Machine Learning (2021) 110:2541–2576
https://doi.org/10.1007/s10994-020-05939-8

Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems

Amarildo Likmeta[1,2] · **Alberto Maria Metelli**[1] · **Giorgia Ramponi**[1] · **Andrea Tirinzoni**[1] · **Matteo Giuliani**[1] · **Marcello Restelli**[1]

Received: 15 March 2020 / Revised: 17 September 2020 / Accepted: 22 December 2020 /
Published online: 14 March 2021

Abstract
In real-world applications, inferring the intentions of expert agents (e.g., human operators) can be fundamental to understand how possibly conflicting objectives are managed, helping to interpret the demonstrated behavior. In this paper, we discuss how inverse reinforcement learning (IRL) can be employed to retrieve the reward function implicitly optimized by expert agents acting in real applications. Scaling IRL to real-world cases has proved challenging as typically only a fixed dataset of demonstrations is available and further interactions with the environment are not allowed. For this reason, we resort to a class of truly batch model-free IRL algorithms and we present three application scenarios: (1) the high-level decision-making problem in the highway driving scenario, and (2) inferring the user preferences in a social network (Twitter), and (3) the management of the water release in the Como Lake. For each of these scenarios, we provide formalization, experiments and a discussion to interpret the obtained results.

Keywords Inverse reinforcement learning · Model-free IRL · Truly batch IRL · IRL for real life · Multiple experts IRL · Non-stationary IRL

1 Introduction

Reinforcement learning (RL, Sutton and Barto 2018) is nowadays an established approach to address a variety of real-world sequential decision making problems. Successful results have been achieved in numerous fields such as robotics (eg., Kober et al. 2013; Levine et al. 2016), recommender systems (eg., Shani et al. 2005; Warlop et al. 2018), financial trading (eg., Dempster and Romahi 2002; Nevmyvaka et al. 2006; Buehler et al. 2019), and autonomous driving (eg., Kiran et al. 2020).

Editors: Yuxi Li, Alborz Geramifard, Lihong Li, Csaba Szepesvari, Tao Wang.

✉ Amarildo Likmeta
amarildo.likmeta2@unibo.it

[1] Politecnico di Milano, Milan, Italy

[2] Università di Bologna, Bologna, Italy

The crucial component of any application of RL is the definition of the *reward function*, which evaluates the quality of the agent's action in each state. In real-world scenarios, it is often difficult to design a suitable reward function, able to induce the desired behavior. This is because the reward function is a *succinct* representation of the task (Sutton and Barto 2018), more abstract and connected to "what" objectives (or *intentions*) the agent is optimizing rather than "how". Indeed, it is typically easier to observe the behavior of an expert agent, possibly a human operator, who plays an optimal policy w.r.t. an unknown reward function. The goal of Inverse reinforcement learning (IRL, Ng and Russell 2000a; Abbeel and Ng 2004) is to recover a reward function that explains the expert's behavior. IRL can be of enormous importance in real-world applications as it might help justify and interpret the expert's choices and identify some trade-offs that a hypothetical human operator makes, even implicitly. Even more than RL, IRL roots its natural motivations in real-life applications. Indeed, the experts are usually humans and the demonstrations come from observing the human who is performing the task. While *imitating* human behavior is relatively simple, *interpreting* its decisions is a rather complex task, also considering that for a human being to communicate precisely these motivations might be hard.

IRL belongs to the broader class of Imitation Learning (IL, Osa et al. 2018) algorithms, whose high-level purpose is to "learn from demonstrations". While IRL has the goal of producing a reward function, other techniques, such as Behavioral Cloning (BC, Argall et al. 2009), are meant to output an *imitating policy*, i.e., a policy that performs actions similarly, in some metric sense, to those demonstrated by the expert. Although BC is typically simpler and can be cast into a supervised learning problem, the produced policy is typically non-transferable to different environments. Instead, the reward function generated by an IRL method encodes the general expert's intentions and, therefore, can be employed even under shifts in the environment dynamics. Thus, contrary to the imitating policy, such a reward function can be employed to perform forward RL in the original environment, *transferred* to different domains, or used in simulation. For a detailed review of the state of the art in IL refer to the recent survey (Osa et al. 2018).

Despite its potential benefits, scaling IRL to real-world applications has historically demonstrated to be more challenging than RL. The most widespread applications are limited to the domains where the environment can be accessed or based on simulation, such as robotics (eg., Ratliff et al. 2006), path planning (eg., Ziebart et al. 2008; Boularias et al. 2011), or simulated car driving (eg., Abbeel and Ng 2004). The fundamental reasons behind this slower development can be ascribed to the peculiar requirements needed for applying IRL to real-world scenarios, which are frequently not met by common IRL algorithms. Those requirements can be summarized as follows:

- *Batch setting*. When dealing with a real application, we cannot always assume to have access to the environment. Thus, we must account for the fact that only a batch of demonstrations collected by observing the expert is available. Further interaction with the environment might be impossible, even for just collecting additional data.
- *Model-Free setting*. In addition to the batch requirement, in real-world applications, no model of the environment dynamics is usually available (even if available it might be overly simplified to be used effectively) and no interaction is allowed to learn it implicitly or explicitly.

Consequently, the range of IRL algorithms that can be actually employed for these applications is rather small, which we refer to as *truly batch model-free*. At the best of our knowledge, they are limited to two categories: the ones that make use of structured

classification (Klein et al. 2012, 2013) and those based on the policy gradient (Pirotta and Restelli 2016; Metelli et al. 2017; Tateo et al. 2017; Ramponi et al. 2020). The previous requirements are necessary for most real-world scenarios; however, there may be additional challenges:

– *Multiple experts*. The available data might come from different experts (e.g., different human operators), possibly by playing different policies and/or optimizing different objectives. Therefore, the IRL algorithm should be able to group/cluster agents based on the demonstrated intentions.
– *Non-Stationarity*. The environment in which the data collection process is carried out might change over time as well as the policy demonstrated by the expert. Thus, a viable IRL method must identify the time points at which the agent's intention changes and deal with them appropriately.

In this paper, we present three case studies of IRL in real-world scenarios. We employ Σ-GIRL (Ramponi et al. 2020), a newly introduced batch model-free IRL approach that is based on the policy gradient and extends GIRL (Pirotta and Restelli 2016), taking into account the uncertainty of the gradient estimates, that is presented in Sect. 3.1. Then, we introduce two extensions of Σ-GIRL: the first one for dealing with the multiple-intention setting (MI-Σ-GIRL, Sect. 3.2) that was already introduced in Ramponi et al. (2020) and the second one to address the non-stationarity of the reward function (NS-Σ-GIRL, Sect. 3.3), which is a novel algorithmic contribution of this work. The subsequent sections are devoted to the illustration of the case studies. For each of them, we present the setting, the modelization, the design of the reward function class, the experimental results and their interpretation. We start with two scenarios in which we address the problem of IRL from multiple experts. In Sect. 5, we aim at inferring the intentions of humans driving along the highway; while in Sect. 6, we consider multiple Twitter users that act in the social network by reposting tweets. Then, we move to a case study in which we tackle the non-stationarity of the expert's objectives. This application, presented in Sect. 7, consists in recovering the intentions of a human operator in charge of controlling the water release of the Como Lake dam. Finally, we present in Sect. 8 a discussion of the obtained results, highlighting the strengths and weaknesses of our approach and possible open questions.

2 Preliminaries

In this section, we introduce the basic concepts about sequential decision-making problems (Sect. 2.1), we formalize the RL and IRL problems (Sect. 2.2), and we introduce the specific parametric setting we will employ (Sect. 2.3). Given a set $\mathcal{X}$, we denote with $\mathscr{P}(\mathcal{X})$ the set of all probability distributions over $\mathcal{X}$.

2.1 Sequential decision-making

We model the agent-environment interaction by means of a Markov Decision Process (MDP, Puterman 1994). An MDP is a 6-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space respectively, $P : \mathcal{S} \times \mathcal{A} \to \mathscr{P}(\mathcal{S})$ is the transition model that for each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ provides the probability distribution of the next state $P(\cdot|s, a)$, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that provides

the reward $R(s, a)$ collected by the agent when performing an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, $\gamma \in [0, 1]$ is the discount factor, and $\mu \in \mathscr{P}(\mathcal{S})$ is the probability distribution of the initial state. We denote with $\mathcal{M} \setminus R$ the MDP devoid of the reward function and with $\mathcal{R} = \{R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}\}$ being the set of all reward functions for a given MDP $\mathcal{M}$.

The behavior of an agent acting in an MDP $\mathcal{M}$ is modeled by means of a Markovian stationary *policy* $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$ that provides for each state $s \in \mathcal{S}$ the probability distribution of the action played by the agent $\pi(\cdot|s)$. A policy π is deterministic if it prescribes a single action for each state. We denote with Π the set of all Markovian stationary policies.

The execution of a policy $\pi \in \Pi$ in an MDP $\mathcal{M}$ generates a sequence of state-action pairs denoted by $\tau = (S_0, A_0, \dots, S_{T-1}, A_{T-1}, S_T)$ and called *trajectory* such that $S_0 \sim \mu$, $A_t \sim \pi(\cdot|S_t)$, $S_{t+1} \sim P(\cdot|S_t, A_t)$ for all $t \in \{0, \dots, T-1\}$ and T denotes the trajectory length.

2.2 Reinforcement learning and inverse reinforcement learning

We now focus on the formalization of the reinforcement learning (RL, Sutton and Barto 2018) and the inverse reinforcement learning (IRL, Ng and Russell 2000a) problems.

Let $\mathcal{M} \setminus R$ be an MDP without reward function, given a policy $\pi \in \Pi$ and a reward function $R \in \mathcal{R}$, we define the *expected return* $J_{\mathcal{M}}(\pi, R)$ as the expected discounted sum of the rewards collected by executing π in the environment:

$$J_{\mathcal{M}}(\pi, R) = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu \\ A_t \sim \pi(\cdot|S_t) \\ S_{t+1} \sim P(\cdot|S_t, A_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t R(S_t, A_t) \right], \tag{1}$$

where we made the unusual choice of making explicit the dependence on the reward function R, which will turn useful in the following. For a fixed reward function $R \in \mathcal{R}$, we can look at the expected return $J_{\mathcal{M}}(\pi, R)$ as an index of the performance of a policy $\pi \in \Pi$ in the MDP $\mathcal{M}$. This viewpoint directly leads to the standard formulation of the RL problem.

(RL Problem) *Let $\mathcal{M} \setminus R$ be an MDP without reward function and let $R^E \in \mathcal{R}$ be a reward function. The RL problem consists in finding an optimal policy, i.e., any policy $\pi^*_{R^E} \in \Pi$ maximizing the expected return $J_{\mathcal{M}}(\pi, R^E)$*:

$$\pi^*_{R^E} \in \mathop{arg\,max}_{\pi \in \Pi} J_{\mathcal{M}}(\pi, R^E). \tag{2}$$

We made explicit the dependence of the optimal policy $\pi^*_{R^E}$ on the reward function R^E since different reward functions may induce different optimal policies. The problem presented above admits, in general, multiple solutions (Sutton and Barto 2018), although a deterministic Markovian stationary optimal policy always exists (Puterman 1994). Typically, when tackling the RL problem, we are interested in finding just *one* optimal policy and not the whole set of optimal policies.

In the IRL setting, however, we take a different perspective. We are given an expert's policy π^E, i.e., the policy of an agent who behaves optimally w.r.t. some unknown reward function R^E. Our goal consists in finding a reward function, not necessarily equal to R^E, such that π^E turns out to be an optimal policy. We will refer to these reward functions as *compatible*.

(IRL Problem) *Let $\mathcal{M} \setminus R$ be an MDP without reward function and let $\pi^E \in \Pi$ be an expert policy. The IRL problem consists in finding a compatible reward function, i.e., any reward function $R^*_{\pi^E} \in \mathcal{R}$ that makes the expert's policy π^E optimal:*

$$R^*_{\pi^E} \in \left\{ R \in \mathcal{R} : \pi^E \in \underset{\pi \in \Pi}{arg\,max}\ J_{\mathcal{M}}(\pi, R) \right\}. \tag{3}$$

Like the RL problem, the IRL problem admits multiple solutions. However, in the IRL setting the multiplicity of solutions is more critical, leading to the well-known *ambiguity problem* (Ng and Russell 2000a, b). For instance, the constant reward function $R(s,a) = c \in \mathbb{R}$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ makes any policy (so also π^E) an optimal policy. Clearly, among all the possible reward functions that make π^E optimal, not all of them have the same ability to "discriminate", i.e., to capture that variations of π^E must be suboptimal.[1] As a consequence, assessing the quality of a reward function is a challenging task, especially when, as in real-world scenarios, it is not possible to use the recovered reward to perform forward learning.

In practice, however, the RL problem cannot be solved exactly as the dynamics of the environment modeled by P and the reward function R are unknown. Thus, interaction with the environment is necessary to learn the optimal policy. Similarly, in the IRL setting the expert's policy π^E is unknown, but a set of demonstrated trajectories $D = \{\tau_i\}_{i=1}^n$ generated by running π^E in the environment $\mathcal{M}$ is usually available.

