underlying decision-making problem as a *Markov decision process* (MDP), a model of a discrete-time process wherein an agent's actions may stochastically influence its environment. In an MDP, at step t, the system (which includes the agent and its environment) is known to be in a *state* $s_t \in \mathcal{S}$; the agent selects an action $a_t \in \mathcal{A}$ and is awarded a real-valued *reward*; and the system jumps to state s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$. Formally, an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of discrete states and actions respectively, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function such that $T(s, a, s') = P(s'|s, a)$, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. Optimally solving an MDP involves finding a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ with $\pi(s, a) = P(a \,|\, s)$, that maximises the expected sum of rewards over a fixed number of decisions h. This expectation can be expressed as the *value* of policy π:

$$V^{\pi}(s) \coloneqq E\{\sum_{t=1}^{h} R(s_t, a_t) \,|\, s_1 = s\}. \tag{1}$$

Given data obtained from interacting with an MDP, RL methods seek a policy that maximises this expectation for a given reward function. By contrast, given data obtained from *expert* interactions with an MDP, IRL seeks the reward function that the expert was maximising. IRL is useful in the many situations in which the expert cannot easily formalize her objectives as a reward function. Instead, the *learner*, an autonomous agent, can learn the reward function from the expert's behaviour and use it to imitate her.

IRL is formalised as an 'incomplete' MDP $\langle \mathcal{S}, \mathcal{A}, T \rangle$, also known as an MDP/R. Though initially unknown, the reward function is parametrised by K *feature functions*, $\phi_k(s, a)$:

$$R(s, a) = \sum_{k=1}^{K} w_k \phi_k(s, a), \tag{2}$$

where $w = [w_1 \, w_2 \, \ldots \, w_k]^T$ is the weight vector that IRL aims to learn.

Since w is independent of states and actions, (1) and (2) imply a parametric form of the value function:

$$V^{\pi}(s) = \sum_{k=1}^{K} w_k \left(E\{\sum_{t=1}^{h} \phi_k(s_t, a_t) \,|\, s_1 = s\} \right) \tag{3}$$

$$=: \sum_{k=1}^{K} w_k \mu_k^{\pi, 1:h}|_{s_1 = s}, \tag{4}$$

where $\mu_k^{\pi, 1:h}|_{s_1=s}$, the *feature expectation*, is the expected accumulation of instances of feature ϕ_k between steps 1 and h under policy π given that $s_1 = s$. The step indices are omitted for simplicity when this expectation is over the full horizon $\{1, \ldots, h\}$.

The learner learns from a dataset of N trajectories, $\mathcal{D} = \{\tau_1, \tau_2, ...\tau_N\}$. Each trajectory $\tau_i = \langle (s_1^{\tau_i}, a_1^{\tau_i}), (s_2^{\tau_i}, a_2^{\tau_i}), \ldots, (s_h^{\tau_i}, a_h^{\tau_i}) \rangle$ is a state-action sequence of length h generated by the expert. Given $\mathcal{D}$, the learner can compute the *empirical feature expectation*, the average accumulated instances of each feature in $\mathcal{D}$:

$$\widetilde{\mu}_k^{\mathcal{D}} \coloneqq \frac{1}{N} \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{h} \phi_k(s_t^{\tau}, a_t^{\tau}). \tag{5}$$

Note that $\widetilde{\mu}_k^{\mathcal{D}}$ is state independent and implicitly estimates an expectation across the expert's initial state. To fairly compare the feature expectation of some policy π to $\widetilde{\mu}_k^{\mathcal{D}}$, we obtain an analogously state-independent measure of π's feature expectation by marginalising out s_1:

$$\mu_k^{\pi}|_{\mathcal{D}} \coloneqq \sum_{s \in \mathcal{S}} P_{\mathcal{D}}(s_1 = s) \mu_k^{\pi}|_{s_1 = s}, \tag{6}$$

where $P_{\mathcal{D}}(s_1 = s) = N_1(s)/N$ is the maximum likelihood estimate of the expert's initial state distribution, given that $N_1(s)$ is the number of trajectories with $s_1 = s$.

The learner aims to find w such that the distance between the vectors $\widetilde{\mu}^{\mathcal{D}} = [\widetilde{\mu}_1^{\mathcal{D}} \ \ldots \ \widetilde{\mu}_k^{\mathcal{D}}]^T$ and $\mu^{\pi}|_{\mathcal{D}} = [\mu_1^{\pi}|_{\mathcal{D}} \ \ldots \ \mu_k^{\pi}|_{\mathcal{D}}]$ is minimised according to some metric, while also generalising well to unseen initial conditions.

IRL methods are typically iterative. An initial guess for w is fed to a planner that produces a policy π that is optimal given w. Using π, trajectories are generated from the same initial states as in $\mathcal{D}$ to compute $\mu^{\pi}|_{\mathcal{D}}$, which is then compared to $\widetilde{\mu}^{\mathcal{D}}$. Finally, w is updated to reduce to distance between the two expectations, and the process repeats.

The task of finding w can be formulated as an optimisation problem that requires the expert trajectories to have maximal value [1] or that do so by some margin [14]. Alternatievely one may employ Bayesian inference [12, 4] to compute a posterior distribution over the weights. In this paper, we concentrate on methods that learn maximum-entropy policies [22, 20]. A maximum-entropy approach is attractive because it is probabilistic and thus robust to noise and randomness in the actions of the expert, and results in convex optimisation problems. The methods work by solving the following constrained optimisation problem:

$$\text{find:} \quad \max_{\pi} H(\mathcal{A}^h || \mathcal{S}^h) \tag{7}$$

$$\text{subject to:} \quad \widetilde{\mu}_k^{\mathcal{D}} = \mu_k^{\pi}|_{\mathcal{D}} \quad \forall k \tag{8}$$

$$\text{and:} \quad \sum_{a \in \mathcal{A}} \pi(s, a) = 1 \quad \forall s \in \mathcal{S} \tag{9}$$

$$\text{and:} \quad \pi(s, a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{10}$$

where $H(\mathcal{A}^h || \mathcal{S}^h)$, the *causal entropy*, is the conditional entropy of the action sequence $\mathcal{A}^h$, causally conditioned on the state sequence $\mathcal{S}^h$:

$$H(\mathcal{A}^h || \mathcal{S}^h) = -\sum_{t=1}^{h} \sum_{\substack{s_{1:t} \in \mathcal{S}^t \\ a_{1:t} \in \mathcal{A}^t}} P(a_{1:t}, s_{1:t}) \log(P(a_t|s_t)), \tag{11}$$

where:

$$\begin{aligned} P(a_{1:t}, s_{1:t}) &= P(s_{1:t-1}, a_{1:t-1}) P(s_t|s_{t-1}, a_{t-1}) P(a_t|s_t) \\ &= P(s_{1:t-1}, a_{1:t-1}) T(s_{t-1}, a_{t-1}, s_t) \pi(s_t, a_t). \end{aligned}$$

The constraints require that π consists of proper probability distributions that match the empirical feature expectations.

