Model-Free Preference-based Reinforcement Learning

Christian Wirth and **Johannes Fürnkranz** and **Gerhard Neumann**
Technische Universität Darmstadt, Germany

Abstract

Specifying a numeric reward function for reinforcement learning typically requires a lot of hand-tuning from a human expert. In contrast, preference-based reinforcement learning (PBRL) utilizes only pairwise comparisons between trajectories as a feedback signal, which are often more intuitive to specify. Currently available approaches to PBRL for control problems with continuous state/action spaces require a known or estimated model, which is often not available and hard to learn. In this paper, we integrate preference-based estimation of the reward function into a model-free reinforcement learning (RL) algorithm, resulting in a model-free PBRL algorithm. Our new algorithm is based on Relative Entropy Policy Search (REPS), enabling us to utilize stochastic policies and to directly control the greediness of the policy update. REPS decreases exploration of the policy slowly by limiting the relative entropy of the policy update, which ensures that the algorithm is provided with a versatile set of trajectories, and consequently with informative preferences. The preference-based estimation is computed using a sample-based Bayesian method, which can also estimate the uncertainty of the utility. Additionally, we also compare to a linear solvable approximation, based on inverse RL. We show that both approaches perform favourably to the current state-of-the-art. The overall result is an algorithm that can learn non-parametric continuous action policies from a small number of preferences.

Introduction

One major limitation of reinforcement learning is that a numeric reward function needs to be specified by the user. This is particularly true for complex control tasks as they occur in robotics where the reward function often consists of several hand-tuned terms. Hence, in recent years, the community has worked on rendering reinforcement learning algorithms more applicable by avoiding a hand-coded definition of the reward function. One of these approaches is *preference-based reinforcement learning (PBRL)*. PBRL uses only pairwise preferences over policies, trajectories, states or actions (Akrour, Schoenauer, and Sebag 2012; Wirth and Frnkranz 2013b; Wilson, Fern, and Tadepalli 2012). Many PBRL approaches rely on a model of the system dynamics (Akrour et al. 2014), but often, accurate models are not available and are also hard to learn. Hence, a model-free approach is desirable. Additionally, directed exploration of the utility function is used often, which is hard to perform if the model is unknown or the state-action space is continuous and possibly high-dimensional.

In this paper, we show an algorithm for learning a continuous action policy without requiring knowledge of the model or maintaining an explicit approximation of it. The preferences are used to estimate the expert's utility function. In contrast to traditional PBRL methods, our method is able to use data from interactions with the environment in an online fashion to improve the policy as well as the estimate of the utility function. It can, nevertheless, achieve better performance. The utility function is learned using a Bayesian approach, estimating the uncertainty. An approach that could be used for exploration. Additionally, we compare with a computationally less expensive linear approximation based on ideas of Ng and Russell (2000).

Preliminaries

MDP\R. Abbeel and Ng (2004) have introduced the notion of *Markov decision processes without rewards* (MDP\R). A MDP\R is defined by a quadruple (S, A, δ, γ). We are given *states* $S \subseteq \mathbb{R}^{D_S}$ and *actions* $A \subseteq \mathbb{R}^{D_A}$, represented by feature vectors $\boldsymbol{\phi}(s) \in \mathbb{R}^{D_{\boldsymbol{\phi}(s)}}$ and $\boldsymbol{\varphi}(s, a) \in \mathbb{R}^{D_{\boldsymbol{\varphi}(s,a)}}$. The *state transition function* $\delta(s'|s, a)$ is assumed to be probabilistic. The parameter $\gamma \in [0, 1)$ is the discount factor. A *policy* $\pi(a|s)$ is a distribution that assigns probabilities to actions choices based on the current state. A *trajectory* is an alternating sequence of states and actions $\boldsymbol{\tau} = \{s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n\}$. 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.

Feature Expectations. A *feature expectation* is the (discounted) sum of features, expected to be realized by a certain policy or trajectory (Puterman 2005; Abbeel and Ng 2004). The feature averages of a trajectory $\boldsymbol{\tau}_i$ are

$$\boldsymbol{\psi}(\boldsymbol{\tau}_i) = \sum_{t=0}^{|\boldsymbol{\tau}_i|} \gamma^t \boldsymbol{\phi}(s_{t,i}), \qquad (1)$$

with t as the time-step. The feature expectations

$$\boldsymbol{\psi}(\pi) = \mathbb{E}\left(\sum_{t=0}^{|\boldsymbol{\tau}|} \gamma^t \boldsymbol{\phi}(s_t)\right), \tag{2}$$

for a policy are then the features of the trajectories, expected to be realized by the policy.

