A unified framework for imitation-like behaviors

Francisco S. Melo and **Manuel Lopes** and **José Santos-Victor** and **Maria Isabel Ribeiro**[1]

Abstract. In this paper, we combine the formal methods from reinforcement learning with the paradigm of imitation learning. The extension of the reinforcement learning framework to integrate the information provided by an expert (demonstrator) has the important advantage of allowing a clear decrease of the time necessary to learn certain robotic tasks. Hence, learning by imitation can be interpreted as a mechanism for fast skill transfer. Another contribution of this paper consists in showing that our formalism is able to model different types of imitation-learning that are described in the biological literature. It thus unifies in the same abstract model what used to be addressed as separate behavioral patterns. We illustrate the application of these methods in simulation and with a real robot.

1 INTRODUCTION

In the early days of behavioral sciences, several processes used by animals to acquire new skills were often dismissed as "mere imitation". As the knowledge of animal behavior, psychology and neurophysiology evolved, imitation has been promoted and is now considered a sophisticated cognitive capability that few species are capable of [1]. This change in the interpretation was accompanied by the discovery of several phenomena resulting in imitation-like behavior, *i.e.*, in a repetition of an observed pattern of behavior.

In social learning, a learner uses information provided by an *expert* to improve its own learning. For example, if the learner is able to observe the actions taken by a second subject, it can bias its exploration of the environment, improve its model of the world or even mimic parts of the other agent's behavior. This process, generally dubbed as *imitation*, makes cultural transfer of knowledge fast and reliable—acquired knowledge enables fast learning. Cultural spreading becomes thus possible by a *Lamarckian* principle, where animals learn how to act by imitating others and having the same manneriisms as their peers. Through imitation, new discoveries are learnt by each individual very efficiently, simply by observation and behavior matching.

"Real" imitation occurs when a new action is added to the agent's repertoire after having seen a demonstration. It is not enough to repeat an action after having seen it. In fact, this phenomenon can often be explained by reinforcement learning (or learning by trial-and-error). Although some social skill is usually developed when learning by trial-and-error, there is no real imitation (where *new* skills are acquired by simple observation). The concept of *imitation* is far from clear and led biologists to define several mechanisms to explain different types of *imitation-like* behaviors.

In this paper, we analyze several such imitation-like behaviors. We show how each can be modeled using a common formalism. This formalism borrows the fundamental concepts and methods from the reinforcement learning framework [2]. By considering different ways by which an expert can provide information to the learner, we feature different types of learning from observation and formalize each of the aforementioned behaviors in a reinforcement learning (RL) context.

We recall that RL addresses the problem of a decision-maker faced with a sequential decision problem and using evaluative feedback as a performance measure. The evaluative feedback provided to the decision-maker consists of a *reinforcement signal* that quantitatively evaluates the immediate performance of the decision-maker. To optimally complete the assigned task, the decision-maker must *learn* by *trial-and-error*: only sufficient exploration of its environment and actions ensure that the task is properly learnt. Therefore, in the standard RL formalism, the reinforcement signal is a fundamental element that completely describes the task to be learnt. If the agent knows how the reinforcement is assigned, it should be able to learn the task by trial-and-error (given enough time) and the information from an expert can, at most, speed up the learning process.

In real imitation as considered above, the learner should be able to acquire a new skill/learn a new task from the observations. However, and unlike the situation described in the previous paragraph, it generally should not be able to do this *without* the information provided by the expert. Considering everything stated so far, we could argue that, from a RL perspective, this corresponds to the *learning of the reinforcement function*.

Imitation has been proposed as a method to program the complex robotic systems existing today [3, 4, 5]. Programming highly-complex robots is a hard task *per se*; if a robot is capable of learning by observation and imitation, the task of programming it would be greatly simplified. To the extent of our knowledge, no systematic computational model has been proposed to formally describe imitation-like behaviors. The formalism proposed in this paper aims at fulfilling such gap. So far, the mainstream of the research in imitation aimed at individually clarifying/modeling several fundamental mechanisms individually: body correspondence [6, 7], imitation metrics [8], view-point correspondence [9] and task representation [10].

In this paper, we propose a formalism to address learning from observations. In this formalism, several types of information provided by an expert are integrated in a RL framework in different ways. We consider different assumptions on the information provided and on the way this information is integrated in the learning process, and show that this results in different imitation-like behaviors. It is our belief that the formal approach in this paper contributes to disambiguate several important concepts and clarify several issues arising in the literature on learning by imitation.

The paper is organized as follows. Section 2 reviews the main concepts in imitation learning. We describe several models of imitation in biological and artificial systems, as well as some computational problems arising in the context of imitation. Section 3 describes the framework of RL and introduces the fundamental notation. We pro-

[1] Institute for Systems and Robotics,Instituto Superior Técnico, Lisboa, Portugal.*E-mail*: {fmelo, macl, jasv, mir}@isr.ist.utl.pt

ceed in Section 4 by analyzing several methods to use expert information to speed learning. We show these methods to fall within specific classes of imitation-like behavior. To do this we describe how imitation and reinforcement can be combined and describe two simple methods to achieve this. We illustrate some of the methods in the paper in Section 5 and conclude the paper in Section 6.

2 IMITATION LEARNING

Several different mechanisms can result in a imitation-like behavior. One agent may perform an action after having seen it, but the mechanisms leading to it may be very different. Even when asking someone to imitate a hand movement, the results may vary substantially depending on the individual in question [11, 12]. From the study of imitation in animals, several mechanisms were proposed to describe an "imitative behavior", [1, 13, 4, 3]:

1. **Stimulus Enhancement** describes the general tendency to respond more vigorously toward those parts of the environment within which a conspecific is seen to interact. Seeing what are the important parts of the environment and which objects might be useful can speed up learning;
2. **Contextual Learning** describes the situation when an action is not learned, but the perception of a new object property can produce the desire to act upon it. If, for example, an animal sees someone throwing a coconut, it will learn the possibility of throwing it. In the context of our work, contextual imitation would amount to learning to employ an action, already known, in different circumstances.
3. **Response facilitation** is described in [1] as "a kind of social effect that selectively enhances responses: watching a conspecific performing an act, often resulting in a reward, increases the probability of an animal doing the same." Large flocks of birds fly in perfect synchronization. They are not imitating each other, but simply doing the same to protect themselves against predators.
4. **Emulation** can also lead to a behavioral match. Observing an action and the corresponding result might bring a desire to obtain the same goal. Learning that a coconut can be smashed to reach the inside will give the desire to eat the inside and thus producing the same behavior.