The optimisation problem is solved using the method of Lagrange multipliers. First, the equality constraint (8) is relaxed, yielding a Lagrangian:

$$\mathcal{L}(\pi, w) \coloneqq H(\mathcal{A}^h||\mathcal{S}^h) + \sum_{k=1}^{K} w_k(\mu_k^\pi|_\mathcal{D} - \widetilde{\mu}_k^\mathcal{D}), \tag{12}$$

and in turn a less constrained optimisation problem:

$$\text{find:} \quad \min_w \left\{ \max_\pi \left(\mathcal{L}(\pi, w) \right) \right\} \tag{13}$$

$$\text{subject to:} \quad \sum_{a \in \mathcal{A}} \pi(s, a) = 1 \quad \forall s \in \mathcal{S} \tag{14}$$

$$\text{and:} \quad \pi(s, a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{15}$$

Differentiating $\mathcal{L}(\pi, w)$ with respect to π at time t results in [20, p. 186]:

$$\nabla_{\pi(s_t,a_t)}\mathcal{L}(\pi, w) = P(s_{1:t}, a_{1:t-1}) \Bigg(- \log(\pi(a_t, s_t)) + H(\mathcal{A}^{t+1:h}||\mathcal{S}^{t+1:h}) + \sum_{k=1}^{K} w_k E \left\{ \sum_{\tau=1}^{h} \phi_k(s_t, a_t)|s_{1:t}, a_{1:t-1} \right\} \Bigg). \tag{16}$$

Setting the gradient to zero and solving for π gives:

$$\pi(s_t, a_t) \propto \exp\left(H(\mathcal{A}^{t:h}||\mathcal{S}^{t:h}) + \sum_{k=1}^{K} w_k \mu_k^{\pi,t:h}|_{s_t,a_t} \right). \tag{17}$$

Due to (4), the rightmost term in (17) is a measure of value for a given state and action, parameterised by w. [20] showed that solving (17) while satisfying (14) and (15) amounts to solving a soft Bellman equation:

$$\begin{aligned} Q_w(s, a)^{soft} &= \sum_{k=1}^{K} w_k \phi_k(s, a) + \sum_{s'} T(s, a, s') V_w(s'), \\ V_w(s)^{soft} &= \log \sum_a exp(Q_w(s, a)), \\ \pi(s, a) &= \exp(Q_w(s, a) - V_w(s)). \end{aligned} \tag{18}$$

Once π is computed for a given w, w is updated via gradient descent, by noting that:

$$\nabla_w \mathcal{L}(\pi, w) = \mu^\pi|_\mathcal{D} - \widetilde{\mu}^\mathcal{D}, \tag{19}$$

where $\mu^\pi|_\mathcal{D}$ is computed by rolling out policy π from the initial state distribution $P_\mathcal{D}(s_1)$ over h steps, and taking an expectation over the features accumulated at each step. The optimisation process terminates once the weight vector converges to the optimal solution.

3. METHOD

In this section, we propose *inverse reinforcement learning from failure* (IRLF), a novel IRL algorithm that exploits failed demonstrations. IRLF is applicable in settings where the apprentice has access to a dataset $\mathcal{F}$ of failed demonstrations, in addition to a dataset $\mathcal{D}$ of successful ones.

A critical challenge in developing such a method is deciding how to interpret failed trajectories. While successful trajectories may not be optimal, it is clear that the agent should try to imitate them, which in our case means matching feature expectations. By contrast, a failed trajectory is more ambiguous because it is unclear what about it is wrong: was the entire trajectory incorrect, or was it almost correct but wrong with respect to one particular feature? A key characteristic of the method we propose is that it works well in both these cases.

The first step is to formulate a new constrained optimisation problem analogous to (7)–(10). In addition to requiring that the feature expectations of the learned policy match the empirical expectations of $\mathcal{D}$, we now also want those feature expectations to be *dissimilar* to the empirical expectations of $\mathcal{F}$. Although successful and failed demonstrations are semantic opposites, incorporating the latter into IRL proves to be nontrivial.

A straightforward approach is to add inequality constraints to the optimisation problem:

$$|\widetilde{\mu}_k^\mathcal{F} - \mu_k^\pi|_\mathcal{F}| > \alpha_k \quad \forall k, \tag{20}$$

where $\widetilde{\mu}_k^\mathcal{F}$ is the empirical expectation of feature k according to $\mathcal{F}$, computed analogously to (5), and α_k is a variable to be maximised as part of the optimisation objective, which becomes:

$$\max_{\pi,\alpha} H(\mathcal{A}^h||\mathcal{S}^h) + \sum_{k=1}^{K} \alpha_k. \tag{21}$$

However, this formulation is problematic because (20) is not a convex constraint. In fact, this approach is equivalent to maximising the L_1 norm of $\widetilde{\mu}^\mathcal{F} - \mu^\pi|_\mathcal{F}$, which is nonconvex.

A convex relaxation of the above approach can be formulated by removing the extra constraint in (20) and instead adding to the objective the inner product of a parameter vector $\theta = [\theta_1 \ \ldots \ \theta_k]^T$ and the difference in feature expectations, yielding the following objective function:

$$\max_{\pi,\theta} H(\mathcal{A}^h||\mathcal{S}^h) + \sum_{k=1}^{K} \theta_k(\mu_k^\pi|_\mathcal{F} - \widetilde{\mu}_k^\mathcal{F}). \tag{22}$$

However, this approach is also problematic because it complicates the maximisation of the Lagrangian. We now need to find a critical point with respect to *both* π and θ, which is analytically involved and numerically expensive, since it requires calculating $\mu^\pi|_\mathcal{F}$ each time that π is updated via the soft Bellman backup in (18).