The Preference Case. Instead of using rewards as evaluative feedback, we assume to have access to trajectory preferences in the form $\boldsymbol{\tau}_i \succ \boldsymbol{\tau}_j$ where we define ζ as the set of all observed preferences. The preferred and dominated trajectories of the k-th preference are also denoted as $\boldsymbol{\tau}^{\succ_k}$ and $\boldsymbol{\tau}^{\prec_k}$. We want to find the policy that maximizes the realization probability for the set of undominated trajectories. As it is difficult to determine if a trajectory is dominated, without knowledge of all possible preferences, a reasonable approach is to compute a policy maximizing the probability for the currently dominating trajectories. Therefore, we want to find a policy

$$\arg\max_{\pi} \sum_{k} P^{\pi}(\boldsymbol{\tau}^{\succ_k}) - P^{\pi}(\boldsymbol{\tau}^{\prec_k}), \tag{3}$$

with $P^{\pi}(\boldsymbol{\tau})$ as the probability that policy π will realize trajectory $\boldsymbol{\tau}$. However, our policy also needs to generate new trajectories that are possibly dominating the currently undominated trajectories. Hence, our policy needs to explore new, significantly different trajectories that can be used to create informative preferences.

Online Preference-Based Reinforcement Learning

Our approach for solving the given problem is a four-step cycle:

1. approximation of the expert's utility function,
2. policy improvement
3. collection of new trajectories and
4. requesting new preferences.

The first step can be solved by viewing the preferences as constraints over the space of possible reward or utility functions. It is termed a utility function, because it is subject to drift induced by the changing set of observed preferences, but it is not used to model risk or threshold effects. Based on the given utility, we can use RL methods to compute a new sampling policy that improves on the expected return. The new policy is then used to compute new trajectories for requesting new preferences as additional transition samples for the reinforcement learning step. By comparing trajectories from this new policy with our current set of undominated trajectories, we gain access to new preferences, possibly improving the utility function estimate in the next iteration of the loop.

For determining the expert's utility function (step 1), we use *preference-based inverse reinforcement learning* (PBIRL), an approach derived from the ideas of (Akrour, Schoenauer, and Sebag 2012; Akrour et al. 2014), because they are among the most efficient PBRL algorithms currently known. Our approach differs in the definition of the optimization problem as well as how the utility function is used. The PBIRL algorithm can be directly used within a RL algorithm: PBIRL is used to estimate the utility function while the RL algorithm is used to perform a policy improvement step.

For step 2, we propose a new algorithm called *actor critic relative entropy policy search* (AC-REPS), which allows us to directly control the exploration-exploitation trade-off by bounding the relative entropy between the old and the new policy. Traditional exploration approaches such as SoftMax or ϵ-greedy action selection can control this trade-off only indirectly and are therefore much harder to tune. Bounding the relative entropy is a well-known strategy in policy search (Peters, Mülling, and Altun 2010).

Following the new stochastic policy, we can now sample a new set of trajectories that is potentially superior to the current set of undominated trajectories (step 3) and request new preference (step 4).

Details of this procedure and how we realise the elements of this cycle are given in the following sections.

Step 1: Preference-based Inverse Reinforcement Learning (PBIRL)

The problem of approximating the utility function (step 1) from preferences is closely related to *inverse reinforcement learning* (IRL). Both settings have access to predefined trajectories. In the IRL case, the trajectories are given by the expert and usually assumed to be near optimal, i.e. the shown trajectories are implicitly preferred over most unseen trajectories, defining implicit preferences. In the PBIRL case, the trajectories are generated by the algorithm itself with pairwise preferences requested from a human expert, i.e. both trajectories of a preference pair are explicitly known. Hence, both approaches can be formalized by similar algorithms that use the available preferences as constraints for the utility function that we want to estimate. We will first formulate a Bayesian version of the PBIRL problem, that is able to capture the uncertainty of the utility estimate. Subsequently, we also introduce a linear approximation, that is computational less demanding.