Although the mechanisms just described produce imitative behavior, they do not exactly correspond to imitation learning, in the sense that no "new actions" are learned from scratch or added to the existent repertoire. On the other hand, there is a second set of processes leading to imitative behavior where learning of new actions does actually occur. This is called *production learning* [13] and, as it is the most-powerful way of imitation, the "true-imitation" [3].

Byrne distinguishes two cases of production learning, namely **action-level** and **program-level** learning [13]:

- **Action-level learning** is defined as: "The indiscriminate copying of the actions of the teacher without mapping them onto more abstract motor representation." [3]. This is a perfect copy of the motions, if the kinematics of the systems are the same, even the joint level trajectories are the same.
- **Program-level learning** defined for the cases where not only the superficial motion is copied but when a broader description of the sequences, goals and the hierarchical structure of the behavior is inferred by the learner [14].

From the examples above we can see that many situations dubbed as imitation do not involve any actual learning, but only simultaneous/similar action. Response facilitation is just the equal answer that similar agents give when they are at the same state. Emulation and contextual learning can be explained as an improvement of the world model. The result of some action, or its relevant use in a given situation is added to the possible actions. In stimulus enhancement some task learning occurs, but the action is learned by trial-and-error, the demonstration only providing partial knowledge. In imitation, we expect the agent to learn how to complete the task or even the task itself.

2.1 Some implementation issues

Imitation cannot be reduced to supervised learning, where the agent is given the input and correct output. In imitation, the agent is given a set of observations of the environment and corresponding adequate actions. It must then *translate* this information *in terms of its own body*. This is the first difficulty in imitation: the observation is made from a different point-of-view. The different actions performed then must be *recognized* and *mapped to the agent's different capabilities*. Finally, the agent must *infer the important parts of the demonstration*. In imitation, all these problems must be carefully addressed, this being the reason why imitation is considered a complex cognitive task. We now discuss each of these three steps in detail.

Due to the problem of "seeing the world from another's viewpoint", the observed actions must be translated into the referential frame of the learner due to the different perceptual viewpoints, *i.e.*, the learner must perform a "mental rotation" to place the demonstrator's body (*allo-image*) in correspondence with the learner's own body (*ego-image*) [15, 9, 16].

Furthermore, when considering the problem of learning by imitation there is some *correspondence* assumed between the body of the demonstrator and that of the imitator. The *correspondence problem* is precisely defined as the mapping between the actions, states and effects of the demonstrator and those of the imitator. It is particularly relevant if the actions are performed by a specific body and should be replicated by a different body. Even when considering similar bodies, contextual knowledge or training may imply that the demonstrator and the imitator cannot use one same object in the same ways. And if this is not the case, there are always small differences in kinematics, size, dynamics or context that require the correspondence problem to be solved. This problem can be addressed using different methodologies. Examples include algebraic approaches [17], trajectory balance correction [18] and matching the effects of the actions [7].

Finally, it is necessary to *evaluate* the performance of the imitator. In other words, an agent needs a metric that, in a sense, allows it to determine if the imitation was successful or not. And, as expected, different metrics can will lead to different results. These *imitation metrics* evaluate how well the imitator was able to grasp underlying goal of the demonstrated task. These metrics can be selected using an algebraic formulation [8], by optimizing the learnt trajectories [19] or by considering the visual process involved [9].

Figure 1 combines the previous elements in an illustrative architecture that summarizes the relation between these elements of imitation learning [5]. In this paper we do not address the fundamental problems of view-point transformations or recognition. Instead, we assume that the learner receives the processed output of the blocks computing the VPT and performing the recognition, and focus in the problem of learning.

As will soon become apparent, we provide a unified framework to address imitation learning and reinforcement learning. We show that, in this setting, there is an imitation metric that arises naturally from the formulation of the problem of imitation. Furthermore, we describe several situations where such metric does not arise naturally from the problem formulation. We identify in each such situation

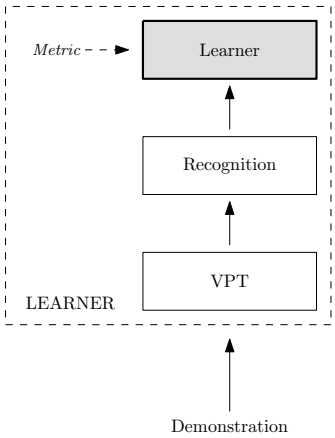

Figure 1. Architecture for the imitator.

a particular instance of *imitation-like behavior*, where the learning agent "appears" to imitate the demonstrator but where no actual imitation takes place.

3 REINFORCEMENT LEARNING

The general purpose of RL is to find a "good" mapping that assigns "perceptions" to "actions". Simply put, this mapping determines how a decision-maker reacts in each situation it encounters, and is commonly known as a *policy*. The use of evaluative feedback, by means of a *reinforcement signal*, allows the decision-maker to gradually grasp the *underlying purpose* of the task it must complete while optimizing the way of completing it.

In this section we describe *Markov decision processes*, the standard framework used to address RL problems. We also review some solution methods that we later employ in the context of imitation.

3.1 Markov decision processes

Let $\{X_t\}$ denote a controlled Markov chain, where the parameter t is the discrete time, and X_t takes values in a finite set $\mathcal{X}$, known as the *state-space*.

The distribution of each r.v. X_{t+1} is conditionally dependent on the past history $\mathcal{F}_t$ of the process according to the probabilities

$$\mathbb{P}\left[X_{t+1} = y \mid \mathcal{F}_t\right] = \mathbb{P}\left[X_{t+1} = y \mid X_t = x, A_t = a\right] = \\ = \mathsf{P}_a(x, y).$$

We note that the transition kernel P depends at each time instant t on a parameter A_t, which takes values in a finite set $\mathcal{A}$. This parameter provides a decision-maker with a mechanism to "control" the trajectories of the chain by influencing the corresponding transition probabilities. We generally refer to the sequence $\{A_t\}$ as the *control process*; we refer to A_t as the *action at time instant* t and to $\mathcal{A}$ as the *action-set*.

Every time a transition from a state $x \in \mathcal{X}$ to a state $y \in \mathcal{X}$ occurs under a particular action $a \in \mathcal{A}$, the decision-maker is granted a numerical *reinforcement* $r(x, a, y)$. This reinforcement provides the evaluative feedback that the decision-maker must use to learn the desired task. The decision-maker must determine the control process $\{A_t\}$ maximizing the *expected total discounted reward*, as given by the functional

$$J(\{A_t\}, x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x\right],$$

where we denoted by R_t the reinforcement received at time t, given by $r(X_t, A_t, X_{t+1})$. Throughout the paper, we admit that the rewards are bounded, *i.e.*, , $|r(x, a, y)| \leq \mathcal{R}$ for some constant $\mathcal{R}$. Also, and to simplify the discussion, we admit r to be constant on the second and third parameters. The parameter $0 < \gamma < 1$ is a discount factor.