To avoid these difficulties, we propose a different formulation. The main idea is to create new equality constraints of the form $\mu_k^\pi|_\mathcal{F} - \widetilde{\mu}_k^\mathcal{F} = z_k$ with $z_k \in \mathbb{R}$, and introduce the z_k variables as terms in the optimisation objective. The complete constrained optimisation problem is thus:

$$\max_{\pi,\theta,z} \quad H(\mathcal{A}^h||\mathcal{S}^h) + \sum_{k=1}^{K} \theta_k z_k - \frac{\lambda}{2}||\theta||^2 \tag{23}$$

$$\text{subject to:} \quad \mu_k^\pi|_\mathcal{D} = \widetilde{\mu}_k^\mathcal{D} \quad \forall k \tag{24}$$

$$\text{and:} \quad \mu_k^\pi|_\mathcal{F} - \widetilde{\mu}_k^\mathcal{F} = z_k \quad \forall k \tag{25}$$

$$\text{and:} \quad \sum_{a \in \mathcal{A}} \pi(s, a) = 1 \quad \forall s \in \mathcal{S} \tag{26}$$

$$\text{and:} \quad \pi(s, a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{27}$$

where λ is a constant. Intuitively, the first term in the objective seeks to maximise π's causal entropy, while the second term balances this against maximising dissimilarity between π's feature expectations and the empirical expectations in $\mathcal{F}$; the third term regularises to discourage large values of θ.

The main advantage of this formulation is that π and θ become decoupled for a given z, making maximisation of the

Lagrangian feasible while preserving convexity. Specifically, the new Lagrangian is:

$$\mathcal{L}(\pi,\theta,z,w^{\mathcal{D}},w^{\mathcal{F}}) = H(\mathcal{A}^h||\mathcal{S}^h) - \frac{\lambda}{2}||\theta||^2 + \sum_{k=1}^{K}\theta_k z_k + \sum_{k=1}^{K} w_k^{\mathcal{D}}(\mu_k^{\pi}|_{\mathcal{D}} - \widetilde{\mu}_k^{\mathcal{D}}) + \sum_{k=1}^{K} w_k^{\mathcal{F}}(\mu_k^{\pi} - \widetilde{\mu}_k^{\mathcal{F}}|_{\mathcal{F}} - z_k). \tag{28}$$

The less constrained optimisation problem of (13)–(15) remains unchanged except that maximisation of the Lagrangian is now with respect to π, θ, and z. Next, we differentiate the new Lagrangian with respect to the primal variables, beginning with θ and z:

$$\nabla_{\theta_k}\mathcal{L}(\pi,\theta,z,w^{\mathcal{D}},w^{\mathcal{F}}) = z_k - \lambda\theta_k, \tag{29}$$

$$\nabla_{z_k}\mathcal{L}(\pi,\theta,z,w^{\mathcal{D}},w^{\mathcal{F}}) = \theta_k - w_k^{\mathcal{F}}. \tag{30}$$

Setting both derivatives to zero yields:

$$z_k = \lambda w_k^{\mathcal{F}} \text{ and } \theta_k = w_k^{\mathcal{F}}. \tag{31}$$

Plugging this back into the Lagrangian gives:

$$\max_{z,\theta}\mathcal{L}(\pi,\theta,z,w^{\mathcal{D}},w^{\mathcal{F}}) =: \mathcal{L}_{z,\theta}(\pi,w^{\mathcal{D}},w^{\mathcal{F}}) = H(\mathcal{A}^h||\mathcal{S}^h) - \frac{\lambda}{2}||w^{\mathcal{F}}||^2 + \sum_{k=1}^{K} w_k^{\mathcal{F}}(\mu_k^{\pi}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}} - \lambda w_k^{\mathcal{F}}) + \sum_{k=1}^{K} w_k^{\mathcal{D}}(\mu_k^{\pi}|_{\mathcal{D}} - \widetilde{\mu}_k^{\mathcal{D}}). \tag{32}$$

Finally, we differentiate $\mathcal{L}_{z,\theta}$ with respect to π:

$$\nabla_{\pi(s_t,a_t)}\mathcal{L}_{z,\theta}(\pi,w^{\mathcal{D}},w^{\mathcal{F}}) = P(s_{1:t},a_{1:t-1})\big(-\log(\pi(a_t,s_t)) + H(\mathcal{A}^{t+1:h}||\mathcal{S}^{t+1:h}) + \sum\nolimits_{k=1}^{K}(w_k^{\mathcal{D}}+w_k^{\mathcal{F}})E\big\{\sum\nolimits_{\tau=1}^{h}\phi_k(s_t,a_t)|s_{1:t},a_{1:t-1}\big\}\big) \tag{33}$$

$$\pi(s_t,a_t) \propto \exp\left(H(\mathcal{A}^{t:h}||\mathcal{S}^{t:h}) + \sum_{k=1}^{K}(w_k^{\mathcal{D}}+w_k^{\mathcal{F}})\mu_k^{\pi,t:h}|_{s_t,a_t}\right). \tag{34}$$

Intuitively, (34) implies that the value of π now depends on *both* Lagrangian multipliers $w^{\mathcal{D}}$ and $w^{\mathcal{F}}$. We can maximise with respect to π using a soft backup method analogous to (18) with the crucial difference that the reward function is now $\sum_{k=1}^{K}(w_k^{\mathcal{D}}+w_k^{\mathcal{F}})\phi_k(s,a)$. Using the resulting policy π^*, we can define the dual objective:

$$L^*(w^{\mathcal{D}},w^{\mathcal{F}}) \coloneqq \max_{\pi,\theta,z}\Big(\mathcal{L}(\pi,\theta,z,w^{\mathcal{D}},w^{\mathcal{F}})\Big) = H^{\pi^*}(\mathcal{A}^h||\mathcal{S}^h) - \frac{\lambda}{2}||w^{\mathcal{F}}||^2 + \sum_{k=1}^{K} w_k^{\mathcal{D}}(\mu_k^{\pi^*}|_{\mathcal{D}} - \widetilde{\mu}_k^{\mathcal{D}}) + \sum_{k=1}^{K} w_k^{\mathcal{F}}(\mu_k^{\pi^*}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}} - \lambda w_k^{\mathcal{F}}). \tag{35}$$

