Preference-Based Policy Learning

Riad Akrour, Marc Schoenauer, and Michele Sebag

TAO
CNRS − INRIA − Université Paris-Sud
FirstName.Name@inria.fr

Abstract. Many machine learning approaches in robotics, based on reinforcement learning, inverse optimal control or direct policy learning, critically rely on robot simulators. This paper investigates a simulator-free direct policy learning, called *Preference-based Policy Learning* (PPL). PPL iterates a four-step process: the robot demonstrates a candidate policy; the expert ranks this policy comparatively to other ones according to her preferences; these preferences are used to learn a policy return estimate; the robot uses the policy return estimate to build new candidate policies, and the process is iterated until the desired behavior is obtained. PPL requires a good representation of the policy search space be available, enabling one to learn accurate policy return estimates and limiting the human ranking effort needed to yield a good policy. Furthermore, this representation cannot use informed features (e.g., how far the robot is from any target) due to the simulator-free setting. As a second contribution, this paper proposes a representation based on the agnostic exploitation of the robotic log.

The convergence of PPL is analytically studied and its experimental validation on two problems, involving a single robot in a maze and two interacting robots, is presented.

1 Introduction

Many machine learning approaches in robotics, ranging from direct policy learning [3] and learning by imitation [7] to reinforcement learning [20] and inverse optimal control [1, 14], rely on the use of simulators used as full motion and world model of the robot. Naturally, the actual performance of the learned policies depends on the accuracy and calibration of the simulator (see [22]).

The question investigated in this paper is whether robotic policy learning can be achieved in a simulator-free setting, while keeping the human labor and computational costs within reasonable limits. This question is motivated by machine learning application to swarm robotics [24, 29, 18]. Swarm robotics aims at robust and reliable robotic systems through the interaction of a large number of small-scale, possibly unreliable, robot entities. Within this framework, the use of simulators suffers from two limitations. Firstly, the simulation cost increases more than linearly with the size of the swarm. Secondly, robot entities might differ among themselves due to manufacturing tolerance, entailing a large variance

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 12–27, 2011.

among the results of physical experiments and severely limiting the accuracy of simulated experiments.

The presented approach, inspired from both energy-based learning [21] and learning-to-rank [12, 4], is called *Preference-based policy learning* (PPL). PPL proceeds by iterating a 4-step process:

- In the *demonstration* phase, the robot demonstrates a candidate policy.
- In the *teaching* or feedback phase, the expert ranks this policy compared to archived policies, based on her agenda and prior knowledge.
- In the *learning* phase, a policy return estimate (energy criterion) is learned based on the expert ranking, using an embedded learning-to-rank algorithm. A key issue concerns the choice of the policy representation space (see below).
- In the *self-training* phase, new policies are generated, using an adaptive trade-off between the policy return estimate and the policy diversity w.r.t. the archive. A candidate policy is selected to be demonstrated and the process is iterated.

PPL initialization proceeds by demonstrating two policies, which are ordered by the expert.

A first contribution of the PPL approach compared to inverse reinforcement learning [1, 14] is that it does not require the expert to know how to solve the task and to demonstrate a perfect behavior (see also [27]); the expert is only required to know whether some behavior is more able to reach the goal than some other one. Compared to learning by demonstration [7], the demonstrated trajectories are feasible by construction (whereas the teacher and the robot usually have different degrees of freedom). Compared to policy gradient approaches [20], the continued interaction with the expert offers some means to detect and avoid the convergence to local optima, through the expert's feedback.

In counterpart, PPL faces one main challenge. The human ranking effort needed to yield a competent policy must be limited; the sample complexity must be of the order of a few dozen to a couple hundred. Therefore the policy return estimate must enable fast progress in the policy space, which requires the policy representation space to be carefully designed (a general concern, as noted by [1]). On the other hand, the simulator-free setting precludes the use of any informed features such as the robot distance to obstacles or other robots. The second contribution of the paper is an original representation of the policy space referred to as *behavioral representation* (BvR), built as follows. Each policy demonstration generates a robotic log, reporting the sensor and actuator values observed at each time step. The robotic logs are incrementally processed using ϵ-clustering [11]; each cluster is viewed as a sensori-motor state (sms). A demonstration can thus be represented by a histogram, associating to each sensori-motor state the fraction of time spent in this sms. The BvR representation complies with a resource-constrained framework and it is agnostic w.r.t. the policy return and the environment. Although the number of states exponentially increases with the clustering precision ϵ and the intrinsic dimension of the sensori-motor space, it can however be controlled to some extent as ϵ is set by the expert. BvR refinement is a perspective for further research (section 5).

PPL is analytically studied on the artificial RiverSwim problem [25] and a convergence proof is given. PPL is also experimentally validated on two problems relevant to swarm robotics. The first one concerns the reaching of a target location in a maze. The difficulty is due to perceptual aliasing (non-Markovian environment) as different locations are perceived to be similar due to the limited infra-red sensors of the robot. The second problem involves two interacting robots. The task is to yield a synchronized behavior of the two robots, controlled with the same policy. The difficulty is again due to the fact that sensors hardly enable a robot to discriminate between its relative and an obstacle.

The paper is organized as follows. Section 2 discusses related work with regards to resource-constrained robotics. Section 3 gives an overview of the PPL approach and presents a convergence study. Section 4 describes the experimental setting and reports on the results of the approach. The paper concludes with some perspectives for further research.

2 Related Work

Policy learning is most generally tackled as an optimization problem. The main issues regard the search space (policy or state/action value function), the definition of the objective, and the exploration of the search space, aimed at optimizing the objective function. The infinite horizon case is considered throughout this section for notational convenience.