A *Markov decision process* (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$ is the state-space, $\mathcal{A}$ is the action-space, P represents the transition probabilities for the controlled chain and r is the reinforcement function.

3.2 Dynamic programming and stochastic approximation

We define a *policy* as being a state-dependent decision-rule, and denote it as a mapping $\delta_t : \mathcal{X} \times \mathcal{A} \longrightarrow [0, 1]$ assigning a probability $\delta_t(x, a)$ to each state action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$. The value $\delta_t(x, a)$ represents the probability of $A_t = a$ when $X_t = x$. A policy δ independent of t is dubbed as *stationary*, and as *deterministic* if for each $x \in \mathcal{X}$ there is an $a \in \mathcal{A}$ such that $\delta_t(x, a) = 1$. In the latter case, we abusively denote by $\delta_t(x)$ the action determined by δ_t when $X_t = x$.

The *value function* associated with a policy δ_t is defined as a mapping $V^{\delta_t} : \mathcal{X} \longrightarrow \mathbb{R}$ defined for each state $x \in \mathcal{X}$ as

$$V^{\delta_t}(x) = J(\{A_t\}, x),$$

where the control process $\{A_t\}$ is generated from $\{X_t\}$ according to δ_t. Given an MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, there is at least one deterministic, stationary policy δ^* such that

$$V^{\delta^*}(x) \geq V^{\delta_t}(x),$$

for any policy δ_t and any state $x \in \mathcal{X}$. This policy can, in turn, be obtained from V^{δ^*} as

$$\delta^*(x) = \arg\max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^{\delta^*}(y)\Big].$$

Any such policy is *optimal* and the corresponding value function V^{δ^*} is simply denoted by V^*. Clearly, V^* verifies the recursive relation

$$V^*(x) = \max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y)\Big],$$

known as the *Bellman optimality equation*. Notice that $V^*(x)$ is the expected total discounted reward along a trajectory of the Markov chain starting at state x obtained by following the optimal policy δ^*.

From V^* we define a function $Q^* : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$ as

$$Q^*(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y).$$

The value $Q^*(x, a)$ is the expected total discounted reward along a trajectory of the chain verifying $X_0 = x$ and $A_0 = a$, obtained by following the optimal policy for $t \geq 1$.

Summarizing, we have the following relations

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a); \tag{1a}$$

$$Q^*(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) \max_{b \in \mathcal{A}} Q^*(y, b); \tag{1b}$$

$$\delta^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a). \tag{1c}$$

Now given any functions $v : \mathcal{X} \longrightarrow \mathbb{R}$ and $q : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$, we consider the operators

$$(\mathbf{T}v)(x) = \max_{a \in \mathcal{A}} \Big[r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x,y) v(y) \Big]$$

and

$$(\mathbf{H}q)(x,a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x,y) \max_{b \in \mathcal{A}} q(y,b).$$

It is straightforward to see that V^* and Q^* are fixed points of the operators $\mathbf{T}$ and $\mathbf{H}$. Each of these operators is a contraction in a corresponding norm and thus a simple fixed-point iteration can be used to determine V^* and Q^*.

The use of either $\mathbf{T}$ or $\mathbf{H}$ to determine V^* or Q^* by fixed-point iteration is a process known as *value iteration*. It is a dynamic programming approach that is often used to determine the function V^* and Q^* from which the optimal policy δ^* can be computed.

When this is not the case, *i.e.*, when P and r are unknown, many methods have been proposed that asymptotically converge to the desired functions [20, 2]. In this paper, we use use one of the most studied methods in the RL literature: the Q-learning algorithm [21]. This method uses sample trajectories of the Markov process, $\{x_t\}$, control process, $\{a_t\}$ and corresponding rewards $\{r_t\}$ to estimate the function Q^*. These estimates are updated according to the Q-learning update

$$\begin{aligned} Q_{t+1}(x_t,a_t) = (1 - \alpha_t(x_t,a_t)) Q_t(x_t,a_t) + \\ + \alpha_t(x_t,a_t) \big[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(x_{t+1},b) \big]. \end{aligned} \tag{2}$$

This algorithm will converge to Q^* w.p.1 as long as $\sum_t \alpha_t(x,a) = \infty$ and $\sum_t \alpha_t^2(x,a) < \infty$ for every $(x,a) \in \mathcal{X} \times \mathcal{A}$. This requires in particular that every state-action pair be infinitely often (there is sufficient exploration of the environment and the agent's actions).

4 LEARNING PARADIGMS USING EXPERT INFORMATION

In the previous sections we described two learning paradigms: learning by imitation and learning by reinforcement. In this section we move towards a combined learning framework, the *learning by observation and reinforcement* (LOR) framework. To this purpose, we consider a learning agent that must learn how to perform a sequential task using some prior knowledge and information provided by an expert.

The formalism considered herein borrows the fundamental ideas from the reinforcement learning framework described in the previous section, thus providing a unified framework to address both classes of learning processes. The fundamental assumptions usually considered in the reinforcement learning framework are:

- The task to be learnt can be described as a mapping from the set of states of the environment to the set of possible actions (a *policy*);
- The environment is *stationary*.

The first assumption simply states that in the same state of the environment the agent should always perform the same action. We remark that this assumption bears yet another important implication. If, as stated, the task to be accomplished can be fully described using a policy, then there is a reward function such that the *policy to be learnt is the optimal policy with respect to this reward function*, in the sense described in Section 3.

The second assumption above simply means that the policy used to fulfill the task should not change with time (the environment always responds to the agent's actions in the same way).

In what follows, we will consider two fundamental situations:

(i) The imitator knows the task to be learnt, but does not know how to perform this task;
(ii) The imitator does not know the task to be learnt.

From everything stated so far, it should be clear that, in terms of our formalism, the situation in (i) simply means that there is a previously defined reward function, known by the agent (since the reward function defines the task). Notice that if the agent is aware of this function, it can learn to perform the task by trial-and-error, given sufficient time. Clearly, the situation in (ii) means that *there is no reward function defined a priori*. This, of course, implies that the agent will not be able to learn any task without any further information.[2]

We analyze how different types of information provided by an expert can be integrated in learning the desired task. As will soon become apparent, models for the imitation-like behaviors described in Section 2 arise naturally in the LOR framework. We also show that, in the more complex scenario of an unknown task, it is possible to provide a natural interpretation for the used algorithm in terms of imitation metrics. The first case, where the agent does know the task, does not correspond to "real" imitation behavior as defined in Section 2. Only in the second situation, where the task is not defined beforehand, can we speak about true imitation. We further comment on this issue at the end of the section.

