

Bridging the Gap between Imitation Learning and Inverse Reinforcement Learning

Bilal Piot, Matthieu Geist, and Olivier Pietquin, *Senior Member, IEEE*

Abstract—Learning from Demonstrations (LfD) is a paradigm by which an apprentice agent learns a control policy for a dynamic environment by observing demonstrations delivered by an expert agent. It is usually implemented as either Imitation Learning (IL) or Inverse Reinforcement Learning (IRL) in the literature. On the one hand, IRL is a paradigm relying on Markov Decision Processes (MDPs), where the goal of the apprentice agent is to find a reward function from the expert demonstrations that could explain the expert behavior. On the other hand, IL consists in directly generalizing the expert strategy, observed in the demonstrations, to unvisited states (and it is therefore close to classification, when there is a finite set of possible decisions). While these two visions are often considered as opposite to each other, the purpose of this paper is to exhibit a formal link between these approaches from which new algorithms can be derived. We show that IL and IRL can be redefined in a way that they are equivalent, in the sense that there exists an explicit bijective operator (namely the inverse optimal Bellman operator) between their respective spaces of solutions. To do so, we introduce the set-policy framework which creates a clear link between IL and IRL. As a result, IL and IRL solutions making the best of both worlds are obtained. In addition, it is a unifying framework from which existing IL and IRL algorithms can be derived and which opens the way for IL methods able to deal with the environment's dynamics. Finally, the IRL algorithms derived from the set-policy framework are compared to algorithms belonging to the more common trajectory-matching family. Experiments demonstrate that the set-policy-based algorithms outperform both standard IRL and IL ones and result in more robust solutions.

Index Terms—Learning from Demonstrations, Inverse Reinforcement Learning, Imitation Learning.

I. INTRODUCTION

Because of the growing interest in robotics and other complex systems (such as interactive systems), new machine learning methods for model-free control have been the subject of many researches this last decade. Especially, Learning from Demonstrations (LfD) [1]–[4] is a promising paradigm where limited programming skills are required and almost no *ad hoc* modeling is necessary. LfD consists in providing examples of the optimal behavior (in the form of a sequence of actions) to a learning agent so as to make it reproducing a similar behavior, even in unseen situations. The learning agent is then called the *apprentice* and the demonstrating agent is called the *expert*. This paradigm, because of its promising applications, has already been used in several domains such as robotics [4] or human-machine interaction [5].

In many cases, the LfD problem is placed in the framework of Markov Decision Processes (MDP) [6], [7]. In machine learning, this is a standard framework for describing a dynamical system and learning an optimal controller for it. The system is described as a set of states, actions and transitions and the learned behavior takes the form of a *policy*, mapping from states to actions. Usually, this is done via Reinforcement Learning [8] where, after each action, the learning agent is provided with an immediate numerical feedback (called a *reward*) depending on the quality of the action. In consequence, the agent learns a policy that maximizes the cumulative reward over time. In the MDP framework, solving the LfD problem thus consists in learning a policy performing as well as the expert agent but using a finite set of demonstrations instead of the reward signal. The type of guidance used in LfD (demonstration of expert actions in given states) is thus very different from the one used in RL (numerical feedback).

In the literature, one can find two main trends of implementation of LfD: Imitation Learning (IL) [1], [2], [9], [10] or Inverse Reinforcement Learning (IRL) [11], [12]. While these two approaches are often considered in opposition, the main contribution of this paper is to propose a unifying framework, namely the *set-policy framework*, exhibiting a formal link between their spaces of solutions.

IL consists in directly learning the mapping between states and actions (the policy), considering this task as a supervised learning problem. Most of the time, the action set is finite, and IL can be reduced to a Multi-Class Classification (MCC) problem [1], [10], [13] where each state is labeled with the corresponding expert action. MCC algorithms are easy to implement, can be non-parametric and even extended to relational domains [14]. MCC-based IL has also been shown to be theoretically sound in the finite [10] and infinite [15] horizon settings. However, MCC algorithms do not take into account the dynamics of the underlying MDP, which is an important drawback. Indeed, an MCC algorithm will not learn on the basis of the consequences of its actions when applied to the dynamical system but relies only on demonstrations. To alleviate this limitation, some rare IL methods integrate information about the dynamics by using a regularized classification method [16] or a kernel-based classification technique [17] where the kernel is determined using MDP metrics [18]. This often leads to much more complex algorithms. We do not consider here IL methods relying on the assumption that it is possible to query the expert for specific demonstrations during the learning process [19], [20].

On the other hand, IRL [11], [12] relies on the assumption that the expert's policy is optimal with respect to an

B. Piot is a member of Univ. Lille, Centrale Lille, Inria, CNRS, UMR 9189 - CRISTAL, F-59000 Lille, France, bilal.piot@univ-lille1.fr

M. Geist, UMI 2958, Georgia Tech - CNRS, CentraleSupélec, Université Paris-Saclay, 91190 Evry, France, matthieu.geist@centralesupelec.fr

O. Pietquin is a member of IUF and Univ. Lille, Centrale Lille, Inria, CNRS, UMR 9189 - CRISTAL, F-59000 Lille, France, olivier.pietquin@univ-lille1.fr

unknown reward function. In this case, the first aim of the apprentice is to learn a reward function that explains the observed expert behavior. Then, using direct reinforcement learning, it optimizes its policy according to this reward and hopefully behaves as well as the expert. Learning a reward has some advantages over learning a policy immediately. First, the reward can be analyzed so as to better understand the expert's behavior. Second, it allows adapting to perturbations in the dynamics of the environment [15], [21]. In other words, it is transferable to other environments. Third, it allows improving with time through real interactions and without requiring new demonstrations. However, a major issue is that an MDP must be solved to obtain the optimal policy with respect to the learned reward. Another issue is that the IRL problem is ill-posed as every policy is optimal for the null reward (which is obviously not the reward one is looking for). More generally, a good IRL algorithm should discard reward functions for which too many actions are optimal (trivial rewards).

The seminal IRL algorithms were incremental and needed to repeatedly solve MDPs [22]–[24]. These incremental methods can be seen as a family of algorithms that iteratively build a reward generating policies for which the trajectories (in the MDP) gets *closer* to the ones of the expert [25]. Here, we refer to this family as the trajectory-matching framework. Each step of these algorithms requires solving an MDP for an intermediate reward (one exception is the Relative Entropy algorithm [26]), which is a difficult problem in general. In addition, these algorithms usually assume a linear parameterization of the reward functions with respect to some features. The choice of features is highly problem-dependent and requires a careful engineering work. Avoiding these drawbacks, recent IRL algorithms such as Structured Classification for IRL (SCIRL) [27], Relative Entropy [26] or Cascaded Supervised IRL (CSI) [28] directly learn a reward function without solving any MDP.

To sum up, it comes out that IL suffers from ignoring the dynamics of the environment and IRL from being ill-posed. The major claim of this paper is that the proposed set-policy framework tackles these two issues and provides a best-of-both-world solution. More precisely, we show that it is possible to slightly modify the definitions of both IL and IRL problems such that they are equivalent in the sense that there exists an explicit bijective operator (namely the *inverse optimal Bellman operator*) between their respective spaces of solutions. Doing so, IRL and IL benefit from each other, IRL becomes well-defined and IL takes the dynamics into account. Besides, this new paradigm allows us to easily derive existing IRL algorithms such as SCIRL and CSI. It also provides some insights on how it is possible to obtain new IL algorithms that take into account the underlying dynamics such as Reward Regularized Classification for Apprenticeship Learning (RCAL) [16].

The remaining of the paper is organized as follows. First, the MDP, IRL and IL frameworks are presented in Sec. II. Then, the set-policy framework, which is the main contribution of this paper, is introduced in Sec. III. From this, the SCIRL, CSI and RCAL algorithms are derived again in Sec. IV. Finally, IRL algorithms relying on the set-policy framework

are compared to trajectory-matching algorithms in a generic experiment (see Sec. V) which uses randomly generated MDPs called Garnet [29]. Comparison is made according to different levels of knowledge of the dynamics and given several levels of quality of the feature space. Experiments show that combining both IRL and IL results in an increased robustness of algorithms and improved performances.

II. BACKGROUND

In this section, we briefly present the concepts of MDP, IRL and IL. The notations are given as we go along and summarized in Sec. VI-C.

A. Markov Decision Processes

In this section, MDPs are briefly described. The reader can refer to [6] for further details. In this paper, we will consider finite MDPs¹. An MDP² models the interactions of an agent evolving in a dynamic environment and is represented by a tuple $M_R = \{S, A, R, P, \gamma\}$ where $S = \{s_i\}_{1 \leq i \leq N_S}$ is the finite state space with $N_S \in \mathbb{N}^*$ states, $A = \{a_i\}_{1 \leq i \leq N_A}$ is the finite action space with $N_A \in \mathbb{N}$ actions, $R \in \mathbb{R}^{S \times A}$ is the reward function³ where $R(s, a)$ is the local benefit of doing action a in state s , $\gamma \in]0, 1[$ is a discount factor and $P \in \Delta_S^{S \times A}$ is the Markovian dynamics which gives the probability⁴, $P(s'|s, a)$, to step in s' by performing action a in state s . A Markovian and stationary policy π is an element of Δ_A^S and defines the behaviour of an agent. In particular, $\pi(a|s)$ gives the probability of doing the action a in the state s . When $\forall s \in S, \text{Card}(\text{Supp}(\pi(\cdot|s))) = 1$ which means that for all states only one action is chosen⁵, the policy $\pi \in \Delta_A^S$ is said deterministic and can be identified to an element $\pi_D \in A^S$ such that: $\forall s \in S, \pi_D(s) = \text{Supp}(\pi(\cdot|s))$. In the remaining, if π is deterministic, with an abuse of notation, π will represent $\pi \in \Delta_A^S$ or its counterpart $\pi_D \in A^S$ depending on the context.

In order to quantify the quality of a policy π with respect to the reward R , the *quality function*, also called the state-action value function, is defined. For a given MDP M_R and a given policy π , the quality function $Q_R^\pi \in \mathbb{R}^{S \times A}$ maps each state-action couple (s, a) to the expected and discounted cumulative reward for starting in state s , doing the action a and following the policy π afterwards. It can be formalized as $Q_R^\pi(s, a) = \mathbb{E}_{s,a}^\pi[\sum_{t=0}^{+\infty} \gamma^t R(s_t, a_t)]$, where $\mathbb{E}_{s,a}^\pi$ is the expectation over the distribution of the admissible trajectories (s_0, a_0, s_1, \dots) obtained by executing the policy π starting from $s_0 = s$ and $a_0 = a$. In addition, the function $Q_R^* \in \mathbb{R}^{S \times A}$ defined as $Q_R^* = \max_{\pi \in \Delta_A^S} Q_R^\pi$ is called the optimal quality function. For ease of writing, for each $Q \in \mathbb{R}^{S \times A}$ and each π , we define $f_Q^* \in \mathbb{R}^S$ such that

¹This work could be easily extended to measurable state spaces as in [30] for instance; we choose the finite case for the ease and clarity of exposition.