Among the most studied approaches is **reinforcement learning** (RL) [26], for its guarantees of global optimality. The background notations involve a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function r defined on the state space $r : \mathcal{S} \mapsto \mathbb{R}$, and a transition function $p(s, a, s')$, expressing the probability of arriving in state s' on making action a in state s under the Markov property. A policy $\pi : (\mathcal{S}, \mathcal{A}) \mapsto \mathbb{R}$ maps each state in $\mathcal{S}$ on some action in $\mathcal{A}$ with a given probability. The policy return is defined as the expectation of cumulative reward gathered along time, conditionally to the initial state s_0. Denoting $a_h \sim \pi(s_h, a)$ the action selected by π in state s_h , $s_{h+1} \sim p(s_h, a_h, s)$ the state of the robot at step $h + 1$ conditionally to being in state s_h and selecting action a_h at step h, and r_{h+1} the reward collected in s_{h+1}, then the policy return is

$$J(\pi; s_0) = \mathbb{E}_{\pi, s_0} \left[\sum_{h=0}^{\infty} \gamma^h r_h \right]$$

where $\gamma < 1$ is a discount factor enforcing the boundedness of the return, and favoring the reaping of rewards as early as possibly in the robot lifetime. The so-called Q-value function $Q_\pi(s, a)$ estimates the expectation of the cumulative reward gathered by policy π after selecting action a in state s. As estimation errors cumulate along time, the RL scalability is limited with respect to the size of the state and action spaces. Moreover in order to enforce the Markov property, the description of state s must provide any information about the robot past trajectory relevant to its further decisions − thereby increasing the size of the state space.

As RL suffers from both the scalability issue and the difficulty of defining *a priori* a reward function conducive to the task at hand [19], a way to "seed" the search with a good solution was sought, referred to as **inverse optimal control**. One possibility, pioneered by [3] under the name of behavioral cloning and further developed by [7] under the name of programming by demonstration relies on the exploitation of the expert's traces. These traces can be used to turn policy learning into a supervised learning problem [7]. Another possibility is to use the expert's traces to learn a reward function, along the so-called inverse reinforcement learning approach [19]. The sought reward function should admit the expert policy π^* (as evidenced from his traces) as solution, and further enforce some return margin between the actual expert actions and other actions in the same state, in order to avoid indeterminacy [1] (since a constant reward function would have any policy as an optimal solution). The search space is carefully designed in both cases. In the former approach, an "imitation metric" is used to account for the differences between the expert and robot motor spaces. In the latter one, a few informed features (e.g. the average speed of the vehicle, the number of times the car deviates from the road) are used together with their desired sign (the speed should be maximized, the number of deviations should be minimized) and the policy return is sought as a weighted sum of the feature expectations. A min-max relaxation thereof is proposed by [27], maximizing the minimum policy return over all weight vectors. Yet another approach, inverse optimal heuristic control [14] proposes a hybrid approach between behavioral cloning and inverse optimal control, addressing the low-dimensionality restrictions on optimal inverse control. Formally, an energy-based action selection model is proposed using a Gibbs model which combines the reward function (as in inverse optimal control) and a logistic regression function (learned using behavioral cloning from the available trajectories).

The main limitation of inverse optimal control lies in the way it is seeded: the expert's traces can hardly be considered to be optimal in the general case, even more so if the expert and the robot live in different sensori-motor spaces. While [27] gets rid of the expert's influence, its minmax approach yields very conservative policies, as the relative importance of the features is unknown.

A main third approach aims at **direct policy learning**, using policy gradient methods [20] or global optimization methods [28]. Direct policy learning most usually assumes a parametric policy space Θ; policy learning aims at finding the optimal θ^* parameter in the sense of a policy return function J:

$$\text{Find } \theta^* = \arg\max \{J(\theta), \theta \in \Theta\}$$

Depending on J, three cases are distinguished. The first case is when J is analytically known on a continuous policy space $\Theta \subset \mathbb{R}^D$. It then comes naturally to use a gradient-based optimization approach, gradually moving the current policy θ_t along the gradient ∇J [20] ($\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta J(\theta_t)$). The main issues concern the adjustment of α_t, the possible use of the inverse Hessian, and the rate of convergence toward a (local) optimum. A second case is when J is only known as a computable function, e.g. when learning a Tetris policy [28]. In such cases, a wide scope optimization method such as Cross-Entropy Method [8] or

population-based stochastic optimization [23] must be used. In the third case, e.g. [7, 14], J is to be learned from the available evidence.

This brief and by no means exhaustive review of related work suggests that the search for an optimal policy relies on three main components. The first one clearly is the expert's support, usually provided through a reward function directly defined on the state space, or inferred from the expert's demonstrations. The second one is a simulator or forward model, enabling the robot to infer the reward function in the IOC case, and more generally to consider the future consequences of its exploratory actions; the lack of any planning component seemingly is the main cause for the limitations of behavioral cloning, forcluding the generalization of the provided demonstrations. The third one is a low-dimensionality representation of the state or policy search spaces.

With regards to swarm robotics however, a good quality simulator or forward model is unlikely to be available anytime soon, for the reasons discussed in the introduction. The expert's support is also bound to be limited, for she can hardly impersonate the swarm robot on the one hand, and because the desired behavior is usually unknown on the other hand[1]. The proposed approach will thus be based on a less demanding interaction between the policy learning process and the expert, only assuming that the former can *demonstrate* policies and that the latter can emit *preferences*, telling a more appropriate behavior from a lesser one.

3 Preference-Based Policy Learning

Only deterministic parameterized policies with finite time horizon H will be considered in this section, where H is the number of time steps each candidate policy is demonstrated.

3.1 PPL Overview and Notations

Policy π_θ, defined from its parameter $\theta \in \Theta \subseteq \mathbb{R}^D$, stands for a deterministic mapping from the state space $\mathcal{S}$ onto the action space $\mathcal{A}$. A *behavioral representation* is defined from the demonstrations (section 3.2) and used to learn the policy return estimate. PPL proceeds by maintaining a policy and constraint archive, respectively denoted Π_t and $\mathcal{C}_t$ along a four-step process. At step t,

- A new policy π_t is demonstrated by the robot and added to the archive ($\Pi_t = \Pi_{t-1} \cup \{\pi_t\}$);
- π_t is ranked by the expert w.r.t. the other policies in Π_t, and the set $\mathcal{C}_t$ is accordingly updated from $\mathcal{C}_{t-1}$;
- The policy return estimate J_t is built from all constraints in $\mathcal{C}_t$ (section 3.3);
- New policies are generated; candidate policy π_{t+1} is selected using an adaptive trade-off between J_t and an empirical diversity term (section 3.4), and the process is iterated.