We consider each of the two situations (i) and (ii) in Subsections 4.1 and 4.2, respectively.

4.1 Known task

We consider that the interaction of the learning agent and the environment can be described as a controlled Markov chain, as in Section 3. This means that, at each time t, the state of the environment will move from a state $X_t = x$ to a state $X_{t+1} = y$ depending on the action A_t of the agent and according to the transition probabilities $\mathsf{P}_a(x,y)$. We suppose that an expert provides the learning agent with some information on how the task can be completed. We refer to such information generally as a *demonstration* and analyze how can this information be used in the learning process. We consider four distinct cases:

(i) The demonstration consists of a sequence of states,

$$\mathcal{H} = \{x_1, \ldots, x_N\},$$

obtained by following the optimal policy;
(ii) The demonstration consists of a sequence of state-actions pairs,

$$\mathcal{H} = \{(x_1,a_1), \ldots, (x_N,a_N)\},$$

"hinting" on which should be the optimal action a_i at each state x_i visited;
(iii) The demonstration consists of a sequence of transition triplets,

$$\mathcal{H} = \{(x_1,a_1,y_1), \ldots, (x_N,a_N,y_N)\},$$

providing the imitator with information on the behavior of the environment;

[2] We could argue that the situation in (ii) means that the agent *does not know* the reward function, but that the latter is defined. We do not adopt such position for the simple reason that, if a reward function *is* defined, the agent can still learn by trial-and-error and, therefore, there is no significative difference from (i).

(iv) The demonstration consists of a sequence of transition-reward tuples,

$$\mathcal{H} = \{(x_1, a_1, r_1, y_1), \ldots, (x_N, a_N, r_N, y_N)\},$$

providing the imitator with information on the behavior of the environment and on how the task should be completed.

First of all, we remark that, since we assume knowledge of r, (iii) and (iv) are redundant. Nevertheless, we will consider how to address the two situations distinctly, noting that in (iv) allows to address situations in which r is unknown.

We must further detail the idea behind each of the previous classes of demonstrations. The first situation, (i), addresses situations in which the learning agent *can not observe/recognize the actions of the demonstrator* but only their effect in the environment. This information will show the agent how the state of the environment should evolve when the optimal policy is implemented. A sequence as described in (ii) illustrates *how the task can be completed*. Each pair (x_i, a_i) is related through some deterministic policy δ that is "close" to optimal. Sequences as those described in (iii) and (iv) illustrate *the dynamics of the environment* in terms of transitions and transitions-rewards, respectively. Unlike the sequences described in (ii), it is not assumed that x_i and a_i in each tuple (x_i, a_i, y_i) or (x_i, a_i, r_i, y_i) are related by some policy.[3]

Another important aspect is that, at this stage, we are not concerned with the particular way by which the sequences $\mathcal{H}$ in (ii) through (iv) are obtained. Consider for example the situation in (ii). It may occur that the demonstrator illustrates how the task is completed by demonstrating the action to be chosen in an arbitrary set of states $\{x_1, \ldots, x_N\}$. Or, it may happen that the sequence of states $\{x_1, \ldots, x_N\}$ is actually a sample path of the process obtained with the control sequence $\{a_1, \ldots, a_{N-1}\}$.

We also remark that, in all 3 cases listed above, we assume that the imitator is able to perceive the information in the sequences $\mathcal{H}$ unambiguously. We could admit *partial observability*, meaning that the imitator was able to observe the states, actions and/or rewards in the sequences $\mathcal{H}$ only up to some degree of accuracy. This would imply that the imitator would have to *estimate* what the actual state, action and/or reward would have been. This, of course, would be the actual case in practical situations. Nevertheless, consideration of partial observability adds no useful insight to our formalization of the imitation problem and significantly complicates the presentation.

The four methods presented below all provide an initial estimate Q_0 for Q^* that integrates the information provided by the demonstration. We will see that this informed initialization brings a significant improvement in the learning performance of the agent.

Method 4.1.1: Sequence of states

Consider a sequence of states

$$\mathcal{H} = \{x_0, \ldots, x_N\},$$

obtained according to the optimal policy. As stated, this first scenario comprises situations where the learning agent is not able to observe/recognize the actions performed by the expert. Nevertheless, the sequence of states $\mathcal{H}$ provides the learning agent with an idea on how the environment evolves "under" the optimal policy.

[3] We make this distinction as each of the sequences described in (i) through (iv) provides the imitator with different information, to be used in different ways. This is not limiting in any way, as discussed below.

Therefore, if the transition model is known, the agent can compute

$$Q_0(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x, y) V^*(y),$$

where V^* is computed as $V^* = (\mathbf{I} - \gamma \mathsf{P}^*)^{-1} r$. The matrix $\mathbf{I}$ denotes the identity and the transition matrix P^* represents the transition model for the optimal policy, estimated from $\mathcal{H}$ as

$$\mathsf{P}^*(x, y) = \frac{N(x, y)}{\sum_{z \in \mathcal{X}} N(x, z)},$$

where $N(x, y)$ denotes the number of times that a transition from x to y occurred in $\mathcal{H}$. This method is similar to that proposed in [22].

Method 4.1.2: Sequence of state-action pairs

Consider a sequence of state-action pairs

$$\mathcal{H} = \{(x_1, a_1), \ldots, (x_N, a_N)\}.$$

Each demonstrated pair (x_i, a_i) provides significant information on the *optimal policy* at x_i. And even if the policy partially defined by $\delta(x_i) = a_i$ is not optimal, it is expectable that it is "close" to optimal. It is therefore reasonable that the imitator *uses δ as an initial policy to perform the task*. And, as it acquires further experience on the task, it should be able to improve from this initial policy, if there is room for such improvement. To incorporate this information in the initial estimate for Q^*, we set $Q_0(x_i, a_i) = 1$ for $i = 1, \ldots, N$ and 0 otherwise.