²MDPs will be indexed by their reward function R which means that the other parameters are fixed and R is seen as a variable.

³Let X and Y be two non empty sets, Y^X is the set of functions from X to Y .

⁴If X and Y are finite, Δ_X the set of distributions over X and Δ_X^Y is the set of functions from Y to Δ_X .

⁵ $\text{Card}(X)$ represents the cardinal of a finite set X and $\text{Supp}(\alpha)$ is the support of α : $\text{Supp}(\alpha) = \{x \in X, \alpha(x) > 0\}$ where $\alpha \in \Delta_X$

$\forall s \in S, f_Q^*(s) = \max_{a \in A} Q(s, a)$ and $f_Q^\pi \in \mathbb{R}^S$ such that $\forall s \in S, \forall a \in A, f_Q^\pi(s) = \sum_{a \in A} \pi(a|s) Q(s, a)$. From the quality function Q_R^π and the optimal quality function Q_R^* , we can define the *value function* $V_R^\pi = f_{Q_R^\pi}^\pi$ and the optimal value function $V_R^* = f_{Q_R^*}^*$. Furthermore, one can show that Q_R^π and Q_R^* are fixed points of the two following contracting operators T_R^π and T_R^* [6], $\forall Q \in \mathbb{R}^{S \times A}, \forall (s, a) \in S \times A$:

$$\begin{aligned} T_R^\pi Q(s, a) &= R(s, a) + \gamma \mathbb{E}_{P(\cdot|s, a)}[f_Q^\pi], \\ T_R^* Q(s, a) &= R(s, a) + \gamma \mathbb{E}_{P(\cdot|s, a)}[f_Q^*]. \end{aligned} \quad (1)$$

Finally, a policy π is said *greedy* with respect to a function Q if: $\forall s \in S, \text{Supp}(\pi(\cdot|s)) \subset \arg\max_{a \in A} Q(s, a)$.

If π is deterministic, π is greedy with respect to a function Q if: $\forall s \in S, \pi(s) \in \arg\max_{a \in A} Q(s, a)$.

Dynamic Programming denotes a set of techniques to find optimal policies given an MDP which are characterized by Theorem 1. Several variants of this theorem are proven in [6], [7].

Theorem 1 (Characterization of optimal policies). *For a given MDP $M_R = \{S, A, R, P, \gamma\}$, a policy $\pi \in \Delta_A^S$ is said optimal if and only if:*

$$\begin{aligned} V_R^\pi = V_R^* &\Leftrightarrow \forall s \in S, \text{Supp}(\pi(\cdot|s)) \subset \arg\max_{a \in A} [Q_R^\pi(s, a)], \\ &\Leftrightarrow \forall s \in S, \text{Supp}(\pi(\cdot|s)) \subset \arg\max_{a \in A} [Q_R^*(s, a)]. \end{aligned}$$

This theorem states that greedy policies with respect to the optimal quality function or to their own quality functions are optimal. It is central (see Sec. III) in our set-policy framework.

B. Inverse Reinforcement Learning (IRL)

IRL is a method that aims at finding a reward function R that could explain the expert policy π_E (which is considered optimal with respect to an unknown reward R_E) from demonstrations. Demonstrations are provided in the form of sampled transitions (possibly but not necessarily forming trajectories) of the expert policy. In general, IRL occurs in a batch setting where the dynamics P of the MDP is unknown and where no interaction with the MDP is possible while learning. Thus, only transitions sampled from an MDP and without rewards $\{S, A, P, \gamma\}$ are available to the apprentice. More formally, a batch IRL algorithm receives as inputs a set D_E of expert sampled transitions $D_E = (s_k, a_k, s'_k)_{1 \leq k \leq N_E}$ where $s_k \in S$, $a_k \sim \pi_E(\cdot|s_k)$ ⁶, and $s'_k \sim P(\cdot|s_k, a_k)$ and if available a set of non-expert sampled transitions $D_{NE} = (s_l, a_l, s'_l)_{1 \leq l \leq N_{NE}}$ where $s_l \in S$, $a_l \in A$, and $s'_l \sim P(\cdot|s_l, a_l)$ which provides partial information on the dynamics.

The goal of an IRL algorithm is to compute a reward R for which all expert actions *and only* expert actions are optimal:

$$\forall s \in S, \arg\max_{a \in A} Q_R^*(s, a) = \text{Supp}(\pi_E(\cdot|s)).$$

Indeed, preventing non-expert actions to be optimal with respect to the reward R is crucial to imitate the expert. To

do that, one must search for a *non-trivial reward*, that is a reward for which the expert policy is optimal and for which only expert actions are optimal or at least a subset of expert actions:

$$\forall s \in S, \arg\max_{a \in A} Q_R^*(s, a) \subset \text{Supp}(\pi_E(\cdot|s)). \quad (2)$$

The IRL literature thus consists of methods aiming at finding non-trivial rewards [22]–[24], [26], [27]. Once the reward is learned, the MDP $M_R = \{S, A, R, P, \gamma\}$ must be solved to compute the apprentice policy, which is a problem as such.

C. Imitation Learning (IL)

IL designates methods trying to generalize the policy π_E observed in the expert data set D_E to any situation. More formally, the aim of an IL algorithm is to find, for each state s , the set of actions that would have been performed by the expert $\text{Supp}(\pi_E(\cdot|s))$, or at least a subset of $\text{Supp}(\pi_E(\cdot|s))$, from the expert data set D_E and maybe some information on the dynamics P . Most often, IL is realized in a batch setting where only available information is D_E and D_{NE} . The IRL and IL batch settings thus share the same input sets.

Batch IL is often reduced to Multi-class Classification (MCC) [10], [15]. Basically, MCC learns a mapping $f \in Y^X$ between inputs $x \in X$ and labels $y \in Y$ (Y being finite) given a training set $(x_k, y_k)_{1 \leq k \leq N}$. Especially, score-based MCC learns a score function $L(x, y)$ so that the mapping is obtained by $f(x) = \arg\max_y L(x, y)$. In theory, MCC consists in finding a decision rule $f \in \mathfrak{H} \subset Y^X$ (where \mathfrak{H} is an hypothesis space) that minimizes the empirical risk:

$$\frac{1}{N} \sum_{k=1}^N \mathbf{1}_{\{y_k \neq f(x_k)\}}. \quad (3)$$

IL is reduced to MCC by identifying X to S and Y to A and the expert data set $D_{CE} = (s_k, a_k)_{1 \leq k \leq N_E}$ extracted from $D_E = (s_k, a_k, s'_k)_{1 \leq k \leq N_E}$ is used as the training set. A state s_k is thus seen as an input and the associated action a_k as a label for the classification algorithm. Then, the mapping f is identified to the decision rule $\pi \in A^S$ which can be interpreted as a policy.

In practice, minimizing the empirical risk is computationally hard and practitioners use convex surrogates [31]. In addition, one can notice that the empirical risk in Eq. (3) doesn't involve the dynamics P of the MDP. The dynamical nature of the environment is therefore ignored which is an important drawback of IL. However, MCC methods are easy to use and have been justified theoretically in a finite and infinite horizon settings [9], [10], [15], [19].

From now on, only score-based IL methods will be considered. They can also be reduced to score-based MCC methods [13], [16]. Because the learned score function L associates a score $L(x, y) \in \mathbb{R}$ to a pair (x, y) , we identify it to a Q -function ($Q \in \mathbb{R}^{S \times A}$) which associates a value $Q(s, a)$ to a pair (s, a) representing how good action a is in state s . The decision rule π associated to the score function Q consists in taking the action with the best score: $\pi(s) = \arg\max_{a \in A} Q(s, a)$. It is interesting to notice that the

⁶Let $x \in X$, the notation $x \sim \nu$ means that x is a realization of a random variable which is sampled according to $\nu \in \Delta_X$

same greedy process allows deriving a deterministic optimal policy π^* from an optimal quality function Q_R^* (see Th. 1) in the reinforcement learning framework. Thus, one can wonder if a score function can be interpreted as an optimal quality function and, if it is the case, which reward corresponds to this score function when the expert policy is applied. So, let us answer this question.

The goal of a score-based IL method is to compute a score function $Q \in \mathbb{R}^{S \times A}$ that gives the highest scores to expert actions:

$$\forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) = \operatorname{Supp}(\pi_E(\cdot|s)), \quad (4)$$

or at least some of them:

$$\forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) \subset \operatorname{Supp}(\pi_E(\cdot|s)) \text{ (similar to Eq. (2))},$$

which also means that the support (or at least a fraction of the support) of the expert policy is obtained by being greedy with respect to the score function Q . The score function Q can thus be directly interpreted as an optimal quality function with respect to a reward R (Q_R^*) if R verifies:

$$R = Q(s, a) - \gamma \mathbb{E}_{P(\cdot|s,a)}[f_Q^*]. \quad (5)$$

Indeed, with this definition of R , we have $Q(s, a) = R + \gamma \mathbb{E}_{P(\cdot|s,a)}[f_Q^*] = T_R^* Q(s, a)$ (from Eq. (1)) and by uniqueness of the fixed point of T_R^* , this means that $Q = Q_R^*$. The fact that a score function Q , verifying Eq (5), can be seen as an optimal quality function Q_R^* means that the expert actions and only the expert actions are optimal with respect to the reward R . Thus, R can be seen as the target of an IRL method. This naturally establishes a link between IL and IRL methods that we are going to study in a deeper fashion through the set-policy framework.

III. THE SET-POLICY FRAMEWORK

This section first defines the notion of set-policy. Then, score-based IL and IRL problems are cast within this framework. Moreover, it is shown that there exists a bijection between the space of solutions of the IRL problem and the one of the score-based IL problem obtained by the *inverse optimal Bellman operator* J^* (defined in Def. 9). This is the main result of the paper. It provides insights to better understand and to easily derive (see Sec. IV) existing algorithms. In addition, two IRL meta-algorithms that use the link between a score-function and an optimal quality function are provided.

Theorem 1 states that a Markovian stationary policy is optimal if and only if, for each state s , it selects an action in the specific set $\operatorname{argmax}_{a \in A} [Q_R^*(s, a)]$. Thus, to characterize the optimality of a policy π , the necessary and sufficient information is $\{\operatorname{Supp}(\pi(\cdot|s))\}_{s \in S}$, which is a set of finite and non-empty sets of actions. Therefore, it is quite natural to consider functions which associate a state to a non empty and finite set of actions. We call these functions *set-policies*.

Definition 1 (Set-policy). A set-policy $\bar{\pi}$ is an element⁷ of $(\mathcal{P}(A) \setminus \emptyset)^S$. The set of set-policies is noted $\bar{\Pi} = (\mathcal{P}(A) \setminus \emptyset)^S$. It contains $(2^{N_A} - 1)^{N_S}$ elements.