2.3 Parametric setting with linear reward

In many real-world scenarios, especially when dealing with continuous state spaces (and possibly continuous action spaces), it is convenient to resort to a parametric representation of the policy space (Deisenroth et al. 2013). More formally, a policy $\pi_{\boldsymbol{\theta}}$ belongs to a space of parametric differentiable policies, defined as:[2]

$$\Pi_{\Theta} = \left\{ \pi_{\boldsymbol{\theta}} : \mathcal{S} \to \mathscr{P}(\mathcal{A}),\ \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d \right\}, \tag{4}$$

where Θ is the policy parameter space. As in Pirotta and Restelli (2016), we restrict our treatment of IRL to the case in which the expert's policy π^E can be represented within Π_{Θ}, i.e., there exists $\boldsymbol{\theta}^E$ such that $\pi^E(\cdot|s) = \pi_{\boldsymbol{\theta}^E}(\cdot|s)$ almost surely for all $s \in \mathcal{S}$.

Similarly, we model the reward function as a parametric mapping $R_{\boldsymbol{\omega}}$, and we enforce the additional constraint of being a linear mapping defined in terms of a feature function $\boldsymbol{\phi}$. More formally, we define a space of linear reward functions as:

$$\mathcal{R} = \left\{ R_{\boldsymbol{\omega}} = \boldsymbol{\omega}^T \boldsymbol{\phi} : \boldsymbol{\omega} \in \mathbb{R}^q_{\geq 0},\ \|\boldsymbol{\omega}\|_1 = 1 \right\}, \tag{5}$$

where $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^q$ is a (state-action) feature function. The simplex constraint on the reward weights $\boldsymbol{\omega}$ (i.e., $\boldsymbol{\omega} \in \mathbb{R}^q_{\geq 0}$ and $\|\boldsymbol{\omega}\|_1 = 1$) allows to avoid the ambiguity of rescaling rewards by a constant (Pirotta and Restelli 2016).[3]

[1] This problem has been partially formalized in the notion of *policy rank* (Metelli et al. 2017).

[2] The differentiability requirement will be necessary for employing policy gradient methods (Sutton et al. 2000; Peters and Schaal 2008).

[3] For any $\alpha \in \mathbb{R}_{>0}$, the reward functions R and αR induce the same optimal policies.

In this setting, we abbreviate the expected return $J_{\mathcal{M}}(\pi_{\boldsymbol{\theta}}, R_{\boldsymbol{\omega}})$ as $J_{\mathcal{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$, highlighting the dependence on the policy parameters $\boldsymbol{\theta}$ and on the reward parameters $\boldsymbol{\omega}$. Exploiting the linearity of the reward function, the expected return decomposes as:

$$J_{\mathcal{M}}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu \\ A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t) \\ S_{t+1} \sim P(\cdot|S_t, A_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t R_{\boldsymbol{\omega}}(S_t, A_t) \right] = \boldsymbol{\omega}^T \boldsymbol{\psi}(\boldsymbol{\theta}), \tag{6}$$

where $\boldsymbol{\psi}(\boldsymbol{\theta})$ denotes the *feature expectations* (Abbeel and Ng 2004), which are defined in terms of the feature function $\boldsymbol{\phi}$ and on the played policy $\pi_{\boldsymbol{\theta}}$ as:

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu \\ A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t) \\ S_{t+1} \sim P(\cdot|S_t, A_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t \boldsymbol{\phi}(S_t, A_t) \right]. \tag{7}$$

Thus, the expected return is a linear combination, through the weights $\boldsymbol{\omega}$, of the feature expectations. This view allows $J_{\mathcal{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$ to be interpreted as a linear scalarization of a multi-objective problem, in which the different objectives (or intentions in the IRL jargon) are represented by $\boldsymbol{\psi}(\boldsymbol{\theta})$.

3 Gradient-based inverse reinforcement learning

In this section, we revise the class of IRL algorithms, named truly batch model-free, which employ techniques based on the policy gradient (Sutton et al. 2000; Peters and Schaal 2008) to recover the reward function optimized by the expert (eg., Pirotta and Restelli 2016; Metelli et al. 2017; Tateo et al. 2017; Ramponi et al. 2020). The main advantage of these approaches that make them suitable for tackling real-world scenarios is that they do not need to have access to the environment (or to a model of it) and are able to output a reward function using only a dataset of trajectories generated by the expert's policy. Unlike widely known IRL methods, they do not need to solve the forward RL problem in order to assess the quality of each candidate reward function (thus saving a lot of computational time, especially in complex and high-dimensional RL problems) and no interaction is necessary to collect additional data. If $\pi_{\boldsymbol{\theta}} \in \Pi_{\Theta}$ is differentiable w.r.t. to its parameters $\boldsymbol{\theta}$, the policy gradient can be expressed as (Sutton et al. 2000; Peters and Schaal 2008):

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) = \mathop{\mathbb{E}}_{\substack{S_0 \sim \mu, \\ A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t), \\ S_{t+1} \sim P(\cdot|S_t, A_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t R_{\boldsymbol{\omega}}(S_t, A_t) \sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_l|S_l) \right] = \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega},$$

where $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) = \left(\nabla_{\boldsymbol{\theta}}\psi_1(\boldsymbol{\theta})| \dots |\nabla_{\boldsymbol{\theta}}\psi_q(\boldsymbol{\theta})\right) \in \mathbb{R}^{d\times q}$ is the Jacobian matrix of the feature expectations $\boldsymbol{\psi}(\boldsymbol{\theta})$ w.r.t. to the policy parameters $\boldsymbol{\theta}$. When the expert's policy $\pi_{\boldsymbol{\theta}^E} \in \boldsymbol{\Pi}_{\boldsymbol{\Theta}}$ is an optimal policy for the reward function $R_{\boldsymbol{\omega}^E}$, $\boldsymbol{\theta}^E$ is a *stationary point* of the expected return $J(\boldsymbol{\theta}, \boldsymbol{\omega}^E) = \left(\boldsymbol{\omega}^E\right)^T \boldsymbol{\psi}(\boldsymbol{\theta})$ and, thus, the gradient of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^E, \boldsymbol{\omega}^E) = \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)\boldsymbol{\omega}^E$ must vanish (first-order necessary conditions for optimality Nocedal and Wright 2006). In other words, the weight vector $\boldsymbol{\omega}^E$, associated to the reward function optimized by the expert, belongs to the *null space* of the Jacobian $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$. This leads to the condition:[4]

$$\textit{if} \quad \boldsymbol{\theta}^E \in \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} J_{\mathcal{M}}(\boldsymbol{\theta}, \boldsymbol{\omega}^E) \quad \textit{then} \quad \boldsymbol{\omega}^E \in \text{null}\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)\right). \tag{8}$$

We call all the $\boldsymbol{\omega}^E$ reward vectors that satisfy the above equation *weak compatible*, respect to Eq. (3). There are two problems that have to be addressed before applying this condition, both deriving from the fact that we have access neither to the explicit representation of the expert's policy π^E nor to the environment model, but just to a dataset $D = \{\tau_i\}_{i=1}^n$ of trajectories of length T generated by the expert's policy $\pi_{\boldsymbol{\theta}^E}$:

(1) *Behavioral cloning.* To compute the Jacobian $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$ it is necessary to have access to a parametric representation of the expert's policy, to calculate the scores $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}^E}$. Starting from the dataset of trajectories $D = \{\tau_i\}_{i=1}^n$ generated by $\pi_{\boldsymbol{\theta}^E}$, we can employ a Maximum Likelihood (ML) procedure to get an estimate $\widehat{\boldsymbol{\theta}}^E$ of the expert's policy parameters $\boldsymbol{\theta}^E$

$$\widehat{\boldsymbol{\theta}}^E \in \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{i=1}^{n}\sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(A_{i,t}|S_{i,t}). \tag{9}$$

The ML estimate is known to be consistent under mild assumptions, i.e., $\widehat{\boldsymbol{\theta}}^E \to \boldsymbol{\theta}^E$ as the number of trajectories n grows to infinity (Casella and Berger 2002). Other approaches based on Bayesian techniques (e.g., maximum a posteriori) are suitable when prior information on $\boldsymbol{\theta}^E$ is available (Tateo et al. 2017).

(2) *Jacobian estimation.* Given a policy parametrization $\boldsymbol{\theta}$, it is possible to get an unbiased estimate of the Jacobian matrix by resorting to sample-based estimators for standard policy gradient methods, such as REINFORCE (Williams 1992) and G(PO)MDP (Baxter and Bartlett 2001). For the sake of completeness, we report below the G(PO)MDP-like estimator, defined for all $u \in \{1, \dots, d\}$ and $v \in \{1, \dots, q\}$ as:[5]

[4] In principle, it is not guaranteed that the null space contains a unique vector even under the simplex constraint (Eq. (5)). The multiplicity of the solutions is typically a symptom of a bad feature design. Indeed, it is always possible to remove one or multiple features to obtain a unique weight vector fulfilling the condition.

[5] The concentration properties of this estimator, being a straightforward extension, can be derived from those of G(PO)MDP (Papini et al. 2019).

$$\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}_{uv}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \left(\sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}_u} \log \pi_{\boldsymbol{\theta}}(A_{i,l}|S_{i,l}) \right) \gamma^t \left(\boldsymbol{\phi}_v(S_{i,t}, A_{i,t}) - b_{uvt} \right), \tag{10}$$

where b_{uvt} is a *baseline* that can be employed to reduce the variance of the estimate and obtained extending the classical one employed in G(PO)MDP (Deisenroth et al. 2013 Equation 2.17) for the Jacobian:

$$b_{uvt} = \frac{\mathbb{E}\left[\left(\sum_{l=0}^{t-1} \nabla_{\boldsymbol{\theta}_u} \log \pi_{\boldsymbol{\theta}}(A_l|S_l) \right)^2 \gamma^t \boldsymbol{\phi}_v(S_t, A_t) \right]}{\mathbb{E}\left[\left(\sum_{l=0}^{t-1} \nabla_{\boldsymbol{\theta}_u} \log \pi_{\boldsymbol{\theta}}(A_l|S_l) \right)^2 \right]},$$

where the expectation is taken w.r.t. the randomness of the trajectories. Being an average of n independent trajectories, $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ concentrates around its true value $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ as n grows to infinity. Furthermore, thanks to the central limit theorem, its distribution is asymptotically Gaussian (Casella and Berger 2002).

The approximations introduced by estimating the expert's policy parameters $\boldsymbol{\theta}^E$ via behavioral cloning and by using samples to compute $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ prevent the direct application of condition (8) for the determination of the expert's weights. This is due to the fact that the estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ might result full rank even if the true Jacobian has a rank smaller than q, leading to a zero-dimensional null space. We will discuss in the following section how to deal with this problem.

3.1 Σ-gradient inverse reinforcement learning

In this section, we revise the recently presented Σ-Gradient inverse reinforcement learning (Σ-GIRL, Ramponi et al. 2020), which is able to solve the IRL problem in a fully batch model-free setting, accounting also for the uncertainty on the Jacobian estimate. The basic idea is to look at the Jacobian estimate $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ as a noisy version of the true Jacobian $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$. For this purpose, we model $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ as a Gaussian random matrix $\mathcal{N}\left(\mathbf{M}, \frac{1}{n} \boldsymbol{\Sigma}\right)$, which is justified by the central limit theorem, being the estimated Jacobian a sample mean.

Since there exists a weight vector $\boldsymbol{\omega}^E$, which defines the reward function optimized by the expert, such that $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega}^E = \mathbf{0}$, whenever $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$ is full rank, we are allowed to move its components in order to get a new estimate $\mathbf{M}$ having non-empty null space. Using the Gaussian likelihood model, we formulate the IRL problem as the problem of finding the weights $\boldsymbol{\omega}$ and the new Jacobian $\mathbf{M}$ that jointly maximize the likelihood of the estimated Jacobian.[6] This leads to the optimization problem:

$$\min_{\substack{\boldsymbol{\omega} \in \mathbb{R}^q_{\geq 0} \\ \|\boldsymbol{\omega}\|_1 = 1}} \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} \right\|^2_{\left[(\boldsymbol{\omega} \otimes \boldsymbol{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \boldsymbol{I}_d) \right]^{-1}}, \qquad (\Sigma\text{-GIRL})$$

where $\otimes$ denotes the Kronecker product and $\boldsymbol{I}_d$ is the identity matrix of order d. Clearly, we need to specify the noise model encoded by the covariance matrix $\boldsymbol{\Sigma}$. In practice, the sample covariance matrix $\widehat{\boldsymbol{\Sigma}}$ is often used in the experiments after applying some necessary

[6] Refer to Section 4 in Ramponi et al. (2020)) for the detailed derivation.

correction to enforce the well-conditioning (Ledoit and Wolf 2004). For a specific choice of $\boldsymbol{\Sigma}$, we reduce to the objective function of GIRL (Pirotta and Restelli 2016):

$$\min_{\substack{\boldsymbol{\omega} \in \mathbb{R}^q_{\geq 0} \\ \|\boldsymbol{\omega}\|_1 = 1}} \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} \right\|_2^2. \qquad \text{(GIRL)}$$

Finally, we can employ the Gaussian likelihood model to define the likelihood of dataset D, used to compute $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$, given the weight vector $\boldsymbol{\omega}$. We will denote this quantity as $p(D|\boldsymbol{\omega})$:[7]

$$p(D|\boldsymbol{\omega}) = \max_{\substack{\mathbf{M} \in \mathbb{R}^{d \times q} \\ \mathbf{M}\omega = \mathbf{0}}} \frac{\sqrt{n}}{\sqrt{(2\pi)^{dq} \det(\boldsymbol{\Sigma})}} e^{-\frac{n}{2} \left\| \mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M} \right) \right\|^2_{\boldsymbol{\Sigma}^{-1}}}, \tag{11}$$

where vec denotes the vectorization operator, that, given a matrix, recovers a vector obtained by stacking its columns.