Method 4.1.3: Sequence of transition triplets

We now consider a sequence of transition triplets

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}.$$

As mentioned above, this sequence provides the imitator with *information on the behavior of the environment*. Clearly this is only useful if the transition probabilities are not known *a priori*. If this is the case, the information provided by the demonstrator can be used to improve the model of the environment by setting

$$\hat{\mathsf{P}}_a(x, y) = \frac{N(x, a, y)}{\sum_{z \in \mathcal{X}} N(x, a, z)},$$

where $N(x, a, y)$ denotes the number of times that the triplet (x, a, y) was observed in $\mathcal{H}$. This estimated transition model $\hat{\mathsf{P}}$ with the function r can be used to perform value iteration and obtain an initial estimate Q_0 for the learning algorithm.

Method 4.1.4: Sequence of transition-reward tuples

Finally, we consider a sequence of transition-reward tuples

$$\mathcal{H} = \{(x_1, a_1, r_1, y_1), \ldots, (x_N, a_N, r_N, y_N)\}.$$

This sequence provides the imitator with information on the behavior of the environment and on *the task*. This means that the tuples in $\mathcal{H}$ can be used to perform N iterations of Q-learning using (2). The resulting Q-function provides the initial estimate Q_0 for the learning algorithm.

4.2 Unknown task

In this subsection, we use the exact same formulation considered in Subsection 4.1 above, but suppose that *no reward mechanism is defined*. This means that the imitator is no longer able to learn the task by trial-and-error if no demonstration is available.

However, if a demonstrator provides the imitator with some information on how the task can be completed, the imitator can *build* its own reward function and use it to learn how to perform the task. We also refer to such information generally as a *demonstration*.

Unlike in the previous situation, we only consider two scenarios: We consider four distinct cases.

(i) The demonstration consists of a sequence of states,

$$\mathcal{H} = \{x_1, \ldots, x_N\}\,.$$

(ii) The demonstration consists of a sequence of transition triplets,

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}\,,$$

providing the imitator with information on the behavior of the environment.

Notice that, since there is no reward function defined, it is not possible to consider the situation where transition-reward tuples are observed. Also, and unlike Subsection 4.1, we now assume that the transition triplets in $\mathcal{H}$ considered in (ii) are obtained *using the policy to be learnt*. Therefore, (ii) includes both (ii) and (iii) from the previous subsection.

Method 4.2.1: Sequence of states

Consider a sequence of states

$$\mathcal{H} = \{x_0, \ldots, x_N\}\,,$$

obtained according to the optimal policy. As in Subsection 4.1, this scenario comprises situations where the learning agent is not able to observe/recognize the actions performed by the expert.

We interpret the sequence of states in $\mathcal{H}$ as providing the learner with information *on the goal* of the task. In particular, we consider that $\mathcal{H}$ represents a *possible trajectory to a goal state*. Therefore, the learner will memorize the last state visited, x_N, as the goal state and build a simple reinforcement function defining the task "reach the goal state as fast as possible". An example of one such reward function is

$$r(x) = \begin{cases} +10 & \text{if } x = x_N; \\ -1 & \text{otherwise.} \end{cases}$$

The agent can now apply any preferred method to determine the optimal policy. For example, it can use value iteration if P is known, or Q-learning otherwise. The learner will thus learn a policy that will partially replicate the demonstration observed.

Method 4.2.2: Sequence of transition triplets

We now consider a sequence of transition triplets

$$\mathcal{H} = \{(x_1, a_1, y_1), \ldots, (x_N, a_N, y_N)\}$$

obtained using the "optimal policy". As in Subsection 4.1, this sequence can be used to improve the model of the environment. This model of the environment can, in turn, be used to determine the reward function that best translates the policy partially defined by $\delta(x_i) = a_i$, $i = 1, \ldots, N$. The approach considered here differs from that used in Method 4.2.1 in that the reward function is no longer built by considering only one final state. Instead, the learning agent will use the *whole demonstration* and apply inverse reinforcement learning to build the reward function [23]. We will show that this procedure is fundamentally different from the previous ones, and corresponds to "real imitation" in the sense of Section 2.

4.3 Classification of the learning paradigms

So far in this section we formalized several different methods by which an agent can use the information provided by an expert in learning how to accomplish a task. However, as discussed in Section 2, there are several learning paradigms that do exhibit imitative behavior but which cannot be truly classified as "imitation". And, as we show in the continuation, most of the methods described above actually fall in one of the following categories:

- Stimulus enhancement;
- Contextual learning;
- Response facilitation;
- Emulation.

We start with the Method 4.1.1. In this method, the learning agent seeks to replicate the *effect* of the actions of the demonstrator. This will actually lead to an initial replication of the demonstrator's policy, but the process by which this behavioral match is attained is *emulation*.

In Method 4.1.2, the imitator uses the demonstration to *bias its learning strategy*. Therefore, this method is actually a *stimulus enhancement* mechanism: the imitator observes some actions that can be useful for the task and uses this information to speed learning.

In Method 4.1.3, the imitator uses the demonstration to *improve its model of the world*. This means that the imitator gains further knowledge on what the consequences of some of its actions may be. We can classify this as a subtle form of *contextual learning*.

A similar thing occurs in Method 4.1.4. In this method, however, the imitator further observes *the rewards* obtained by the imitator. It realizes not only the consequences of some actions but also on *how these actions contribute to complete the task*. This use of the reward information allows us to realize that Method 3 combines contextual learning with *response facilitation*.

Notice that, in all these methods, the agent already knows the task to be learnt. This means that, with enough time, the agent could learn the task without any help from a demonstrator. Furthermore, independently of the policy used in the demonstration, the agent will eventually learn the correct policy, completely disregarding the demonstration if necessary. This means that the demonstration only provides a means for the agent to speed up its own learning process. Therefore, it is not surprising that all these situations do not correspond to "true-imitation" behaviors.

Moving to the the methods in Subsection 4.2, we start by noticing that, in Method 4.2.1 the agent seeks to replicate the final *effect* of the actions of the demonstrator. In fact, in this method, the agent focuses all its learning in *replicating the effect* observed in the demonstration (in terms of final state), displaying a flagrant example of *emulation*.

On the other hand, Method 4.2.2 seeks to *extrapolate the task behind the actions of the demonstrator*. From this information, the agent builds a reward function that will eventually lead to a replication of the demonstrator's policy. However, the actual method for computing this reward function (and, thus, realizing the task to be learnt) provides important insights into the problem of imitation, that we discuss next.

4.4 Inverse reinforcement learning and imitation metrics

As argued in Section 2, "true" imitation will occur if a broad description of the action sequences, goals and hierarchical structure of the desired behavior is inferred by the learner. As we have seen, in the RL formalism, the goals and structure of the desired behavior are "encoded" in the reward function. Therefore, learning the reward function and using it to determine the optimal policy would fit the above description of true imitation.