[1] Swarm policy learning can be viewed as an inverse problem; one is left with the problem of e.g. designing the ant behavior in such a way that the ant colony achieves a given task.

PPL is initialized from two policies with specific properties (section 3.5), which are ordered by the expert; the policy and constraint archives are initialized accordingly. After detailing all PPL components, an analytical convergence study of PPL is presented in section 3.6.

3.2 The Behavioral Representation

In many parametric settings, e.g. when policy π is characterized as the weight vector $\theta \in \Theta$ of a neural net with fixed architecture, the Euclidean metric on Θ is poorly informative of the policy behavior. As the sought policy return estimate is meant to assess policy behavior, it thus appears inappropriate to learn J_t as a mapping on Θ. The proposed *behavioral representation* (BvR) fits the robot bounded memory and computational resources by mapping each policy onto a real valued vector ($\mu : \pi \mapsto \mu_\pi \in \mathbb{R}^d$) as follows. A robot equipped with a policy π can be thought of as a data stream generator; the sensori-motor data stream SMS(π) consists of the robot sensor and actuator values recorded at each time step. It thus comes naturally to describe policy π through compressing SMS(π), using an embedded online clustering algorithm. Let SMS(π) be given as $\{x_1, \ldots x_H\}$ with $x_h \in \mathbb{R}^b$, where b is the number of sensors and actuators of the robot and x_h denotes the concatenation of the sensory and motor data of the robot at time step h. An ϵ-clustering algorithm ($\epsilon > 0$) [11] is used to incrementally define a set $\mathcal{S}$ of sensori-motor states s_i from all SMS(π) data streams. For each considered policy π, in each time step, x_h is compared to all elements of $\mathcal{S}$, and it is added to $\mathcal{S}$ if $min\{||x_h - s_i||_\infty,\ s_i \in \mathcal{S}\} > \epsilon$. Let n_t denote the number of states at time t. BvR is defined as follows:

$$\begin{array}{ll} \text{Given SMS}(\pi) = \{x_1, \ldots x_H\} \text{ and } & \mathcal{S} = \{s_1, \ldots s_{n_t}\} \\ \mu : \pi \mapsto \mu_\pi \in [0,1]^{n_t} & \mu_\pi[i] = \frac{|\{x_h \ s.t. \ ||x_h - s_i||_\infty < \epsilon\}|}{H} \end{array}$$

BvR thus differs from the feature counts used in [1, 14] as features are learned from policy trajectories as opposed to being defined from prior knowledge. Compared to the discriminant policy representation built from the set of policy demonstrations [17], the main two differences are that BvR is independent of the policy return estimate, and that it is built online as new policies explore new regions of the sensori-motor space. As the number of states n_t and thus BvR dimension increases along time, BvR consistency follows from the fact $\mu_\pi \in [0,1]^{n_t}$ is naturally mapped onto $(\mu_\pi, 0)$ in $[0,1]^{n_t+1}$ (the i-th coordinate of μ_π is 0 if s_i was not discovered at the time π was demonstrated).

The price to pay for the representation consistency is that the number of states exponentially increases with precision parameter ϵ; on the other hand, it increases like $\epsilon^{b'}$ where b' is the intrinsic dimension of the sensori-motor data. A second limitation of BvR is to be subject to the initialization noise, meant as the initial location of the robot when the policy is launched. The impact of the initial conditions however can be mitigated by discarding the states visited during the burn-in period of the policy.

3.3 Policy Return Estimate Learning

For the sake of notational simplicity, μ_i will be used instead of μ_{π_i} when no confusion is to fear. Let the policy archive Π_t be given as $\{\mu_1, \dots \mu_t\}$ at step t, assuming with no loss of generality that μ_is are ordered after the expert preferences. Constraint archive $\mathcal{C}_t$ thus includes the set of $\frac{t(t-1)}{2}$ ordering constraints $\mu_i \succ \mu_j$ for $i > j$.

Using a standard constrained convex optimization formulation [4, 13], the policy return estimate J_t is sought as a linear mapping $J_t(\mu) = \langle \mathbf{w}_t, \mu \rangle$ with $\mathbf{w}_t \in \mathbb{R}^{n_t}$ solution of (P):

$$(P) \begin{cases} \text{Minimize} & \frac{1}{2} \, ||\mathbf{w}||^2 + C \sum_{i,j=1, i>j}^{t} \xi_{i,j} \\ \text{subject to} & (\langle \mathbf{w}, \mu_i \rangle - \langle \mathbf{w}, \mu_j \rangle \geq 1 - \xi_{i,j}) \text{ and } (\xi_{i,j} \geq 0) \text{ for all } i > j \end{cases}$$

The policy return estimate J_t features some good properties. Firstly, it is consistent despite the fact that the dimension n_t of the state space might increase with t; after the same arguments as above, the $\mathbf{w}_t$ coordinate related to a state which has not yet been discovered is set to 0. By construction, it is independent on the policy parameterization and can be transferred among different policy spaces; likewise, it does not involve any information but the information available to the robot itself; therefore it can be computed on-board and provides the robot with a self-assessment.