⁷If X is a non-empty set, then we note $\mathcal{P}(X)$ the powerset of X

Definition 2 (Inclusion and Equality for set-policies). Let $\bar{\pi}_1$ and $\bar{\pi}_2$ be two sets-policies, $\bar{\pi}_1 \subset \bar{\pi}_2$ if:

$$\forall s \in S, \bar{\pi}_1(s) \subset \bar{\pi}_2(s).$$

Moreover, $\bar{\pi}_1 = \bar{\pi}_2$ if $\bar{\pi}_1 \subset \bar{\pi}_2$ and $\bar{\pi}_2 \subset \bar{\pi}_1$.

To each policy π , we can associate a set-policy $\bar{\pi}$ called the *set-policy associated to π* and defined as follows:

Definition 3 (Associated set-policy). Let $\pi \in \Delta_A^S$, $\bar{\pi} \in \bar{\Pi}$ is called the *set-policy associated to π* and:

$$\forall s \in S, \operatorname{Supp}(\pi(\cdot|s)) = \bar{\pi}(s).$$

The associated set-policy of π indicates, for each state, the set of actions that might be chosen by π .

If π is a deterministic policy, it is interesting to remark that the set policy $\bar{\pi}$ associated to π is:

$$\forall s \in S, \operatorname{Supp}(\pi(\cdot|s)) = \bar{\pi}(s) = \pi_D(s).$$

Thus, the associated set-policy can be seen as a generalization of $\pi_D \in A^S$ for non-deterministic policies. Moreover, to each MDP M_R , we associate a special set-policy that we name *optimal set-policy generated by M_R* and defined as follows:

Definition 4 (Generated set-policy). Let $R \in \mathbb{R}^{S \times A}$, $\bar{\pi}_R^* \in \bar{\Pi}$, called the *optimal set-policy generated by M_R* , is defined as:

$$\forall s \in S, \bar{\pi}_R^*(s) = \operatorname{argmax}_{a \in A} [Q_R^*(s, a)].$$

The optimal set-policy generated by M_R indicates, for each state, the set of optimal actions to choose in order to optimize the reward R . Thanks to definitions 3 and 4, Theorem 1 can be rewritten: $V_R^\pi = V_R^* \Leftrightarrow \bar{\pi} \subset \bar{\pi}_R^*$.

Now, we have the tools to define the IRL problem in the set-policy framework. Originally, the IRL problem [12] consists in finding the unknown reward function R_E for which the expert policy π_E is optimal. Thus, we have $V_{R_E}^{\pi_E} = V_{R_E}^*$, which means via Theorem 1 that $\bar{\pi}_E \subset \bar{\pi}_{R_E}^*$. A first and naive approach, to solve the IRL problem in the set-policy framework, would be to find a reward function R for which the expert policy π_E is optimal:

$$V_R^{\pi_E} = V_R^* \Leftrightarrow \bar{\pi}_E \subset \bar{\pi}_R^*. \quad (6)$$

However, this approach is not safe in the sense that we can find R satisfying Eq. (6) such that $\bar{\pi}_{R_E}^* \subset \bar{\pi}_R^*$, which means that non-optimal actions for the original reward R_E could be optimal for the reward R as shown in Fig. 1(a). For instance,

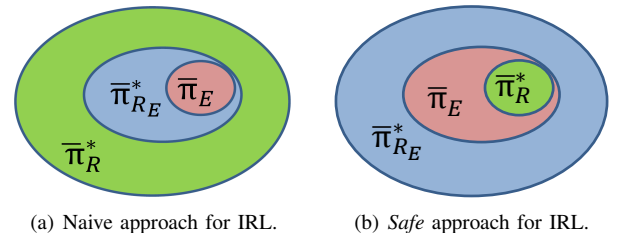


Fig. 1. Comparison between naive and safe-IRL.

the null reward function, for which all actions are optimal,

satisfies Eq. (6). Thus, this approach should not be considered. A second approach, to solve the IRL problem in the set-policy framework, would be to find a reward function R such that all optimal actions are also expert actions:

$$\pi_R^* \subset \pi_E \subset \pi_{R_E}^*. \quad (7)$$

This is the *safe* approach to IRL because, if one manages to find R satisfying Eq. (7), then optimal actions for the reward R are also optimal for the original reward R_E as shown in Fig. 1(b). However, it is likely that some actions shown by the expert are not optimal for R .

Definition 5 (Safe-IRL in the set-policy framework). *From the data sets D_E and D_{NE} , find a reward $R \in \mathbb{R}^{S \times A}$ such that $\pi_R^* \subset \pi_E$.*

Finally, it is possible to combine the two preceding approaches in order to find a reward R such that the optimal set-policy generated by M_R is equal to the set-policy π_E associated to the observed expert policy π_E . This is what we call IRL in the set-policy framework.

Definition 6 (IRL in the set-policy framework). *From the data sets D_E and D_{NE} , find a reward $R \in \mathbb{R}^{S \times A}$ such that $\pi_R^* = \pi_E$.*

This is the ideal case where all the expert actions and only the expert actions are optimal.

Now, the problem is to characterize, for a given MDP M_R and a given set-policy π , the set $C_\pi = \{R \in \mathbb{R}^{S \times A}, \pi = \pi_R^*\}$. Indeed by definition, the set of rewards C_{π_E} are the solutions of the IRL problem in the set-policy framework and the set of rewards $\bigcup_{\pi \subset \pi_E} C_\pi$ are the solutions of the safe-IRL problem. As $C_{\pi_E} \subset \bigcup_{\pi \subset \pi_E} C_\pi$, it is easier to solve the safe-IRL problem. Characterizations of the IRL solutions set have already been done for deterministic policies [12] and for stationary and Markovian policies [32]. However, the set-policy framework allows proposing the notion of safe-IRL which is new and central in order to retrieve recent IRL algorithms.

In addition, the set-policy framework allows defining easily the notion of score-based IL. Indeed, for a fixed set-policy π , let us consider the set H_π of score functions such that:

$$H_\pi = \{Q \in \mathbb{R}^{S \times A}, \forall s \in S, \arg\max_{a \in A} Q(s, a) = \pi(s)\}. \quad (8)$$

The set H_π contains all the score functions Q for which:

$$\forall s \in S, \arg\max_{a \in A} Q(s, a) = \pi(s) = \text{Supp}(\pi(\cdot|s)).$$

Thus, by definition (see Eq. (4)), H_{π_E} is the set of score-based solutions of the IL problem for the policy π_E associated to the set policy π_E . So, in the set-policy framework IL can be defined as:

Definition 7 (Definition of IL in the set-policy framework). *From the data sets D_E and D_{NE} , find $Q \in H_{\pi_E}$.*

Moreover, it is also possible to define the safe-IL solutions set for the expert policy π_E which is $\bigcup_{\pi \subset \pi_E} H_\pi$. This set contains all the score functions Q verifying:

$$\forall s \in S, \arg\max_{a \in A} Q(s, a) \subset \pi_E(s) = \text{Supp}(\pi_E(\cdot|s)).$$

Definition 8 (Definition of safe-IL in the set-policy framework). *From the data sets D_E and D_{NE} , find $Q \in \bigcup_{\pi \subset \pi_E} H_\pi$.*

Those score-functions select only expert actions. This is interesting as it guarantees to obtain an optimal behavior with respect to the original reward R_E for which the expert is optimal.

Now, that we have formally defined the notions of IRL and score-based IL in the set-policy framework, a formal link can be established between them. To do so, let us introduce the operator J^* , called the inverse optimal Bellman operator [32], which is a function from $\mathbb{R}^{S \times A}$ to $\mathbb{R}^{S \times A}$ (here we also define the inverse Bellman operator J^π for a given policy $\pi \in \Delta_A^S$).

Definition 9 (Definitions of J^* and J^π). $\forall Q \in \mathbb{R}^{S \times A}, \forall (s, a) \in S \times A$:

$$J^*Q(s, a) = Q(s, a) - \gamma \mathbb{E}_{P(\cdot|s, a)}[f_Q^*], \quad (9)$$

$$J^\pi Q(s, a) = Q(s, a) - \gamma \mathbb{E}_{P(\cdot|s, a)}[f_Q^\pi]. \quad (10)$$

The operator J^* represents the one-to-one relation existing between the optimal quality function Q_R^* and its reward function R as shown in Theorem 2.

Theorem 2 (Properties of J^* and J^π). *The inverse optimal Bellman operator J^* is a bijection from $\mathbb{R}^{S \times A}$ to $\mathbb{R}^{S \times A}$ and we have:*

$$\forall Q \in \mathbb{R}^{S \times A}, Q = Q_R^*, \text{ with } R = J^*Q,$$

which also means that $\forall R \in \mathbb{R}^{S \times A}, (J^)^{-1}R = Q_R^*$. In addition, for a given policy $\pi \in \Delta_A^S$, the inverse Bellman operator J^π is a bijection from $\mathbb{R}^{S \times A}$ to $\mathbb{R}^{S \times A}$ and we have:*

$$\forall Q \in \mathbb{R}^{S \times A}, Q = Q_R^\pi, \text{ with } R = J^\pi Q,$$

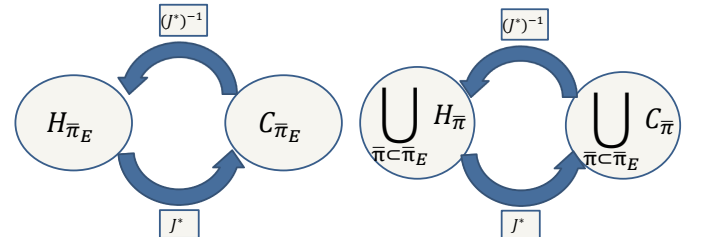
which also means that $\forall R \in \mathbb{R}^{S \times A}, (J^\pi)^{-1}R = Q_R^\pi$.

Proof of Theorem 2 is given in appendix VI-A. It is straightforward via the inverse Bellman operator properties that:

$$J^*(H_\pi) = \{R \in \mathbb{R}^{S \times A}, \forall s \in S, \arg\max_{a \in A} [Q_R^*(s, a)] = \pi(s)\},$$

$$= \{R \in \mathbb{R}^{S \times A}, \pi_R^* = \pi\} = C_\pi.$$

In particular, $J^*(H_{\pi_E}) = C_{\pi_E}$. So, the IRL solutions set C_{π_E} is the image of the IL solutions set H_{π_E} by the operator J^* as shown in Fig. 2(a). Contrary to C_{π_E} , it is easy to characterize



(a) Formal link between IL solutions set and IRL solutions set. (b) Formal link between safe-IL solutions set and safe-IRL solutions set.

Fig. 2. Equivalence between IL and IRL problems.

concretely an element of $H_{\bar{\pi}_E}$ as it is shown by Eq. (8). This characterization does not depend on the dynamics P and only on the set-policy $\bar{\pi}_E$.