Finally, to solve the dual, i.e., minimise L^*, we differentiate it with respect to $w^{\mathcal{D}}$ and $w^{\mathcal{F}}$:

$$\nabla_{w_k^{\mathcal{D}}}L^*(w^{\mathcal{D}},w^{\mathcal{F}}) = \mu_k^{\pi^*}|_{\mathcal{D}} - \widetilde{\mu}_k^{\mathcal{D}}, \tag{36}$$

$$\nabla_{w_k^{\mathcal{F}}}L^*(w^{\mathcal{D}},w^{\mathcal{F}}) = \mu_k^{\pi^*}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}} - \lambda w_k^{\mathcal{F}}. \tag{37}$$

Setting (37) to zero yields:

$$w_k^{\mathcal{F}} = \frac{\mu_k^{\pi^*}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}}}{\lambda}. \tag{38}$$

(36) implies that $w_k^{\mathcal{D}}$ can be updated using gradient descent, since the value of $\mu_k^{\pi^*}|_{\mathcal{D}}$ will change in the next iteration. The minimising solution for $w_k^{\mathcal{F}}$ is found analytically. (38) shows how IRLF pushes the apprentice away from the failed demonstrations. If $\mu_k^{\pi^*}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}}$ is positive, then $w_k^{\mathcal{F}}$ will also be positive. Since $w^{\mathcal{F}}$ is part of the reward function, on the next iteration, this will encourage a new π^* that increases $\mu_k^{\pi^*}|_{\mathcal{F}} - \widetilde{\mu}_k^{\mathcal{F}}$ further. The reverse occurs when $w^{\mathcal{F}}$ is negative.

A key characteristic of IRLF is that it also handles cases where the failed trajectories are only 'partial' failures, i.e., when the failed trajectories are similar to the successful ones with respect to some features and dissimilar with respect to others. For example, suppose there are only two features and $\widetilde{\mu}_1^{\mathcal{D}}$ is similar to $\widetilde{\mu}_1^{\mathcal{F}}$ but $\widetilde{\mu}_2^{\mathcal{D}}$ is dissimilar to $\widetilde{\mu}_2^{\mathcal{F}}$. It might seem that the updates to $w_k^{\mathcal{D}}$ and $w_k^{\mathcal{F}}$ with respect to the first feature would cancel each other out, thereby preventing IRLF from successfully imitating the successful trajectories with respect to that feature. However, that is not the case.

As IRLF iterates, the gradient descent update on $w_1^{\mathcal{D}}$ will bring the model and the data closer with respect to the first feature. Due to the similarity between $\widetilde{\mu}_1^{\mathcal{D}}$ and $\widetilde{\mu}_1^{\mathcal{F}}$, the analytical update on $w_1^{\mathcal{F}}$ will be smaller. Therefore, the inluence of the failed demonstrations with respect to that feature will be small. The opposite will happen with respect to the second feature. Since, $\widetilde{\mu}_2^{\mathcal{D}}$ and $\widetilde{\mu}_2^{\mathcal{F}}$ are disimilar, the update $w_2^{\mathcal{F}}$ will have an increasingly greater influence on the overall reward associated with that feature. Thus, with respect to the second feature, IRLF will be pulled towards the successful trajectories and pushed away from the failed ones. At the same time, this will not prevent IRLF from being pulled towards the successfull trajectories with respect to the first feature. Our experiments in the next section empirically validate this characteristic of IRLF.

Another convenient characteristic of IRLF is that it does not require the feature sets for successful and failed demonstrations that make up $\mu^{\pi^*}|_{\mathcal{F}}$ and $\mu^{\pi^*}|_{\mathcal{D}}$ to be the same. A designer is therefore free to represent failed demonstrations with a different feature set that is more informative than the one used for succesful demonstrations. Exploiting this property in practice is an interesting direction for future work.

However, a potential problem with IRLF is that, because (36) is updated incrementally while (38) is solved analytically, performance may oscillate, as small changes in w_k^s may be accompanied by large changes in w_k^f . This is especially true because the feature expectations of π are influenced by the updates on both $w^{\mathcal{D}}$ and $w^{\mathcal{F}}$. Fortunately, we can make IRLF more stable by annealing λ across iterations. Specifically, we start with a large value of λ, meaning that failed demonstrations are essentially ignored, and decrease it by a factor of α_λ on each iteration until a user-specified floor λ_{min} is reached. Intuitively, IRLF begins by ignoring the failed demonstrations and gradually takes them into account more and more.

Algorithm 1 describes IRLF procedurally. First, we compute the empirical feature expectations (lines 1-2) for both datasets $\mathcal{D}$ and $\mathcal{F}$ using (5). The initial state distributions are also computed (lines 3-4) by taking the normalised counts of the initial states in the two datasets. Then, we initialise

the reward function (lines 5-8) used to find a policy (line 9). Using this policy, the model feature expectations are calculated as described by [21]. Next, we perform the updates following (36) and (38) (lines 12-13). Finally, we incrementally reduce λ (lines 14-15) to improve stability.

Algorithm 1 IRLF$(\mathcal{S}, \mathcal{A}, T, \phi, \mathcal{D}, \mathcal{F}, \alpha, \alpha_\lambda, \lambda, \lambda_{min})$

1: $\widetilde{\mu}^{\mathcal{D}} \leftarrow$ `empiricalFE`$(\mathcal{D})$ {using (5)}
2: $\widetilde{\mu}^{\mathcal{F}} \leftarrow$ `empiricalFE`$(\mathcal{F})$
3: $P^{s_1}_{\mathcal{D}} \leftarrow$ `initialStateDistribution`$(\mathcal{D})$
4: $P^{s_1}_{\mathcal{F}} \leftarrow$ `initialStateDistribution`$(\mathcal{F})$
5: $w^{\mathcal{F}}_k \leftarrow 0 \quad \forall k \in \{1, \ldots, K\}$
6: Initialize $w^{\mathcal{D}}$ randomly
7: **repeat**
8: $\quad R(s,a) \leftarrow (w^{\mathcal{D}} + w^{\mathcal{F}})^T \phi(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$
9: $\quad \pi \leftarrow$ `softPlan`$(\mathcal{S}, \mathcal{A}, T, R)$ {using (18)}
10: $\quad \mu^{\pi}|_{\mathcal{D}} =$ `calculateFE`$(\pi, T, P^{s_1}_{\mathcal{D}})$
11: $\quad \mu^{\pi}|_{\mathcal{F}} =$ `calculateFE`$(\pi, T, P^{s_1}_{\mathcal{F}})$
12: $\quad w^{\mathcal{D}} \leftarrow w^{\mathcal{D}} - \alpha(\mu^{\pi}|_{\mathcal{D}} - \widetilde{\mu}^{\mathcal{D}})$
13: $\quad w^{\mathcal{F}} \leftarrow \frac{(\mu^{\pi}|_{\mathcal{F}} - \widetilde{\mu}^{\mathcal{F}})}{\lambda}$
14: $\quad$ **if** $\lambda > \lambda_{min}$ **then**
15: $\quad\quad \lambda \leftarrow \alpha_\lambda \lambda$
16: $\quad$ **end if**
17: **until** convergence
18: **return** R, π

