Preference-Based Reinforcement Learning: A preliminary survey

Christian Wirth and Johannes Fürnkranz

Knowledge Engineering, Technische Universität Darmstadt, Germany
{cwirth,fuernkranz}@ke.tu-darmstadt.de

Abstract. Preference-based reinforcement learning has gained significant popularity over the years, but it is still unclear what exactly preference learning is and how it relates to other reinforcement learning tasks. In this paper, we present a general definition of preferences as well as some insight how these approaches compare to reinforcement learning, inverse reinforcement learning and other related approaches. Additionally, we are offering a coarse categorization of preference-based reinforcement learning algorithms and a preliminary survey based on this allocation.

1 Introduction

Preference-based reinforcement learning (PBRL) is usually considered as learning a policy for a *Markov decision processes (MDP)*, which satisfies a given set of preferences over trajectories or partial trajectories, but this is not a clearly defined problem. There is no common definition for preferences or a consensus what can be assumed concerning the underlying decision process. Therefore, we will try to identify similarities and differences by relating PBRL to conventional reinforcement learning. In both scenarios, MDPs are assumed, but are not necessary. Their advantage lies in the well-defined theory of MDPs which comes with a large amount of theoretical and empirical analysis, allowing to build on existing work. All algorithms for PBRL, known to the authors, are using MDPs, hence we are also sticking to this assumption. However, the common definition for MDPs like the one given by Sutton and Barto [15] implies a notation of rewards, concerning single state-action pairs. Those rewards are not available or not existent in preference-based approaches, hence we are turning to a reward-free definition (MDP\R) [1], which is used by *inverse reinforcement learning (IRL)* approaches in general.

After formalising the notion of MDP\R in section 2, we will take a closer look at what preferences are and how they can be described in section 3. In section 4, we will see that the key step for defining a learning problem based on an MDP\R and a preference relation on sequences of state-action pairs is the definition of a preference relation over policies, which can either be explicitly modeled or be defined via a utility function. Existing preference-based reinforcement learning algorithms can be categorised by this difference. In section 5, we are presenting algorithms employing a utility function while the approaches in section 6 are

explicitly modelling a relation between trajectory- and policy-preferences. Before we conclude the paper in section 8, we also take a brief look at the relation between the different reinforcement learning subtasks and PBRL in section 7.

2 MDP\R

Abbeel and Ng [1] have introduced the notion of *Markov decision processes without rewards (MDP\R)*. A (finite-state) MDP\R is defined by a quadruple (S, A, δ, γ). Given are a set of *states* $S = \{s_i\}$, *actions* $A = \{a_j\}$, and a probabilistic *state transition function* $\delta : S \times A \times S \to [0, 1]$ such that $\sum_{s'} \delta(s, a, s') = 1$ for all $(s, a) \in S \times A$.

A *policy* $\pi : S \times A \to [0, 1]$ is a probabilistic function that assigns probabilities to state action pairs s.t.$\sum_{a'} \pi(s, a') = 1$ for all $s \in S$. A *trajectory* is an alternating sequence of states and actions

$$T = \{s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n\}.$$

The space of all trajectories is denoted as Υ, the set of trajectories that can be generated with a given policy π is denoted as Υ^π. Most MDPs have goal states and/or a finite horizon, resulting in finite length trajectories. In the following, we assume that all trajectories start in the same *start state* s_0. Multiple start states can be included by using a single, virtual s_0 with a null-action that has probabilistic transitions to the true start states. Let $A(s)$ denote the set of actions that are possible in state s, i.e., $A(s) = \{a \in A | \sum_{s' \in S} \delta(s, a, s') > 0\}$. A *terminal state* is a state in which no action is available, i.e., where $A(s) = \emptyset$. A trajectory is called *complete* if it leads from a start state s_0 to a terminal state s_n. As a slight generlization of trajectories, we will, in the following, also consider *state-action sequences* $C : \mathbb{N} \to S \times A$, where $C(n)$ denotes the n-th state-action pair in the sequence.