Finally, although J_t is defined at the global policy level, $\mathbf{w}_t$ provides some valuation of the sensori-motor states. If a high positive weight $\mathbf{w}_t[i]$ is associated to the i-th sensori-motor state s_i, this state is considered to significantly and positively contribute to the quality of a policy, *comparatively to the policies considered so far*. Learning J_t thus significantly differs from learning the optimal RL value function V^*. By definition, $V^*(s_i)$ estimates the maximum expected cumulative reward ahead of state s_i, which can only increase as better policies are discovered. In contrast, $\mathbf{w}_t[i]$ reflects the fact that visiting state s_i is conducive to discovering better policies, comparatively to policies viewed so far by the expert. In particular, $\mathbf{w}_t[i]$ can increase *or decrease* along t; some states might be considered as highly beneficial in the early stages of the robot training, and discarded later on.

3.4 New Policy Generation

The policy generation step can be thought of in terms of self-training. The generation of new policies relies on black-box optimization; expected-global improvement methods [6] and gradient methods [20] are not applicable since the policy return estimate is not defined on the parametric representation space Θ. New policies are thus generated using a stochastic optimization method, more precisely a $(1 + \lambda)$-ES algorithm [2]. Formally, a Gaussian distribution $\mathcal{N}(\theta_c, \Sigma)$ on Θ is maintained, updating the center θ_c of the distribution and the covariance matrix Σ after the parameter θ of the current best policy. For n_g iterations, λ policies π are drawn after $\mathcal{N}(\theta_c, \Sigma)$, they are executed in order to compute their

behavioral description μ_π, and the best one after an optimisation criterion $\mathcal{F}$ (see below) is used to update $\mathcal{N}(\theta_c, \Sigma)$. After n_g iterations of the policy generation step, the best policy after $\mathcal{F}$ denoted π_{t+1} is selected and demonstrated to the expert, and the whole PPL process is iterated.

The criterion $\mathcal{F}$ used to select the best policy out of the current λ policies is meant to enforce some trade-off between the exploration of the policy space and the exploitation of the current policy return estimate J_t (note that the discovery of new sensori-motor states is worthless after J_t since their weight after $\mathbf{w}_t$ is 0). Taking inspiration from [2], $\mathcal{F}$ is defined as the sum of the policy return estimate J_t and a weighted exploration term E_t, measuring the diversity Δ of the policy w.r.t the policy archive Π_t:

$$\begin{aligned} \text{Maximize } \mathcal{F}(\mu) &= J_t(\mu) + \alpha_t E_t(\mu) \qquad \alpha_t > 0 \\ \text{with } E_t(\mu) &= \min \{\Delta(\mu, \mu_u),\ \mu_u \in \Pi_t\} \\ \alpha_t &= \begin{cases} c.\alpha_{t-1} & \text{if } \pi_t \succ \pi_{t-1} \qquad c > 1 \\ \frac{1}{c^{1/p}}\alpha_{t-1} & \text{otherwise} \end{cases} \end{aligned}$$

Parameter α_t controls the **exploration vs exploitation tradeoff**, accounting for the fact that both the policy distribution and the objective function J_t are non stationary. Accordingly, α_t is increased or decreased by comparing the empirical success rate (whether π_t improves on all policies in the archive) with the expected success rate of a reference function (usually the sphere function [2]). When the empirical success rate is above (respectively below) the reference one, α_t is increased (resp. decreased). Parameter p, empirically adjusted, is used to guarantee that p failures cancel out one success and bring α_t back to its original value.

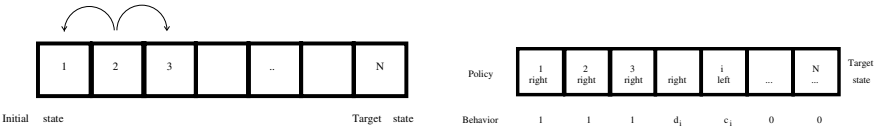
The **diversity function** $\Delta(\mu, \mu')$ is defined as follows. Let a, b, c respectively denote the number of states visited by μ only, by μ' only, and by both μ and μ'. Then $\Delta(\mu, \mu')$ is set to $\frac{a+b-c}{\sqrt{a+c}\sqrt{b+c}}$; it computes and normalizes the symmetric difference minus the intersection of the two sets of states visited by μ and μ'.

3.5 Initialization

The PPL initialization is challenging for policy behaviors corresponding to uniformly drawn θ usually are quite uninteresting and therefore hard to rank. The first two policies are thus generated as follows. Given a set P_0 of randomly generated policies, π_1 is selected as the policy in P_0 with maximal information quantity ($J_0(\mu) = -\sum_{i=1}^{d} \mu[i] \log \mu[i]$). Policy π_2 is the one in P_0 with maximum diversity w.r.t. π_1 ($\pi_2 = argmax \{\Delta(\mu, \pi_1), \mu \in P_0\}$). The rationale for this initialization is that π_1 should experiment as many distinct sensorimotor states as possible; and π_2 should experiment as many sensorimotor states different from that of π_1 as possible, to facilitate the expert ordering and yield an informative policy return estimate J_2. Admittedly, the use of the information quantity and more generally BvR only make sense if the robot faces a sufficiently rich environment. In an empty environment, i.e. when there is nothing to see, the robot can only visit a single sensori-motor state.

3.6 Convergence Study

PPL convergence is analytically studied and proved for the artificial RiverSwim problem [25]. This problem involves N states and two actions, going *left* or *right*; starting from the leftmost state, the goal is to reach the rightmost state (Fig. 1). While the policy parametric representation space is $\{\text{left, right}\}^N$, the



(a) State and Action Space (b) Parametric and Behavioral representation

Fig. 1. The RiverSwim problem. The parametric representation of a policy π specifies the action (go left or right) selected in each state. The behavioral representation μ_π is $(1, ...1, 0, ..0)$, with the rightmost 1 at position $\ell(\pi)$, defined as the leftmost state associated to the *left* move.

policy π behavior is entirely characterized from the leftmost state i where it goes left, denoted $\ell(\pi)$. In the following, we only consider the boolean BvR, with $\mu_\pi = \{1, 1, .., 1, 0, ..0\}$ and $\ell(\pi)$ the index of the rightmost 1. For notational simplicity, let μ_π and $\mu_{\pi'}$ be noted μ and μ' in the following. By definition, the expert prefers the one out of μ and μ' which goes farther on the right ($\mu \succ \mu'$ iff $\ell(\mu) > \ell(\mu')$).