4. EXPERIMENTS

To evaluate IRLF, we consider three domains. Two of these concern the task of navigating a mobile robot in a social environment, a key challenge problem in social robotics [11]. Because social rules are difficult to quantify, IRL is well suited to this task [6, 18]. We first consider learning in a simulated navigation domain and then repeat the process using data gathered from experiments on a real telepresence robot.[1]

Our third domain is the *Factory* problem, a well known decision-theoretic benchmark domain[3]. We compare IRLF to the original *maximum casual entropy IRL* [22], which we henceforth refer to simply as IRL.

4.1 Simulated Navigation Domain

First, we consider a simulated navigation domain, shown in Figure 1, in which a robot (blue) navigates its environment to reach a target (red) while avoiding a moving obstacle (green). The state space contains all possible combined (x, y) positions of the obstacle and the robot as well as five possible orientations for the obstacle, with one of them being stopped. The action space is $\mathcal{A} = \{up, down, left, right, stay\}$. Binary state features are computed by discretising the displacement between the (x, y) coordinates of the robot to both the obstacle and the target into five possible values for each of the four dimensions, yielding a complete feature vector $\phi(s) \in \{0, 1\}^{20}$ for any $s \in \mathcal{S}$. The transition model for this world is deterministic. The action chosen by the robot results in the agent moving to that direction with probability 1. The moving obstacle maintains its initial orientation throughout the episode. When the obstacle moves beyond the edge of the grid, it reappears on the opposite side, while the robot remains in the same cell until a valid action is chosen.

[1]Code to replicate the experiments in Sections 4.1 and 4.2 can be found at : `https://github.com/KyriacosShiarli/IRLF_aamas2016_Replicate.git`.

Figure 1: Simulated social navigation task.

We first manually define two ground truth reward functions, $R^{\mathcal{D}^*} = w^{\mathcal{D}^*}\phi(s,a)$, $R^{\mathcal{F}^*} = w^{\mathcal{F}^*}\phi(s,a)$, described further below. Then, we sample initial test states from a uniform distribution over the state space, over which we define $P^{s_1}_{test}$, for all experimental runs. These initial conditions, along with the optimal maximum-entropy policies for $R^{\mathcal{D}^*}$ and $R^{\mathcal{F}^*}$, allow us to compute feature expectations $\widetilde{\mu}^{\mathcal{D}_{test}}$ and $\widetilde{\mu}^{\mathcal{F}_{test}}$ respectively. For each experimental run, we then sample a set of initial training states, over which we define $P^{s_1}_{train}$, and generate the respective feature expectations $\widetilde{\mu}^{\mathcal{D}_{train}}$, $\widetilde{\mu}^{\mathcal{F}_{train}}$. These feature expectations are used to learn reward functions and their corresponding policies using each algorithm under evaluation. Each learned policy π^* is then executed from initial states sampled from $P^{s_1}_{test}$ to determine $\mu^{\pi^*}|_{test}$, the feature expectations for the policy at those initial conditions. Finally, we compute the values of each policy, at those initial conditions, with respect to the two reward functions, $V^{\pi^*}_{\mathcal{D},test} = (w^{\mathcal{D}^*})^T \mu^{\pi^*}|_{test}$, $V^{\pi^*}_{\mathcal{F},test} = (w^{\mathcal{F}^*})^T \mu^{\pi^*}|_{test}$. A good algorithm will yield a high $V^{\pi^*}_{\mathcal{D},test}$ and a low $V^{\pi^*}_{\mathcal{F},test}$.

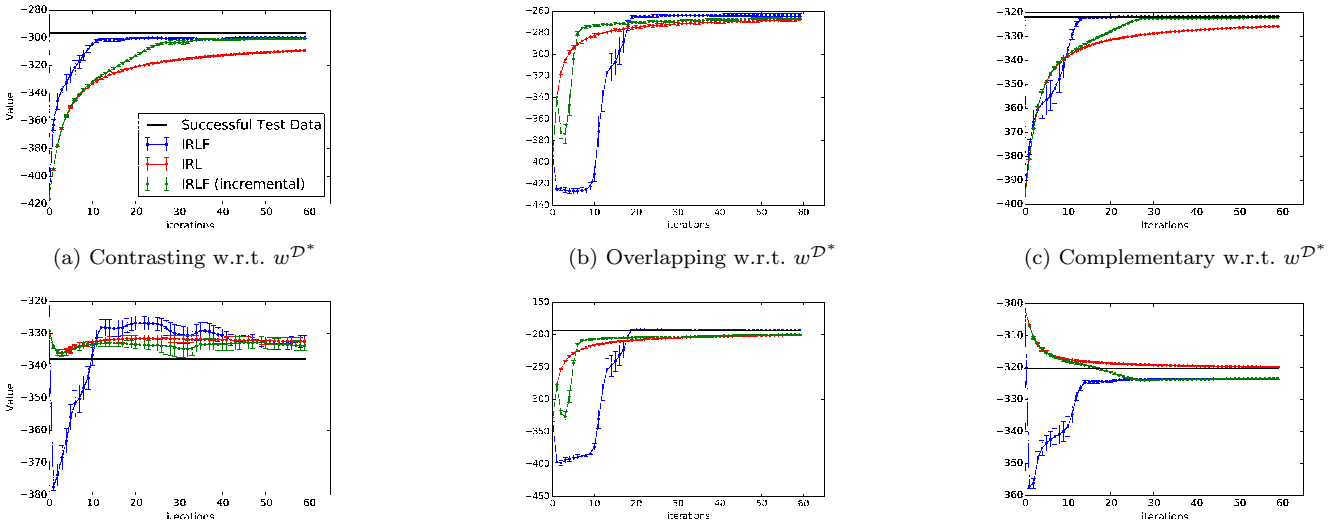
Within this domain, we consider three scenarios that differ in how $R^{\mathcal{D}^*}$ and $R^{\mathcal{F}^*}$ are defined. In the *contrasting scenario*, $w^{\mathcal{D}^*}$ rewards reaching the target and avoiding the obstacle, while $w^{\mathcal{F}^*}$ rewards being in the same cell as the obstacle. This scenario examines the value of completely failed demonstrations when the successful demonstrations already show the complete desired behaviour.