We assume that states and actions are represented by a k-dimensional normalized feature vector of the form $\phi : S \times A \to [0, 1]^k$ over states and actions. The standard tabular representation of a state-action space can, e.g., be obtained with the feature set

$$\phi(s, a)^{(i,j)} = \begin{cases} 1 & \textbf{if } s = s_i \text{ and } a = a_j \\ 0 & \textbf{else} \end{cases} \tag{1}$$

$\gamma \in [0, 1)$ is a discount factor, which is relevant if we assume an underlying, hidden reward function as is, e.g., the case in inverse reinforcement learning.

2.1 Rewards & Utility

Several of the algorithms surveyed here assume that a numeric utility can be calculated for a trajectory. In particular, we assume that a *utility function* $U^* : S \times A \to \mathbb{R}^m$ exists, but is not known. Of course, the result is non-scalar if $m > 1$,

but the value of m is usually not known. We further assume this function to be linear in terms of the features, i.e., $U^*(s,a) = \mathbf{W}^* \cdot \phi(s,a)$.

In the case of classic reinforcement learning, the utility function corresponds to the reward function. However, as pointed out by Akrour et al. [2], a reward function, as defined in the common MDP scenario is not changing, but the utility of a trajectory can change with the occurrence of new preferences.

2.2 Feature Expectations

The idea behind *feature expectations* is to break down the utility of a sequence into a (linear) sum over the state-action features ϕ, as described by Abbeel and Ng [1]. $C(t)$ is the state-action pair, encountered at index t in the sequence C. $|C|$ is the number state-action pairs in the sequence. As mentioned, the last state is not forming a pair in the case of complete trajectories and is therefore omitted. The utility of a sequence C is

$$U(C) = \sum_{t=0}^{|C|} \gamma^t U(C(t)) = \sum_{t=0}^{|C|} \gamma^t \mathbf{W} \cdot \phi((s,a)_t) = \mathbf{W} \cdot \sum_{t=0}^{|C|} \gamma^t \phi((s,a)_t) \tag{2}$$

Let us now define the vector $\mu(C)$ as the weighted sum of all values for the individual feature vectors in the sequence.

$$\mu(C) = \sum_{t=0}^{|C|} \gamma^t \phi((s,a)_t) \tag{3}$$

The feature expectations for a policy π can now be defined via the expected value of $\mu(T)$ over all trajectories T that can be observed when starting in start state s_0 and following policy π thereafter.

$$\mu(\pi) = \mathbb{E}\left[\mu(T)|\pi\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi((s,a)_t)|\pi\right] \tag{4}$$

If we know the feature expectations of a policy, we can use (2) to determine its utility as

$$U(\pi) = \mathbf{W} \cdot \mu(\pi) \tag{5}$$

$\mu(\pi)$ can, e.g., be approximated by calculating a sample based estimate for the expectation.

3 Preference-Based Feedback

One of the remaining questions is, how to define preferences in general. We are suggesting a formal representation based on sequences of state-action pairs $C(t) : \mathbb{N} \to S \times A$. A *preference* is a relation $C_1 \succeq C_2$ between two sequences C_1, C_2. We also use the following notations:

$C_1 \succ C_2$**:** The first sequence is *strictly preferred*, i.e., $C_1 \succeq C_2$ but not $C_2 \succeq C_1$.
$C_1 \prec C_2$**:** The second sequence is *strictly preferred*, i.e., $C_2 \succ C_1$.
$C_1 \sim C_2$**:** Both sequences are *indifferent*, i.e., both $C_1 \succeq C_2$ and $C_2 \succeq C_1$ hold.
$C_1 \perp C_2$**:** The sequences are *incomparable*, i.e., neither $C_1 \succeq C_2$ nor $C_2 \succeq C_1$ holds.

In many cases, the preference relation is assumed to be a total order, i.e., for each pair C_1 and C_2, either $C_1 \succ C_2$ or $C_2 \succ C_1$, or, in other words, there are no incomparable pairs of sequences. This is, e.g., the case if we assume that the preference relation can be modeled with an underlying, single-dimensional utility function, i.e.,

$$C_1 \succ C_2 \Leftrightarrow U(C_1) \geq U(C_2) \tag{6}$$

Total orders guarantee a single optimal solution. If there are incomparable pairs $C_1 \perp C_2$, the preferences form a partial order. Partial orders do not have a unique, highest rank, preventing the determination of a single optimum. Instead, a set of non-dominated solutions, the so-called *Pareto front* can be determined.