In the RiverSwim setting, problem (P) can be solved analytically (section 3.3). Let archive Π_t be given as $\{\mu_1 \dots \mu_t\}$ and assume wlog that μ_is are ordered by increasing value of $\ell(\mu_i)$. It comes immediately that constraints related to adjacent policies ($\langle \mathbf{w}, \mu_{\mathbf{i+1}} \rangle - \langle \mathbf{w}, \mu_i \rangle \geq 1 - \xi_{i,i+1}$ for $1 \leq i < t-1$) entail all other constraints, and can be all satisfied by setting $\sum_{k=\ell(\mu_i)+1}^{\ell(\mu_{i+1})} \mathbf{w}[k] = 1$; slack variables $\xi_{i,j}$ thus are 0 at the optimum.

Problem (P) can thus be decomposed into $t-1$ independent problems involving disjoint subsets of indices $]\ell(\mu_i), \ell(\mu_{i+1})]$; its solution $\mathbf{w}$ is given as:

$$\mathbf{w}[k] = \begin{cases} 0 & \text{if } k < \ell(\mu_1) \text{ or } k > \ell(\mu_t) \\ \frac{1}{\ell(\mu_{i+1}) - \ell(\mu_i)} & \text{if } \ell(\mu_i) < k \leq \ell(\mu_{i+1}) \end{cases}$$

Proposition 1: In the RiverSwim setting, $J_t(\mu) = \langle \mathbf{w}_t, \mu \rangle$ where $\mathbf{w}_t$ satisfies: $\mathbf{w}_t[k] = 0$ if $k \leq \ell(\mu_1)$ or $k > \ell(\mu_t)$; $\mathbf{w}_t[k] > \frac{1}{N}$ if $\ell(\mu_1) < k \leq \ell(\mu_t)$; $\sum_{k=1}^{i=N} \mathbf{w}_t[k] = t$. □

Policy generation (section 3.4) considers both the current J_t and the diversity E_t respective to the policy archive Π_t, given as $E_t(\mu) = min\{\Delta(\mu, \mu_u), \mu_u \in \Pi_t\}$. In the RiverSwim setting, it comes:

$$\Delta(\mu, \mu_u) = \frac{|\ell(\mu) - \ell(\mu_u)| - \min(\ell(\mu), \ell(\mu_u))}{\sqrt{\ell(\mu)\ell(\mu_u)}}$$

For $i < j$, $\frac{|i-j|-\min(i,j)}{\sqrt{ij}} = \sqrt{\frac{j}{i}} - 2\sqrt{\frac{i}{j}}$. It follows that the function $i \mapsto \frac{\sqrt{|\ell(\mu)-i|}-\min(\ell(\mu),i)}{\sqrt{\ell(\mu)\, i}}$ is strictly decreasing on $[1, \ell(\mu)]$ and strictly increasing on $[\ell(\mu), +\infty)$. It reaches its minimum value of -1 for $i = \ell(\mu)$. As a consequence, for any policy μ in the archive, $E_t(\mu) = -1$.

Let μ_t and μ^* respectively denote the best policy in the archive Π_t and the policy that goes exactly one step further right. We will now prove the following result:

Proposition 2: The probability that μ^* is generated at step t is bounded from below by $\frac{1}{eN}$. Furthermore, after μ^* has been generated, it will be selected according to the selection criterion $\mathcal{F}_t$ (section 3.4), and demonstrated to the expert.

Proof
Generation step: Considering the boolean parametric representation space $\{0,1\}^N$, the generation of new policies proceeds using the standard bitflip mutation, flipping each bit of the current μ_t with probability $\frac{1}{N}$. The probability to generate μ^* from μ_t is lower-bounded by $\frac{1}{N}(1-\frac{1}{N})^{\ell(\mu_t)}$ (flip bit $\ell(\mu_t)+1$ and do not flip any bit before that). Furthermore, $(1-\frac{1}{N})^{\ell(\mu_t)} > (1-\frac{1}{N})^{N-1} > \frac{1}{e}$ and hence the probability to generate μ^* from μ_t is lower-bounded by $\frac{1}{eN}$.
Selection step. As shown above, $E_t(\mu^*) > -1$ and $E_t(\mu) = -1$ for all $\mu \in \Pi_t$. As the candidate policy selection is based on the maximization of $\mathcal{F}_t(\mu) = \langle \mathbf{w}_t, \mu \rangle + \alpha_t E_t(\mu)$, and using $\langle \mathbf{w}_t, \mu_t \rangle = \langle \mathbf{w}_t, \mu_{t+1} \rangle = t$, it follows that $\mathcal{F}_t(\mu_t) < \mathcal{F}_t(\mu^*)$, and more generally, that $\mathcal{F}_t(\mu) < \mathcal{F}_t(\mu^*)$ for all $\mu \in \Pi_t$. Consider now a policy μ that is not in the archive though $\ell(\mu) < t$ (the archive does not need to contain all possible policies). From Proposition 1, it follows that $\langle \mathbf{w}, \mu \rangle < t - \frac{1}{N}$. Because $E_t(\mu)$ is bounded, there exists a sufficiently small α_t such that $\mathcal{F}_t(\mu) = \langle \mathbf{w}_t, \mu \rangle + \alpha_t E_t(\mu) < \mathcal{F}_t(\mu^*)$. □

Furthermore, thanks to the monotonicity of $E_t(\mu)$ w.r.t. $\ell(\mu)$, one has $\mathcal{F}_t(\mu_{t+i}) < \mathcal{F}_t(\mu_{t+j})$ for all $i < j$ where $\ell(\mu_{t+i}) = \ell(\mu_t) + i$. Better RiverSwim policies will thus be selected along the policy generation and selection step.