From IRL to PBIRL. Ng and Russell (2000) presented the first algorithm for IRL. It is based on the idea that the value of the demonstrated policy V^{π^*} should be higher than for any other policy V^{π_i}. They assume that the reward function $r(s) = \boldsymbol{w}^T\boldsymbol{\phi}(s)$ is linear in a given feature space. Due to this linearity, the resulting value V^{π} is also linear in the *feature expectations* $\boldsymbol{\psi}(\pi)$, i.e.,

$$\hat{V}^{\pi} = \boldsymbol{w}^T\boldsymbol{\psi}(\pi), \tag{4}$$

where $\psi(\pi)$ is defined in Eq. (2). The constraints $V^{\pi^*} > V^{\pi_i}$ now translate into linear constraints on $\boldsymbol{w}$, i.e, $\boldsymbol{w}^T\boldsymbol{\psi}(\pi^*) > \boldsymbol{w}^T\boldsymbol{\psi}(\pi_i)$.

IRL can be easily extended to the PBIRL case. First, we do not compare policies but trajectories, i.e. instead of using feature expectations over policies we use feature averages over trajectories $\boldsymbol{\psi}(\boldsymbol{\tau})$. Note that feature averages over trajectories do not require knowledge of the MDP dynamics, unlike the feature expectations of IRL.

We also don't have access to optimal trajectories $\boldsymbol{\tau}^{\pi^*}$, but only to pairwise feedback $\boldsymbol{\tau}^{\succ_k} \succ \boldsymbol{\tau}^{\prec_k}$. Therefore, we have to rewrite the objective of the constraints as $\boldsymbol{w}^T\boldsymbol{\psi}(\boldsymbol{\tau}^{\succ_k}) > \boldsymbol{w}^T\boldsymbol{\psi}(\boldsymbol{\tau}^{\prec_k})$. For ease of notation, we use

$$d(\boldsymbol{w}, k, \zeta) = \boldsymbol{w}^T\big(\boldsymbol{\psi}(\boldsymbol{\tau}^{\succ_k}) - \boldsymbol{\psi}(\boldsymbol{\tau}^{\prec_k})\big), \tag{5}$$

as the preference difference function from here on. We can now define an optimization problem, based on the mentioned constraints. Because of the binary feedback, we want to minimize the 0-1 loss

$$\min_{\boldsymbol{w}} \sum_{k=0}^{|\zeta|} \mathbb{I}\big(d(\boldsymbol{w}, k, \zeta) \leq 0\big). \tag{6}$$

The result of the optimization is a new realization of $\boldsymbol{w}$ that specifies the reward $r(s) = \boldsymbol{w}^T\boldsymbol{\phi}(s)$.

However, this formulation is subject to three problems: (i) it must be possible to approximate the utility with a linear function, (ii) the value difference of a preference can become arbitrary small, and (iii) the problem can have multiple solutions. For overcoming the first problem, we utilize a tabular model for the feature space in discrete domains. In this case, the linear function is not an approximation, but an exact representation (Geramifard et al. 2013). In continuous domains, our feature space consists of radial basis functions. To ensure a selection of basis functions that is able to capture the properties of the sampled parts of the feature space, we use a subset of observed state/action samples as centers.

Bayesian PBIRL. As mentioned, minimizing the 0-1 loss is not sufficient for determining a solution to the preference problem. To prevent arbitrary small differences and determine a single, best solution, we follow the approach of Ng and Russell (2000), also utilized in (Akrour, Schoenauer, and Sebag 2012; Akrour et al. 2014) and most other PBRL approaches. Besides minimizing the 0-1 loss, we also try to maximize the utility difference of the observed preferences. The optimization of 0-1 loss is computationally difficult, but can be performed with Bayesian approaches. This has the additional advantage of encapsuling uncertainty in the optimization problem, enabling handling of noisy preferences, as analysed by Akrour et al. (2014). We consider the problem of computing the weight vector $\boldsymbol{w}$ via the given preferences ζ, which can be solved by the Bayesian formulation:

$$\Pr(\boldsymbol{w}|\zeta) \propto \Pr(\boldsymbol{w}) \Pr(\zeta|\boldsymbol{w}). \tag{7}$$

