Learning from Trajectory-Based Action Preferences

Christian Wirth[1] and Johannes Fürnkranz[1]

***Abstract*— Conventional reinforcement learning algorithms depend on the availability of a numerical feedback signal. In many domains, this is not readily available and, in fact, constitutes an additional parameter of the problem setting. As a consequence, a fair amount of engineering is required in order to find a reasonable configuration of reward signal and algorithm parameters that facilitates an efficient solution of the problem. Preference-based reinforcement learning has been recently proposed to circumvent such problems by replacing the definition of quantitative reward signals with the observation of qualitative preferences over trajectories. In this paper, we propose a novel algorithm which converts preferences over trajectories into preferences over actions, with the goal of directly identifying preferred actions in a state. In addition, we also aim for a reduction of the number and the sensitivity of hyper-parameters, thereby reducing the required amount of experimentation that is needed for arriving at a good solution. The performance of this new approach is evaluated by two common reinforcement learning benchmark problems. The results are promising, showing convergence rates that are comparable to SARSA, but are more robust and obtained with considerably less demand on tuning the setup.**

I. Introduction

Most common methods for reinforcement learning are using feedback in the form of numerical rewards. This form of feedback is very informative, which eases the learning process, but it is often not easy to define. Numeric reward definition often requires detailed knowledge of the task at hand. In some domains, there is no natural choice, so that certain outcomes have to be associated with arbitrary reward signals. As an example, consider the cancer therapy domain studied by Zhao *et al.* [11], in which a negative reward of -60 has been assigned to the death of the patient in a medical treatment. Even for classical benchmark problems such as the inverted pendulum or the mountain car, there are many choices for modeling the reinforcement signal. The most natural choice, giving a positive reinforcement if the problem has been solved (i.e., the car gets up the mountain) is not sufficient because solution length is a crucial factor (the quicker the car manages to get up the mountain, the better). Thus, one has to use step penalties, trace decay or a reinforcement signal that depends on solution length. Again, these choices result in different results and convergence speeds. Moreover, these choices interact with the definition of features representing a state, possibly the parameterizable policy, as well as the parametrization for the learning algorithm itself. Our main goal is to overcome those problems for easing the use of reinforcement learning and to expand its applicability to domains where numerical feedback is not readily available.

The growing field of preference learning [6] enables an intuitive form of feedback, which does not require numerical values. For reinforcement learning, preference-based feedback signals can be modeled as pairwise comparisons of trajectories [7]. This form of feedback does not require detailed knowledge about the problem domain and can often be given by non-expert users. We also only expect feedback based on complete trajectories. For keeping the process as simple as possible, states are only represented in a tabular way and the policy learned is not parametrized. Additionally, we are considering approaches requiring only a minimal amount of hyper-parameters for tuning the algorithm.

II. Problem Definition

In this paper, we formally define a problem setting, which we call *preference-based sequential decision processes* (PSDP). While this setting shares many similarities with Markov decision processes, the basic scenario in which most reinforcement learning algorithms operate, there are some important differences, most notably the absence of a numerical reward signal.

A PSDP $\{S, A, \delta, \succ\}$ is defined by a state space $S = \{s_i\}, i = 1 \ldots |S|$, a finite action space $A = \{a_j\}, j = 1 \ldots |A|$, a stochastic state transition function $\delta : S \times A \times S \rightarrow [0, 1]$, and a preference relation $\succ$ that is defined over trajectories through this state space. Each state s is associated with a set of available actions $A(s)$. A trajectory is a state/action sequence $T = (s_0, a_0, s_1, ..., a_{n-1}, s_n)$ where $s_i \in S, a_i \in A(s_i)$, and $s_n \in S^F$ is an absorbing (final) state. A state is an absorbing state, if there are no actions available, i.e., the set of all absorbing states is $S^F = \{s \in S | A(s) = \emptyset\}$. A (stochastic) policy π is a probability distribution over the actions in each states, i.e., $\pi : S \times A \leftarrow [0, 1]$, where $\sum_{a' \in A(s)} \pi(s, a') = 1$.