4 Experimental Validation

This section reports on the experimental validation of the PPL approach. For the sake of reproducibility, all reported results have been obtained using the publicly available simulator Roborobo [5].

4.1 Experiment Goals and Experimental Setting

The experiments are meant to answer three main questions. The first one concerns the feasibility of PPL compared to baseline approaches. The use as surrogate optimization objective of the policy return estimate learned from the expert preferences, is compared to the direct use of the expert preferences. The performance indicator is the *speed-up* in terms of sample complexity (number of calls

to the expert), needed to reach a reasonable level of performance. The second question regards the merits of the behavioral representation comparatively to the parametric one. More specifically, the point is to know whether BvR can enforce an effective trade-off between the number of sensori-motor states and the perceptual aliasing, enabling an accurate policy return estimate to be built from few orderings. A third question regards the merits of the exploration term used in the policy generation (self-training phase, section 3.4), and the entropy-based initialization process (section 3.5), and how they contribute to the overall performance of PPL. Three experiments are considered to answer these questions. The first one is an artificial problem which can be viewed as a 2D version of the RiverSwim [25]. The second experiment, inspired from [16], involves a robot in a maze and the goal is to reach a given location. The third experiment involves two robots and the goal is to enforce a coordinated exploration of the arena by the two robots. In all experiments the robotic architecture is a Cortex-M3 with 8 infra-red (IR) sensors and two motors respectively controlling the rotational and translation speed. The IR range is 6 cm; the surface of the arena is 6.4 square meters.

Every policy is implemented as a 1-hidden-layer feed-forward neural net, using the 8 IR sensors (and a bias) as input, with 10 neurons on the hidden layer, and the two motor commands as output ($\Theta = \mathbb{R}^{121}$). A random policy is built by uniformly selecting each weight in $[-1, 1]$. BvR is built using norm L_∞ and $\epsilon = 0.45$; the number of sensori-motor states, thus the BvR dimension, is below 1,000. The policy return estimate is learned using SVM^{rank} library [13] with linear kernel and default parameter $C = 1$ (section 3.3). The optimization algorithm used in the policy generation (section 3.4) and initialization (section 3.5) is a $(1+\lambda)$-ES [23] with $\lambda = 7$; the variance of the Gaussian distribution is initialized to .3, increased by a factor of 1.5 upon any improvement and decreased by a factor of $1.5^{-1/4}$ otherwise. The $(1+\lambda)$-ES is used for n_g iterations in each policy generation step, with $n_g = 10$ in the maze (respectively $n_g = 5$ in the two-robot) experiment; it is used for 20 iterations in the initialization step. All presented results are averaged over 41 independent runs. Five algorithms are compared: PPL_{BvR} with entropy-based initialization and exploration term, where the policy return estimate relies on the behavioral representation; PPL_{param}, which only differs from PPL_{BvR} as the policy return estimate is learned using the parametric representation; *Expert-Only*, where the policy return estimate is replaced by the true expert preferences; *Novelty-Search* [16] which can be viewed as an exploration-only approach[2]; and $\text{PPL}_{w/o\ i}$ which differs from PPL_{BvR} as it uses a random uniform initialization.

4.2 2D RiverSwim

In this 4×4 grid world, the robot starts in the lower left corner of the grid; the sought behavior is to reach the upper right corner and to stay there. The expert

[2] In each time step the candidate policy with highest average diversity compared to its k-nearest neighbors in the policy archive is selected, with $k = 1$ for the 2D RiverSwim, and $k = 5$ for both other experiments.

preference goes toward the policy going closer to the goal state, or reaching earlier the goal state. The action space includes the four move directions with deterministic transitions (the robot stays in the same position upon marching to the wall). For a time horizon $H = 16$, the BvR includes only 6334 distinguishable policies (to be compared with the actual 4^{16} distinct policies). The small size of this grid world allows one to enumeratively optimize the policy return estimate, with and without the exploration term. Fig. 2 (left) displays the true policy return *vs* the number of calls to the expert. In this artificial problem, *Novelty-Search* outperforms PPL and discovers an optimal policy in the 20-th step. PPL discovers an optimal policy at circa the 30-th step. The exploration term plays an important role in PPL performance; Fig. 2 (right) displays the average weight α_t of the exploration term. In this small-size problem, a mere exploration strategy is sufficient and typically *Novelty-Search* discovers two optimal policies in the first 100 steps on average; meanwhile, PPL discovers circa 25 optimal policies in the first 100 steps on average.

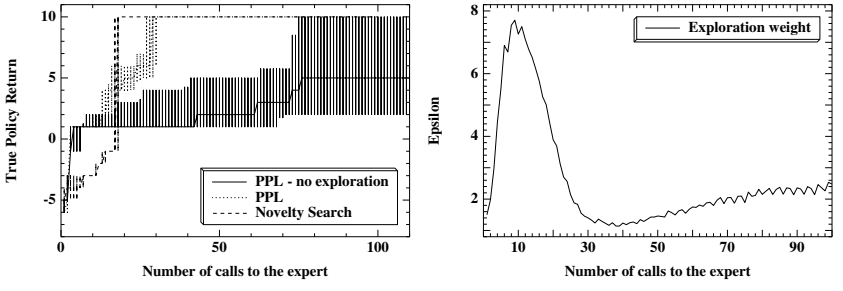


Fig. 2. 2D RiverSwim: Performance of PPL and *Novelty-Search*, averaged over 41 runs with std. dev (left); average weight of the exploration term α_t in PPL (right)

4.3 Reaching the End of the Maze

In this experiment inspired from [15], the goal is to traverse the maze and reach the green zone when starting in the opposite initial position (Fig. 3-left). The time horizon H is 2,000; an oracle robot needs circa 1,000 moves to reach the target location. As in the RiverSwim problem, the expert preference goes toward the policy going closer to the goal state, or reaching earlier the goal state; the comparison leads to a tie if the difference is below 50 distance units or 50 time steps. The "true" policy return is the remaining time when it first reaches the green zone, if it reaches it, minus the min distance of the trajectory to the green zone, otherwise. The main difficulty in this maze problem is the severe perceptual aliasing; all locations situated far from the walls look alike due to the IR sensor limitations.