Remark 1 (On the Suboptimality of the Expert) In principle, if no knowledge about the reward function optimized by the expert is available, we are unable to detect whether the expert is suboptimal. This is because, we can always design a reward function in which the demonstrated behavior is optimal (unless the expert's contradicts itself). Instead, if we assume that the reward function optimized by the expert lies in our class of reward functions, i.e., it is linear, and we are unable to find a weight vector making the gradient vanish, we can conclude that the expert is suboptimal. In such a case, similarly to Pirotta and Restelli (2016), instead of looking at the null space of the Jacobian, we will recover the reward that induces the minimum change in the policy parameters, i.e., the reward that better explains the expert demonstrated behavior.

3.2 Dealing with multiple experts and intentions

Algorithm 1 Multiple-Intention Σ-GIRL (MI-Σ-GIRL)

input: datasets $\mathbf{D} = (D_1, \ldots, D_m)$, number of clusters k, number of iterations N_{ite}
output: optimal parameters $\Omega = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_k, \alpha_1, \ldots, \alpha_k)$

Initialize Ω^0 randomly
for $t = 1, \ldots, N_{\text{ite}}$ **do**
  E-step: Compute $z_{ij} = \frac{\alpha_j^{t-1} p(D_i|\boldsymbol{\omega}_j^{t-1})}{\sum_{h=1}^k \alpha_h^{t-1} p(D_i|\boldsymbol{\omega}_h^{t-1})}$
  M-step: Optimize $\Omega^t \in \arg\max_{\boldsymbol{\omega}_j, \alpha_j} Q(\Omega, \Omega^{t-1})$,
    where $Q(\Omega, \Omega^{t-1}) := \sum_{j=1}^k \sum_{i=1}^m z_{ij} \left(\log \alpha_j + \log p(D_i|\boldsymbol{\omega}_j) \right)$
end for
return $\Omega^{N_{\text{ite}}}$

In several applications, we have access to demonstrations generated by multiple experts who possibly optimize different objectives (i.e., different reward functions). The corresponding IRL problem, which consists in recovering the reward function optimized by each

[7] The notation is taken from (Barratt 2018).

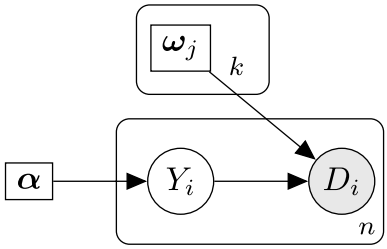


Fig. 1 Plate notation of the probabilistic model employed for the clustering procedure. α_j with $j \in \{1, \dots, k\}$ are the prior probabilities on the cluster assignment

expert, is commonly refered to as IRL about multiple intentions (MI-IRL, Babes et al. (2011)). Formally, suppose we have a set $\{E_1, \dots, E_m\}$ of m experts, each of which demonstrates a policy $\pi^{E_i} \in \boldsymbol{\Pi}_{\boldsymbol{\Theta}}$ by means of n_i trajectories, $D_i = \{\tau_1, \dots, \tau_{n_i}\}$. Furthermore, there exist $k \leq m$ unknown reward functions $\{R_{\boldsymbol{\omega}_1}, \dots, R_{\boldsymbol{\omega}_k}\}$ such that the i-th expert optimizes $R_{\boldsymbol{\omega}_{r_i}}$, where $r_i \in \{1, \dots, k\}$ are the unknown expert-intention assignments. The goal is to recover the set of k rewards together with the corresponding assignments (Fig. 1). In the remaining, we assume that we know the identity of the expert who generates each trajectory and the number of intentions k.

We now revise the approach by Ramponi et al. (2020), which extends the Σ-GIRL algorithm to the MI-IRL setting. We note that a simple solution would be to run Σ-GIRL (or any other IRL algorithm) independently on the sets of trajectories demonstrated by each different expert. However, this solution is likely to yield poor performance when each expert provides very small amounts of data, as is common in real-world scenarios. A more data-efficient solution is to cluster the given trajectories (or equivalently the experts) according to their underlying intention (Babes et al. 2011) so that it is possible to run the IRL algorithm on larger datasets (the clusters). Ramponi et al. (2020) build exactly on top of this idea. Since computing the clusters requires the intentions to be known and vice versa, the authors propose an expectation-maximization (EM) framework that maximizes the total likelihood of the data. In the E-step, the algorithm uses the current estimates of the reward weights to compute the probabilities z_{ij} that the i-th expert optimizes the j-th estimated reward. In the M-step, the algorithm uses the current probabilities z_{ij} to update the reward weights. This can be done, for each reward weight, by solving a weighted version of the Σ-GIRL objective:

$$\min_{\substack{\boldsymbol{\omega}_j \in \mathbb{R}^q_{\geq 0} \\ \left\|\boldsymbol{\omega}_j\right\|_1 = 1}} \sum_{i=1}^{m} z_{ij} n_i \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}_i(\boldsymbol{\theta}) \boldsymbol{\omega}_j \right\|^2_{\left[(\boldsymbol{\omega}_j \otimes \boldsymbol{I}_d) \boldsymbol{\Sigma}_i (\boldsymbol{\omega}_j \otimes \boldsymbol{I}_d)^T\right]^{-1}}, \quad j \in \{1, \dots, k\}. \tag{12}$$

The two steps are then repeated until convergence. The final output of the algorithm are the estimated reward weights together with the corresponding "soft" expert assignments (i.e., the probabilities z_{ij}). Refer to Algorithm 1 for the pseudocode of Multiple-Intention Σ-GIRL (MI-Σ-GIRL).

3.3 Dealing with non-stationary experts

In many real-world scenarios, the expert who controls a system (e.g., a human operator) might modify its behavior over time. This is because its objectives might change or the environment might evolve. We can interpret this phenomenon as a form of non-stationarity

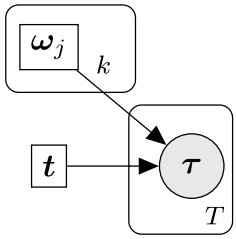


Fig. 2 Plate notation of the probabilistic model employed for the change-point detection procedure

in the expert's intentions. In this section, we formalize the problem of IRL with a non-stationary expert's reward function. Our setting assumes that we have access to a *lifelong trajectory* $\boldsymbol{\tau} = (\tau_1 | \dots | \tau_T)$ obtained from the concatenation of T trajectories $D = \{\tau_i\}_{i=1}^{T}$.[8] Within the lifelong trajectory $\boldsymbol{\tau}$ the expert displays a non-stationary behavior since it optimizes $k \leq T$ reward functions $\boldsymbol{R} = (R_{\boldsymbol{\omega}_1}, \dots, R_{\boldsymbol{\omega}_k})$, where k is referred to as *number of regimes*. In particular, there exists a set of indexes $\mathcal{T} = \{t_0, t_1, \dots, t_k\}$ with $1 = t_0 < t_1 < \cdots < t_{k-1} < t_k = T$, inducing the intervals $I_j = \{t_{j-1}, \dots, t_j - 1\}$, such that for each $j \in \{1, \dots, k\}$ the set of trajectories $D_j = \{\tau_i\}_{i \in I_j}$, made of $n_j = t_j - t_{j-1} + 1$ trajectories, are generated by the expert who optimizes the same reward function $R_{\boldsymbol{\omega}_j}$. We assume to know the number of regimes k, the subdivision of the lifelong trajectory $\boldsymbol{\tau}$ in the T trajectories $D = \{\tau_i\}_{i=1}^{T}$, but not the set of indexes $\mathcal{T}$, nor the reward functions $\boldsymbol{R}$. Clearly, we expect that for different intervals I_j not only the reward function changes, but also the policy performed by the expert (Fig. 2).

A naïve solution would be to treat this problem as a multiple-intention IRL problem in which each expert E_i generated the dataset consisting of a single trajectory $D_i = \{\tau_i\}$ for $i \in \{1, \dots, T\}$. However, this approach has at least two drawbacks. First, the estimate of the reward function will likely be very noisy since only one trajectory is available for each expert. Second, we are totally disregarding that the expert's intention changes sequentially. Thus, it would be unrealistic to cluster non-contiguous intervals.

For these reasons, we take inspiration from the *change-point detection* algorithms (Aminikhanghahi and Cook 2017) and we adapt it to the non-stationary IRL setting. Given a dataset D of trajectories and a reward weight $\boldsymbol{\omega}$ we employ the likelihood function $p(D|\boldsymbol{\omega})$ defined in Eq. (11), we define the likelihood of the lifelong trajectory $\boldsymbol{\tau}$ as the product of the likelihoods of the individual trajectories τ_i:

$$\mathcal{L}(\Omega|\boldsymbol{\tau}) = p(\boldsymbol{\tau}|\Omega) = \prod_{i=1}^{T} \sum_{j=1}^{k} p(\tau_i|\boldsymbol{\omega}_j) \mathbb{1}_{\{i \in I_j\}}, \tag{13}$$

where $\Omega = (\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_k, t_1, \dots, t_{k-1})$ is the concatenation of the parameters. Now we can derive the objective function that we seek to optimize for the parameters Ω:

[8] The granularity of the subdivision of the lifelong trajectory to the T sub-trajectories is a design choice, based on the knowledge of the environment. For instance, in the Como Lake case study, given the cycle-stationarity of the environment, each sub-trajectory is associated to one year of data. The length of the sub-trajectories determines the agent's planning horizon employed in the IRL process.

$$
\begin{aligned}
Q(\Omega) = \log \mathcal{L}(\Omega|\boldsymbol{\tau}) &= \sum_{j=1}^{k} \sum_{i=1}^{T} \mathbb{1}_{\{i \in I_j\}} \log p(\tau_i|\boldsymbol{\omega}_j) \\
&= \sum_{j=1}^{k} \sum_{i \in I_j} \log p(\tau_i|\boldsymbol{\omega}_j) \\
&= \sum_{j=1}^{k} \log p(D_j|\boldsymbol{\omega}_j),
\end{aligned}
$$

where we recall that $D_j = \{\tau_i\}_{i \in I_j}$. In order to optimize the objective function $Q(\Omega)$ we adapt the change-point detection algorithm *Opt* which employs a dynamic programming approach to determine the optimal solution to the identification of the change points $\mathcal{T}$ (Bellman 1958; Aminikhanghahi and Cook 2017; Truong et al. 2020). The adaptation of this algorithm to our non-stationary IRL problem is reported in Algorithm 2, which we name *Non-Stationary Σ-GIRL* (NS-Σ-GIRL). It is worth noting that the optimization of such objective consists in solving $O(T^2)$ IRL problems, one for each $1 \leq u < v \leq T$:

$$
\min_{\substack{\boldsymbol{\omega}_{uv} \in \mathbb{R}^q_{\geq 0} \\ \|\boldsymbol{\omega}_{uv}\|_1 = 1}} (v-u) \sum_{i=u}^{v-1} \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}_i(\boldsymbol{\theta}) \boldsymbol{\omega}_{uv} \right\|^2_{[(\boldsymbol{\omega}_{uv} \otimes \boldsymbol{I}_d) \boldsymbol{\Sigma}_i (\boldsymbol{\omega}_{uv} \otimes \boldsymbol{I}_d)^T]^{-1}}. \tag{14}
$$

Algorithm 2 Non-Stationary Σ-GIRL

input: dataset $\boldsymbol{\tau} = (\tau_1|\tau_2|\dots|\tau_T)$,number of regimes k
output: optimal parameters $\Omega = (\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_k, t_1, \dots, t_{k-1})$

```
for u = 1, ..., T − 1 do
    for v = u + 1, ..., T do
        Define D_uv = {τ_u, ..., τ_{v−1}}
        Perform BC to find the policy parameters θ_uv
        Optimize ω*_uv ∈ arg max_ω log p(D_uv|ω)
        C_1(u, v) = log p(D_uv|ω*_uv)
    end for
end for
for l = 2, ..., k − 1 do
    for u = 1, ..., T − l do
        for v = u + l, ..., T do
            C_l(u, v) = max_{u+l−1⩽t<v} {C_{l−1}(u, t) + C_1(t + 1, v)}
        end for
    end for
end for
t_k = T
for l = k, ..., 2 do
    t_{l−1} ∈ arg max_{l−1⩽t<t_l} C_{l−1}(1, t) + C_1(t + 1, t_l)
    ω_l = ω*_{t_{l−1} t_l}
end for
return Ω
```

4 Related works

In recent years, there have been several successful applications of imitation learning methods to real-world problems. Robotics is perhaps the most common example (Osa et al. 2018). In this setting, learning policies on real robots is often prohibitive due to both sample-complexity and safety reasons, while expert demonstrations are fairly simple to obtain. Due to their simplicity, behavioral cloning (BC) methods have received considerable attention. Kober and Peters (2009) trained a robotic arm to hit a ball in the table-tennis game. The arm was guided by a human expert to generate trajectories and the ball-hitting policy was learned directly via BC on these demonstrations. Englert et al. (2013) addressed the same problem but with an under-actuated robot using a model-based BC technique. Abbeel et al. (2010) trained policies to drive an RC helicopter from human-teleoperation trajectories. A similar problem was considered by Ross et al. (2013), who trained a controller for an unmanned aerial vehicle capable of avoiding obstacles (e.g., trees in a forest). Zhang et al. (2018) trained policies for several robotic manipulation tasks (e.g., grasping or pushing objects) directly from images, with demonstrations generated in virtual reality. Finn et al. (2017) used meta-learning to train image-based controllers that adapt to several manipulation tasks using only a single visual demonstration. For a thorough discussion of the applications of BC methods, we refer to the recent surveys by Hussein et al. (2017) and Osa et al. (2018).