The associated learning problems assumes that the learner is able to observe a set P of trajectory preferences, i.e., $P = \{T_i^1 \succ T_i^2\}, i = 1 \ldots |P|$. The goal of the learner is to find a policy, which respects these observed preferences.

III. Learning from Trajectory Preferences

Our approach is mainly based on the idea of identifying action preferences within the given trajectory preferences. For this, we are only comparing pairs of trajectories starting in identical states, because other comparisons are not trivial. We are also limited to defining action preferences for states visited by both trajectories by assuming that, for an identical state that occurs in both trajectories, we prefer the

[1]Knowledge Engineering, Technische Universität Darmstadt, Germany {cwirth, juffi}@ke.tu-darmstadt.de

action encountered in the preferred trajectory over the one encountered in the other trajectory. For keeping track of the amount of received feedback, we maintain a table A^s for each state. Whenever we observe a state-action pair (s, a) in a trajectory T^1, and the same state s followed by a different action a' in a trajectory A', we perform the update

$$A^s(a, a') = A^s(a, a') + 1 \quad \textbf{if } T^1 \succ T^2$$
$$A^s(a', a) = A^s(a', a) + 1 \quad \textbf{if } T^2 \succ T^1$$

Based on the counts $A^s(a, a')$, we can create an index function, ranking the available actions according to the amount of evidence preferring the according action.

$$\kappa_s(a) = \frac{1}{|A(s)| - 1} \sum_{a' \in A(s)} \left(\frac{A^s(a, a') - A^s(a', a)}{A^s(a, a') + A^s(a', a)} + 1 \right)$$

A higher κ_s value means a higher amount of evidence, resulting in a higher rank. Our estimated optimal policy is now $\tilde{\pi}^*(s, a) = \max(\kappa_s(a)), a \in A(s))$.

IV. Generating Trajectory Preferences

For a preferred trajectory to be based on preferable actions, we need to create solutions based on mostly (approximate) optimal decisions, because a solution based on random decision could be better by pure chance, invalidating the correlation to preferable actions. By acting greedily, we employ the best approximation of the optimal policy currently known. Additionally, we want to maximize the number of states visited by both trajectories for increasing the amount of action preferences created. Assuming that all (approximate) optimal solutions are within the same part of the state space, is it possible to increase the amount overlapping states (visited by both trajectories) by acting greedily. Hence, we are acting greedily in all states that can not be used to create an action preference, meaning all states not sampled by trajectories from the current start state.

For solving the related exploration/exploitation dilemma [9], several solutions have been proposed, but the EXP3 action-selection policy [3, 2] is especially interesting, because it is designed for adversarial bandit problems. Assuming stochastic bandits would be another another constraint that has to be validated by the engineer, hence we decided to turn to an adversarial bandit policy for weakening this requirement. The policy was modified to be applicable to our preference scenario and $\eta \in [0, 1/|A|)$ was turned into $\eta \in [0, 1]$, replacing all occurrences by $\frac{\eta}{|A|}$.

$$g^s(a, a') = g^s(a, a') + \frac{1}{EXP3(s, a)} \textbf{if } T^1 \succ T^2$$
$$g^s(a', a) = g^s(a', a) + \frac{1}{EXP3(s, a)} \textbf{if } T^2 \succ T^1$$
$$\tilde{G}_{s,a} = \frac{1}{|A(s)| - 1} \sum_{a' \in A(s)} \left(\frac{g^s(a, a') - g^s(a', a)}{g^s(a, a') + g^s(a', a)} + 1 \right)$$
$$\text{EXP3}(s, a) = (1 - \eta) \frac{\exp(\frac{\eta}{|A|} \tilde{G}_{s,a})}{\sum\limits_{b \in A} \exp(\frac{\eta}{|A|} \tilde{G}_{s,b})} + \frac{\eta}{|A|}$$

This action selector is used for all overlapping states.

We realized two versions of our algorithm, which differ in the way they generate trajectories for training data. The first version, *Unpaired*, does not assume control over the start state s_0, but simply starts in a random state of the set S_0. This is a general version of the algorithm because it can also be applied to problems where it is not possible to recreate states exactly, e.g. manual robotics setups. On the other hand, many other problems allow the exact selection of a start state, like problems that can be solved based on simulations. This allows us to design a different version of the algorithm, *Paired*, which allows to generate M sample trajectories from the same start state at once.