An agent will observe a subset $\zeta = C \times C$ of these preferences based on the visited states and the obtained feedback. Often, special types of preferences are observed, which fit into the proposed framework as follows:

Action Preferences An action preference $(s, a_1) \succ (s, a_2)$ means that it is preferred to select action a_1 opposed to a_2, when in state s. Within our sequence based representation, this is a preference over single element sequences $C_1 \succ C_2, C_1 = \{(s, a_1)\}, C_2 = \{(s, a_2)\}$.

State Preferences A state preference $s_1 \succ s_2$ means that it is preferred to visit state s_1 opposed to state s_2. This can be turned into a set of sequence preferences $\{(s_1, a_1) \succ (s_2, a_2) | \forall a_1 \in A(s_1) \wedge a_2 \in A(s_2)\}$ because the preference is valid no matter what action has been performed in each state.

Trajectory Preferences A trajectory $T = \{s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n\}$ is associated with the sequence $\{(s_0, a_0), \ldots, (s_{n-1}, a_{n-1})\}$, hence we can directly interpret a trajectory preference as a sequence preference.

4 The Learning Problem

We can now define a *preference-based decision process* (PBDP) as a quintuple $PBDP = (S, A, \delta, \Pi, \succ)$. Informally, the learning problem now is to identify a policy π^* in the space of policies Π which generates trajectories that agree with the observed preferences ζ. However, it is non-trivial to specify what it means that one or more policies conform to the observed preferences. A simple approach is to associate them with the trajectories they generate.

The key step that has to be addressed when solving a PBDP is how to lift preferences on the trajectory or sequence level up to preferences on the policy level. In general, we can discriminate two different approaches: *relational approaches* try to define a preference relation $\sqsupset$ on policies, directly utilizing the preferences, whereas *value-based approaches* attempt to associate numerical values with policies. In each case there are several possible approaches, which we briefly discuss in the following.

4.1 Relational Approaches

The goal of relational approaches to preference-based reinforcement learning is to define a relation $\sqsupset$: $\Pi \times \Pi$ over the space of policies, without determining a value for them. Several options are possible, we list a few of them in the following.

Dominance A simple way of defining a (partial) preference relation on policies is to postulate that a policy π_1 is preferred over a policy π_2 if it is guaranteed that π_1 always produces better trajectories than π_2.

$$\pi_1 \sqsupset \pi_2 \Leftrightarrow \forall T_1 \in \Upsilon^{\pi_1}, T_2 \in \Upsilon^{\pi_2} : T_1 \succ T_2. \tag{7}$$

Obviously, this is a quite conservative definition. For example, if every policy has non-zero probabilities for all possible actions in a state, no policy will be preferred over any other policy.

On the other hand, when policies and state transitions are deterministic, i.e., when each policy generates a unique trajectory from each starting state, this is the most natural choice.

Stochastic Dominance [8] suggested the use of *stochastic dominance* as a somehwat less conservative alternative. Essentially, a policy π_1 dominates a policy π_2, if the probability that for any given trajectory T, the probability that π_1 generates a trajectory T_1 that is at least as good as T is greater than the probability that π_2 generates a trajectory T_2 that is at least as good as T.

$$\pi_1 \sqsupset \pi_2 \Leftrightarrow \forall T \in \Upsilon : \Pr(T_1 \succ T) \geq \Pr(T_2 \succ T), \text{ where } T_1 \in \Upsilon^{\pi_1}, T_2 \in \Upsilon^{\pi_2} \tag{8}$$

Probabilistic Dominance Another way to weaken dominance is to only require that the probability of generating the preferred trajectory is higher than the probability of generating the dominated trajectory.

$$\pi_1 \sqsupset \pi_2 \Leftrightarrow \forall T_1 \in T^{\pi_1}, T_2 \in T^{\pi_2} : \Pr(T_1 \succ T_2) \geq \Pr(T_2 \succ T_1) \tag{9}$$

Note that this relation is not transitive, i.e., cyclic relationships $\pi_1 \sqsupset \pi_2 \sqsupset \pi_3 \sqsupset \pi_1$ are possible.

4.2 Value-Based Approaches

Value-based approaches assume that each policy can be associated with a numerical utility value $U(\pi)$. This is usually achieved by assuming a utility function, that can be described as a function over state-action features, as discussed in section 2.2. Its now possible to either use $U(\pi)$ for searching in the policy space or to use the implicitly given utility $U(s, a)$ as state-action value for defining a policy, comparable to a Q-function in classic reinforcement learning.