Moreover, $\{H_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition of $\mathbb{R}^{S \times A}$ (see appendix VI-B for the proof).

Theorem 3. $\{H_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition of $\mathbb{R}^{S \times A}$. In addition, for each $\bar{\pi} \in \bar{\Pi}$, $H_{\bar{\pi}}$ has an infinite cardinal.

Therefore, as J^* is a bijection and as for each policy $\bar{\pi}$, $J^*(H_{\bar{\pi}}) = C_{\bar{\pi}}$, then $\{C_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is also a finite partition of $\mathbb{R}^{S \times A}$. This guarantees that $C_{\bar{\pi}_E}$ is not empty. So, the IRL problem in the set-policy framework has at least one solution. However, the number of solutions is infinite because, for each $\bar{\pi} \in \bar{\Pi}$, the set $H_{\bar{\pi}}$ has an infinite cardinal. Thus, the IRL solution set has to be constrained in order to obtain a unique solution or, at least, restrain the solution set. For instance, one can force a linear parameterization of the reward.

In addition, as $\{H_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition:

$$J^*\left(\bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}\right) = \bigcup_{\bar{\pi} \subset \bar{\pi}_E} J^*(H_{\bar{\pi}}) = \bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}},$$

which means there exists also a formal link between safe-IL solutions $\bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ and safe-IRL solutions $\bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}}$ as shown in Fig. 2(b). This implies, for instance, that if one finds a score-function Q which is a solution of the safe-IL problem, then by applying J^* to Q one finds a reward function $R = J^*Q$ which is a solution of the safe-IRL problem. When faced with batch data D_E and D_{NE} , finding $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ can be done by using a classification method and the application of J^* can be seen as a regression step (see Sec. IV-A).

Now, we propose two canonical meta-algorithms that allow finding or retrieving IRL algorithms via the set-policy framework. These two meta-algorithms are presented in the safe-IRL case but can be adapted to the IRL case by using $H_{\bar{\pi}_E}$ in lieu of $\bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$. We choose to focus on safe-IRL algorithms because they allow retrieving existing algorithms of the literature. The first meta-algorithm consists in two steps. The first step is to find $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ which is a score-based safe-IL step. The second step consists in applying the operator J^* to Q . The result, R , is guaranteed to be in $\bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}}$ as $J^*(\bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}) = \bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}}$. The second algorithm consists in directly searching R such that $(J^*)^{-1}R = Q_R^* \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$. As $(J^*)^{-1}(\bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}}) = \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$, this means that $R \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} C_{\bar{\pi}}$.

Algorithm 1 Meta-Algorithm 1 for safe-IRL

Require: π_E .

- 1: Find $Q \in \mathbb{R}^{S \times A}$, such that $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ (This is a score-based safe-IL step).
 - 2: $R = J^*Q$.
-

Algorithm 2 Meta-Algorithm 2 for safe-IRL

Require: π_E .

- 1: Find $R \in \mathbb{R}^{S \times A}$ such that $(J^*)^{-1}R = Q_R^* \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$.
-

IV. IRL AND IL ALGORITHMS THROUGH THE SET-POLICY FRAMEWORK

In this section, we revisit recent IRL algorithms of the literature in the light of the set-policy framework. This is done by showing how the different steps of the previous meta-algorithms can be computed when faced with the data sets D_E and D_{NE} of the batch IL and IRL settings. Moreover, a general framework to find new score-based IL algorithms is proposed. It combines the main advantages of IRL (using the underlying dynamics of the MDP) and IL (simplicity and efficiency).

A. Cascaded Supervised IRL (CSI)

Here, CSI [28] is derived from meta-algorithm 1. The first step is a score-based safe-IL problem and consists in finding Q such that $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$. In particular, we can search $Q \in H_{\bar{\pi}}$ where $\bar{\pi} \subset \bar{\pi}_E$ and $\forall s \in S, \bar{\pi}(s)$ is a singleton. This means that the deterministic policy $\pi \in A^S$, defined as $\forall s \in S, \pi(s) = \bar{\pi}(s)$, is the only greedy policy with respect to Q . Moreover, as we want to imitate the expert, the deterministic policy π associated to $\bar{\pi}$ must be as similar as possible to π_E . So, we can ask that $\pi(s) \in \operatorname{argmax}_{a \in A} \pi_E(a|s)$. Thus, we are looking for a score function Q such that there exists a deterministic policy π verifying:

$$\begin{aligned} \forall s \in S, \pi(s) &\in \operatorname{argmax}_{a \in A} \pi_E(a|s), \\ \forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) &= \pi(s). \end{aligned}$$

This can be seen as a classification problem. Indeed, given two random variables \mathbf{x} taking its values in X and \mathbf{y} taking its values in Y (X and Y are finite spaces), a classification problem consists in finding a deterministic function $f \in \mathfrak{H} \subset Y^X$, where \mathfrak{H} is an hypothesis space, such that:

$$\forall x \in X, f(x) \in \operatorname{argmax}_{y \in Y} \kappa(x|y),$$

where $\kappa(\cdot|\cdot) \in \Delta_Y^X$ is the transition kernel between the variables \mathbf{x} and \mathbf{y} . If the deterministic function f is derived from a score function $L \in \mathbb{R}^{X \times Y}$ such that $\forall x \in X, \operatorname{argmax}_{y \in Y} L(x, y) = f(x)$ then it is a score-based MCC problem. Here, we are looking for a deterministic function $\pi \in A^S$ which approximates as well as possible the transition kernel $\pi_E \in \Delta_A^S$. As the deterministic function π is derived from a score function Q such that $\forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) = \pi(s)$, then the score-based IL problem can be reduced to a score-based classification problem. In practice, when faced with the input data sets D_E and D_{NE} , a classical score-based MCC algorithm used in IL is the large-margin approach [13], [33]. This algorithm only uses the set D_{CE} extracted from D_E and as a consequence does not take into account the underlying dynamics of the MDP.

The second step of the meta-algorithm 1 is $R = J^*Q$, which can be rewritten as:

$$\forall (s, a) \in S \times A, R(s, a) = \mathbb{E}_{P(\cdot|s, a)}[Q(s, a) - \gamma f_Q^*]. \quad (11)$$

Thus, if the dynamics P is available, R can be computed directly from Q for any state-action couple. In the case where

P is only known through the input data sets D_E and D_{NE} , Eq. (11) can be seen as a regression. Indeed a regression problem consists, given two random variables \mathbf{x} taking its values in X and \mathbf{y} taking its values in \mathbb{R} , in finding a function $f^* \in \mathbb{R}^X$, such that $f^*(x) = \mathbb{E}[\mathbf{y}|\mathbf{x}] = \min_{f \in \mathbb{R}^X} \int_{x \in X, y \in \mathbb{R}} (f(x) - y)^2 \kappa(dx, dy)$ where $\kappa(\cdot, \cdot)$ is the probability distribution of the couple (\mathbf{x}, \mathbf{y}) . This function f^* can also be written $f^*(x) = \mathbb{E}_{P(\cdot|\mathbf{x})}[\mathbf{y}] = \int_{y \in \mathbb{R}} \kappa(y|\mathbf{x}) y dy$ where $\kappa(\cdot|\cdot)$ is the transition kernel between the variables \mathbf{x} and \mathbf{y} . So, $R(s, a)$ has the same form as $f^*(x)$ where $x = (s, a)$, $\mathbf{y} = Q(s, a) - \gamma f_Q^*$ and $\kappa(y|\mathbf{x}) = P(Q(s, a) - \gamma \max_{b \in A} Q(s', b) | s, a)$. When faced with samples $D = (x_i \in X, y_i \in \mathbb{R})_{1 \leq i \leq N}$, a regression algorithm consists in finding a function $f \in \mathcal{H} \subset \mathbb{R}^X$, \mathcal{H} is an hypothesis space, which is a good estimation of f^* . Here, our regression data set $D_R = \{(s_i, a_i), \hat{r}_i\}_{1 \leq i \leq N_R}$ is constructed from $D_E \cup D_{NE} = (s_i, a_i, s'_i)_{1 \leq i \leq N_R}$ where $\hat{r}_i = Q(s_i, a_i) - \gamma \max_{a \in A} Q(s'_i, a)$ and $N_R = N_E + N_{NE}$. It can be performed with a regression tree [34], [35] or by a least-squares method for instance. Thus, it appears that meta-algorithm 1 gives us canonically the algorithm CSI which consists of a first step of classification followed by a second step of regression. CSI is summarized by Algo 3. It is interesting to notice that, by choosing non-parametric MCC and regression algorithms, CSI is made non-parametric and model-free. In addition, CSI has theoretical guarantees which are presented in the original paper [28].

Algorithm 3 The Cascaded-Supervised Learning Algorithm

Require: The sets D_E and D_{NE} .

- 1: Compute the function Q thanks to a score-based IL algorithm.
 - 2: Construct the data set $D_R = \{(s_i, a_i), \hat{r}_i\}_{1 \leq i \leq N_R}$.
 - 3: Return the function \hat{R} thanks to a regression algorithm fed by the data D_R .
-

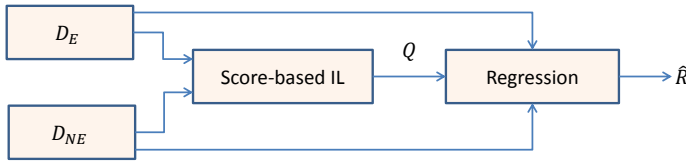


Fig. 3. Illustration of the CSI algorithm.

B. Structured Classification for IRL (SCIRL)

In this section, SCIRL [27] is derived from the meta-algorithm 2. It consists in finding $R \in \mathbb{R}^{S \times A}$ such that $(J^*)^{-1} R \in \bigcup_{\pi \in \pi_E} H_{\pi}$. In the SCIRL framework, the expert policy is supposed to be deterministic, thus $\bigcup_{\pi \in \pi_E} H_{\pi} = H_{\pi_E}$. In addition, as $\forall Q \in H_{\pi_E}, f_Q^{\pi_E} = f_Q^*$ which implies that $\forall Q \in H_{\pi_E}, J^* Q = J^{\pi_E} Q$ and as J^* is a bijection, then:

$$\forall R \in C_{\pi_E}, (J^*)^{-1} R = (J^{\pi_E})^{-1} R.$$

So, we are looking for R such that $(J^{\pi_E})^{-1} R = Q_R^{\pi_E} \in H_{\pi_E}$.