This experiment supports the feasibility of the PPL scheme. A speed-up factor of circa 3 is observed compared to *Expert-Only*; on average PPL reaches the green zone after demonstrating 39 policies to the expert, whereas *Expert-Only*

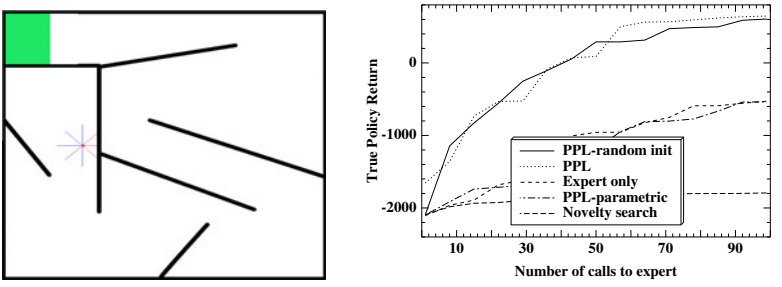


Fig. 3. The maze experiment: the arena (left) and best performances averaged out of 41 runs

needs 140 demonstrations (Fig. 4, left). This speed-up is explained as PPL filters out unpromising policies. *Novelty-Search* performs poorly on this problem comparatively to PPL and even comparatively to *Expert-Only*, which suggests that *Novelty-Search* might not scale up well with the size of the policy space.

This experiment also establishes the merits of the behavioral representation. Fig. 4 (right) displays the average performance of PPL_{BvR} and PPL_{param} when no exploration term is used, to only assess the accuracy of the policy return estimate. As can be seen, learning is very limited when done in the parametric representation by PPL_{param}, resulting in no speedup compared to *Expert-Only* (Fig. 3, right). A third lesson is that the entropy-based initialization does not seem to bring any significant improvement, as PPL_{BvR} and $\text{PPL}_{w/o\ i}$ get same results after the very first steps (Fig. 3).

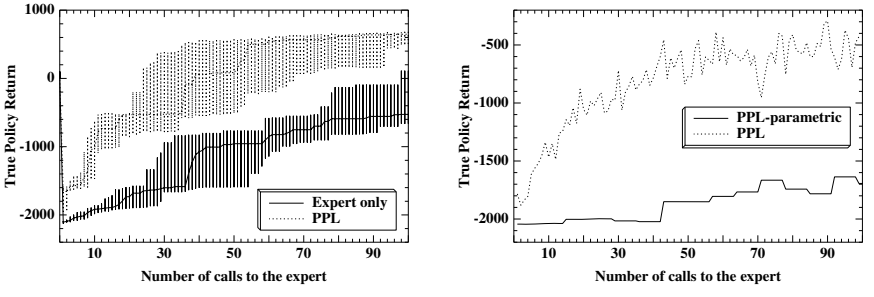


Fig. 4. The maze experiment: PPL vs *Expert-Only* (left); Accuracy of parametric and behavioral policy return estimate(right)

4.4 Synchronized Exploration

This experiment investigates the feasibility of two robots exploring an arena (Fig. 5, left) in a synchronous way, using the same policy. The expert preferences are emulated by associating to a policy the number of distinct 25x25 tiles explored by any of the two robots conditionally to the fact that their distance is below 100

unit distances at the moment the tile is explored. Both robots start in (distinct) random positions. The difficulty of the task is that most policies wander in the arena, making it unlikely for the two robots to be anywhere close to one another at any point in time.

The experimental results (Fig. 5, right) mainly confirm the lessons drawn from the previous experiment. PPL improves on *Expert-Only*, with a speed-up circa 10, while *Novelty-Search* is slightly but significantly outperformed by *Expert-Only*. In the meanwhile, the entropy-based initialization does not make much of a difference after the first steps, and might even become counter-productive in the end.

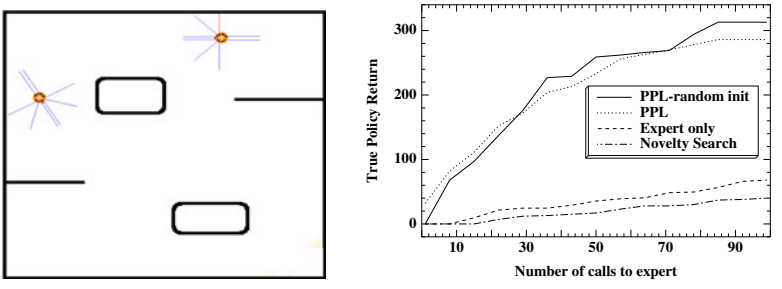


Fig. 5. Synchronous exploration: the arena (left) and the best average performance out of 41 runs (right)

5 Conclusion and Perspectives

The presented preference-based policy learning demonstrates the feasibility of learning in a simulator-free setting. PPL can be seen as a variant of Inverse Optimal Control, with the difference that the demonstrations are done by the robot whereas the expert only provides some feedback (*it's better; it's worse*). PPL, without any assumptions on the level of expertise of the teacher, incrementally learns a policy return estimate. This learning is made possible through the original unsupervised *behavioral representation* BvR, based on the compression of the robot trajectories. The policy return estimate can be thought of as an embedded "system of values", computable on-board. Further research will investigate the use of the policy return estimate within lifelong learning, e.g. to adjust the policy and compensate for the fatigue of the actuators. While the convergence of the approach can be established in the artificial RiverSwim framework [25], the experimental validation demonstrates that PPL can be used effectively to learn elementary behaviors involving one or two robots.