Although IRL methods have also enjoyed many success stories in complex robotics problems, their application in this context is considerably more difficult than BC. In fact, as mentioned in the introduction, in this kind of problems, a model of the environment is hardly ever available in practice, it is difficult or unsafe to interact with the real system, and expert demonstrations are often very limited. Therefore, many traditional IRL techniques are not applicable and model-free and data-efficient (e.g., batch) methods are typically preferred. Among the notable applications, Boularias et al. (2011) used a model-free variant of MaxEnt IRL to learn the "ball-in-a-cup" task, in which a robot must swing a ball connected to a rope into a cup. The task was demonstrated by a human expert only a very small number of times and the resulting controller was shown successfully on a real robot. Bogert and Doshi (2014) proposed an IRL method for a real patrolling problem in which a robot must penetrate the perimeter patrolled by other robots inferring their intentions. Finn et al. (2016) learned about house-keeping tasks (such as moving dishes and pouring liquids) using a model-free IRL algorithm with non-linear reward functions and visual demonstrations.

Autonomous driving is another field where the application of imitation learning (and, in particular, IRL) techniques has received increasing interest. This setting presents even more complications than robotics problems and, thus, the focus is typically on learning policies in simulation. However, there have been many attempts to integrate real-world driving demonstrations and to deploy the resulting controllers to real cars. Several BC approaches have been proposed for learning end-to-end car-driving policies (which map raw sensor data to actions) directly from expert demonstrations. These approaches provide significant evidence of the capabilities of neural network-based controllers but are typically difficult to deploy on real cars due to safety and interpretability reasons. Codevilla et al. (2018) and Dosovitskiy et al. (2017) trained end-to-end image-based policies for complex urban driving domains. The trained models were evaluated in the real-world using a toy car. Similarly, Pan et al. (2017) adopted an end-to-end imitation learning method for off-road autonomous driving that was successfully tested using toy cars.

The application of IRL methods is typically on specific driving problems. Ziebart et al. (2008) considered the problem of predicting driving behavior and route preferences. The authors applied their MaxEnt IRL algorithm to a large dataset of real GPS data from different taxi cabs and showed that it was able to capture the route choices of the drivers. A similar problem was considered by Wulfmeier et al. (2017) who extended the approach of Ziebart et al. (2008) to learn non-linear reward functions. Silver et al. (2013) proposed a maximum-margin method to learn driving maneuvers from human demonstrations and successfully tested the resulting controller on a real-world vehicle. Kuderer et al. (2015) employed the MaxEnt IRL algorithm to learn driving styles from demonstrations in order to optimize the comfort perceived by passengers in the autonomous vehicle. The approach uses data obtained by recording real drivers with different driving styles.

5 Case study 1: Highway driving

5.1 IRL from multiple experts

Highway driving is a widely employed benchmark for RL and IRL algorithms, thanks to the potentially simple representation of the environment and the "few-constrained" possibility in choosing the action. Here we focus on the problem of *high-level* control, where the main decision the agent has to make is when to change lanes. This is a quite common scenario, close to the setting considered in real autonomous vehicles, where the presence of a *low-level controller*, which assures compliance with the safety distance with other vehicles, managing the speed accordingly and maintaining the center of the lane, is assumed. Therefore, the lane-change problem consists in controlling the ego vehicle on the highway and deciding when it is convenient to perform a lane change on the left to overtake, or a lane change on the right, to occupy the rightmost free lane.[9] Driver agents in this setting typically aim at proceeding along the highway as fast as possible, while displaying a "natural" behavior, respectful of driving rules.

In this case study, we consider a mixed real/simulated setting. The demonstrations are collected by human drivers, but the environment in which humans operate is simulated. We employ SUMO simulator, an open-source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks (Krajzewicz et al. 2012). SUMO focuses on high-level control of the car, integrating an internal system that controls the vehicle dynamics. This mimics the low-level controller of autonomous vehicles. Slight changes have been made to the simulator to ensure that the car-follower models employed in the simulator are aligned with low-level controllers used in real autonomous driving systems. For this reason, we believe that our setting is not significantly different from the fully real environment. Furthermore, since we are interested in recovering a reward function which is a transferable element, rather than an imitating policy, the simulated environment is less critical than in the BC case.

In this kind of driving problems, the use of classical IRL algorithms is extremely challenging, since any interaction with the environment (e.g., to learn the optimal policy for a candidate reward function) must be performed in simulation and must account for the

[9] We use the right-hand traffic rules.

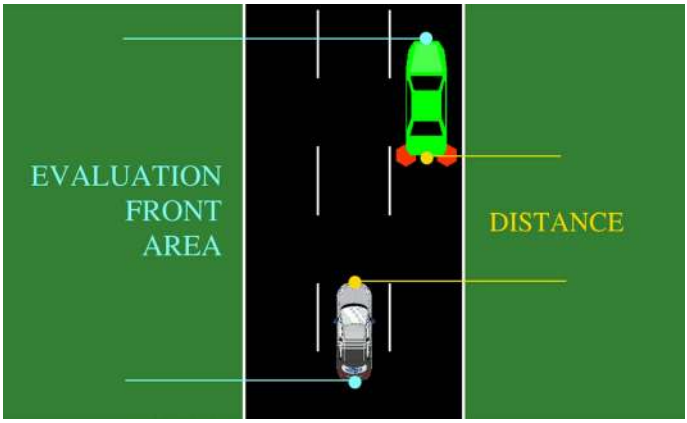


Fig. 3 Range in which the car in front is considered for front distances, highlighted in blue. The value recorded is highlighted in yellow (Color figure online)

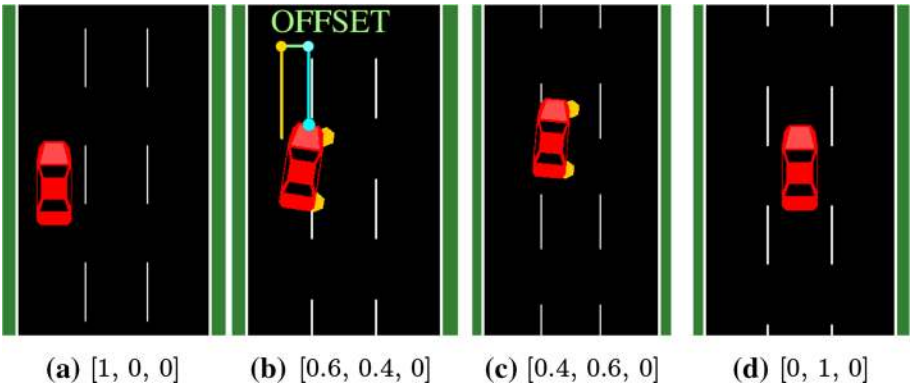


Fig. 4 The ego vehicle lane occupancy during a lane change (Color figure online)

differences with the real vehicle.[10] Σ-GIRL, on the other hand, requires only agents' demonstrations and can identify the reward function that the expert optimizes without interacting with the environment. More specifically, we consider the case where we have interactions from multiple agents and we can identify which agent each of our demonstrations belongs to. The goal is to cluster agents based on their intentions.

The immediate application of the results of IRL in this scenario is in the field of *autonomous driving*. Specifically, we can exploit the clustering of agents based on their intentions to identify agents that demonstrate unwanted behaviors. This allows removing from the dataset demonstrations that would result in an imitation policy showing these unwanted behaviors, such as unsafe driving or non-compliance with driving rules, with a possible benefit in subsequent BC applications. Furthermore, and most importantly, we can use the

[10] Our method uses trajectories collected by human experts in simulation, but we never interact with the simulator to perform forward RL. Thus, the presence of the simulator is less critical for our approach.

identified reward functions to understand the different trade-offs performed by human drivers. Consequently, those rewards can be employed to train an autonomous controller that replicates (and possibly improves) the human behavior, only using the demonstrations of agents that optimize a "safe" reward function. In the following sections, we will refer to the controlled vehicle as the *ego* vehicle.

5.2 System modeling

We present here details about the state representation considered in the lane change scenario. We focus on three-lane highways, but the state space can be generalized to an arbitrary number of lanes. The state is composed of 25 high-level features extracted by the observations of the environment. For two vehicles in each lane, one at the front and one at the rear of the ego vehicle, we record the distance from the ego vehicle. The distance considered is the one from the front bumper of the following vehicle to the rear bumper of the leading vehicle, as shown in Fig. 3. A vehicle is considered in front of the ego vehicle as long as its front bumper is in front of the rear bumper of the ego vehicle. We also record their speeds and their lateral positions inside the corresponding lanes, to know if they are making lane changes. The variables that represent the state of the ego vehicle are its speed, its position over the lanes, a flag indicating whether the ego vehicle is changing lanes, and two flags that check whether the ego vehicle has the free-left or the free-right. The free-left and the free-right are evaluated only for vehicles visible by the sensors, therefore within the visibility range, otherwise, they are true. The position of the ego vehicle in the lanes is represented as occupancy weights. For each lane of the highway, we record in what percentage the ego vehicle is in each lane, considering the offset between the front bumper of the vehicle and the center of the lane. Figure 4 shows an example of a lane change from the third lane to the second lane (lanes ordered from right to left), together with the corresponding occupancy arrays. In Fig. 4a the offset between the front bumper of the car (highlighted in blue) and the center of the lane (highlighted in yellow), is highlighted in green. In this case, the vehicle is 60% in the third lane and 40 % in the second lane.

The action space in the lane-change highway scenario consists of three actions, *car_following*, *lane_change_right* and *lane_change_left* . The *car_following* action leaves control of the car to the low-level controller which follows the planned route, cornering when necessary, but does not make lane changes. Furthermore, it controls the vehicle speed to avoid collisions and maintains a safe distance with the vehicle in front. The controller sets the safety speed considering the vehicles that are in sight of the sensors only and adjusts it respecting the maximum practicable deceleration and acceleration. The remaining two actions are the *lane changes*, left or right. These maneuvers are non-interruptible, once issued, they cannot be reverted. For more details on the environment modelling see Likmeta et al. (2020).