V. Experiments

We are comparing our algorithm to SARSA(λ) [9] in two testing domains, because it is a well known, widely used, but not specialized algorithm. It is required to decrease the SARSA ϵ parameter over time for guaranteeing convergence [8], hence we decided to use an ϵ_n-greedy decay scheme ($\epsilon_n = \min\{1, \frac{c|A|}{d^2 t}\}$), presenting a generic solution to the problem [4]. For keeping our comparison as fair as possible, we are only using terminal rewards, and are setting the step reward to 0. This is in-line with our preference approach, which is also only having access to feedback based on complete solutions (pairs). It should be noted, that even terminal-only numeric rewards are still much more informative than preferences, because they define a graded relation to all other solutions, as opposed to a singular, binary preference relation. Reward-based feedback is also directly available after calculating a single solution, while preferences are requiring a second solution.

Our first testing domain is the well known mountain car problem, parametrized as defined by Sutton and Barto [9]. The state is defined by creating a tabular representation based on a 40 equal-width bins discretization of the position and 20 bins for the velocity. For the SARSA comparison, we are using a terminal reward of 1 and a step reward of 0. SARSA was run in two settings: With $\lambda = 0.\bar{9}$ (exact: $1 - 10^{-15}$), because our own algorithm is also not using any decay and with an optimized λ value for showing a comparison to the best possible setting. The horizon was set to 500.

As a second testing domain are we using the inverted pendulum problem. The parametrization is mostly the same as defined by Dimitrakakis et al. [5], namely $F = -50N/0N/50N \pm 10, m = 2kg, M = 8kg, l = 0.5m, \tau = 0.05, horizon = 1000$. The discretization was performed with 10 equal-width bins each for θ and $\dot{\theta}$. We are also comparing to a decaying ($\lambda < 1$) and a non-decaying ($\lambda = 0.\bar{9}$) SARSA configuration. The terminal feedback was set to a penalty of -1 for the decaying experiment and to a *horizon-stepcount* reward for the non-decaying version, because we need to defined a reward (or decay) which allows for distinguishing solutions by length.

The optimal hyper-parameter set for the algorithms itself have been determined by 1296 (6^4) grid search trials for SARSA and only 50 for our own approach, because of

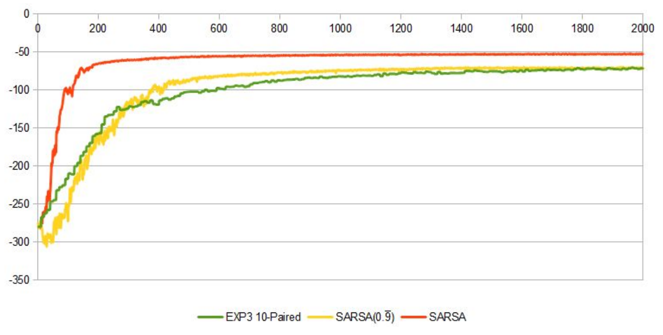

Fig. 1. Mountain Car: Average step-penalty for the best results compared with SARSA for all 2000 episodes

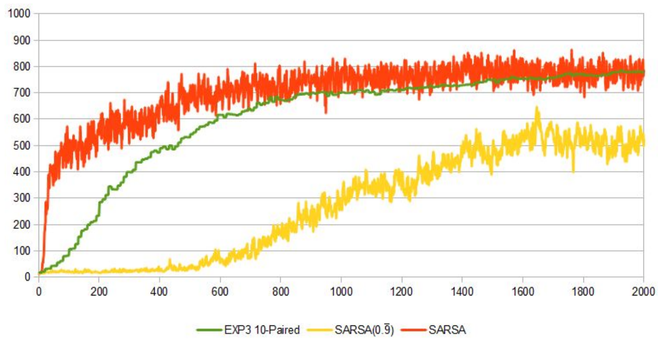

Fig. 2. Inverted Pendulum: Average step-penalty for the best results compared with SARSA for all 2000 episodes