State-based Policy Evaluation State-based policy evaluation defines for each policy π a value $U(\pi, s)$ for each state $s \in S$, which indicates how good it is to follow policy π in state s. One approach is to associate with each state the probability with which a trajectory starting in this state is preferred over some other trajectory in this state.

$$U(\pi, s) = \Pr(T \succ T'), \text{ where } T \in \Upsilon^{\pi}(s), T' \in \Upsilon(s) \tag{10}$$

Thus, a policy π_1 is preferred over some other policy π_2, if π_1's probability of generating a preferred hypothesis is higher than the corresponding probability of π_2.

These probabilities can be estimated via policy roll-outs.

$$U(\pi, s) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(T \succ T'), \text{ where } T \in \Upsilon^{\pi}(s), T' \in \Upsilon^{\pi'}(s), \pi \neq \pi' \tag{11}$$

This is essentially the approach that was followed in [8], where such roll-outs where used to provide training information for a label ranker that learns a model for ranking the actions in each state.

Trajectory-based Policy Evaluation Trajectory-based policy evaluation defines for each trajectory T a value $U(T)$. It seems natural to require that a policy π^* maximises the value of the trajectories it can generate. Hence, we can evaluate a policy based on the expected sum over all trajectories, subject to the probability that the trajectory gets generated by π.

$$U(\pi) = \mathbb{E}\left[\sum_{T \in \Upsilon} U(T) | \pi\right] \tag{12}$$

A trajectory preference is now a sample concerning the value of the trajectory, meaning $T_1 \succ T_2 \Leftrightarrow U(T_1) > U(T_2)$. By assuming a state-action based utility function $U(s, a)$, it is also possible to employ the state-action preferences described in section 3 with e.g. $(s, a_1) \succ (s, a_2) \Leftrightarrow U(s, a_1) > U(s, a_2)$.

5 Value-based Algorithms

Value-based approaches have been predominantly used so far in approaches to preference-based reinforcement learning. As described in the previous section, they evaluate policies with a utility function. We can discriminate two main approaches: algorithms that operate in the policy space (aka as *policy search* in conventional reinforcement learning), utilizing $U(\pi)$, or approaches implicitly defining a policy by a value $U(s, a)$. The PPL approach, presented in section 5.3 belongs to the first category while the two other algorithms (sections 5.1 and 5.2) are concerned with the second variant.

One of the main problems of these approaches is how to utilize the preferences efficiently. Preferences are usually given over complete trajectories, preventing any feedback before a trajectory rollout has finished. Additionally, preferences are not valid for subsequences of the the sampled trajectories, because it is not known how the missing state-action pairs are influencing the utility.

5.1 Preference-based Policy Iteration (PBPI)

Fürnkranz et al. [8] have defined a Monte-Carlo-based approach. The algorithm works within the framework of policy iteration, where the policy is evaluated in each iteration and refined based on the outcome [15]. The basic idea is to determine the utility-value of an state-action pair relative to the other actions available, because the absolute values are not known. This means, $U : S \times A \times A \to [0, 1]$ is used instead of the Q-function $Q : S \times A \to \mathbb{R}$ common in classic Reinforcement learning. The implicit meaning is:

$$U(s, a, a') = \begin{cases} 1 & \textbf{if } Q(s, a) > Q(s, a') \\ 0 & \textbf{else} \end{cases}$$

In each iteration, a limited subset of all available states is evaluated, and its actions are compared in a pairwise manner by always sampling rollouts for two actions for the same state s. The evaluation determines if $Q(s, a) > Q(s, a')$ holds, which is the case if the sampled trajectory starting with (s, a) is preferred over the one starting with (s, a'). This is then a sample for the relative utility function and used as training information for a label ranking algorithm [11] with $\phi(s)$ as features. The optimal action is now determined by the predicted ranking of the actions.

This approach updates only a single action pair for a single state by each preference encountered, requiring a high amount of evaluations for convergence.