Moreover, in the SCIRL framework, the reward is considered as a linear combination of some features which are a finite

set of $p \in \mathbb{N}^*$ functions $(\phi_i)_{i=1}^p$ such that $\phi_i \in \mathbb{R}^{S \times A}$. Thus, for each vector $\theta \in \mathbb{R}^p$, a reward function R_{θ} can be associated such that: $R_{\theta}(s, a) = \sum_{i=1}^p \phi_i(s, a) \theta_i = \theta^{\top} \phi(s, a)$, where $\phi(s, a) = (\phi_i(s, a))_{i=1}^p \in \mathbb{R}^p$. The linear parameterization of the reward imposes a linear parameterization of the quality function $Q_{R_{\theta}}^{\pi_E}$:

$$\begin{aligned} Q_{R_{\theta}}^{\pi_E}(s, a) &= \mathbb{E}_{s,a}^{\pi_E} \left[\sum_{t=0}^{+\infty} \gamma^t R_{\theta}(s_t, a_t) \right], \\ &= \theta^{\top} \mathbb{E}_{s,a}^{\pi_E} \left[\sum_{t=0}^{+\infty} \gamma^t \phi(s_t, a_t) \right] = \theta^{\top} \mu_{\pi_E}(s, a), \end{aligned}$$

where $\mu_{\pi_E}(s, a) = \mathbb{E}_{s,a}^{\pi_E} [\sum_{t=0}^{+\infty} \gamma^t \phi(s_t, a_t)]$ is the so-called expert feature expectation for (s, a) [22]. When faced with the batch data set D_E , the expert feature expectation $\mu_{\pi_E}(s, a)$ can be estimated via an LSTD-like algorithm which is called LSTD- μ [36] and the help of some heuristics [27]. The result of the estimation is $\hat{\mu}_{\pi_E}$ and the estimation of $Q_{R_{\theta}}^{\pi_E}(s, a)$ is therefore $\hat{Q}_{R_{\theta}}^{\pi_E}(s, a) = \theta^{\top} \hat{\mu}_{\pi_E}(s, a)$.

The problem becomes: find a reward R_{θ} in the set $\mathfrak{R} = \{R_{\theta} \in \mathbb{R}^{S \times A}, \theta \in \mathbb{R}^p\}$ such that $\hat{Q}_{R_{\theta}}^{\pi_E} \theta \in H_{\pi_E}$. As seen before (see Sec. IV-A), this kind of problems can be reduced to a score-based MCC algorithm which consists in finding a score function $Q_{\theta} \in H_{\pi_E}$ in the hypothesis set of functions $\Omega \subset \mathbb{R}^{S \times A}$ where:

$$\Omega = \{Q_{\theta} \in \mathbb{R}^{S \times A}, Q_{\theta}(s, a) = \theta^{\top} \hat{\mu}_{\pi_E}(s, a), \theta \in \mathbb{R}^p\}.$$

As in [27], a large margin method [33] can be used to realize this classification step which is the choice done in our experiments for the implementation of SCIRL. The output of this classification algorithm is a vector θ_C and the output of the SCIRL algorithm is the reward R_{θ_C} . So, meta-algorithm 2 leads to SCIRL which consists in an evaluation step of the feature expectation μ_{π_E} followed by a score-based classification algorithm. In [27], the authors give theoretical results on the near-optimality of the expert-policy with respect to the reward output by the algorithm. SCIRL is presented in Algo. 4.

Algorithm 4 The SCIRL Algorithm

Require: The sets D_E and D_{NE} .

- 1: Compute the estimation $\hat{\mu}_{\pi_E}$ via LSTD- μ .
 - 2: Compute θ_C thanks to a score based IL with hypothesis set Ω .
 - 3: Return R_{θ_C} .
-

Contrary to CSI, SCIRL is by nature a parametric algorithm as the reward is linearly parameterized by the features $(\phi_i)_{i=1}^p$. Thus, a good choice of features is important in order to have a correct representation of the reward. This matter is discussed in our experiments where the performance of SCIRL with a *good* representation of the reward versus a *bad* one are compared.

C. A general approach to design IL algorithms

This section provides a general method to derive score-based IL algorithms that makes use of the underlying dynamics of the MDP thanks to the set-policy framework. Notice that,

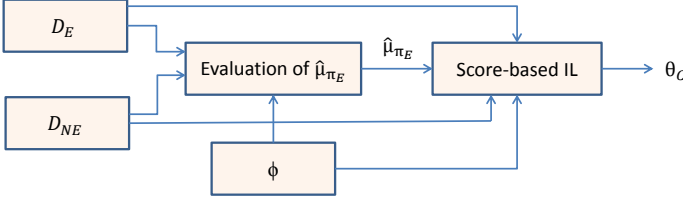


Fig. 4. Illustration of the SCIRL algorithm.

as they are also MCC algorithms, they can be used as a first step of the CSI algorithm. In addition, they combine the main advantages of IL methods (simplicity, efficiency and computing a policy without solving any MDPs) and IRL methods (accounting for the underlying dynamics).

It was shown earlier that score-based IL consists in finding a score function $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$. To obtain a satisfying solution, the hypothesis space is constrained to exhibit two properties : having a small complexity and to contain an element of $\bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$. Indeed, a small complexity reduces the variance of the classification algorithm [37] and having $\mathfrak{H} \cap \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ non-empty guarantees that there is no bias.

The hypothesis space \mathfrak{H} can be chosen, for instance, by constraining the score functions. In addition, to leverage the information of dynamics in the data sets D_E and D_{NE} , the constraint should integrate the dynamics P of the MDP. As seen previously, each score function Q can be seen as an optimal quality function $Q = Q_R^*$ where $R = J^*Q$. The reward associated to Q by J^* , $R = J^*Q$, will also be noted R_Q . To introduce in a natural way the dynamics P appearing in J^* , the reward function R_Q is constrained in lieu of the score function Q . Possible constraints on the reward may depend on some prior such as a state-only dependency, linear parameterization, a Lipschitz condition, a sparsity, *etc.* Thus, let us note $\mathfrak{H} = \{Q \in \mathbb{R}^{S \times A}, R_Q \text{ satisfies a constraint}\}$. In practice, finding $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$ is realized by score-based MCC solved by minimizing a criterion $L(Q, D_{CE})$ (for instance the large margin one [33]) depending on the score-function Q and the dataset D_{CE} :

$$L(Q, D_{CE}) = \frac{1}{N_E} \sum_{i=1}^{N_E} \max_{a \in A} [Q(s_i, a_i) + l(s_i, a, a_i)] - Q(s_i, a_i),$$

where $l \in \mathbb{R}^{S \times A \times A}$ is a margin function that imposes a structure on the score function Q . In order to adapt this algorithm to find $Q \in \mathfrak{H} \cap \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$, one can solve the following optimization problem:

$$\begin{aligned} \min_{Q \in \mathbb{R}^{S \times A}} L(Q, D_{CE}), \\ \text{such that } Q \in \mathfrak{H}. \end{aligned}$$

For instance, for RCAL [16], the constraint consists in choosing an associated reward function R_Q with a small $\mathbf{L}_{1,\nu}$ -norm which favors the sparseness of the reward. The sparse reward constraint is quite natural in the MDP framework as it is the type of rewards that are encountered in practical problems. In addition, in MDPs with sparse rewards, the dynamics plays an important role. Indeed, the agent must

perform a long sequence of actions before receiving a reward [15]. Thus, the principle of RCAL, illustrated in Fig 5, consists in finding a score function in the hypothesis space $\mathfrak{H} = \{Q \in \mathbb{R}^{S \times A}, \|R_Q\|_{1,\nu} \leq \eta\}$, where $\eta \in \mathbb{R}_+^*$ is small and such that $Q \in \bigcup_{\bar{\pi} \subset \bar{\pi}_E} H_{\bar{\pi}}$.

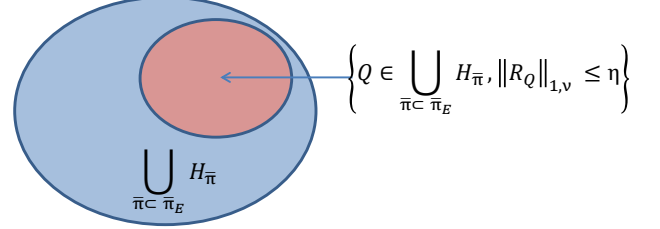


Fig. 5. Search space of RCAL.

In order to integrate the information of dynamics, the score-function can be taken from $\mathfrak{H} = \{Q \in \mathbb{R}^{S \times A}, \|R_Q\|_{1,\nu} \leq \eta\}$ where η is the smallest possible. Thus, we search Q that minimize the following constrained problem:

$$\begin{aligned} \min_{Q \in \mathbb{R}^{S \times A}, \eta \in \mathbb{R}^*} L(Q, D_{CE}) + \lambda \eta, \\ \text{such that } \|R_Q\|_{1,\nu} \leq \eta, \end{aligned}$$

where λ is a trade-off parameter between having a small classification criterion $L(Q, D_{CE})$ and having a small norm $\|R_Q\|_{1,\nu} = \|J^*Q\|_{1,\nu}$. This problem, as the variable η is tight and positive, is equivalent to minimizing the following unconstrained regularized criterion:

$$L(Q, D_{CE}, \lambda) = L(Q, D_{CE}) + \lambda \|R_Q\|_{1,\nu}.$$

However, as the dynamics P is unknown, it is not possible to compute $R_Q = J^*Q$. The idea presented in [16] consists in using a biased proxy of the norm $\|R_Q\|_{1,\nu}$ which is:

$$\frac{1}{N_R} \sum_{i=1}^{N_R} |\hat{r}_i| = \frac{1}{N_R} \sum_{i=1}^{N_R} |Q(s_i, a_i) - \gamma \max_{a \in A} Q(s'_i, a)|,$$

where $(s_i, a_i, s'_i) \in D_R = D_E \cup D_{NE}$ and $\hat{r}_i = Q(s_i, a_i) - \gamma \max_{a \in A} Q(s'_i, a)$ is an unbiased estimate of the reward $R_Q(s_i, a_i) = Q(s_i, a_i) - \gamma \mathbb{E}_{P(\cdot|s_i, a_i)}[f_Q^*]$. Finally, RCAL consists in minimizing:

$$L(Q, D_{CE}, D_R, \lambda) = L(Q, D_{CE}) + \frac{\lambda}{N_R} \sum_{i=1}^{N_R} |\hat{r}_i|.$$

This criterion is not convex, hence there is no guarantee to obtain a global minimum using a (sub-)gradient minimization technique. However, it can be seen as a perturbation of the convex criterion $L(Q, D_{CE})$. Thus, a good start point to minimize $L(Q, D_{CE}, D_R, \lambda)$ can be a minimizer of $L(Q, D_{CE})$. For more details about the soundness of RCAL or about the minimization technique of $L(Q, D_{CE}, D_R, \lambda)$, one can refer to the original paper [16].

V. EXPERIMENTS

A. Motivations

This section reports comparisons of IRL algorithms derived from the trajectory-matching framework [25] to IRL

algorithms from the set-policy framework (CSI and SCIRL) and to IL algorithms (Tree, Classif and RCAL). From the trajectory matching framework, selected algorithms are the Projection algorithm [22] (Proj), the Multiplicative Weight Apprenticeship Learning algorithm [23] (MWAL) and the Max Margin Planning algorithm [38] adapted to IRL [25] (MMP). In addition, performances of three IL algorithms are reported. First, Classif is a large margin classifier [13]. Second, Tree is a classification tree [34]. The last one is RCAL with $\lambda = 0.1$.