In the *overlapping scenario*, $w^{\mathcal{D}^*}$ is as before but $w^{\mathcal{F}^*}$ rewards not only colliding with the obstacle, but also reaching the target. This scenario examines the value of failed demonstrations when they are similar in some respects to the successful demonstrations.

In the *complementary scenario*, $w^{\mathcal{D}^*}$ rewards only reaching the target, while $w^{\mathcal{F}^*}$ only rewards hitting the obstacle. This scenario examines the value of failed demonstrations when the successful demonstrations do not fully disambiguate the desired behaviour.

Figures 2a and 2d compare the performance of IRLF, with and without incremental updates to λ, to that of IRL in the contrasting scenario. Both versions of IRLF successfully utilise failed demonstrations to learn better and faster than IRL in terms of $V^{\pi^*}_{\mathcal{D},test}$. They achieve similar performance to IRL in terms of $V^{\pi^*}_{\mathcal{F},test}$.

Figures 2b and 2e show results for the overlapping scenario. Even in this challenging setting, IRLF learns better than IRL, demonstrating the resilience of our method to the fact that some successful and failed trajectories might be similar and showing that our method can exploit the additional data found in failed trajectories without negative side effects.



(a) Contrasting w.r.t. $w^{\mathcal{D}^*}$ (b) Overlapping w.r.t. $w^{\mathcal{D}^*}$ (c) Complementary w.r.t. $w^{\mathcal{D}^*}$

(d) Contrasting w.r.t. $w^{\mathcal{F}^*}$ (e) Overlapping w.r.t. $w^{\mathcal{F}^*}$ (f) Complementary w.r.t. $w^{\mathcal{F}^*}$

Figure 2: Value over 60 iterations, for 20 runs in contrasting, complementary and overlapping simulated domains w.r.t. $w^{\mathcal{D}^*}$ and $w^{\mathcal{F}^*}$.

Finally, Figures 2c and 2f show similar results for the complementary scenario. The IRLF methods again perform better in terms of $V^{\pi^*}_{\mathcal{D},test}$ but now also perform better in terms of $V^{\pi^*}_{\mathcal{F},test}$. In fact, they outperform even the successful data in terms of $V^{\pi^*}_{\mathcal{F},test}$, a consequence of the complementary nature of the reward functions.

IRLF's performance with respect to $V^{\pi^*}_{\mathcal{F},test}$ (Figure 2 bottom row) illustrates how the algorithm gives priority to successful demonstrations when necessary. For the contrasting scenario, the probability of approaching the obstacle is already low for $w^{\mathcal{D}^*}$; therefore, IRL and IRLF behave similarly with respect to $w^{\mathcal{F}^*}$. In the ovelapping scenario, IRLF gives priority to reaching the target quickly, since this matches the behavior in the successful demonstrations (even though it is discouraged from doing so by the failed demonstrations). Doing so means accumulating more value in terms of $V^{\pi^*}_{\mathcal{F},test}$. In the complementary scenario, since it is possible to satisfy both objectives simultaneously (reaching the target and avoiding obstacles), IRLF finds a reward function that performs well with respect to the succesful demonstrations while at the same time having a lower value with respect to the failed demonstrations.

In all scenarios, IRLF is more stable with incremental updates than without. In the contrasting and complementary scenarios, IRLF with incremental updates learns more slowly, while in the overlapping scenario it learns faster initially but plateaus slightly lower.

In the experiments above, all methods received successful and failed demonstrations from five initial states. Hence, the results do not address how the number of succesful demonstrations given to the two learners affects their performance on the test set. In the complementary scenario, failed demonstrations are obviously important regardless of how many succesful demonstrations are available. In the other scenarios, however, it is not clear whether failed demonstrations are still useful even when successful demonstrations are abundant.

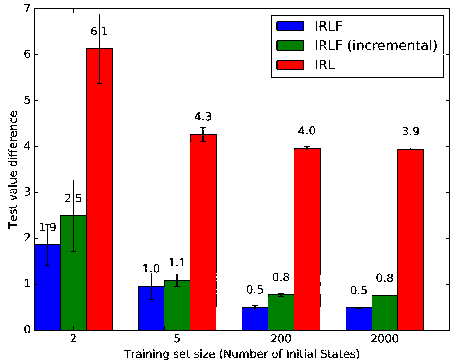
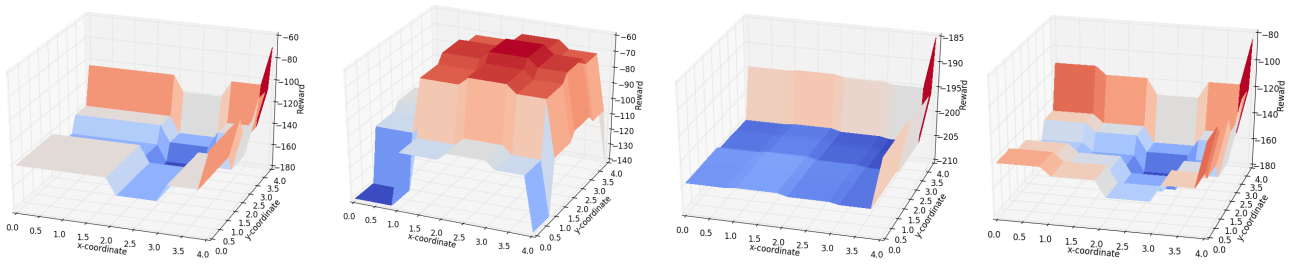


Figure 3: Performance of all three methods in the constrasting scenario for different numbers of initial states.

To test this, we repeated our experiments in the constrasting scenario but with 5, 50, 200 and 2000 initial states for successful demonstrations, while keeping the failed demonstrations to 5. For each algorithm, we plot $(w^{\mathcal{D}^*})^T \widetilde{\mu}^{\mathcal{D}_{test}}$ - $V^{\pi^*}_{\mathcal{D},test}$ after 60 iterations of the algorithm. The smaller this value, the closer the learner comes to replicating the expert on the test set of initial states. The results, shown in Figure 3, demonstrate that IRLF maintains its superiority over IRL even if the number of initial states for which demonstrations are given rises significantly.