The main limitation of the approach comes from the exponential increase of the behavioral representation w.r.t. the granularity of the sensori-motor states. Further work will investigate how to refine BvR, specifically using quad-trees to describe and retrieve the sensori-motor states while adaptively adjusting their granularity. A second perspective is to adjust the length of the self-training

phases, taking inspiration from online learning on a budget [9]. A third research avenue is to reconsider the expert preferences in a Multiple-Instance perspective [10]; clearly, what the expert likes/dislikes might be a fragment of the policy trajectory, more than the entire trajectory.

Acknowledgments. The first author is funded by FP7 European Project *Symbrion*, FET IP 216342, `http://symbrion.eu/`. This work is also partly funded by ANR Franco-Japanese project Sydinmalas ANR-08-BLAN-0178-01.

References

[1] Abbeel, P., Ng, A.Y.: Apprenticeship Learning via Inverse Reinforcement Learning. In: Brodley, C.E. (ed.) Proc. 21st Intl. Conf. on Machine Learning (ICML 2004). ACM Intl. Conf. Proc. Series, vol. 69, p. 1. ACM, New York (2004)

[2] Auger, A.: Convergence Results for the (1,λ)-SA-ES using the Theory of φ-irreducible Markov Chains. Theoretical Computer Science 334(1-3), 35–69 (2005)

[3] Bain, M., Sammut, C.: A Framework for Behavioural Cloning. In: Furukawa, K., Michie, D., Muggleton, S. (eds.) Machine Intelligence, vol. 15, pp. 103–129. Oxford University Press, Oxford (1995)

[4] Bakir, G., Hofmann, T., Scholkopf, B., Smola, A.J., Taskar, B., Vishwanathan, S.V.N.: Machine Learning with Structured Outputs. MIT Press, Cambridge (2006)

[5] Bredeche, N.: `http://www.lri.fr/~bredeche/roborobo/`

[6] Brochu, E., de Freitas, N., Ghosh, A.: Active Preference Learning with Discrete Choice Data. In: Proc. NIPS 20, pp. 409–416 (2008)

[7] Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Trans. on Systems, Man and Cybernetics, Special Issue on Robot Learning by Observation, Demonstration and Imitation 37(2), 286–298 (2007)

[8] de Boer, P.-T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A Tutorial on the Cross-Entropy Method. Annals OR 134(1), 19–67 (2005)

[9] Dekel, O., Shalev-Shwartz, S., Singer, Y.: The Forgetron: A Kernel-Based Perceptron on a Budget. SIAM J. Comput. 37, 1342–1372 (2008)

[10] Dietterich, T.G., Lathrop, R., Lozano-Perez, T.: Solving the Multiple-Instance Problem with Axis-Parallel Rectangles. Artif. Intelligence 89(1-2), 31–71 (1997)

[11] Duda, R.O., Hart, P.E.: Pattern Classification and scene analysis. John Wiley and sons, Menlo Park, CA (1973)

[12] Joachims, T.: A Support Vector Method for Multivariate Performance Measures. In: De Raedt, L., Wrobel, S. (eds.) Proc. 22nd ICML. ACM Intl. Conf. Proc. Series, vol. 119, pp. 377–384. ACM, New York (2005)

[13] Joachims, T.: Training Linear SVMs in Linear Time. In: Eliassi-Rad, T., et al. (eds.) Proc. 12th Intl. Conf. KDDM, pp. 217–226. ACM, New York (2006)

[14] Zico Kolter, J., Abbeel, P., Ng, A.Y.: Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion. In: Proc. NIPS 20. MIT Press, Cambridge (2007)

[15] Lehman, J., Stanley, K.O.: Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In: Proc. Artificial Life XI, pp. 329–336 (2008)

[16] Lehman, J., Stanley, K.O.: Exploiting Open-Endedness to Solve Problems through the Search for Novelty. In: Proc. ALife 2008, MIT Press, Cambridge (2008)

[17] Levine, S., Popovic, Z., Koltun, V.: Feature Construction for Inverse Reinforcement Learning. In: Proc. NIPS 23, pp. 1342–1350 (2010)
[18] Liu, W., Winfield, A.F.T.: Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems. Intl. J. Robotic Research 29(14), 1743–1760 (2010)
[19] Ng, A.Y., Russell, S.: Algorithms for Inverse Reinforcement Learning. In: Langley, P. (ed.) Proc. 17th ICML, pp. 663–670. Morgan Kaufmann, San Francisco (2000)
[20] Peters, J., Schaal, S.: Reinforcement Learning of Motor Skills with Policy Gradients. Neural Networks 21(4), 682–697 (2008)
[21] Ranzato, M.-A., Poultney, C.S., Chopra, S., LeCun, Y.: Efficient Learning of Sparse Representations with an Energy-Based Model. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Proc. NIPS 19, pp. 1137–1144. MIT Press, Cambridge (2006)
[22] Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic Grasping of Novel Objects using Vision. Intl. J. Robotics Research (2008)
[23] Schwefel, H.-P.: Numerical Optimization of Computer Models. John Wiley & Sons, New York (1981) 2nd edn. (1995)
[24] Stirling, T.S., Wischmann, S., Floreano, D.: Energy-efficient Indoor Search by Swarms of Simulated Flying Robots without Global Information. Swarm Intelligence 4(2), 117–143 (2010)
[25] Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC Model-free Reinforcement Learning. In: Airoldi, E.M., Blei, D.M., Fienberg, S.E., Goldenberg, A., Xing, E.P., Zheng, A.X. (eds.) ICML 2006. LNCS, vol. 4503, pp. 881–888. Springer, Heidelberg (2007)
[26] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
[27] Syed, U., Schapire, R.: A Game-Theoretic Approach to Apprenticeship Learning. In: Proc. NIPS 21, pp. 1449–1456. MIT Press, Cambridge (2008)
[28] Thiery, C., Scherrer, B.: Improvements on Learning Tetris with Cross Entropy. ICGA Journal 32(1), 23–33 (2009)
[29] Trianni, V., Nolfi, S., Dorigo, M.: Cooperative Hole Avoidance in a Swarm-bot. Robotics and Autonomous Systems 54(2), 97–103 (2006)