Algorithms were tested on different tasks in their respective original papers. Some of those tasks were toy problems and others were real ones. Unlike toy problems, in real problems the dynamics P of the MDP is often unknown, a canonical linear parameterization of the reward is often not provided and real problems are much larger. The knowledge of the dynamics P through the data set D_{NE} and features $(\phi_i)_{i=1}^p$ are key ingredients of most IRL algorithms. This may not be the case for IL algorithms which can be non-parametric. Also, they use only the expert state-action couples data set D_{CE} . So, it is interesting to see if the IRL algorithms are suited for real problems where those ingredients are not perfectly provided.

Thus, this experiment try to establish how the knowledge of the dynamics P and a good linear parameterization of the reward can affect the performance of the IRL algorithms. That is why a generic task is chosen for comparison. It consists in generating randomly MDPs and test the algorithms on a large set of those problems. Those MDPs, called Garnet [29], which are slightly modified [16] from the ones presented in the original paper, are representative of the kind of MDPs that might be encountered in practice (see Sec. V-B for details). They are finite MDPs where the dynamics P is known and a tabular basis can be used. Thus, an optimal policy can be computed for each Garnet via the policy iteration algorithm, and the quality and value functions can be evaluated for any policy and any reward. It is also easy to sample transitions of a Garnet for any policy. It is also possible to give the dynamics P and a tabular basis to the user. As an optimal policy can be exactly computed, the trajectory matching framework, often requiring to solve repeatedly MDPs, can easily be applied. In addition, when P is known, CSI can use directly the operator J^* in order to retrieve the reward. When P is not provided, a batch RL algorithm such as LSPI (Least Squares Policy Iteration) [39] or Fitted-Q [40] can be used as an approximate MDP solver for the IRL algorithms of the trajectory matching framework. In our experiments, the batch RL algorithm used is LSPI and the estimation of the expert feature expectation for SCIRL is done via LSTD- μ [36].

Besides, a choice of features must be done for most IRL algorithms (one exception is CSI). The features chosen are either a tabular basis in order to guarantee a perfect representation of the reward or Radial Basis Functions (RBF) (see Sec. V-B for details). Notice that the only IRL algorithm without a feature choice is CSI. Indeed, a non-parametric version of this algorithm is presented here. Its classification step is realized by a non-parametric version of RCAL and the regression step by a regression tree.

Algorithms are compared in four different settings. The first setting stands for the perfect setting where P and a tabular

basis are provided to the user. In the second setting, the dynamics P is unknown and the tabular basis is provided. In the third setting, the dynamics is known and the RBF basis is used. The fourth setting approximates the real case scenario where the dynamics is unknown and the RBF basis is used.

B. The Garnet experiment

This experiment focuses on stationary Garnet problems, which are a class of randomly constructed finite MDPs representative of the kind of finite MDPs that might be encountered in practice. A stationary Garnet problem is characterized by 3 parameters: $Garnet(N_S, N_A, N_B)$. The parameters N_S and N_A are the number of states and actions respectively, and N_B is a branching factor specifying the number of next states for each state action pair. In this experiment, the Garnets present a topological structure relative to real dynamical systems. Those systems are generally multi-dimensional state spaces MDPs where an action leads to different next states close to each other. The fact that an action leads to close next states can model the noise in a real system for instance. Thus, problems such as the highway simulator [27], the mountain car or the inverted pendulum (possibly discretized) are particular cases of this type of Garnets. For those particular Garnets, the state space is composed of d dimensions ($d = \{1, 2, 3\}$ in this particular experiment) and each dimension i has a finite number of elements x_i ($x_i = 10$). So, a state $s = [s^1, s^2, \dots, s^i, \dots, s^d]$ is a d -uple where each component s^i can take a finite value between 1 and x_i . In addition, the distance between two states s, s' is $\|s - s'\|_2^2 = \sum_{i=1}^d (s^i - s'^i)^2$. Thus, we obtain MDPs with a state space size of $\prod_{i=1}^d x_i$. For instance, when $d = 3$, the state space size is $N_S = 1000$. The number of actions is fixed to $N_A = 5$. For each state action couple (s, a) , the N_B next states ($N_B = 5$) are chosen randomly via a Gaussian distribution of d dimensions centered in s where the covariance matrix is the identity matrix of size d , I_d , multiplied by a term σ (here $\sigma = 1$). The term σ allows handling the smoothness of the MDP: if σ is small the next states s' are close to s and if σ is large, the next states s' can be very far from each other and also from s . The probability of going to each next state s' is generated by partitioning the unit interval at $N_B - 1$ cut points selected randomly. A sparse expert reward R_E is built by choosing $\frac{N_S}{10}$ states (uniform random choice without replacement) where $R_E(s, a) = 1$, elsewhere $R_E(s, a) = 0$. For each Garnet problem, it is possible to compute an expert policy $\pi_E = \pi^*$ and the expert value function $V_{R_E}^{\pi_E}$ via the policy iteration algorithm.

In this experiment, 100 Garnets $\{G_p\}_{1 \leq p \leq 100}$ were generated as explained before. For each Garnet G_p , 10 data sets $\{D_E^{p,q}\}_{1 \leq q \leq 10}$ are generated, composed of L_E trajectories of H_E sampled transitions $(s_i, \pi_E(s_i), s'_i)$ of the expert policy π_E and 10 data sets $\{D_{NE}^{p,q}\}_{1 \leq q \leq 10}$ of L_{NE} trajectories of H_{NE} sampled transitions of the random policy (for each state, the action is uniformly chosen over the set of actions) (s_i, a_i, s'_i) . Each trajectory begins from a state picked uniformly over the state space, this uniform distribution is noted ρ . There are 4 settings considered. In the first setting the IRL algorithm uses the knowledge of the dynamics P and

a tabular basis to parameterize the reward function. In the second setting, P is not provided but replaced by $D_{NE}^{p,q}$. Thus LSPI is used as an MDP solver. LSPI needs a set of sampled transitions of the form (s_i, a_i, r_i, s'_i) as input. Here, this can be easily provided to optimize a reward \hat{R} as we can use the data set $D_R^{p,q} = D_E^{p,q} \cup D_{NE}^{p,q}$ where to each transition (s_i, a_i, s'_i) of $D_R^{p,q}$ we add the information $r_i = \hat{R}(s_i, a_i)$. In the third setting, the reward is parameterized by an RBFs basis and P is provided. Finally, in the fourth setting the dynamics is unknown and an RBF basis is used. It is important to note that the provided basis (tabular or RBFs) are used by LSPI and LSTD- μ too. The RBFs basis consists in choosing N_{RBF} states $(s_j^c)_{1 \leq j \leq N_{RBF}}$ without replacement from the state space which are called centered states (the choice can be done such that the centered states are uniformly distributed). Then, it is possible to define a notion of similarity between couples (s_j^c, a) and (s, b) by the following formula $\forall 1 \leq j \leq N_{RBF}, \forall (s, a, b) \in S \times A \times A$:

$$\exp\left(-\frac{\|s_j^c - s\|_2^2}{\sigma_\phi}\right)\delta_A(a, b),$$

where $\sigma_\phi \in \mathbb{R}_+$ is a parameter that controls the width of the RBF around the center s_j^c . Finally, let us define a feature extractor ϕ that represents the RBFs basis. ϕ is a matrix of size $(N_S N_A, N_C N_A)$ such that:

$$\phi(i + (N_S - 1)p, j + (N_C - 1)q) = \exp\left(-\frac{\|s_j^c - s_i\|_2^2}{\sigma_\phi}\right)\delta_A(a_p, a_q).$$

The RBF basis represented by ϕ is an approximation of the tabular basis because if $N_{RBF} = N_S$ and $\sigma_\phi \rightarrow 0$, then the RBF basis is exactly the tabular basis. In the experiments, the centered states are chosen uniformly distributed over the states space such that $N_C = \frac{N_S}{2}$ and $\sigma_\phi = 1$. Recall that the trajectory matching IRL algorithms need to solve repeatedly MDPs. Here, we fix the number of solving steps to 10.

For each data set $D_E^{p,q}$ and $D_{NE}^{p,q}$, an IRL algorithm A outputs a reward $R_A^{p,q}$. Performances are compared according to $T_A^{p,q}$ which represents the normalized error between the expert policy and the optimal policy with respect to $R_A^{p,q}$:

$$T_A^{p,q} = \frac{\mathbb{E}_\rho[\|V_{R_E}^{\pi_E} - V_{R_E}^{\pi_A^{p,q}}\|]}{\mathbb{E}_\rho[\|V_{R_E}^{\pi_E}\|]}.$$

For IL algorithms, for each data sets $D_E^{p,q}$ and $D_{NE}^{p,q}$, the criterion of performance is $T_A^{p,q}$:

$$T_A^{p,q} = \frac{\mathbb{E}_\rho[\|V_{R_E}^{\pi_E} - V_{R_E}^{\pi_A}\|]}{\mathbb{E}_\rho[\|V_{R_E}^{\pi_E}\|]},$$

where π_A is the policy output by the IL algorithm A . Therefore, the lower $T_A^{p,q}$ is, the better is the performance.

For a given algorithm A , the mean performance criterion T_A is $\frac{1}{1000} \sum_{p=1}^{1000} \sum_{q=1}^{10} T_A^{p,q}$. For each algorithm, the standard deviation $\text{std}_A^p = (\frac{1}{10} \sum_{q=1}^{10} (T_A^{p,q} - \frac{1}{10} \sum_{q=1}^{10} T_A^{p,q})^2)^{\frac{1}{2}}$ and the mean standard deviation is $\text{std}_A = \frac{1}{100} \sum_{p=1}^{100} \text{std}_A^p$ are provided. Fig. 6 shows the performance criterion T_A and the mean standard deviation std_A for the first setting when $d = 3$. Here, $L_E = 5$ and H_E is the evolving parameter.

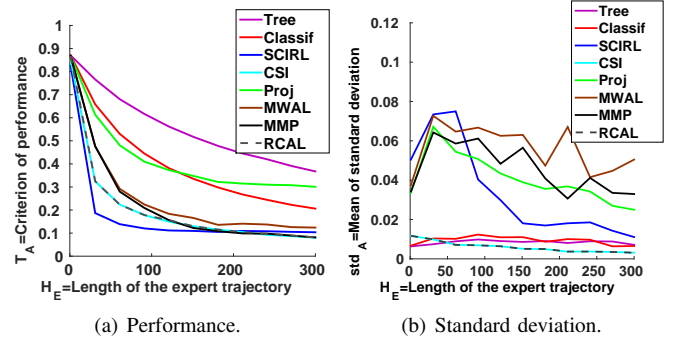


Fig. 6. Garnet Experiment: First setting with $d = 3$ and $L_E = 5$.