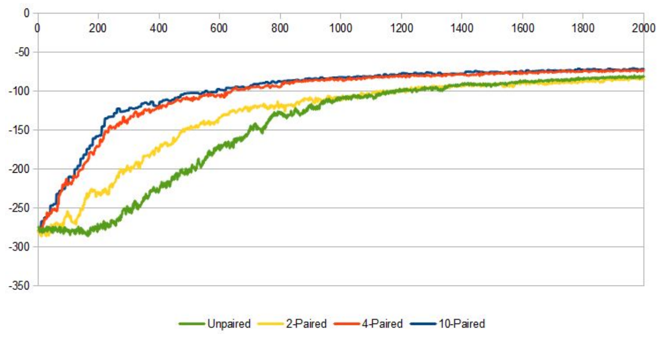

Fig. 3. Mountain Car: Average step-penalty for EXP-3 with different pairing for all 2000 episodes

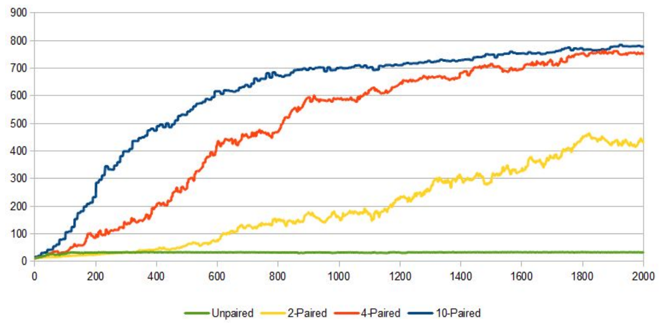

Fig. 4. Inverted Pendulum: Average step-penalty for EXP-3 with different pairing for all 2000 episodes

the lower parameter count. The values have been uniformly selected within the observed min/max values of the 20 best trials out of 250 random trials for SARSA, with the random trials sampling within given $[0, 1]$ ranges, except for $\epsilon.c$ which was uniformly sampled within $[1, 50]$. For reducing the amount of parameters further, we also set λ to the best value encountered in the random trials for the grid search trials. This two step approach was chosen for reducing the amount of required experiments without using any assumptions. Nevertheless, it was necessary to reduce the amount of trials for the SARSA(λ) inverted pendulum configuration because of time constraints. The amount of grid search trials was reduced to 81 (3^4). A single trial is an average over 100 repeated runs in all cases.

VI. Results

Figure 1 show the results for the best configuration found for SARSA, SARSA($0.\bar{9}$), and EXP3 in the mountain car domain. The best result was picked out of all available results by comparing the terminal policy quality. Note that the values reported here are negative, because of the step-penalty, as opposed to the step-reward used in the inverted pendulum problem. We can see that preference-based reinforcement learning converges more slowly than SARSA for the first steps, but that was to be expected due to the lower information content available within the feedback for each solution.

The results for the inverted pendulum, shown in Figure 2, are similar.

Figures 3 and 4 show comparisons between the different strategies for generating trajectory preferences (*Paired* with $M = 2$, 4, and 10 pairings, and *Unpaired*). We can see that requesting multiple states at once is beneficial for the result. In particular, we are not losing convergence speed, despite the lower amount of starting states sampled. The faster early convergence of higher pairings can be explained by more information earlier on, because we do not need to wait for a re-occurrence of the same start state, like in the unpaired scenario, or at least we can identify more action preferences before considering a new start state when using higher pairings. It should be noted, that, for the inverted pendulum, the unpaired approach was not converging towards the optimal solution within the given episode count.

Concerning the parameter sensitivity, we evaluated the standard deviation and average quality of the different approaches, considering the area under the learning curve LCA, which is the sum of rewards/penalties over all episodes, and the terminal policy quality t (Table I). The values calculated for our own approach are over all 50 trials, while excluding the first 250 random search trials for the SARSA test. This should exclude probably useless values and be more representative for the standard deviation encountered by domain experts setting up the algorithm. It becomes clear, that we are not only having less parameters to consider, but the result is also less sensitive to tuning them. The parameter EXP3-η has limited influence on the convergence speed of the algorithm, and only marginal effect on the terminal

TABLE I

STANDARD DEVIATION FOR ALL PARAMETERIZATIONS AFTER 2000 EPISODES FOR THE AREA UNDER THE LEARNING CURVE (LCA) AND THE TERMINAL STEPCOUNT (t)