For $\boldsymbol{w}$, we utilize a multivariate, Gaussian prior $N(\boldsymbol{w}|\boldsymbol{0}, \sigma^2\boldsymbol{I})$. To reduce the amount of free parameters, we fixed the covariance matrix to a uniformly scaled identity matrix. Our likelihood function

$$\begin{aligned}\Pr(\zeta|\boldsymbol{w}) = \prod_{k=0}^{|\zeta|} \Bigg(& \frac{|\zeta|-1}{|\zeta|} L_{01}(\zeta|\boldsymbol{w}, k) \\ & + \frac{1}{|\zeta|} L_{\text{sig}}(\zeta|\boldsymbol{w}, k)\Bigg), \qquad (8)\\ L_{01}(\zeta|\boldsymbol{w}, k) =& \mathbb{I}\big(d(\boldsymbol{w}, k, \zeta) > 0\big), \\ L_{\text{sig}}(\zeta|\boldsymbol{w}, k) =& \frac{1}{1+\exp(-m \cdot d(\boldsymbol{w}, k, \zeta))},\end{aligned}$$

is a combination of the 0-1 loss $L_{01}(\zeta|\boldsymbol{w}, k)$, rephrased as a maximization problem, and a sigmoid function $L_{\text{sig}}(\zeta|\boldsymbol{w}, k)$. The shape factor m defines the steepness of the sigmoid function.

The scaled sum guarantees that the 0-1 loss always dominates the sigmoidal distance term, hence it is always more beneficial to fulfill another preference than to increase the margins. Increasing the influence of the sigmoidal term would introduce a tradeoff between margin maximization and preference fulfillment. The proposed formulation is difficult to solve analytically, because of the multivariate, Gaussian prior and the indicator function required for the 0-1 loss. However, it is possible to employ *elliptic slice sampling*[1] (Murray, Adams, and MacKay 2010) for sampling from the posterior distribution.

For determining the utility functions weight vector, we consider a maximum likelihood and a posterior approach. As we only have access to samples, it is reasonable to utilize the most likely sample (maximum likelihood). Additionally, considering $r(s) = \int \Pr(\boldsymbol{w}|\zeta)\boldsymbol{w}^T\boldsymbol{\phi}(s) = \mu_{\boldsymbol{w}}^T\boldsymbol{\phi}(s)$, we also compare to the sample based mean of $\Pr(\boldsymbol{w}|\zeta)\boldsymbol{w}^T$.

Linear PBIRL. The formulation above is computationally expensive and subject to tuning the shape factor as well as the prior. Hence, we also present a computationally less demanding version, which can be solved via linear programming. To that end, we adapt the classic IRL algorithm (Ng and Russell 2000) to utilize preferences over trajectories. This comes at the cost of not being able to optimize the 0-1 loss, but only an approximation. The original version uses boundary constraints for each element of the linear weight vector $\mathbf{w}$ and turns the value constraints into soft constraints with a penalty function c:

$$\begin{aligned}&\max \sum_{i=1}^{k} c\Big(\boldsymbol{w}^T\big(\boldsymbol{\psi}(\pi^*) - \boldsymbol{\psi}(\pi_i)\big)\Big), \\ &s.t. |\boldsymbol{w}_j| \leq 1, i = 1, ..., d, \\ &c(x) = \begin{cases} x & \text{if } x > 0 \\ 2x & \text{else} \end{cases}.\end{aligned} \tag{9}$$

The function $c(\dot{})$ can also be phrased as an linear program, due to its similarity to the absolute value problem. For introducing the trajectory based preferences, we have to rewrite

[1] http://homepages.inf.ed.ac.uk/imurray2/pub/10ess/

the objective of the IRL problem given in Equation (9) as

$$\max \sum_{k=0}^{|\zeta|} c\Big(d(\boldsymbol{w}, k, \zeta)\Big). \tag{10}$$

The result is again a realisation of $\boldsymbol{w}$, specifying the utility function.

Step 2: Actor Critic Relative Entropy Policy Search

In order to perform a policy improvement step (step 2), we have to deal with the following requirements. We do not want to assume a known (or approximated) model of the MDP because such an assumption is limiting in many settings. Moreover, we want to be able to use our policy improvement step for continuous valued policies with a large, maybe even infinite number of parameters, such as a Gaussian process (GP; Rasmussen and Williams 2005).

We will resort to random exploration strategies that can be implemented by a stochastic policy $\pi(a|s)$. Therefore, we developed a new actor critic algorithm that permits the use of non-parametric policies such as GPs. As our algorithm is based on the relative entropy policy search (REPS) algorithm (Peters, Mülling, and Altun 2010), we will call our algorithm *actor critic relative entropy policy search* (AC-REPS).

Our algorithm consists of three steps, which are described in detail in the following sections:

1. Estimate the Q-function using the current estimate of the utility function.
2. Compute new sample probabilities that maximize the Q-values while staying close to the old policy, in terms of Kullback-Leibler(KL) distance. This approach limits the greediness of the new policy.
3. Fit a new policy to these probabilities.