In Fig. 6(a), one can observe that CSI and SCIRL perform better when the amount of data is small. This is particularly true for SCIRL. However, when the number of expert data grows, MMP and MWAL manage to obtain the same performance as CSI and SCIRL. We also observe that IRL algorithms outperform IL algorithms at the exception of RCAL which has the same performance as CSI (because RCAL is used as a first step of CSI and the regression step is done by directly applying the bijective operator J^*). IRL algorithms use the information of dynamics to improve their performance which is not the case of IL algorithms (at the exception of RCAL). In Fig. 6(b), we observe that Classif, RCAL, SCIRL, Tree, and CSI have low standard deviations which is not the case for MMP, MWAL and Proj. Thus, it seems that the algorithms from the set-policy framework are more stable. Similar results are obtained when the size of the MDP is different. In Fig 7, one can observe the results for $d = 2$ ($N_S = 100$). In Fig. 8, the performance and the standard

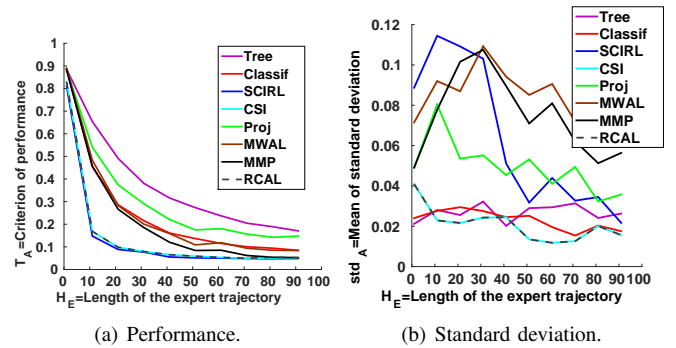


Fig. 7. Garnet Experiment: First setting with $d = 2$ and $L_E = 5$.

deviation are plotted for the second setting (tabular basis and unknown dynamics). Here, the dynamics is only known through the data set D_{NE} where $L_{NE} = 200$ and $H_{NE} = 5$. This is a batch IRL setting where a perfect representation of the reward function is used but a too limited amount of non-expert data is provided, preventing to have a good model of the dynamics. SCIRL manages to obtain good results in that configuration compared to the other IRL algorithms. Indeed, it does not use the non-expert data to estimate its feature expectations but rather uses heuristics [27]. Thus, it appears that when a good representation of the reward function is provided, SCIRL performs well. The performance deteriora-

tion of MMP, MWAL and Proj comes from the use of the approximate MDP solver (LSPI). As they repeatedly use it, the LSPI errors propagate and deteriorate their performance. This is not the case for set-policy-based algorithms that directly learn a reward. However, CSI suffers from a lack of non-expert data to regress correctly the reward. Indeed, RCAL which is used as first step of CSI performs well, thus the bad performance of CSI comes from the regression step. The performance of Tree and Classif stay the same as they do not take into account the dynamics.

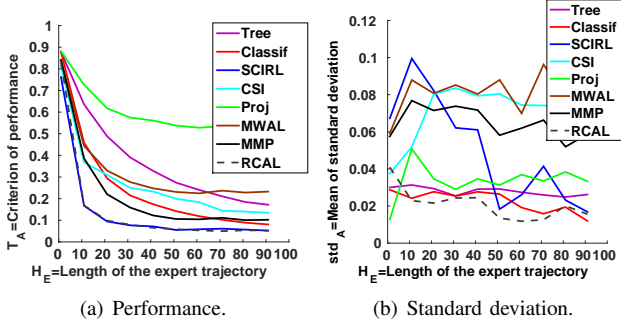


Fig. 8. Garnet Experiment: Second setting with $L_{NE} = 200$, $L_E = 5$, $H_{NE} = 5$ and $d = 2$.

In Fig. 9, the performance and the standard deviation are plotted for the third setting (dynamics known and RBF basis). We observe here, that CSI obtains better results than the other IRL algorithms. The main reason is that CSI is non-parametric. Therefore, it manages to find automatically a good representation of the reward function which is not the case for the other IRL algorithms. Moreover, CSI uses the knowledge of the dynamics P . RCAL has the same performance of CSI as the regression is done by directly applying J^* . Fig. 10

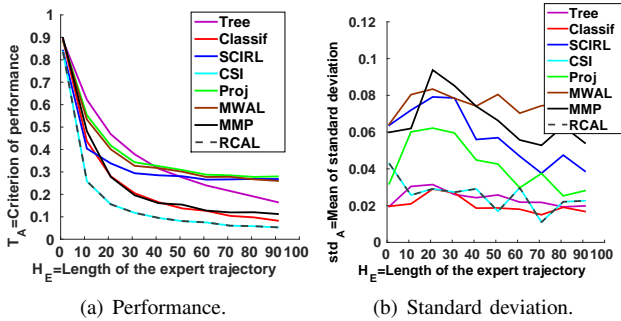


Fig. 9. Garnet Experiment: Third setting with $L_E = 5$ and $d = 2$.

shows the performance and the standard deviation for the fourth setting (unknown dynamics and RBF basis). Here, no IRL algorithm manages to outperform Classif and RCAL. We voluntarily restricted the number of non-expert sampled transitions ($L_{NE} = 200$, $H_{NE} = 5$) to exhibit a case where IRL algorithms have not enough data on the dynamics to perform well. However, CSI performs better than Tree and if more non-expert sampled transitions are added it also beats Classif. Thus, IRL algorithms have a poor performance when provided with few non-expert transitions and when features are not carefully engineered. This shows the importance of the

choice of features and the need to have as much as possible information on the dynamics. Finally, it is important to note

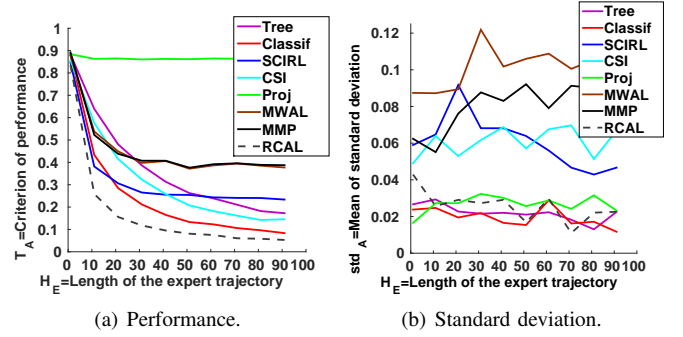


Fig. 10. Garnet Experiment: Fourth setting with $L_{NE} = 200$, $L_E = 5$, $H_{NE} = 5$ and $d = 2$.

that SCIRL and CSI are computationally more efficient than IRL algorithms from the trajectory matching framework in all settings. This is explained mostly because they do not solve MDPs repeatedly.

C. Discussion

In these experiments, it was shown that IRL algorithms perform well when the dynamics and a good representation of the reward are provided. Moreover, CSI and SCIRL, derived from the set-policy framework, outperform IRL algorithms from the trajectory matching family. When the features are well chosen and the dynamics is unknown, SCIRL manages to obtain good results and when the features are poorly chosen and the dynamics is known, CSI performs well. When the features are not well chosen and the dynamics is only known through few non-expert sampled transitions, IRL algorithms do not reach better results than basic IL algorithms. However, if we add more non-expert sampled transitions, CSI manages, as in Fig. 9, to perform better than IL algorithms. These non-expert sampled transitions could be easily obtained, for instance, via an online RL algorithm trying to optimize a first reward obtained by CSI. Then, the estimation of the reward could be improved by these non-expert sampled transitions collected by the online algorithm and so on. Finally, RCAL shows good performance in every setting because it combines the advantages of MCC algorithms (it can be non-parametric and does not need a good parameterization of the reward) and those of IRL (it uses the knowledge of the dynamics).

VI. CONCLUSION AND PERSPECTIVES

In this paper, we have presented an original paradigm, called the set-policy framework, that establishes a formal link between score-based IL methods and IRL methods. This is done via the inverse optimal Bellman operator J^* which maps a score-function Q to a reward R such that $Q = Q_R^*$. We also defined the notion of safe-IL solutions and safe-IRL solutions where safe means that they only search for optimal actions as shown by the expert. In addition, two recent IRL algorithms, namely SCIRL and CSI, were derived from this framework. They use the link between score-functions

and optimal quality functions. As a direct consequence, those algorithms have the particularity to directly compute a reward function without solving repeatedly MDPs, contrary to most algorithms in the literature. This framework also results in non-parametric algorithms such as CSI which consists in a first step of classification followed by a regression step.

The experiments showed that IRL algorithms provide better results than IL algorithms (at the exception of RCAL) when a good representation of the reward and the dynamics P are given. Moreover, in the batch IRL setting (the dynamics is unknown), SCIRL manages to obtain good results if a good reward representation is provided. If the features of the reward are not well chosen, the parametric IRL algorithms do not provide better results than IL algorithms. This shows that the features choice is important in the batch IRL framework. However, if sufficient information on the dynamics is provided through non-expert data, CSI manages to obtain better results than classification without any feature choice as it is non-parametric. Thus, depending on the problem configuration, one can choose CSI in lieu of SCIRL and vice versa.

Finally, the set-policy framework may be an interesting paradigm to develop new IL algorithms and analyze them. Indeed, we provide a general method to derive IL algorithms that take into account the dynamics of the MDP. It consists in imposing a constraint on the set of rewards associated to the score functions we are looking for. The constraint choice can depend on some prior about the expert reward function such as a linear parameterization of the reward, a Lipschitz reward, a sparse reward, *etc.* An application of this general method is RCAL where sparsity of the reward is chosen as a constraint. RCAL shows promising results on each of the proposed settings as it combines the best of IL and IRL.

A. Proof of theorem 2

Proof. Let $R \in \mathbb{R}^{S \times A}$. Thanks to Eq. (9) and the uniqueness of the fixed point of T_R^* :

$$J^*Q = R \Leftrightarrow Q = T_R^*Q \Leftrightarrow Q^* = Q.$$

This means that the inverse image of every singleton $R \in \mathbb{R}^{S \times A}$ under J^* exists, is unique and equal to Q_R^* . In addition, thanks to Eq. (10) and the uniqueness of the fixed point of T_R^π :

$$J^\pi Q = R \Leftrightarrow Q = T_R^\pi Q \Leftrightarrow Q_R^\pi = Q.$$

This means that the inverse image of every singleton $R \in \mathbb{R}^{S \times A}$ under J^π exists, is unique and equal to Q_R^π . \square

B. Proof of theorem 3

Proof. First, let $\bar{\pi}_1$ and $\bar{\pi}_2$ be two different set-policies and let show that $H_{\bar{\pi}_1} \cap H_{\bar{\pi}_2} = \emptyset$. We recall that:

$$\forall \bar{\pi} \in \bar{\Pi}, H_{\bar{\pi}} = \{Q \in \mathbb{R}^{S \times A}, \forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) = \bar{\pi}(s)\}.$$

Let us suppose that there is an element $Q \in H_{\bar{\pi}_1} \cap H_{\bar{\pi}_2}$. This means that: $\forall s \in S, \operatorname{argmax}_{a \in A} Q(s, a) = \bar{\pi}_1(s) = \bar{\pi}_2(s)$. So $\bar{\pi}_1 = \bar{\pi}_2$ which is not the case. Thus, $H_{\bar{\pi}_1} \cap H_{\bar{\pi}_2} = \emptyset$. Then, let $\bar{\pi} \in \bar{\Pi}$ and let show that $H_{\bar{\pi}} \neq \emptyset$. Let $Q_{\bar{\pi}} \in \mathbb{R}^{S \times A}$ be the following function $\forall (s, a) \in S \times A, Q_{\bar{\pi}}(s, a) = \mathbf{1}_{a \in \bar{\pi}(s)}$.