Notice that we consider Method 4.2.1 to be emulation because two completely different sequences ending in a common final state will lead the learning agent to infer the exact same reward function. This means that, as stated in the previous subsection, this method seeks to replicate the effect of the actions of the demonstrator rather than to extrapolate the task behind the actions of the demonstrator.

On the other hand, Method 4.2.2 does seek to extrapolate this information from the demonstration. To better realize how this method operates, we provide a brief description of its working [23].

Given the model of the environment (namely the transition probabilities in P), the inverse reinforcement learning method used (dubbed *Bayesian inverse reinforcement learning*—BIRL) searches the space of possible reward functions. To this purpose, the method considers a fine discretization of the referred space of reward functions. Then, given any initial reward function, the method evaluates the optimal Q-function, Q^*, for this reward function and evaluates the *likelihood of the demonstrated policy being optimal* given Q^*. This likelihood also takes into consideration a numerical parameter describing the *confidence on the optimality of the demonstrated policy*. The method will thus output the most likely reward given the demonstrated policy (obtained from $\mathcal{H}$) and the confidence parameter.

We emphasize several important aspects of this approach. First of all, this method considers the demonstration *as a whole*, instead of focusing on particular aspects. Therefore, the reward thus determined will more accurately the task "behind" the demonstration. On the other hand, the likelihood function used to compare different reward functions as well as the confidence parameter naturally provide an imitation metric for the problem. The inclusion of the confidence parameter is an important aspect that allows the agent to realize how strict it should follow the provided demonstration. A low confidence parameter will result in a learnt policy significatively more different from the demonstrated policy than a high confidence parameter.

Also notice that considering imitation metrics makes no sense in the other methods. In the methods in Subsection 4.1 the demonstration is only used to speed the learning. The agent is not trying to replicate the demonstration but to optimize its policy with respect to the pre-defined rewards. In Method 4.2.1, on the other hand, the agent is simply trying to reach the final state observed in the demonstration. Once again, is not trying to replicate the demonstration but to optimize the policy leading it to this goal state.

The reward function thus constructed will provide adequate evaluative feedback on the task and the imitator can use this evaluative feedback to optimize its own policy. We emphasize that, without the demonstration, the imitator *has no knowledge on the task*. The reward function built from the demonstration is, therefore, *new knowledge* that describes the task at hand and allows the imitator to learn how to perform it in an optimal fashion.

4.5 Discussion

With the methods above we conclude the presentation of the LOR framework. Within this framework, we model an agent's environment as a controlled Markov chain $\{X_t\}$. The demonstration provided by an expert is, in turn, described as a sequence $\mathcal{H}$ which can take various forms, depending on the information provided. The formalism considered herein borrows fundamental ideas from reinforcement learning and provides a unified framework to address both classes of learning processes.

We notice that the MDP model considered in this paper is the simplest model used in reinforcement learning. We are interested in establishing a unified framework to address both learning by imitation and reinforcement and thus focus on this simpler model for the sake of clarity. In Section 6 we briefly comment on how the fundamental framework considered herein can be extended to accommodate richer RL models (such as POMDPs).

As argued in Section 2, imitation cannot be reduced to supervised learning and, therefore, the framework presented here should not be seen as simple a combination of supervised learning and reinforcement learning.[4] Instead, it should be seen as a formalism to describe learning processes in which imitation and reinforcement learning can be properly modeled.

It is possible to find other works in the literature that combine learning by imitation and reinforcement. In [22], imitation arises *implicitly* in non-interactive multiagent scenarios. In it, a learning agent uses the trajectories observed from other agents to speed the learning of its individual task (which is generally independent of that of the others). In yet another example, [25], a learning method is proposed that learns a reinforcement function and dynamic model from the demonstration of an expert (human executer). This is then combined with a model-free, task-level direct learner to compensate for modeling errors.

Our work is fundamentally different from those considered above in that our aim is to understand how can the problem of imitation be modeled and how can imitative-like behaviors be distinguished with a formal perspective. Nevertheless, several methods described in our paper can be seen as simplified versions of the methods described in those papers.

Also, as argued in Section 2, we considered that in order for the learning mechanism to be properly classified as *imitation*, it should be able to *realize* the task from the demonstration. However, it should be flexible enough to feature two possible behaviors: to *replicate* the exact behavior of the demonstrator or, instead, to *perceive* the purpose of the task and, eventually, optimize beyond whatever it observed. As discussed in the previous subsection, the use of Method 4.2.2 verifies all these requisites. On the other hand, each of the remaining methods exhibits one of the above features, not all. This is the reason why we classified them as imitation-like.

Finally, we remark that the classical inverse reinforcement learning algorithms [26, 27] also determine a reward function given a policy. The difference from these methods to the one used here is that BIRL allows the policy to be only *partially specified* and *suboptimal*. This is an important advantage in the problems considered herein.

5 EXPERIMENTS

We conducted several simple experiments to evaluate the performance of proposed methods against that of simple trial-and-error. We evaluated each of the methods described in Section 4.

[4] Such approach is adopted, for example, in [24], where a supervisor is combined with an actor-critic learning architecture.

The task considered is a simple recycling game, where a robot must separate the objects in front of him according to its shape (Fig. 2). In front of the robot are two slots (Left and Right) where 3 types of objects can be placed: Large Ball, Small Ball and Box. The boxes should be dropped in the corresponding container and the small balls should be kicked out of the table. The large balls should be touched upon. Every time a large ball is touched, all objects are removed from the table.

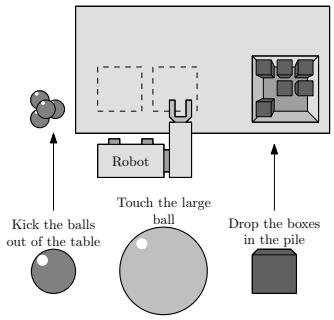

Figure 2. Simple recycling game.

The robot has, therefore, 6 possible actions: Touch Left (TL), Touch Right (TR), Kick Left (KL), Kick Right (KR), Grasp Left (GL) and Grasp Right (GR). We notice that, if the robot kicks a ball on the right while an object is lying on the left, the ball will remain in the same spot. The robot receives a reward of $+10$ every time the table is empty and -1 every other time.

The correct policy for this game is to touch the large ball, if there is any, or get rid of the object on the left and then of the object on the right (there are some situations where the order is not important). Every time the table is emptied, the game is restarted.

We tested the performance of the 4 Methods in Subsection 4.1 when the optimal policy is demonstrated and a suboptimal policy is demonstrated. We compared the performance of an agent using the information provided by the demonstration with that of an agent that has no previous information on the task. In all situations we allowed both agents to learn for 200 time steps using an ε-greedy policy with decaying ε.