5.2 A Policy Iteration Algorithm for Learning from Preference-based Feedback

The algorithm by Wirth and Fürnkranz [18] is closely related to PBPI, but differs in some essential points. The main advantage lies in a possibility to re-use a preference feedback for updating multiple states, which is achieved by introducing a probability theorem that states the probability of a state-action selection being the cause for the observed preference. Additionally, a modified version of the EXP-3 [5] policy for exploration exploitation tradeoff was utilized for further improvements. However, these approaches are not directly comparable, because the PBPI algorithm assumes a parameterized state space, whereas Wirth and Fürnkranz [18] uses a simple tabular representation which is not suited for generalization over states, rendering the usage of a ranker infeasible.

5.3 Preference-based Policy Learning (PPL)

A well-known example of preference based policy search is the approach by Akrour et al. [2, 3]. They describe states by the features of the *sensori-motor-state (SMS)* of the underlying robotics agent system. An SMS is defined as the combination of the measured sensor values with the actuator values. For simplifying the problem, all encountered SMS are clustered, which reduces the amount of states in the MDP\R. This representation is closely related to feature expectations, as described in sec. 2.2, but not identical because of the clustering. The clustering results in ϕ being a dynamic function, due to the clustering only dependent on the *already observed* SMS. A clustering of all available SMS could be seen as discretization of the state space. In spite of this difference, we are still considering those clusters as tabular features in the form of eq. (2).

The first step of PPL is to determine the weight vector $\mathbf{w}$ of the scalar utility version of eq. (2). This can be achieved by solving eq. (13) with $T_i \succ T_j$ as the preference of trajectory T_i over trajectory T_j (with the trajectories being complete in this case). C_i and C_j are the state, action sequences of T_i and T_j. A preference is interpreted as constraint on the utility function.

$$\begin{aligned} &\min \frac{1}{2}||\mathbf{w}||^2 + c \sum_{i,j,C_i \succ C_j \in \zeta} \xi_{i,j} \\ &\text{s.t. } (\langle \mathbf{w}, \mu(C_i) \rangle - \langle \mathbf{w}, \mu(C_j) \rangle \geq 1 - \xi_{i,j} \text{ and } (\xi_{i,j} \geq) \text{ for all } C_i \succ C_j \in \zeta \end{aligned} \tag{13}$$

This formulation is conveniently the same optimization problem as solved by SVM^{Rank} [12], which is used to solve this problem. New policies are now generated using an evolutionary strategy, more specifically the $(1+\lambda)-ES$ algorithm by Auger [6]. For determining the value of an policy, its utility estimate $U(\pi)$ (sec. 2.2) is calculated summed up with an exploration term E. This trades off the expected utility, as measure for exploitation, with additional exploration.

Concerning details of the exploration function and how the tradeoff with the utility is realised, we want to point the interested reader to [2], because it is not essential for understanding the approach.

It should be noted, the search is not performed in the policy space directly, but in the feature space ϕ. The optimization is preformed for determining $\mathbf{w}^*$, which is dependent on the feature space, and only the resulting utility function is used for evaluating the policies.

6 Relation-based Algorithms

Relation-based approaches compare policies, without determining a utility. Considering that a policy can be described by the set of all trajectories it can generate (or the probability distribution over the trajectories in the stochastic case), we can use trajectories as samples for the policy. Hence, it is sufficient to compare the trajectories created by different policies to determine an approximation of their relation.

The main difference between the algorithms in this class is how the policy relation is defined relative to the observed samples of the preference relations and how to create new policies based on this information. It should be noted, all algorithms presented in this class are requiring parameterized policies, because this enables generalization of policy relations to the complete policy space.

6.1 Preference-based Evolutionary Direct Policy Search

The algorithm of Busa-Fekete et al. [7] is closely related to the work of Akrour et al. [2]. It also employs an evolutionary strategy for optimizing the parameters of a parametric policy, but CMA-ES [9] was used in this case. The main difference lies in the evaluation of the policies, which is performed directly in policy space. Each candidate policy of the current iteration is used to sample a limited amount of trajectories. The pairwise preference relation is now used to estimate how often π_i yields a trajectory that is preferred over a trajectory produced by π_j. Using a racing algorithm witch utilizes Hoeffeding bounds enables the determination of a ranking for the policies based on the fraction of dominating trajectories [10]. This ranking is then use within the CMA-ES framework to create new policies.