5.3 Reward design

The lane change scenario is a classic example of a *multi-objective task*. Humans consider several objectives, corresponding to the reward features, while driving along highways, including: (1) going as fast as possible, (2) occupying the rightmost free lane, (3) avoiding useless lane changes, and (4) keep safety distances with other vehicles. To encode these

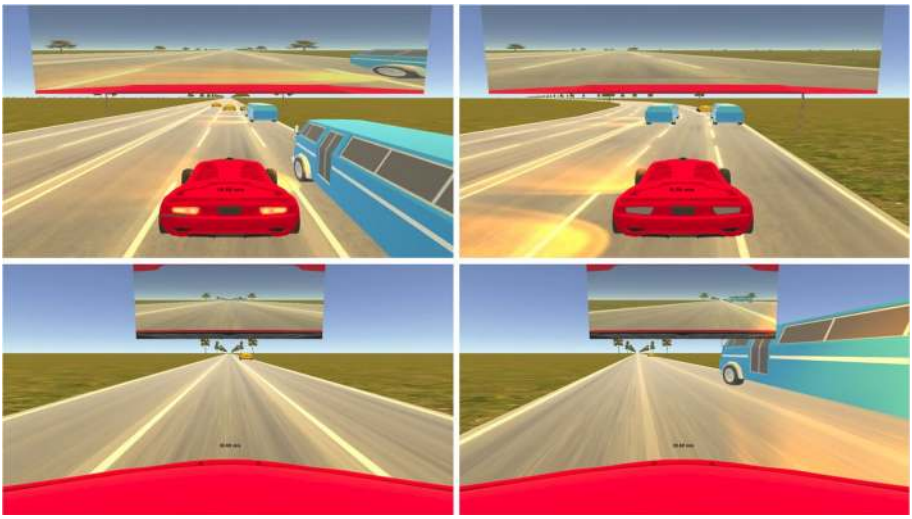


Fig. 5 3D inteface used to collect the human demonstrations

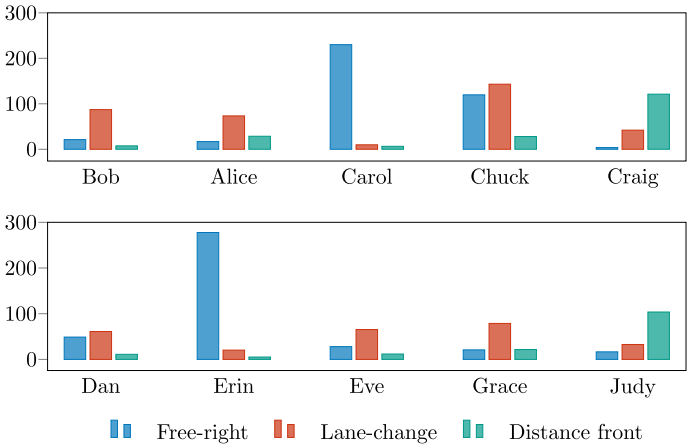


Fig. 6 Feature expectations of the human agents in the highway task

objectives we employ three reward features. All the features are meant as punishment, so they have negative values:

- *Free-right* (ϕ^R): to encode the objective of occupying the rightmost lane we use a binary feature, activated during the timesteps when the agent could perform a lane change on the right.
- *Lane-change* (ϕ^L): a binary feature is used to encode the objective of avoiding too many lane changes. Since the lane change is non-interruptable and lasts 3s (30 timesteps), this punishment is given entirely at the beginning of the lane change and has a high value (30).

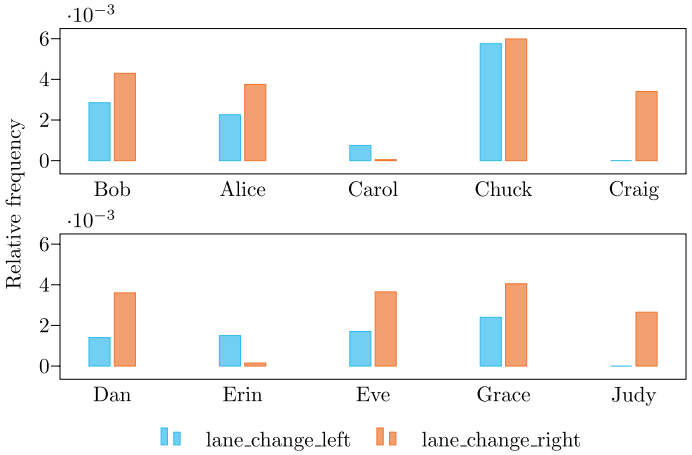


Fig. 7 Distributions of lane changes for each agent. The y-axis reports the fraction of demonstrated actions where either a lane change to the left (in blue) or one to the right (in orange) was performed (Color figure online)

– *Distance front* (ϕ^D): a feature to encode both the safety objective and the maintenance of the high-speed profile. This is a feature that incorporates the distance of the ego vehicle from the vehicle in front of it. It grows linearly with the distance to the front vehicle, the higher the distance of the ego vehicle from the vehicle in front the higher is its value. It has the highest value (0) when there is no vehicle in front. This objective also encodes the high-speed objective, since it is the low-level controller that regulates the speed of the ego vehicle when it is about to violate safety distances. Without any vehicle ahead, the ego vehicle continues to accelerate until it reaches the maximum allowed road speed.

5.4 Data description

In the SUMO simulator, we model scenarios with different road topologies and traffic intensities, randomizing the flow of vehicles, to ensure the generation of sufficiently general and realistic situations. We set the control frequency to 10 Hz for all our experiments, which means that we choose an action to be performed every 100 ms. During the simulation, SUMO provides information about the other vehicles around the ego vehicle. More specifically, we can query SUMO for the positions and velocities of all the cars in the simulation. This information is also available for the decision-making module in a real car, being provided by the sensing module. To collect the dataset we built a 3D-interface on top of the SUMO traffic simulator, connected to the traffic simulator. The 3D interface, shown in Fig. 5, was used by human drivers to collect trajectories. The dataset consists of demonstrations provided by 10 different drivers. Each set of demonstrations consists of 50 trajectories each of 400 steps, recorded at 10 Hz, resulting in trajectories equal to 40 s of driving time, for a total of 5.5 hours of driving.

The agents show different behaviors. To grasp an initial understanding of the differences, we show in Fig. 6 the feature expectations for all the agents considered. In Appendix A.1, we also show some 2D visualizations of the trajectories of some of the experts.

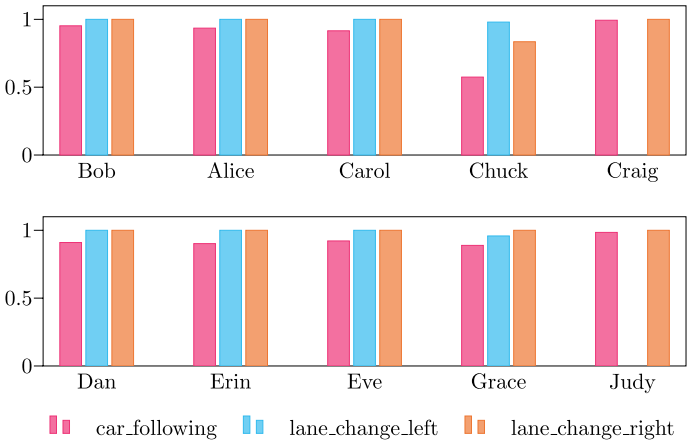


Fig. 8 Accuracy (fraction of correctly-predicted actions) of the BC models in the AD task (Color figure online)

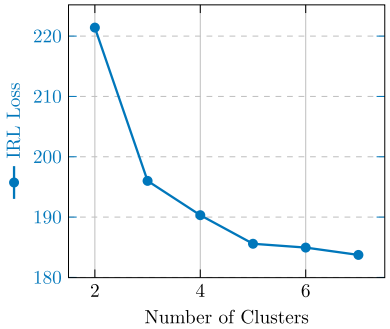


Fig. 9 IRL loss (Eq. 12) in the Highway domain as a function of the number of clusters

Finally, it is worth noting that the distribution of the actions in the dataset is highly unbalanced. We want to identify the intentions that drive human agents in changing lanes while driving, but most of the actions in the dataset are car-following. Figure 7 shows the distributions of lane changes for each agent.

5.5 Results

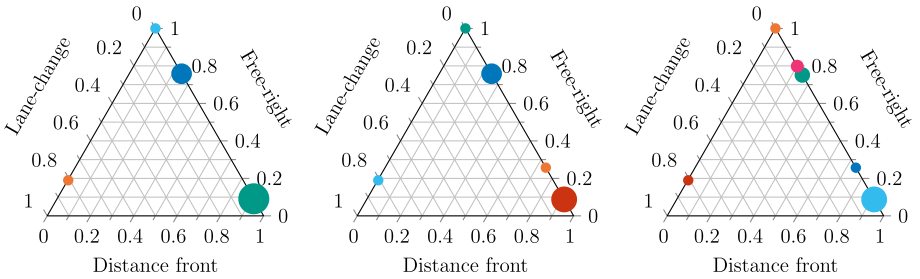
The BC phase is performed by means of a one-layer neural network, with 8 hidden units and a Boltzmann output layer to represent the policy model for the AD task. Different architectures were explored, but the simpler models were unable to accurately predict the agents' behaviors and more complex models did not offer substantial improvements. We recall that in this task, the BC dataset is highly unbalanced, with most of the actions in the dataset being car-following (NOP) and only a small portion being lane changes. To deal with this problem, we employed oversampling over the minority classes. Figure 8 shows the accuracy of all agents' policies derived via BC, for each action separately. We can notice that the BC models generally predict the agents' behaviors well, except for the 4th agent, which seems to have a more non-deterministic response to the state. For agents Craig and Judy, the column corresponding to the lane-change left is not represented since the respective agents never performed that action.

Table 1 The reward weights learned by Σ-GIRL in the AD task

	Reward features			N. agents
	Free-right	Lane-change	Distance front	
Cluster 1	0.76	0.00	0.24	3
Cluster 2	0.09	0.00	0.91	5
Cluster 3	1.00	0.00	0.00	1
Cluster 4	0.19	0.81	0.00	1

Table 2 Cluster assignment made by Σ-GIRL in the AD task

	Agents
Cluster 1	Eve, Grace, Alice
Cluster 2	Carol, Erin, Bob, Dan, Chuck
Cluster 3	Craig
Cluster 4	Judy



(a) 4 Clusters. **(b)** 5 Clusters **(c)** 6 Clusters

Fig. 10 Visualization of the weights of the clusters (Color figure online)

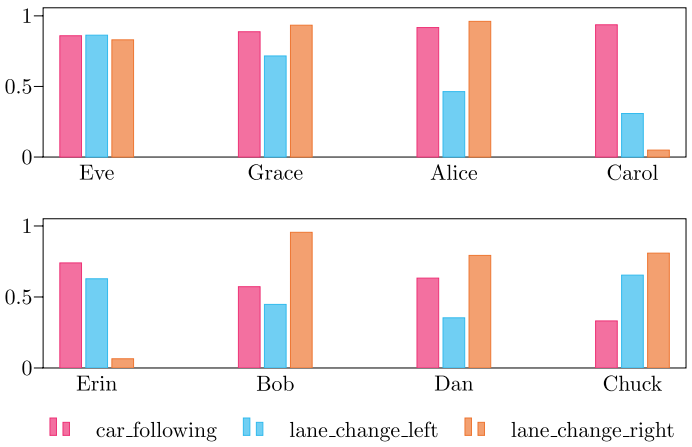


Fig. 11 Intra-cluster BC evaluations (Color figure online)

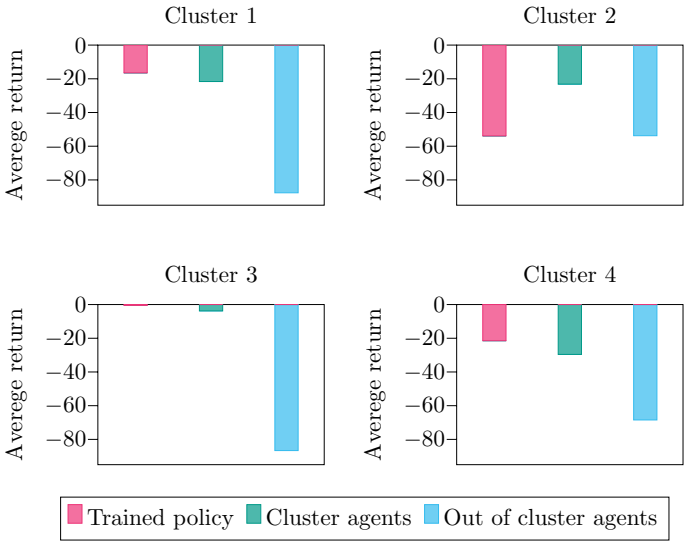


Fig. 12 Average return of the policy trained with the reward function of each cluster and mean of the average return of the experts divided based on whether they have been assigned to the cluster (Color figure online)

Clustering results We employ Multiple-Intention Σ-GIRL, as described in Sect. 3.2, with $k = 4$ clusters. The results are summarized in Tables 1 and 2, with a visualization of the reward weights in Fig. 10a. We can identify four clear clusters. The first cluster includes three agents showing the "best" behavior. These agents keep the right lane while overtaking slow vehicles. This translates into a high weight for the free-right objective, since it is a binary feature activated when we could perform a lane change to the right, and some weight for the distance-front objective. As we mentioned earlier, the distance-front objective is related to maintaining high speed, as the low-level controller starts to decelerate to maintain safety distances when the front vehicles are too close. In fact, they start overtaking vehicles only when the low-level controller starts to slow down. It is also interesting to note that the change-lane objective is not given any weight, since changing lanes without motivation is already suboptimal because it decreases speed while changing lane and creates unnecessary free-rights. The next cluster contains agents who rarely occupy the right lane but focus on maintaining a high-speed profile that provides most of the weight to the front-distance feature. Again, useless lane changes are implicitly optimized, because they affect the speed of the ego vehicle and are advantageous only when you employ them to overtake a slow vehicle. Finally, we have two clusters composed of single agents. One of them tends to keep the right lane, but changes lanes more rarely and takes longer time to decide to change lanes, while the last agent focuses only on the free right features and changes lane to the right immediately when given the possibility.

To investigate the robustness of clustering through Σ-GIRL, we increase the number of clusters. By construction, the clustering loss function will always separate clusters (Fig. 9). The remarkable behavior of MI-Σ-GIRL in this problem is that an overestimation of the number of clusters can be easily detected since the weights of the separated clusters will not differ much from the original one. This can be seen in Fig. 10b and c where the newly added clusters are close to the existing ones.