Table 1 provides the percentage of time (out of the 200 time steps) that the agents are able to reach the goal state (empty table). For the sake of comparison, we also provide the performance of a "pure" reinforcement learner.

Table 1. Results obtained with Methods 4.1.1 through 4.1.4 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**
Pure RL	34.6 %	32.4 %
Method 4.1.1	41.5 %	40.5 %
Method 4.1.2	41.5 %	37.5 %
Method 4.1.3	41.5 %	39.0 %
Method 4.1.4	42.0 %	41.0 %

From Table 1 it is evident that the performance of the learning algorithm is improved when considering a demonstration, since the agents were able to reach the goal state (and thus complete the task) more often. To have a clearer understanding of how this translates in terms of the learning process, we present in Figures 3 through 6 the total reward obtained during learning.

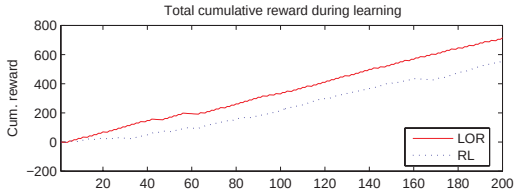

Figure 3. Total reward obtained with Method 4.1.1 over the time-frame of 200 steps when the demonstrator follows an optimal policy.

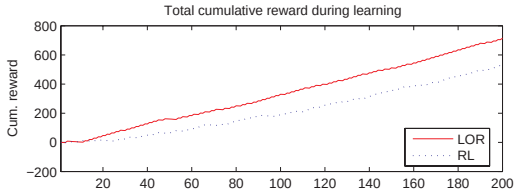

Figure 4. Total reward obtained with Method 4.1.2 over the time-frame of 200 steps when the demonstrator follows an optimal policy.

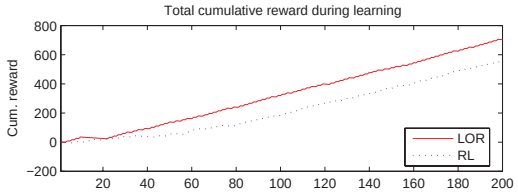

Figure 5. Total reward obtained with Method 4.1.3 over the time-frame of 200 steps when the demonstrator follows an optimal policy.

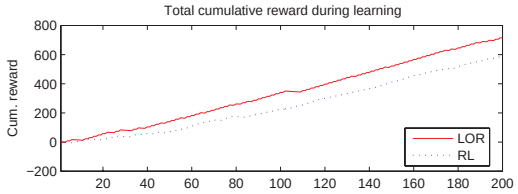

Figure 6. Total reward obtained with Method 4.1.4 over the time-frame of 200 steps when the demonstrator follows an optimal policy.

In the plots, the slope of the performance curve indicates how good is the learnt policy. It is clear that, in all methods, the provided information gives the learning agent a significative advantage: in the beginning of the learning process, the "greedy" action for the agents that were provided a demonstration is much more informed than that of the pure RL learner. This means that the demonstration provides the learner with a *knowledge boost* by improving the estimative of the optimal Q-function and thus speeding up the learning.

Notice that, in all these methods, the demonstration provides only informed initial estimates for Q^*, thus improving the initial performance of the agent. However, since this initial estimate is then properly adjusted by the learning algorithm, the sub-optimality of the demonstrated policy does not affect the performance of the learner.

In a second set of experiments we tested Method 4.2.1 from Subsection 4.2. To evaluate the performance of the method, we explicitly observed the learnt policy when the demonstrated policy is optimal and when it is not. The results are summarized in Table 2. We denoted by 0 the empty slot, by B the large ball, by c the cube and by b the small ball.

Notice that both learnt strategies are optimal. This is due to the fact that, in considering the same final state, the reward function obtained by Method 4.2.1 is the same independently of the actual policy used

Table 2. Learnt policies with Method 4.2.1 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**
$(0,0)$	TL	TL
$(0,B)$	TR	TR
$(0,c)$	GR	GR
$(0,b)$	KR	KR
$(B,0)$	TL	TL
(B,B)	TR	TL
(B,c)	TL	TL
(B,b)	TL	TL
$(c,0)$	GL	GL
(c,B)	TR	TR
(c,c)	GR	GR
(c,b)	GL	GL
$(b,0)$	KL	KL
(b,B)	TR	TR
(b,c)	GR	KL
(b,b)	KL	KL

to demonstrate. And, in this particular case, it matches exactly the reward function considered in the previous examples, thus giving rise to the same policy.

Finally, we tested Method 4.2.2 from Subsection 4.2. As in the previous experiment, we evaluate the performance of the method by explicitly observing the learnt policy when the demonstrated policy is optimal and when it is not. The results are summarized in Table 3. In the third column we also present the results obtained with Method 4.2.2 using an optimal policy, but where the model is also estimated from the demonstration. The table elements in bold denote "suboptimal" actions.

Table 3. Learnt policies with Method 4.2.2 using optimal and suboptimal demonstrations.

	Optimal	**Suboptimal**	**No Model**
$(0,0)$	TL	TL	TL
$(0,B)$	TR	**TL**	**TL**
$(0,c)$	GR	**TR**	GR
$(0,b)$	KR	KR	KR
$(B,0)$	TL	**KL**	TL
(B,B)	TR	**GL**	TL
(B,c)	TL	**TR**	TL
(B,b)	TL	**TR**	**TR**
$(c,0)$	GL	**TL**	GL
(c,B)	TR	**GL**	**TL**
(c,c)	GR	GR	**TR**
(c,b)	GL	**KR**	**KL**
$(b,0)$	KL	KL	**TR**
(b,B)	TR	**KL**	**KL**
(b,c)	GR	**TL**	**TL**
(b,b)	KL	**TR**	**TR**

We emphasize the policy obtained with Method 4.2.2 when the demonstrated policy is suboptimal (and the agent has little confidence on the observed policy). Recall that this method determines a likely reward function for which demonstrated policy, we expect the performance of this method to be affected by the sub-optimality of the demonstrated policy. Notice that the policy learnt from a suboptimal demonstration is even worse than that learnt in the absence of a model with an optimal demonstration (third column of Table 3).

To conclude this section, we present the images obtained by experimenting Method 4.2.1 in a real robot. The robot is capable of recognizing the actions Grasp, Touch and Kick as well as the objects on the table (to details refer to [7]). Figure 7 presents the robot following the task it learned after having observed it.