To shed more light on IRLF's behaviour, Figure 4 plots the original and learned reward functions for IRL and IRLF for the contrasting scenario. In the original reward function $w^{\mathcal{D}^*}$ (Figure 4a), the obstacle is in cell [2,2] and the goal in cell [4,4], which explains the dips and spikes in those locations. The reward function learned by IRL (Figure 4c) is flat in the area of the obstacle. However, IRLF, by employing $w^{\mathcal{F}^*}$ (Figure 4b) learns a reward function (4d) that properly assigns a low reward to the obstacle and its surroundings. The reward function resulting from IRLF can therefore generalise better to unseen initial conditions and environments.



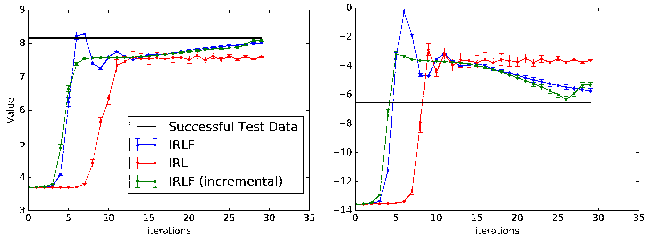
(a) $w^{\mathcal{D}^*}$ (b) $w^{\mathcal{F}^*}$ (c) IRL (d) IRLF

Figure 4: The original reward and learned reward functions for the two algorithms. Obstacle is static at [2,2]

4.2 Simulated Factory Domain

Our second simulated domain is the *Factory* benchmark problem proposed by Dearden & Boutilier [3]. In this domain, an autonomous agent is tasked with building an object according to specifications, which involves some sequence of shaping, painting, cleaning and assembly operations on its parts. Each individual operation costs some time and may have a probabilistic outcome, possibly resulting in unwanted side effects on the condition of each of the parts. Furthermore, there are precedence conditions on some of these operations – for instance, parts must be shaped before they can be assembled. The goal of this domain is to produce the object in the most cost-efficient way.

While this problem has been previously tackled from the perspective of off-line decision-theoretic planning, it is a suitable domain to demonstrate the applicability of learning from demonstration, and in particular of IRLF. In a real manufacturing problem, human demonstrations on how to properly execute the manufacturing task, as well as failed demonstrations, in which the resulting object did not meet the quality specifications, would be readily available. Furthermore, consider a situation where the conditions in the factory problem may be subject to changes (for example, in the shapes or sizes of the parts) which may affect the optimal manufacturing policy. Learning a reward function, as opposed to learning a policy from demonstration, would make it easier for an autonomous agent to adapt to these changes.



(a) Value w.r.t. $w^{\mathcal{D}^*}$ (b) Value w.r.t. $w^{\mathcal{F}^*}$

Figure 5: Value over 30 iterations, for 15 runs of learning in the Factory domain.

Our instantiation of the Factory problem follows Dearden & Boutilier's description [3] (*c.f.* Section A.2). It has 512 states, factored into 9 binary variables, and 10 actions. The expert demonstrations were drawn from the optimal policy for that domain according to its original additive rewards

Feature Index i	Proposition	$w_i^{\mathcal{D}^*}$	$w_i^{\mathcal{F}^*}$
1	AClean	0.1	1
2	BClean	0.1	-0.1
3	APainted	0.2	0.4
4	BPainted	0.2	0
5	Joined	0.4	-2

Table 1: Reward functions for the Factory domain.

($w_i^{\mathcal{D}^*}$, Table 1), while the failed demonstrations were drawn from a policy derived from a different reward function ($w_i^{\mathcal{F}^*}$, Table 1). Note that we consider only 5 features, corresponding to those state factors that have non-zero rewards. Each demonstration starts from random initial conditions and the evaluation proceeds in an identical manner to Section 4.1.

Figure 5 shows the results, which are consistent with our previous observations. Figure 5a shows that the reward function learned using IRLF is capable of yielding a policy that achieves a higher $V^{\pi^*}_{\mathcal{D},test}$ than IRL, and very close to that of the original policy. Furthermore, Figure 5b shows that IRLF is capable of achieving lower $V^{\pi^*}_{\mathcal{F},test}$ than IRL. These results not only demonstrate the stability and reliability of IRLF, but further confirm the usefulness of failed demonstrations in inverse reinforcement learning.

4.3 Real Navigation Domain

We also evaluate the performance of IRLF on a social navigation dataset gathered using is a commercially available telepresence robot that has been augmented with extra sensors and processors. The collected dataset consists of 47 trajectories of an expert navigating the robot towards an interaction target while avoiding a moving or standing person in a socially appropriate way. The positions of the people and the robot were accurately recorded using a motion capture system covering an area of $5m \times 5m$. A sample of the collected trajectories are shown in Figure 6.

For safety reasons, the failed dataset $\mathcal{F}$, which consists of 59 trajectories, was collected by driving the robot in a simulator in a way that intentionally violates social norms, e.g., running into, following, or standing next to people.

To model this domain as an MDP, we define a state space S consisting of discretised positions for both the obstacle and the robot, as well as the discretised orientation of the obstacle, yielding a total of 837,808 states. The action space consists of eight possible directions for robot movement, and an action that stops the robot. Using $\mathcal{D}$, we learned a fac-

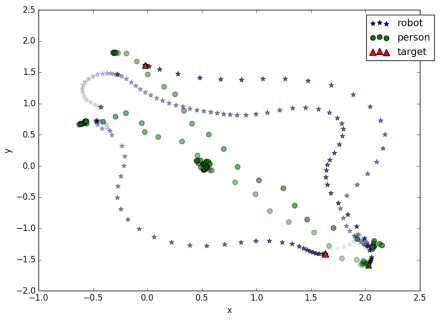


Figure 6: A sample of trajectories from the real navigation domain data set. Darker dots occur later in time.

tored stochastic transition function T. Furthermore, we defined a feature set $\phi(s, a)$ of 1401 binary features, encoding the relative position of the robot from both the target and the obstacle, as well as their relative orientations.

Applying IRL and IRLF to this domain is straightforward. However, measuring performance is not, because we do not have access to the ground truth reward function needed to compute a policy's value. In previous work on IRL, researchers have taken multiple approaches to evaluation. Some have considered simulated domains [8, 17, 15] as in our results above, where a ground truth reward function is available. Others have considered real-world data but relied on qualitative evaluation [14], domain-specific performance measures [9] or ad-hoc performance measures [18].