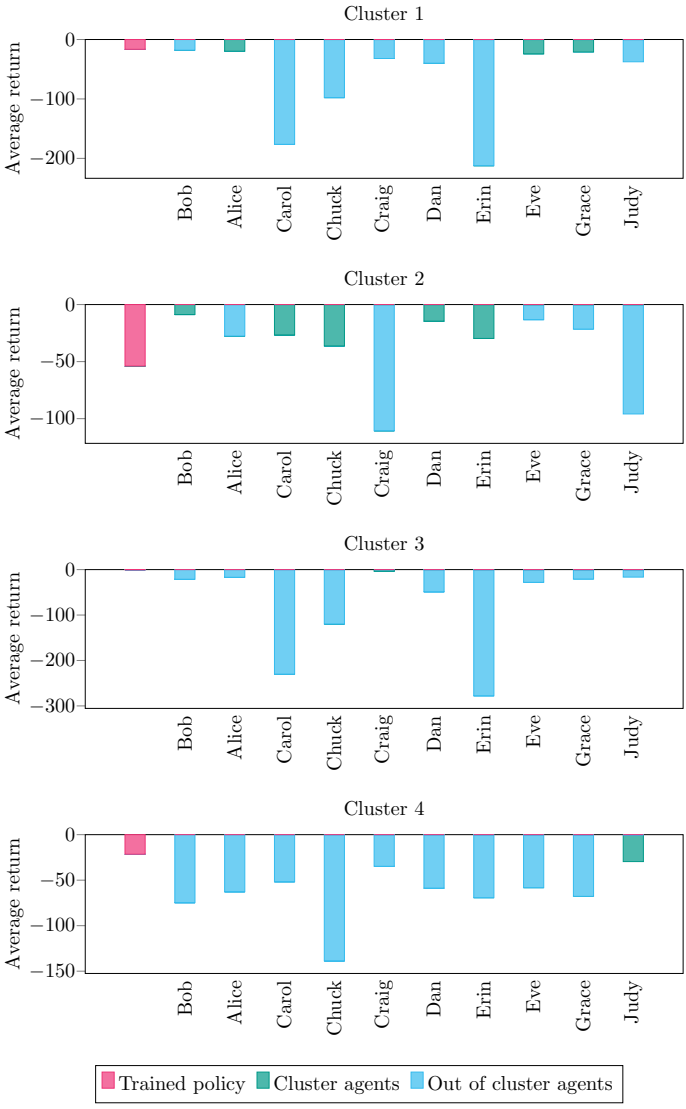


Fig. 13 Average return of the policy trained with the reward function of each cluster and average return of each expert evaluated with the reward function of each cluster (Color figure online)

BC fails to generalize We investigate how transferable the BC policies are between agents belonging to the same cluster. To this purpose, we took the policies trained to imitate each agent separately and tested them on all demonstrations of their identified clusters (excluding their own). Figure 11 shows the accuracies of the BC policies of each agent. We only show 8 agents as the other two are alone in their respective clusters. We can clearly see that BC may easily fail to generalize across agents who might show slightly different behaviors while optimizing the same reward functions. Recall, in fact, that for any reward function, there might exist multiple optimal policies. Hence, agents with the same intent

Table 3 Cluster assignment based on the feature expectations

	Agents
Cluster 1	Bob, Alice, Dan, Eve, Grace
Cluster 2	Carol, Erin
Cluster 3	Craig, Judy
Cluster 4	Chuck

can demonstrate different policies, and BC tends to "overfit" to the specific behavior. IRL methods, on the other hand, can correctly group different behaviors into the same intent.

Next we investigate how "good" the recovered reward functions fit the behaviour of the agents assigned to them. Unfortunately, since we do not have the "true" rewards it is hard to do a quantitative evaluation of the results since we cannot compute the error on the recovered weights. Nonetheless, given the presence of the simulator, we can train agents that optimize the recovered reward functions. We used the methods described in Likmeta et al. (2020), which use parametrized rule-based policies to represent the agent's policy, and have shown good performance in this Highway environment. We trained one agent for each cluster and we compare the expected return of these trained policies, with the average return of each agent in that specific reward function. For details on the training procedure see Appendix A.2. The results are shown in Fig. 12. Here we show for each of the 4 clusters, the return of the policy trained in the environment with the reward function of the cluster in pink, compared to the average return of the agent assigned to that cluster under the reward function of the cluster in green and the return of the other agents (assigned to other clusters) under the same reward function, in blue. So every subplot of the figure shows performances of different policies all evaluated with the same reward which means that their performances are comparable (comparisons accross clusters cannot be made since they have different reward functions). As can be expected, in all clusters, the agents not assigned to that cluster perform poorly. The best performing policy is generally the policy explicitly optimizing that reward function, except in Cluster 2 where the RL training procedure failed to recover a good policy. And finally the returns of the policies assigned to the cluster, have a policy close to the policies which explicitly optimize that return. The results of Cluster 2, where the trained policy is the one performing the worst, show also the general difficulty of evaluating the reward functions recovered from our algorithms, since we are not guaranteed that an agent trained with the given reward functions will achieve the optimal performcance. Figure 13 shows in detail the performances of each agent in each cluster.

Clustering via feature expectations Finally, we compare the clustering performed using Σ-GIRL, with a clustering based on the features expectations of the agents. To this end, we cluster the feature expectations of the agents using a *K-means* with $k = 4$ clusters. Table 3 shows the results of this clustering. Compared to the clustering done by IRL, we can see that the first cluster contains agents that show very heterogeneous behaviors. This cluster contains the three "good" agents, identified by the IRL clustering, together with two agents that show a greater preference for keeping a high-speed profile (Bob and Dan). It also separates agent Chuck from the other agents who show a preference towards maintaining a high-speed and the left lane, because her feature expectations are "far" from the others, even though the objectives are the same.

6 Case study 2: Twitter IRL from multiple experts

Social networks like Twitter and Facebook are actively used by millions of users every day. Inferring the users' interests and intentions is a relevant problem in this context with a variety of possible applications (Piao and Breslin 2018). For instance, understanding why users perform certain actions, like posting a message or clicking on an ad, and what their preferences are, allows the system to provide personalized recommendations and, in general, improve the user experience. Similarly, inferred intentions might help detect and counter dangerous agents, such as bots or fake profiles, who could harm the system or its users. Several learning-based techniques have been designed for this problem (Saravia et al. 2017; Song et al. 2015; Xu et al. 2011; Sadri et al. 2019). We refer the reader to an interesting survey by Piao and Breslin (2018). To the best of our knowledge, the only previous work that has applied IRL to this problem is Das and Lavoie (2014), in which the authors presented an IRL-based algorithm to infer the intentions of Reddit users. Although almost no previous work has been proposed in this direction, we believe that IRL is a natural and relevant alternative to address this problem. In fact, common existing techniques typically focus on learning the users' behavior, i.e., how users will respond to certain stimuli, in order to understand what their interests are. However, form our perspective, social network users are learning agents who act in order to maximize certain objectives, and inferring these objectives is what really informs us about their interests and behavior.

Here we explore the adoption of IRL methods, precisely the MI-Σ-GIRL algorithm, to the problem of inferring the users' intentions on Twitter. In particular, we try to answer the following questions: "Why does a user decide to retweet a post? What is her intention in deciding to post the tweet?" This problem poses several challenges from the IRL perspective. First, we do not have a simulator of the environment and the interaction with the social network might be time-prohibitive and, in some cases, illegal. Therefore, model-free and batch algorithms are required. Furthermore, although lots of data are available for free, collecting this data is very time-consuming and requires significant preprocessing (cleaning, filtering, anonymizing, etc.). Finally, the problem involves a huge amount of agents whose behaviors and intentions depend on those of other agents.

6.1 System modeling

We now describe our simplified model of the user-Twitter interaction. Among the several actions that a user can perform on Twitter, we restrict our attention to the most common one: re-tweeting a post. In our model, a user observes a tweet (generated by another user) and has to decide whether to re-tweet it or not. Intuitively, this simple model allows us to capture most of the relevant interests and intentions of Twitter users. In fact, there exist several reasons why a user might decide to re-tweet a post or not. For instance, the user might be personally interested in the content/topic of the post, or she might think that the post would be appreciated by other Twitter users, or she might simply intend to re-tweet everything (e.g., a spam bot).

In each episode of interaction with the social network, the agent observes a sequence of tweets and must decide for each one whether to retweet it or not. The state encodes information about the last observed tweet and about the agent's past behavior. It is modeled by three variables: the *popularity* of the tweet, the *number of retweets* recently performed by the agent, and the *retweet time*. The *popularity score* encodes the likelihood that the general community

Table 4 The reward weights learned by Σ-GIRL: popularity score of a retweet, number of retweets in a window T, and retweet time (δ_{time})

	Reward features			N. agents
	Popularity	N. retweets	δ_{time}	
Cluster 1	0.56	0.00	0.44	4
Cluster 2	0.16	0.19	0.65	6
Cluster 3	0.78	0.03	0.19	4

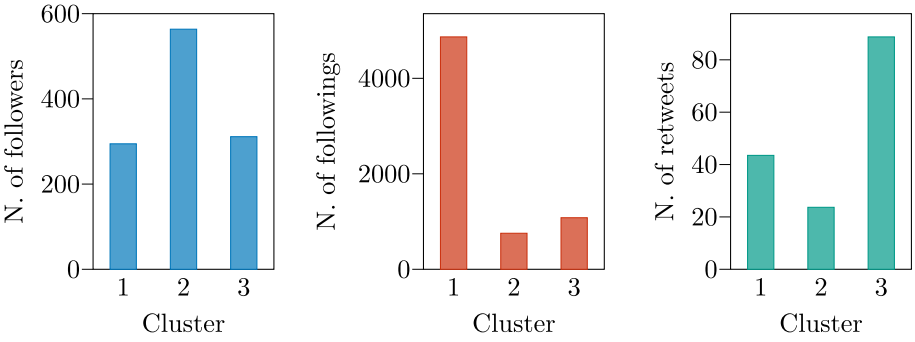


Fig. 14 Twitter clustering statistics. Average number of followers (left), followings (center) and retweets (right) for each cluster

will like the last observed tweet. It is computed as the average of the number of likes to the tweet and the number of retweets,

$$\text{Popularity-score} = \frac{N_{\text{like}} + N_{\text{retweet}}}{2},$$

and then normalized by the average popularity-score of the user's tweets. The *number of retweets* performed by the agent is computed on a retweet window of $T = 10$ steps, i.e., the last 10 observed tweets. Finally, the *retweet time* is a measure proportional to the time elapsed since the last retweet performed by the agent. It is computed as $\delta_{\text{time}} = 0.1 \cdot (t - t_0) - 1$, where t is the first time the agent receives a tweet that she decides to retweet after having retweeted at time $t_0 < t$. State transitions work as follows. The next tweet does not depend on the current one or the agent's actions since it is generated naturally by the environment (i.e., by other users). Note, however, that the popularity score of the retweet might, in fact, depend on the past actions since, for example, agents that retweet interesting content might increase their number of followers and thus the popularity of their retweets. The *retweet time* is reset to zero if the agent performed a retweet in the current step or updated accordingly as described above if the agent did not retweet, and similarly for the number of retweets.

6.2 Reward design

In this domain, the reward features are the same as the state ones, except that the Popularity-score is set to zero whenever the agent does not re-tweet. Intuitively, these features allow us to capture different interesting intentions. For instance, users who want

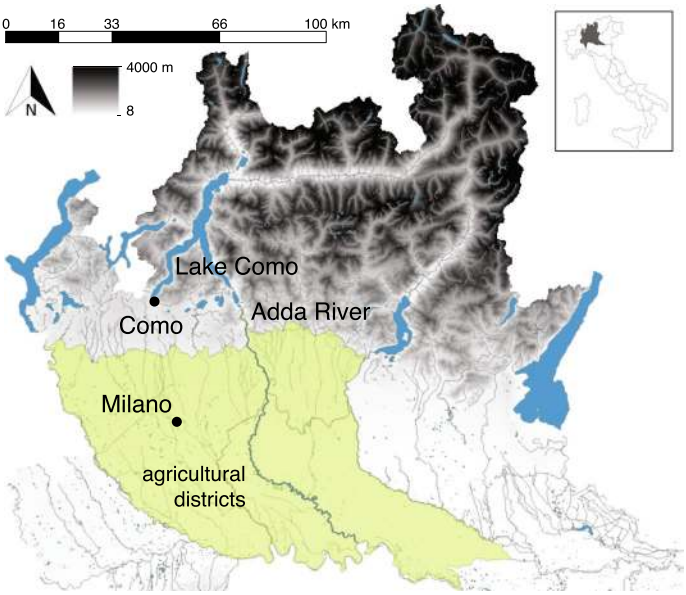


Fig. 15 Map of the Lake Como basin

to share content that is interesting to the community typically focus on the popularity score while keeping reasonable values for the other two features (so that they do not appear spammers). On the opposite side, users who want to spam every tweet focus on the number of retweets, ignoring their popularity.