6.2 Bayesian Preference Learning from Trajectory Preference Queries

Wilson et al. [17] tries to learn the distribution of the policy space, according to the expert preferences. The sequences C are limited to trajectories of length K, with K typically much smaller than the horizon. The posterior distribution of the policy space (6.2) is defined by the expected distance between the observed trajectories and an ones created by an expert policy. This means $C_i \succ C_j \Leftrightarrow f^*(C_i, C_j, \pi^*) < 0$ with the comparison function

$$f^*(C_i, C_j, \pi^*) = \mathbb{E}[d(C_i, C^*)] - \mathbb{E}[d(C_j, C^*)] \tag{14}$$

where C^* is a random trajectory generated by π^*. The trajectory distance function d is defined utilizing the real-valued vector space $\phi(s, a)$:

$$d(C_i, C_j) = \sum_{t=0}^{K} ||\phi((s,a)_{i,t}) - \phi((s,a)_{j,t})||$$

Hence, eq. (14) can be seen as the expected difference of the undiscounted feature expectations between C_i and an optimal trajectory C^*.

It is now possible to determine the (expected) policy distribution, using Bayes theorem:

$$\Pr(\pi^*|\zeta) \propto \Pr(\pi^*) \prod_{\zeta} \Pr(C_i \succ C_j|\pi^*)^{\mathbb{I}(C_i \succ C_j)} (1 - \Pr(C_i \succ C_j|\pi^*))^{1-\mathbb{I}(C_j \succ C_j)}$$

$$\Pr(C_i \succ C_j|(C_i, C_j), \pi^*) = \int_{-\infty}^{\infty} \mathbb{I}(f^*(C_i, C_j, \pi^*) > \epsilon) N(\epsilon|0, \sigma_r^2) d\epsilon$$

$\Pr(C_i \succ C_j|\pi^*)$ denotes the probability of $C_i \succ C_j$ concerning samples of expert policy π^*.

The posterior distribution is approximated using Hybrid Monte Carlo [4], which is a form of Markov Chain Monte Carlo. It should be noted that the experiments have been performed with explicit knowledge of π^* for determining the expert feedback.

7 The Preference Problem Cycle

An interesting question concerns the relation between reinforcement learning (RL), preference-based reinforcement learning (PBRL), inverse reinforcement learning (IRL), and apprenticeship learning. IRL is about learning the reward function of the MDP from demonstrated solutions [13]. Apprenticeship Learning tries to directly recover the policy that was used to create the demonstrated solutions, meaning the demonstrations are assumed to be (quasi-) optimal [1]. This is usually achieved by applying IRL for learning the reward function which allows the application of classic RL algorithms for determining the policy, but this is can also be achieved in other ways. Demonstrated solutions can be seen as an implicit preference with the meaning of "the demonstrated solutions are preferred over all others", turning those approaches into preference-based methods. The main difference is that PBRL does not assume that solutions are demonstrated,