Mountain Car				
	$\overline{LCA}$	$\sigma(LCA)$	$\bar{t}$	$\sigma(t)$
EXP3 Unpaired	-296k	3524	-81.7	1.1
EXP3 2-Paired	-261k	3704	-83.3	1.1
EXP3 4-Paired	-207k	2311	-72.9	0.8
EXP3 10-Paired	-201k	2163	-72.0	0.9
SARSA(λ)	**-143k**	8845	**-64.2**	5
SARSA(0.$\bar{9}$)	-286k	52978	-128	28.7

Inverted Pendulum				
	$\overline{LCA}$	$\sigma(LCA)$	$\bar{t}$	$\sigma(t)$
EXP3 Unpaired	59k	7912	30.3	4.1
EXP3 2-Paired	343k	26025	379.2	33.1
EXP3 4-Paired	999k	45907	744.4	31.2
EXP3 10-Paired	**1184k**	61907	**767.6**	37.4
SARSA(λ)	628k	470354	359.9	239.4
SARSA(0.$\bar{9}$)	274k	123760	194.7	105.1

result. This especially noticeable in the mountain car domain. SARSA, on the other hand, is very sensitive to parameter tuning. Not only is the standard deviation much higher if compared to our novel approach, but the difference between the best results (Figures 1 and 2) and the average result is quite substantial. The SARSA(λ) results for the inverted pendulum should only be considered an approximation, because of the low experiment count (250+81). Especially the standard deviation data is not representative, but was included to show how difficult a useful parametrization is in this domain.

VII. Related Work

The work of Fürnkranz *et al.* [7] has the closest relation to our work. They are also identifying action preferences, but based on a roll-out sampling method. This approach is exploiting less of the available information, because each preference is only used to update a single state. Additionally, they are using a non-tabular state representation, enabling policy generalization to unseen states. The approach of Akrour *et al.* [1] is also related, but they investigate parameterizable policies in continuous state spaces. Besides optimizing a policy they are also working on reducing the amount of preference feedback required, because they are considering problems which can not be easily simulated, requiring a human evaluation. Their approach combines preference learning with active learning. Weng [10] is also trying to relax the requirements for reinforcement learning feedback by introducing ordinal rewards. Ordinal rewards are still more restrictive than preferences due to the requirement for an evaluation on an absolute scale. But the main difference is in the approach itself, because Weng is still trying to map the ordinal rewards to numeric values while we are trying to find a solution that does not assume numerical rewards.

To the authors' knowledge this is the first work trying to directly identify action preferences within trajectory preferences.

VIII. Conclusion

Our results show that preference-based reinforcement learning can offer a performance comparable to SARSA with a fraction of the hyper-parameters and a simpler feedback structure, which substantially decreases the amount of work required for a successful application. Additionally, it should be noted that suboptimal parameterizations for our algorithm are usually also leading to near optimal results, as opposed to the high variance encountered by our SARSA experiments.

The amount of pairings that should be requested can also be determined by the problem domain itself. Our results suggest, that higher amount of pairings are usually beneficial (up to 10), but this also requires a higher amount of preference feedback, which could be a problem in domains requiring a human evaluation of trajectories.

References

[1] R. Akrour, M. Schoenauer, and M. 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)*, Bristol, UK, pp. 116–131, 2012.

[2] J.-Y. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *Proceedings of the 22nd Conference on Learning Theory (COLT-09)*, Montreal, Canada, 2009.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. 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*, pp. 322–331. IEEE Computer Society Press, 1995.

[4] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[5] C. Dimitrakakis and M. G. Lagoudakis. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, 2008.

[6] J. Fürnkranz and E. Hüllermeier (eds.) *Preference Learning*. Springer-Verlag, 2010.

[7] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89(1-2):123–156, 2012.

[8] S. P. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

[9] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[10] P. Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, pp. 282–289, Freiburg, Germany. AAAI Press, 2011.

[11] Y. Zhao, M. Kosorok, and D. Zeng. Reinforcement learning design for cancer clinical trials. *Statistics in Medicine*, 28:3295–3315, 2009.