6.3 Data collection and processing

The dataset was collected using the tweepy API (http://docs.tweepy.org), a Python library for accessing the Twitter API. We selected 14 Twitter accounts, and we obtained all of their followed accounts (5745 in total), using the API. For every of these 5759 (14 + 5745) accounts we collected their tweets and re-tweets from November 2018 to the end of January 2019 using a crawling process. We obtained a total number of 468304 tweets posted by these accounts on the Social Network. We assumed that each user only observes tweets from the accounts she follows, hence ignoring those coming from general (not followed) Twitter users. Furthermore, since a (human) Twitter user is very unlikely to view all the tweets from her followings while generating trajectory data we considered a probability of 0.01 that the agent sees each tweet. We used this process to split the tweet data for each agent into trajectories of 20 tweets, which were used directly to run MI-Σ-GIRL.

6.4 Results

We perform behavioral cloning on the agents' demonstrations employing a two-layer neural network (8 neurons each). Then, we divide the demonstrations in trajectories of size 20, which gives us exactly one retweet window in every trajectory. We apply MI-Σ-GIRL with

$k = 3$ clusters. The results are shown in Table 4, while Fig. 14 reports some statistics of the three clusters found.

We can observe that the users in the first cluster seem to be interested in retweeting posts with high popularity at a high frequency, i.e., they aim at maximizing the popularity score while minimizing the retweet time. This cluster can be interpreted as a grouping of standard Twitter users. This is also confirmed by Fig. 14, which shows that users in this cluster follow many other users while having fewer followers, the standard situation in the social network. The second cluster, on the other hand, groups users who do not aim at retweeting too often. These are users who do not frequently use the social network, as they have few retweets and follow a small number of people. The last cluster is perhaps the most interesting one: these agents tend to retweet all popular tweets. After inspecting the users assigned by the algorithm to this cluster, we found that they are mostly commercial accounts (e.g., bots, companies, or two HR managers). Not surprisingly, they show the intention to post popular tweets, but they are uninterested in following other accounts, as Fig. 14 highlights. For completeness, we show in Appendix B.1 the results of clustering based on feature expectations.

7 Case study 3: Como Lake Dam IRL from non-stationary expert

Lake Como is a sub-alpine lake in northern Italy, characterized by an active storage capacity of 254 Mm^3 fed by a 4552 km^2 catchment (Fig. 15). The main tributary and only emissary of the lake is the Adda river, the fourth longest Italian river, whose sublacual part originates in the southeastern branch of the lake and feeds eight run-of-the-river hydroelectric power plants and serves a dense network of irrigation canals belonging to four irrigation districts, with a total irrigated area of 1400 km^2. The southwestern branch of the lake constitutes a dead-end exposed to flooding events, particularly in the city of Como which is the lowest point of the lake shoreline. The hydro-meteorological regime is characterized by scarce discharge in winter and summer, and peaks in late spring and autumn due to snowmelt and rainfall, respectively. Snowmelt from May to July is the most important contribution to the seasonal lake storage. The agricultural districts downstream prefer to store snowmelt in the lake to satisfy the peak summer water demands, when the natural inflow is insufficient to meet irrigation requirements. Yet storing such water increases the lake level and, consequently, the flood risk, which could instead be minimized by keeping the lake level as low as possible. The lake regulation has, therefore, to balance flood protection to the lake shores and water supply to downstream users.

While the objectives that the human operator in charge of deciding the water release are known, their relative importance is unknown and it might change over time. In this setting, IRL can help in understanding the operator preferences. This knowledge could lead to the future development of artificial systems helping the operator by suggesting a suitable water release amount or even a fully automatic controller. Additionally, exploiting NS-$\mathcal{Z}$-GIRL, we can capture the possible variations in the operator preferences over time.

While RL is receiving growing attention in the water community, the application IRL is still in its infancy. In Mason (2018), the Cutting-Plane Inverse Reinforcement Learning algorithm (Pirotta 2016) is first tested in a synthetic case study; the experiments show that CPIRL is able to identify the specific tradeoff underlying a simulated control policy and to distinguish among different formulations of the same objective. The same algorithm is also used to identify changes in the operations of an Alpine hydropower reservoir in response

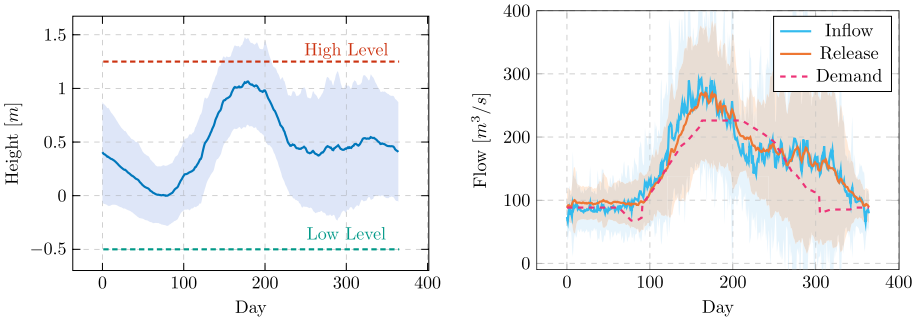


Fig. 16 Lake level and flood thresholds (left); cyclostationary average inflow, release and demand (right). Shaded areas refer to the historical variability over the 1946–2010 period

to the transition from a regulated electric energy market to a free setting. In Mason et al. (2018), a multilateral negotiation process replaces IRL in the identification of the preference among multiple objectives. The method assumes that multiple virtual agents, which independently optimize different objectives, periodically negotiate a compromise policy for the operation of the system. The authors also model preference dynamics via periodic negotiations where the agents' attitudes in each negotiation step are determined by the recent system performance.

7.1 System modeling

The system is modeled as a discrete-time, periodic, nonlinear, stochastic MDP with a continuous state variable representing the water stored in the lake S_t, a continuous action that controls the water released a_t, a state-transition function affected by the stochastic lake inflow q_{t+1} to describe the mass balance equation of the lake storage, i.e.

$$S_{t+1} = S_t + q_{t+1} - r_{t+1}(S_t, a_t, q_{t+1}) \tag{15}$$

where S_t is the lake storage at time t; q_{t+1} the inflow in the time interval $[t, t+1)$, r_{t+1} the water volume released in the same interval, which coincides with the action a_t corrected, where appropriate, with a non-linear release function determining the minimum and maximum releases feasible for the time interval to respect physical and legal constraints (e.g., spills when the lake level exceeds the maximum capacity). In the adopted notation, the time subscript of a variable indicates the time instant when its value is deterministically known. The reservoir storage is known at time t by measuring the lake level h_t and thus is denoted as S_t, while the net inflow is denoted as q_{t+1} because it can be known only at the end of the time interval.

7.2 Reward design

We model the competing interests of flood control and water supply using the following reward functions, with a specific feature accounting for intense drought events:

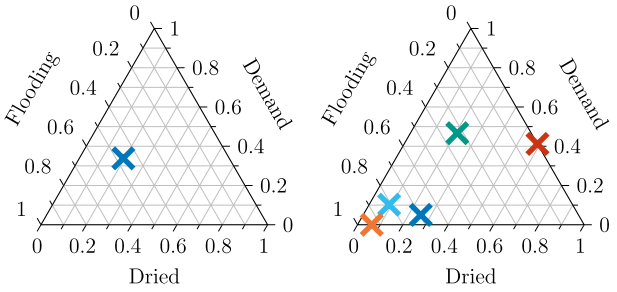


Fig. 17 Graphical representation of the weights recovered for one (left) and five (right) regimes. The crosses use the same color coding as the intervals in Fig. 20 (Color figure online)

– *Water supply deficit* (ϕ^D): the daily water deficit between the lake release r_{t+1} and the water demand d_t of the downstream system:

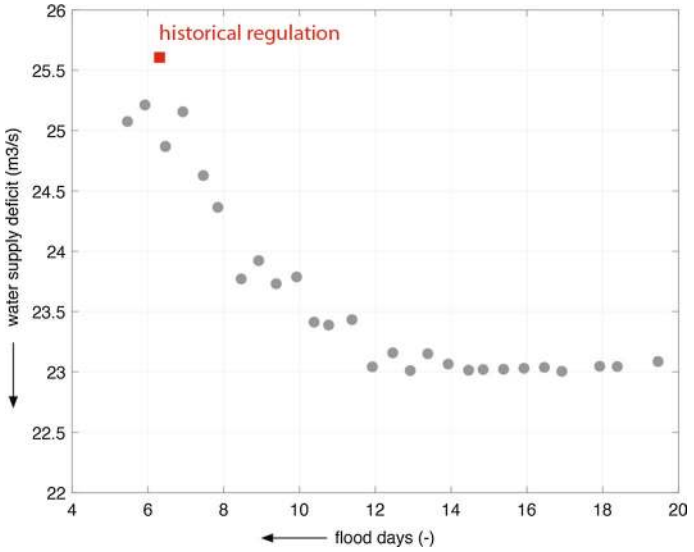


Fig. 18 Comparison of the historical regulation (red square) with the set of Pareto optimal control policies (gray circles) exploring the trade-off between flood control and water supply obtained in Giuliani et al. (2019). The historical regulation reveals a preference for reducing floods, as confirmed by our results in Fig. 17 (Color figure online)

$$\phi_t^D = \max(r_t - d_t, 0). \tag{16}$$

– *Flood risk* (ϕ^F): a penalization function that is large for small releases associated to high lake levels:

$$\phi_t^F = \begin{cases} 0 & \text{if } r_{t+1} > \underline{r}^F \\ -\left(\frac{\overline{r}^F - r_{t+1}}{\overline{r}^F - \underline{r}^F}\right)^2 & \text{if } \underline{r}^F \le r_{t+1} \le \overline{r}^F \\ -1 & \text{otherwise} \end{cases}, \tag{17}$$

with $\overline{r}^F = 120$ and $\underline{r}^F = 5$.

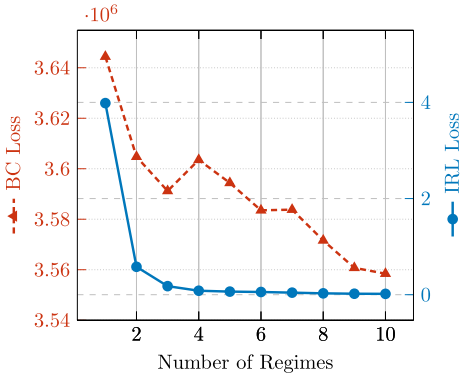


Fig. 19 IRL and BC losses as a function of the number of regimes. The IRL loss is the one of Eq. 14. The BC loss is the cross-entropy of the same intervals computed by IRL

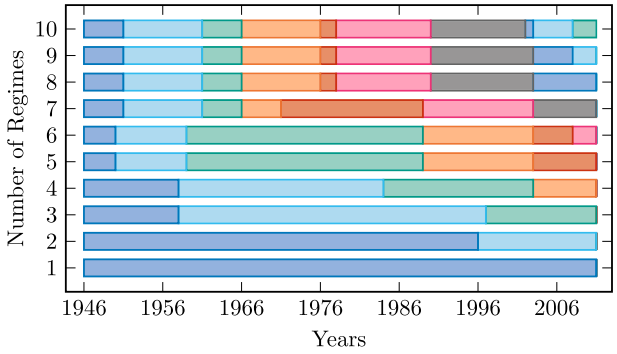


Fig. 20 Identified year intervals for different number of regimes (Color figure online)

– *Drought risk* (ϕ^L): a penalization that is large for large releases associated to low lake levels:

$$\phi_t^L = \begin{cases} 0 & \text{if } r_{t+1} < \underline{r}^L \\ -\left(\frac{r_{t+1}-\underline{r}^L}{\overline{r}^L-\underline{r}^L}\right)^2 & \text{if } \underline{r}^L \le r_{t+1} \le \overline{r}^L \\ -1 & \text{otherwise} \end{cases}, \tag{18}$$

with $\overline{r}^L = 500$ and $\underline{r}^L = 5$:

7.3 Data description

The dataset is composed of the historical trajectory of lake levels, inflows, and releases over the period 1946–2010, which are illustrated in Fig. 16. All data are daily and were provided by Consorzio dell'Adda (www.addaconsorzio.it). In Fig. 16 we show the lake level, together with the inflow and release for a year, averaged over the considered time interval (1946–2010).

7.4 Results

We employed as state representation the concatenation of the lake level h_t, the inflow of the previous day q_t, the demand of the current day d_t and the actions of the previous two days a_{t-1} and a_{t-2}:

$$s_t = \left(h_t, q_t, d_t, a_{t-1}, a_{t-2}\right). \tag{19}$$

For the preliminary BC phase, we employed a Gaussian policy with fixed variance $\sigma^2 = 1$ and mean which is linear in the state s_t.

The human operator has a set of preferences that are unknown. First of all, we focus on the IRL results without any time interval subdivision, i.e., when considering just one regime. The weights recovered in this case are shown in Fig. 17 left. We notice a slight predominance of the interest in controlling the floods, whose feature ϕ^F is weighted with $\omega^F = 0.47$, whereas the remaining weight is divided between the feature of the demand ϕ^D ($\omega^D = 0.34$) and the control of the drought events ϕ^L ($\omega^L = 0.19$). These preferences can be validated by comparing the historical data with the set of Pareto optimal control policies exploring the tradeoff between flood control and water supply obtained in Giuliani et al. (2019). The mapping of the historical regulation and of the Pareto optimal solutions in the space of water supply deficit and flood control illustrated in Fig. 18 shows how the operator is almost Pareto efficient and the historical regulation attains very good performance in terms of flood control at the cost of high values of water supply deficit.