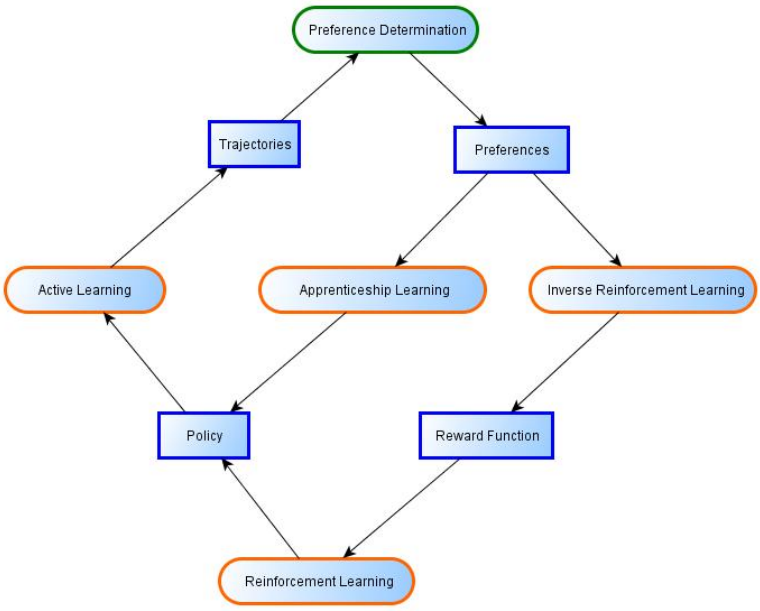

Fig. 1. The Preference Problem Cycle

but that the training information is generated with some sampling policy. This mainly concerns the exploration-/exploitation tradeoff [15] that has to be considered in reinforcement learning, but can also be seen as a case of Active Learning [14]. Together with the preference feedback, determined by some kind of oracle, this results in a cycle, as show in figure 1. The edge labels are describing the output of the last element, which is also the input for the next part of the cycle, together with the globally available information, like the MDP\R. Currently all PBRL publications, known to the authors, are trying to find a solution for every subtask of the cycle, but this is arguably not required. Due to the possibility of modularizing the cycle, we can identify the subtask which are not solvable with already existing methods. In fact, the only new problems are preference-based inverse reinforcement learning and the preference-based apprenticeship learning. Determining an optimal policy according to a reward function can be solved by utilizing classic reinforcement learning. Methods for sampling new trajectories, which are beneficial for the learning process, are also existent but they can probably be improved for preference based algorithms. It should also be noted, that the relation-based methods from section 6 are an instance of preference-based apprenticeship learning, while learning a utility function can be seen as preference-based inverse reinforcement learning.

8 Conclusion

In our view, PBRL can be seen as an algorithm that is in between conventional reinforcement learning and apprenticeship learning. The key difference is that we are not shown optimal (or at least desirable) trajectories as in apprenticeship learning or inverse reinforcement learning, but may also see suboptimal and undesirable policies like in traditional reinforcement learning. On the other hand, unlike in reinforcement learning, we cannot directly observe the utility of these policies but can only observe instances of preferences between trajectories, states or actions. These preferences can be modeled as preferences between state-action sequences, resulting in a generalized representation which unifies several different types of preferences. Other aspects like the definition of optimality still lack a unifying framework.

PBRL algorithms can be divided into two coarse categories: relation-based and value-based approaches where preferences are samples for a policy relation or the value of a trajectory or state-action pair. Both approaches have their own advantages and shortcomings. The most prominent question for relation-based algorithms is how trajectory or other preferences are raised to the level of policies. Value based methods are usually having difficulties to utilize the training information efficiently.

Acknowledgments

This work was supported by the German Research Foundation (DFG) as part of the Priority Programme 1527. We would like to thank Eye Hüllermeier and Robert Busa-Fekete for interesting and stimulating discussions that greatly influenced this paper.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21th international conference on Machine learning (ICML-04)*, page 1, 2004.

[2] Riad Akrour, Marc Schoenauer, and Michèle Sebag. Preference-based policy learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD-11)*, volume 6911 of *LNCS*, pages 12–27. Springer, 2011.

[3] Riad Akrour, Marc Schoenauer, and Michèle Sebag. APRIL: Active preference learning-based reinforcement learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD-12)*, volume 7524 of *LNCS*, pages 116–131. Springer, 2012.

[4] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[5] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-arm bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, 1995.

[6] Anne Auger. Convergence results for the $(1,\lambda)$-SA-ES using the theory of ϕ-irreducible markov chains. *Theoretical Computer Science*, 334(1–3):35 – 69, 2005.

[7] Robert Busa-Fekete, Balazs Szorenyi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. Preference-based evolutionary direct policy search. ICRA Workshop on Autonomous Learning, 2013.

[8] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89(1-2):123–156, 2012. Special Issue of Selected Papers from ECML PKDD 2011.

[9] Nikolaus Hansen and Stefan Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 282–291. Springer, 2004.

[10] Verena Heidrich-Meisner and Christian Igel. Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-09)*, pages 401–408. ACM, 2009.

[11] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172 (16–17):1897–1916, 2008.

[12] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)*, pages 133–142. ACM Press, 2002.

[13] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*, pages 101–103. ACM, 1998.

[14] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[15] Richard S. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

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

[17] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS-12)*, pages 1142–1150, 2012.

[18] Christian Wirth and Johannes Fürnkranz. A policy iteration algorithm for learning from preference-based feedback. In *Proceedings of the 12th International Symposium on Intelligent Data Analysis (IDA-13)*, London, England, October 2013. IOPress.