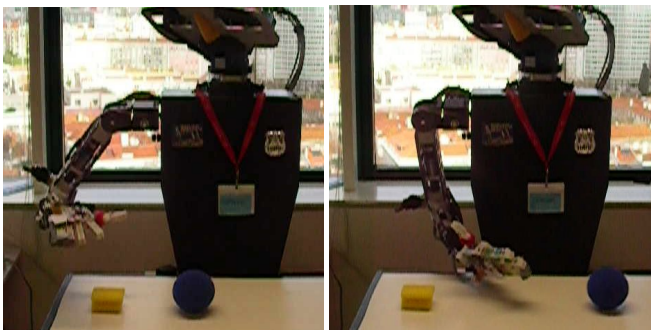

Figure 7. Robot following the learned task.

6 CONCLUSIONS

In this paper, we proposed an unified formalism to address imitation learning and RL problems. Using this formalism, we analyzed several imitation-like learning mechanisms, such as stimulus enhancement, response facilitation, contextual learning and emulation. These mechanisms can lead to imitative behavior without being imitation in the stricter sense of the concept. In this formalism, which we refer as the *learning by observation and reward* (LOR), these behaviors can be summarized as:

- *Stimulus enhancement*: biases exploration using the observed policy;
- *Contextual learning*: uses the observed transitions to improve the model of the world;
- *Response facilitation*: uses the observed transitions/rewards to improve the model of the world and accelerate learning;
- *Emulation*: uses the observed sequence of states, to either replicate the dynamics of the underlying Markov chain or final state.

One of the major contributions of the paper was to unify all of these mechanisms using a common formalism. We showed that this modelation is possible and the resulting behavior of the learner matches the descriptions of the corresponding behaviors in animals. We also discussed that, when learning a task from an expert, there are many sources of information and each of them can be exploited individually or in combination.

The results presented clearly established one of the known advantages of imitation learning: the imitation learner acquired the optimal policy for the problem faster than a learner following a standard trial-and-error learning strategy. We emphasize that, in the discussed cases of imitation-like behavior, the agent would still be able to learn the task on its own—the learner did not *infer* the solution from the demonstration. Instead, the demonstration provided *hints* on how to solve the task that the learner used to learn the task more efficiently.

It is interesting to note that, as these mechanisms do not rely completely on the details of the demonstration, they can also learn the optimal policy even when the demonstration is sub-optimal. The learner can thus look at someone performing a task and then understand the goal of the task and outperform the teacher.

We also emphasize the difference between imitation-like behaviors and "pure" imitation methods. In a pure imitation system, the found solution should not exist in the learner repertoire; or it should not be possible to know the task if it were not for the demonstration. In our formalism this translates into the fact that, without the demonstration, the agent does not know the task (there is no pre-defined reward mechanism). In imitation-like methods this reward function previously exists and the learner can always learn the task on its own.

The demonstrations used throughout the paper do not illustrate the full potential of the different methods, mainly due to the great simplicity of the task considered—the state and action spaces are small and the task is easily defined by a very simple reward function.

In our proposed LOR framework it is not easy to distinguish between action-level and program-level learning, since the important steps of the demonstration are abstract concepts that can be interpreted and implemented in different ways. We intend to address this problem with further detail by defining an hierarchical learner where we can define actions at several "resolutions". We also intend to study the effects of partial observability of state and action in learning by imitation.

Acknowledgements

This work was partially supported by Programa Operacional Sociedade do Conhecimento (POS_C) that includes FEDER funds, and by the EU-Project Robotcub. The first author acknowledges the PhD grant SFRH/BD/3074/2000.

References

[1] Richard Byrne. *The Thinking Ape Evolutionary Origins of Intelligence*. Oxford University Press, 1995.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[3] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[4] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Phil. Trans. of the Royal Society of London: Series B, Biological Sciences*, 358(1431):537–547, 2003.

[5] Manuel Lopes and José Santos-Victor. A developmental roadmap for task learning by imitation in robots. *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, 37(2), 2007.

[6] Hideki Kozima, Cocoro Nakagawa, and Hiroyuki Yano. Emergence of imitation mediated by objects. In *2nd Int. Workshop on Epigenetic Robotics*, 2002.

[7] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning affordances objects: From sensory motor maps to imitation. Technical report, Instituto de Sistemas e Robótica, Lisbon, Portugal, Feb 2007.

[8] Chrystioher L. Nehaniv and Kerstin Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics and Systems*, 32:11–51, 2001.

[9] Manuel Lopes and José Santos-Victor. Visual transformations in gesture imitation: What you see is what you do. In *IEEE Int. Conf. Robotics and Automation*, 2003.

[10] R. Zöllner and R. Dillmann. Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2003.

[11] H. Bekkering, A. Wohlschläger, and M. Gattis. Imitation of gestures in children is goal-directed. *Quarterly J. Experimental Psychology*, 53A:153–164, 2000.

[12] György Gergely, Harold Bekkering, and Ildikó Király. Rational imitation in preverbal infants. *Nature*, 415:755, 2002.

[13] Richard W. Byrne. Imitation of novel complex actions: What does the evidence from animals mean? *Advances in the Study of Bahaviour*, 31:77–105, 2002.

[14] R. W. Byrne and A.E. Russon. Learning by imitation: a hierarchical approach. *Behav Brain Sci*, pages 667–84, 1998.

[15] J.S. Bruner. Nature and use of immaturity. *American Psychologist*, 27:687–708, 1972.

[16] Minoru Asada, Yuichiro Yoshikawa, and Koh Hosoda. Learning by observation without three-dimensional reconstruction. In *Intelligent Autonomous Systems (IAS-6)*, 2000.

[17] C. Nehaniv and K. Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In *European Workshop on Learning Robots*, 1998.

[18] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances,. In *ICRA*, Taipei, Taiwan, 2003.

[19] Aude Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47:2-3, 2004.

[20] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[21] Christopher J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, University of Cambridge, May 1989.

[22] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *J. Artificial Intelligence Research*, 19:569–629, 2003.

[23] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. Proc. 20th Int. Joint Conf. Artificial Intelligence, 2007. (to appear).

[24] M.T. Rosenstein and A.G. Barto. Supervised actor-critic reinforcement learning. In *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. John Wiley & Sons, Inc., 2004.

[25] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *14th International Conference on Machine Learning*, pages 12–20. Morgan Kaufmann, 1997.

[26] Andrew Y. Ng and Stuart J. Russel. Algorithms for inverse reinforcement learning. In *Proc. 17th Int. Conf. Machine Learning*, pages 663–670, 2000.

[27] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. 21st Int. Conf. Machine Learning*, pages 1–8, 2004.