Since none of these approaches is entirely satisfactory, we adopt a different approach. First, we apply IRL to both $\mathcal{D}$ and $\mathcal{F}$ and derive reward functions, $R^{\mathcal{D}}$ and $R^{\mathcal{F}}$ and their respective weights $w^{\mathcal{D}^*}$ and $w^{\mathcal{F}^*}$, which we thereafter treat as ground truth. Because MDPs are generative models, we can use these two separate reward functions to generate data while preserving access to the ground-truth reward functions that are so essential to evaluation. We are therefore able to conduct an evaluation analogous to the done in the simulated navigation domain, with the key difference that the $w^{\mathcal{D}^*}$ and $w^{\mathcal{F}^*}$ are learned from real data.

Figure 7, which shows example trajectories generated by these reward functions, confirms that $w^{\mathcal{D}^*}$ produces human-like trajectories, while $w^{\mathcal{F}^*}$ generates inappropriate behaviour. These two reward functions can therefore be considered to be two separate agents whose demonstrations IRLF uses in order to learn a single reward function in a principled manner.

Figure 8 shows the results of our experiments on the real data, averaged across 6 independent runs. These results demonstrate that both versions of IRLF substantially outperform IRL with respect to $w^{\mathcal{D}^*}$. The performance of all methods is similar with respect to $w^{\mathcal{F}^*}$, which suggests that, in this domain, imitating good behaviour is sufficient to score very low with respect to $w^{\mathcal{F}^*}$. IRLF without incremental updates to λ clearly oscillates, as expected, but still converges, while IRLF with incremental updates is stable and achieves a higher value than IRL. Overall, the consistency of these results with those of the simulated domain provides further confirmation that failed demonstrations are useful for IRL, allowing it to generalise better to new initial conditions. Furthermore, IRLF is a useful method for exploiting this data.

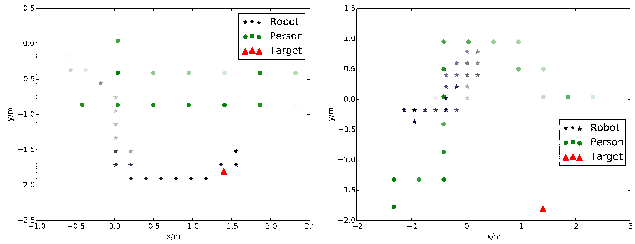
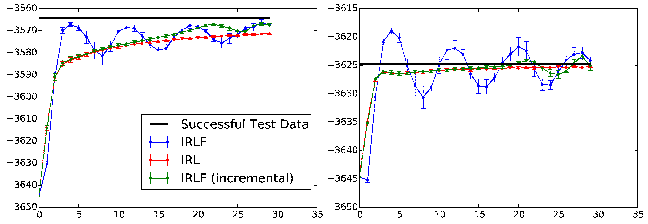


Figure 7: Sample trajectories from $w^{\mathcal{D}^*}$ (top) and $w^{\mathcal{F}^*}$ (bottom). Darker dots occur later in time.



(a) Value w.r.t. $w^{\mathcal{D}^*}$ (b) Value w.r.t. $w^{\mathcal{F}^*}$

Figure 8: Value over 30 iterations, for 6 runs on real data.

5. CONCLUSIONS AND FUTURE WORK

This paper introduced IRLF, the first inverse reinforcement learning algorithm that can learn from successful as well as failed demonstrations simultaneously. Starting from the maximum causal entropy IRL method, we proposed a new constrained optimisation formulation that accommodates both types of demonstrations while remaining convex and derived update rules for learning reward functions and policies. Our simulated navigation experiments investigated the properties of IRLF in cases where the succesful and failed demonstrations are either constrasting, overlapping, or complementary. The results in the overlapping scenario show that IRLF works even if the failed demonstrations are similar to the successful ones. We also presented results on a simulated factory domain as well as on real robot data from a social navigation task. These results clearly suggest that IRLF consistently learns faster and generalises better than the original IRL algorithm in consideration. In future work, we aim to investigate whether other IRL methods can also be extended to exploit failed demonstrations. In addition, we plan to conduct more experiments with real robots, including deploying an IRLF-learned policy on a robot.

6. ACKNOWLEDGEMENTS

This work is funded by the EC-FP7 under grant agreement no. 611153 (TERESA). We would like to thank the Social Robotics group of University Pablo Olavide (UPO) for their help in collecting and processing the data used in the experiments.

REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first International Conference on Machine Learning (ICML)*, page 1. ACM, 2004.

[2] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 897–904, 2011.

[3] R. Dearden and C. Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

[4] C. Dimitrakakis and C. A. Rothkopf. Bayesian multitask inverse reinforcement learning. In *Recent Advances in Reinforcement Learning*, pages 273–284. Springer, 2012.

[5] D. H. Grollman and A. Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[6] P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to navigate through crowded environments. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 981–986. IEEE, 2010.

[7] M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Seattle, USA*, volume 134, 2015.

[8] S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

[9] G. Neu and C. Szepesvári. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2-3):303–337, 2009.

[10] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 663–670, 2000.

[11] E. Pacchierotti, H. I. Christensen, and P. Jensfelt. Embodied social interaction for service robots in hallway environments. In *Field and Service Robotics*, pages 293–304. Springer, 2006.

[12] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. *Urbana*, 51:61801, 2007.

[13] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt. Boosting structured prediction for imitation learning. *Robotics Institute*, page 54, 2007.

[14] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[15] C. A. Rothkopf and C. Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 34–48. Springer, 2011.

[16] C. Sungjoon, K. Eunwoo, L. Kyungjae, and O. Songhwai. Leveraged non-stationary gaussian process regression for autonomous robot navigation. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2015.

[17] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2007.

[18] D. Vasquez, B. Okal, and K. O. Arras. Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison. In *Proceedings of Intelligent Robots and Systems (IROS 2014)*, pages 1341–1346. IEEE, 2014.

[19] J. Zheng, S. Liu, and L. M.-S. Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proceedings of the National Conference on Artificial Intelligence, 28th AAAI Conference on Artificial Intelligence, AAAI 2014, 26th Innovative Applications of Artificial Intelligence Conference, IAAI 2014 and the 5th Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, Quebec City, Canada*, page 2198, 2014.

[20] B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.

[21] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. The principle of maximum causal entropy for estimating interacting processes. *Information Theory, IEEE Transactions on*, 59(4):1966–1980, 2013.

[22] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.