$Q_{\bar{\pi}}$ is clearly in $H_{\bar{\pi}}$. So, $H_{\bar{\pi}} \neq \emptyset$. Moreover for each $x \in \mathbb{R}$, $xQ_{\bar{\pi}}$ is obviously in $H_{\bar{\pi}}$. So the cardinal of $H_{\bar{\pi}}$ is infinite. Finally, the last step to show that $\{H_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition of $\mathbb{R}^{S \times A}$ is: $\bigcup_{\bar{\pi} \in \bar{\Pi}} H_{\bar{\pi}} = \mathbb{R}^{S \times A}$. Let $Q \in \mathbb{R}^{S \times A}$. We define the following set-policy: $\forall s \in S, \bar{\pi}_Q(s) = \operatorname{argmax}_{a \in A} Q(s, a)$. Thus, $Q \in H_{\bar{\pi}_Q}$. So, $\bigcup_{\bar{\pi} \in \bar{\Pi}} H_{\bar{\pi}} = \mathbb{R}^{S \times A}$. In conclusion, $\{H_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition because $\bar{\Pi}$ is a finite set. \square

C. Notations

$(\mathbb{R}, |\cdot|)$ is the real space associated to its canonical norm. Let X and Y be two non empty sets, X^Y is the set of functions from Y to X . For the remaining of this section, X and Y are finite and $\operatorname{Card}(X)$ is the cardinal of X : $X = (x_i)_{1 \leq i \leq \operatorname{Card}(X)}$. We note $\mathcal{P}(X)$ the powerset of X and Δ_X the set of distributions over X . Therefore, Δ_X^Y is the set of functions from Y to Δ_X . Let $\xi \in \Delta_X^Y$ and $y \in Y$, $\xi(y) \in \Delta_X$ is also noted $\xi(\cdot|y)$ and $\forall x \in X, [\xi(y)](x) = \xi(x|y)$. Let $\alpha \in \mathbb{R}^X$ or $\alpha \in \Delta_X$, $\operatorname{Supp}(\alpha) \in \mathcal{P}(X)$ is the support of α : $\operatorname{Supp}(\alpha) = \{x \in X, \alpha(x) > 0\}$. Let $\nu \in \Delta_X$ and $1 \leq p \leq +\infty$, then we can define the \mathbf{L}_p -norm of α noted $\|\alpha\|_p$ and the $\mathbf{L}_{p,\nu}$ -norm of α noted $\|\alpha\|_{p,\nu}$ such that: $\|\alpha\|_p = (\frac{1}{\operatorname{Card}(X)} \sum_{x \in X} |\alpha(x)|^p)^{\frac{1}{p}}$ and $\|\alpha\|_{p,\nu} = (\sum_{x \in X} \nu(x) |\alpha(x)|^p)^{\frac{1}{p}}$. Let $x \in X$, the notation $x \sim \nu$ means that x is a realization of a random variable which is sampled according to ν and $\mathbb{E}_\nu[\alpha] = \sum_{x \in X} \nu(x) \alpha(x)$ is the expectation of α under the distribution ν . Moreover, we note $\delta_X \in \mathbb{R}^{X \times X}$ the function such that for each $(x, x') \in X \times X$, $\delta_X(x, x') = 1$ if $x = x'$ and $\delta_X(x, x') = 0$ if $x \neq x'$.

ACKNOWLEDGMENT

This work has received funding from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement number 270780 (ILHAIRE).

REFERENCES

- [1] D. Pomerleau, "Alvin: An autonomous land vehicle in a neural network," DTIC Document, Tech. Rep., 1989.
- [2] C. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proc. of ICML*, 1997.
- [3] S. Schaal, "Learning from demonstration," in *Proc. of NIPS*, 1997, pp. 1040–1046.
- [4] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [5] O. Pietquin, "Inverse Reinforcement Learning for Interactive Systems," in *Proc. of MLIS 2013*, 2013.
- [6] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.
- [7] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA, 1995, vol. 1, no. 2.
- [8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998.
- [9] S. Ross and J. Bagnell, "Efficient reductions for imitation learning," in *Proc. of AISTATS*, 2010.
- [10] U. Syed and R. Schapire, "A reduction from apprenticeship learning to classification," in *Proc. of NIPS*, 2010.
- [11] S. Russell, "Learning agents for uncertain environments," in *Proc. of COLT*, 1998.
- [12] A. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning," in *Proc. of ICML*, 2000.
- [13] N. Ratliff, J. Bagnell, and S. Srinivasa, "Imitation learning for locomotion and manipulation," in *Proc. of IEEE-RAS International Conference on Humanoid Robots*, 2007.

- [14] S. Natarajan, S. Joshi, P. Tadepalli, K. Kersting, and J. Shavlik, "Imitation learning in relational domains: A functional-gradient boosting approach," in *Proc. of AAAI*, 2011, pp. 1414–1420.
- [15] B. Piot, M. Geist, and O. Pietquin, "Learning from demonstrations: is it worth estimating a reward function?" in *Proc. of ECML*, 2013.
- [16] —, "Boosted and reward-regularized classification for apprenticeship learning," in *Proc. of AAMAS*, 2014.
- [17] F. Melo and M. Lopes, "Learning from demonstration using MDP induced metrics," in *Proc. of ECML*, 2010.
- [18] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite Markov decision processes," in *Proc. of UAI*, 2004, pp. 162–169.
- [19] S. Ross, G. Gordon, and J. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. of AISTATS*, 2011.
- [20] K. Judah, A. Fern, and T. Dietterich, "Active imitation learning via reduction to iid active learning," in *Proc. of UAI*, 2012.
- [21] T. Munzer, B. Piot, M. Geist, O. Pietquin, and M. Lopes, "Inverse reinforcement learning in relational domains," in *Proc. of IJCAI*, 2015.
- [22] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of ICML*, 2004.
- [23] U. Syed and R. Schapire, "A game-theoretic approach to apprenticeship learning," in *Proc. of NIPS*, 2008.
- [24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008, pp. 1433–1438.
- [25] G. Neu and C. Szepesvári, "Training parsers by inverse reinforcement learning," *Machine learning*, vol. 77, no. 2, 2009.
- [26] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proc. of AISTATS*, 2011.
- [27] E. Klein, M. Geist, B. Piot, and O. Pietquin, "Inverse reinforcement learning through structured classification," in *Proc. of NIPS*, 2012.
- [28] E. Klein, B. Piot, M. Geist, and O. Pietquin, "A cascaded supervised learning approach to inverse reinforcement learning," in *Proc. of ECML*. Springer, 2013.
- [29] T. Archibald, K. McKinnon, and L. Thomas, "On the generation of Markov decision processes," *Journal of the Operational Research Society*, 1995.
- [30] A. Farahmand, R. Munos, and C. Szepesvári, "Error propagation for approximate policy and value iteration," *Proc. of NIPS*, 2010.
- [31] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Advances in Computational Mathematics*, vol. 13, no. 1, pp. 1–50, 2000.
- [32] F. Melo, M. Lopes, and R. Ferreira, "Analysis of inverse reinforcement learning with perturbed demonstrations," in *Proc of ECAI*, 2010.
- [33] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proc. of ICML*, 2005.
- [34] L. Breiman, *Classification and regression trees*. CRC press, 1993.
- [35] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, 2006.
- [36] E. Klein, M. Geist, and O. Pietquin, "Batch, off-policy and model-free apprenticeship learning," in *Recent Advances in Reinforcement Learning*. Springer, 2012, pp. 285–296.
- [37] V. Vapnik, *Statistical learning theory*. Wiley, 1998.
- [38] N. Ratliff, J. Bagnell, and M. Zinkevich, "Maximum margin planning," in *Proc. of ICML*, 2006.
- [39] M. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, 2003.
- [40] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," in *Journal of Machine Learning Research*, 2005.



Learning and Interactive Systems group) part of the lab UMI 2958 (GeorgiaTech - CNRS). He defended the 14th of November 2014. From October 2014 to September 2015, he was ATER (postdoctorate) at University Lille 3 and affiliated to the CRISTAL (UMR CNRS 9189) lab's Sequential Learning (Sequel) Team (also INRIA team-project). Since September 2015, he is an assistant professor at University Lille 1 where he teaches computer science and still a member of Sequel.



Matthieu Geist obtained an Electrical Engineering degree and an Msc degree in Mathematics from Supélec (France), both in September 2006, as well as a PhD degree in Mathematics from the University Paul Verlaine of Metz (France) in November 2009. From January 2007 to January 2010, he was a member of the Measure and Control lab (MC cluster) of ArcelorMittal Research and a member of the CORIDA project-team of INRIA. In February 2010, he joined the IMS-MaLIS research group of Supélec (now called CentraleSupélec) as an assistant professor. Since late 2013, he is also an associate member of the UMI 2958 (Georgia Tech - CNRS). His research interests include statistical machine learning (especially reinforcement learning), as well as applications to man-machine interactions. He authored or co-authored more than 60 international publications in these fields.



Olivier Pietquin (M'01 - SM'11) obtained an Electrical Engineering degree from the Faculty of Engineering, Mons (Belgium) in 1999 and a PhD degree in 2004. In 2001, he has been a visiting researcher at the Speech and Hearing lab of the University of Sheffield (UK). Between 2004 and 2005, he was a Marie-Curie Fellow at the Philips Research lab in Aachen (Germany). From 2005 to 2013 he was professor at Supélec (France), and headed several research groups among which the Machine Learning and Interactive Systems group (MaLIS). From 2007 to 2010, he was also a member of the IADI INSERM lab. He was a full member of the GeorgiaTech - CNRS joint lab from 2010 to 2013 and coordinated the computer science department of this international lab. He is now a Full Professor at University Lille 1 and affiliated to the CRISTAL lab's Sequential Learning (Sequel) Team. In 2014, he has been appointed at the *Institut Universitaire de France*. Olivier Pietquin sat on the IEEE Speech and Language Technical Committee from 2009 to 2012. His research interests include machine learning, speech and signal processing and applications to spoken dialog systems. He authored or co-authored over 130 publications in these domains.