Since the available data span a time period of 65 years, we investigate whether the behavior of the lake operator displays a stationary intention or the underlying preferences change over time. As many environmental systems, the Lake Como is a non-stationary system that has undergone several alterations over time, which might have changed the preferences of the regulator. For this purpose, we employ NS-Σ-GIRL, described in Sect. 3.3, by considering a single trajectory the sequence of states and actions observed along a one year period. The results are shown in Fig. 19 and 20. First of all, looking at the IRL loss (Fig. 19) we observe a significant improvement moving from one regime to two, then the loss keeps reducing but with smaller benefits. From an elbow analysis, we can conclude that a number of regimes of 4 or 5 result suitable for the problem. Looking at the BC loss, we notice an overall reduction as well, although not monotonic.

Concerning the interval subdivision, according to the properties of the real domain and the events that occurred, we believe that the most interpretable one is the case with five regimes. The first two time periods (1946–1949 and 1950–1959) can be seen as set-up periods in which the operator tries different policies and learns how to operate the dam, which was constructed in 1946. Overall, this period displays a preference towards reducing flood risk. Moving to the third period (1960-1988) we appreciate a notable change in the intentions trade-off. Indeed, the estimated weights increase the preference toward satisfying the downstream water demand ($\omega^D = 0.47$) while reducing the interest in avoiding the floods ($\omega^F = 0.33$). The notable length of this period can be interpreted as an indicator of the fact that the human regulator has converged to a stable policy. However, starting from 1989 we notice a significant variation of trade-off that becomes largely driven by flood control. In the time interval 1989–2002, most of the weight (0.94) is given to the feature ϕ^F. This change of intention can be justified by the large flood event that occurred in 1987, registering the highest level in the historic records. This event was followed by other floods, which might have further consolidated this conservative behavior that minimizes flood risk. In recent years, the climatic conditions in the region have manifested a

drying trend inducing a further modification of the operator's preferences. The summer of 2003–2005–2006 represent extreme, unprecedented drought events (Giuliani et al. 2016). Our results capture this transition, with the period (2003–2010) that is associated with a new regime that assigns a high weight to reducing drought risk ($\omega^L = 0.59$) and supplying water demand ($\omega^D = 0.41$) while reducing the importance of flood control.

This analysis allows grasping a general overview of how the operator's preferences, which are modeled via intentions, change over time. However, several questions remain open. First, the choice of a suitable number of regimes, by just observing the data is challenging. The elbow analysis can provide some suggestions, but still, we needed the domain expert's knowledge to understand whether a subdivision is reasonable. Second, the five regime setting allows us to provide some interpretation but displays limits as well. Specifically, the subdivision is sometimes inaccurate. We may wonder why the fourth interval (the one in which the flooding control objective is predominant) begins in 1989 instead of 1988, being that the Como flooding event occurred in 1987. Third, the results are significantly dependent on the choice of the features. A suitable feature design is an iterative process that needs to account for both the domain peculiarities and the characteristics of the employed IRL algorithms.

8 Discussion and conclusions

We tackled the problem of inferring the intentions of human operators in several real-world scenarios via IRL algorithms. In these settings, it is important to have algorithms that operate in a model-free, batch manner, since, in most applications, the model of the environment is not available and there is no possibility of interaction as well. We applied the MI-Σ-GIRL algorithm to the Twitter and AD tasks, identifying multiple clusters of agents with different reward functions. Furthermore, we proposed an extension to the Σ-GIRL algorithm to deal with non-stationary intentions of the expert and applied it to the real-world case of the Lake Como dam operation, identifying multiple operating regimes. We interpreted these regimes with the evolution of the dam environment supported by historical data on the climatic events that occurred in the geographical area of the dam.

Although we were able to employ these algorithms in real-life scenarios, it is worth noting that their application should not be seen as a black-box. Being in a fully-batch setting, without further interaction with the environment, these algorithms depend heavily on the system modeling phase. A bad design of the state, action, policy and most importantly, reward space can highly affect the final results. We witnessed this phenomenon in all our applications, and we believe that it is a price to pay when giving away the possibility to interact with the environment. Typically, a bad state, action and policy space design can be detected in the behavioral cloning phase, as usually the accuracy of the imitating policy is low. Reward design is a more delicate phase, as with most of the IRL algorithms. In general it is important to avoid features that can cause a constant expected return under every policy. An example of these kind of features are constant features, features and its negation and constant conic features combination. We have observed that usually a poor reward design results in "extreme" reward weights, where all the weight goes to one of the features. For these reasons, the application of this kind of algorithms requires a close interaction with experts of the specific field of application.

As future work, we intend to extend our approach to deal with settings in which the action space of the demonstrations differs from the action space of the task in which

the reward functions will be applied. For instance, in the car driving problem, we might consider the case where demonstrations come from the low-level control of the vehicle, but the reward function will be applied for the high-level control. Furthermore, we might consider a more extreme scenario, when we do not observe the actions performed by the expert but only their effects on the state of the environment.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10994-020-05939-8.

Author Contributions Not Applicable

Funding Open access funding provided by Alma Mater Studiorum - Università di Bologna within the CRUI-CARE Agreement.

Availability of data and material The data relative to the AD task and the Como dam operation are available in the code repository. The data relative to the Twitter case study are available upon request because they need to be anonymized.Code availability The code of the experiments will be made available at https://github.com/amarildolikmeta/irl_real_life.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*(13), 1608–1639.

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, p. 1, New York, NY, USA. ACM.

Almingol, J., & Montesano, L. (2015). Learning multiple behaviours using hierarchical clustering of rewards. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4608–4613.

Aminikhanghahi, S., & Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and Information Systems*, *51*(2), 339–367.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

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

Barratt, S. (2018). A matrix gaussian distribution. arXiv preprint arXiv:1804.11010.

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, *16*(1), 87–90.

Bogert, K., & Doshi, P. (2014). Multi-robot inverse reinforcement learning under occlusion with interactions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 173–180. Citeseer.

Boularias, A., Kober, J., & Peters, J. (2011). Relative entropy inverse reinforcement learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, *19*(8), 1271–1291.

Casella, G., & Berger, R. L. (2002). *Statistical inference* (Vol. 2). CA: Duxbury Pacific Grove.

Codevilla, F., Miiller, M., López, A., Koltun, V., & Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE.

Das, S., & Lavoie, A. (2014). The effects of feedback on human behavior in social media: An inverse reinforcement learning model. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 653–660. International Foundation for Autonomous Agents and Multiagent Systems.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, *2*(12), 1–142.

Dempster, M. A. H., & Romahi, Y. S. (2002). Intraday fx trading: An evolutionary reinforcement learning approach. In *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 347–358. Springer.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). Carla: An open urban driving simulator. arXiv preprint arXiv:1711.03938.

Englert, P., Paraschos, A., Deisenroth, M. P., & Peters, J. (2013). Probabilistic model-based imitation learning. *Adaptive Behavior*, *21*(5), 388–403.

Finn, C., Levine, S., & Abbeel, P. (2016). Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning—Volume 48*, ICML'16, pp. 49–58. JMLR.org.

Finn, C., Yu, T., Zhang, T., Abbeel, P., & Levine, S. (2017). One-shot visual imitation learning via meta-learning. arXiv preprint arXiv:1709.04905.

Giuliani, M., Li, Y., Castelletti, A., & Gandolfi, C. (2016). A coupled human-natural systems analysis of irrigated agriculture under changing climate. *Water Resources Research*.

Giuliani, M., Zaniolo, M., Castelletti, A., Davoli, G., & Block, P. (2019). Detecting the state of the climate system via artificial intelligence to improve seasonal forecasts and inform reservoir operations. *Water Resources Research*, *55*, 9133–9147.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, *50*(2), 1–35.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2020). Deep reinforcement learning for autonomous driving: A survey. arXiv preprint arXiv:2002.00444.

Klein, E., Geist, M., Piot, B., & Pietquin, O. (2012). Inverse reinforcement learning through structured classification. *Advances in Neural Information Processing Systems*, *25*, 1007–1015.

Klein, E., Piot, B., Geist, M., & Pietquin, O. (2013). A cascaded supervised learning approach to inverse reinforcement learning. In *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases—Volume Part I*, ECMLPKDD'13, pp. 1–16. Springer, Berlin, Heidelberg.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274.

Kober, J., & Peters, J. R. (2009). Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems*, *21*, 849–856.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO—Simulation of Urban Mobility. *International Journal on Advances in Systems and Measurements*, *5*(3&4), 128–138.

Kuderer, M., Gulati, S., & Burgard, W. (2015). Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2641–2646. IEEE.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, *17*(1), 1334–1373.

Likmeta, A., Metelli, A. M., Tirinzoni, A., Giol, R., Restelli, M., & Romano, D. (2020). Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, *131*, 103568.

Mason, E. (2018). *Beyond full rationality: modeling tradeoff dynamics in multi-objective water management*. PhD thesis, Politecnico di Milano, Italy.

Mason, E., Giuliani, M., Castelletti, A., & Amigoni, F. (2018). Identifying and modelling dynamic preference evolution in multipurpose water resources systems. *Water Resources Research*, *54*(4), 3162–3175.

Metelli, A. M., Pirotta, M., & Restelli, M. (2017). Compatible reward inverse reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pp. 2050–2059. Curran Associates, Inc.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 673–680, New York, NY, USA. Association for Computing Machinery.

Ng, A. Y., & Russell, S. J. (2000a). Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 663–670, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ng, A. Y., & Russell, S. J. (2000b). Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670. Morgan Kaufmann.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Berlin: Springer.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., Peters, J., et al. (2018). An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, *7*(1–2), 1–179.

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., & Boots, B. (2017). Agile autonomous driving using end-to-end deep imitation learning. arXiv preprint arXiv:1709.07174.

Papini, M., Pirotta, M., & Restelli, M. (2019). Smoothing policies and safe policy gradients. arXiv preprint arXiv:1905.03231.

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, *21*(4), 682–697.

Piao, G., & Breslin, J. G. (2018). Inferring user interests in microblogging social networks: A survey. *User Modeling and User-Adapted Interaction*, *28*(3), 277–329.

Pirotta, M. (2016). *Reinforcement learning: from theory to algorithms*. PhD thesis, Politecnico di Milano, Italy.

Pirotta, M., & Restelli, M. (2016). Inverse reinforcement learning through policy gradient minimization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1993–1999. AAAI Press.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley.

Ramponi, G., Likmeta, A., Metelli, A. M., Tirinzoni, A., & Restelli, M. (2020). Truly batch model-free inverse reinforcement learning about multiple intentions. In *The 23nd International Conference on Artificial Intelligence and Statistics*.

Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. A. (2006). Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 729–736, New York, NY, USA. ACM.

Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., & Hebert, M. (2013). Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pp. 1765–1772. IEEE.

Sadri, A. M., Hasan, S., & Ukkusuri, S. V. (2019). Joint inference of user community and interest patterns in social interaction networks. *Social Network Analysis and Mining*, *9*(1), 11.

Saravia, E., Wu, S.-C., & Chen, Y.-S. (2017). A dynamic influence keyword model for identifying implicit user interests on social networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 1160–1163.

Shani, G., Heckerman, D., & Brafman, R. I. (2005). An mdp-based recommender system. *Journal of Machine Learning Research*, *6*(Sep), 1265–1295.

Silver, D., Bagnell, J. A., & Stentz, A. (2013). Learning autonomous driving styles and maneuvers from expert demonstration. In *Experimental Robotics*, pp. 371–386. Springer.

Song, X., Nie, L., Zhang, L., Liu, M., & Chua, T.-S. (2015). Interest inference via structure-constrained multi-source multi-task learning. *Twenty-Fourth International Joint Conference on Artificial*. (**Intelligence**).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Adaptive computation and machine learning: MIT Press, second edition.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Solla, S., Leen, T., and Müller, K., editors, *Advances in Neural Information Processing Systems 12*, pp. 1057–1063. MIT Press.

Tateo, D., Pirotta, M., Restelli, M., & Bonarini, A. (2017). Gradient-based minimization for multi-expert inverse reinforcement learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8. IEEE.

Truong, C., Oudre, L., & Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, *167*, 107299.

Warlop, R., Lazaric, A., & Mary, J. (2018). Fighting boredom in recommender systems with linear reinforcement learning. *Advances in Neural Information Processing Systems*, *31*, 1757–1768.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256.

Wulfmeier, M., Rao, D., Wang, D. Z., Ondruska, P., & Posner, I. (2017). Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, *36*(10), 1073–1087.

Xu, Z., Ru, L., Xiang, L., & Yang, Q. (2011). Discovering user interest on twitter with a modified author-topic model. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 422–429. IEEE.

Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., & Abbeel, P. (2018). Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE.

Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (Vol. 3, pp. 1433–1438).