Estimating the Q-Function. For estimating the Q-function, we reuse all the observed state transitions (s_i, a_i, s_i') from the environment. We first compute the new utility $u_i = \boldsymbol{w}^T \boldsymbol{\phi}(s)$ for all transitions. Subsequently, we generate new on-policy actions for all successor states in our data set, i.e., $a_i' \sim \pi(\cdot|s)$. Given these pre-processed transition data and a feature representation of the Q-function, i.e. $Q(s,a) = \boldsymbol{\varphi}(s,a)^T \boldsymbol{\theta}$, the parameter vector $\boldsymbol{\theta}$ of the Q-function can be estimated by the LSTD algorithm (Boyan 1999). To increase the robustness of LSTD, we use the regularization method presented by Hoffman et al. (2012) and include a bias term.

Actor Critic REPS. The policy update of actor critic REPS is inspired by the episodic REPS algorithm (Kupcsik et al. 2013). We want to find a policy $\pi(a|s)$ that optimizes the expected Q-value, but at the same time has a limited Kullback-Leibler(KL) distance to the old policy $q(a|s)$. We optimize over the joint state action distribution $p(s,a) = p(s)\pi(a|s)$ and require that the estimated state distribution $p(s)$ is the same as the state distribution $\mu(s)$ of the current policy, i.e., $p(s) = \mu(s), \forall s$. This set of constraints is implemented by matching feature averages of the distributions $p(s)$ and $\mu(s)$ (Daniel, Neumann, and Peters 2012), i.e. $\int p(s)\boldsymbol{\phi}(s)\, ds = \hat{\boldsymbol{\phi}}$, where $\hat{\boldsymbol{\phi}}$ is the average feature vector of all state samples. Summarizing all constraints, we obtain the following constraint optimization problem

$$\begin{aligned} &\arg\max_p \int p(s,a) Q(s,a)\, ds\, da, \\ &\quad \text{s.t. } \mathrm{KL}(p(s,a)\,||\, q(s,a)) \le \epsilon, \\ &\quad \int p(s)\boldsymbol{\phi}(s)\, ds = \hat{\boldsymbol{\phi}}, \quad \int p(s,a)\, ds\, da = 1, \end{aligned} \tag{11}$$

where $q(s,a) = \mu(s)q(a|s)$ is the current state action distribution. The constraint optimization problem can be solved in closed form by the method of Lagrangian multipliers and has the solution

$$p(s)\pi(a|s) \propto q(s,a) \exp\left(\frac{Q(s,a) - V(s)}{\eta}\right), \tag{12}$$

where $V(s) = \boldsymbol{v}^T \boldsymbol{\phi}(s)$ is a state dependent baseline. The parameters η and $\boldsymbol{v}$ are Lagrangian multipliers that can be obtained efficiently by minimizing the dual function $g(\eta, \boldsymbol{v})$ of the primal optimization problem

$$\begin{aligned} g(\eta, \boldsymbol{v}) =& \epsilon\eta + \boldsymbol{v}^T \hat{\boldsymbol{\phi}} + \\ & \eta \log \sum_i \frac{1}{N} \exp\left(\frac{Q(s_i,a_i) - \boldsymbol{v}^T \boldsymbol{\phi}(s_i)}{\eta}\right), \end{aligned} \tag{13}$$

where we already replaced the integrals with a sum over samples.

The optimization problem is similar to the one of the contextual REPS algorithm presented by Kupcsik et al. (2013), with the difference that we want to maximize the Q-values instead of the returns. For details of the derivation of the given equations, we refer to the above-mentioned papers and the survey (Deisenroth, Neumann, and Peters 2013).

Obtaining a new Exploration Policy. Effectively, the optimization problem given in the previous paragraph is only solvable given a finite set of state action pairs s_i, a_i and their corresponding Q-values Q_i. For these samples, we can obtain a desired probability $p(s_i, a_i) = p(s_i)\pi(a_i|s_i)$ from Eq. (12). Our goal is now to generalize this sample-based representation to the whole state-action space with a new parametric (or non-parametric) policy $\tilde{\pi}$. A standard approach to obtain a generalizing distribution $\tilde{\pi}$ from samples is to minimize the KL between π and $\tilde{\pi}$, i.e.,

$$\begin{aligned} &\mathbb{E}_s \left[\mathrm{KL}\left(\pi(a|s)\,||\,\tilde{\pi}(a|s)\right)\right] = \int p(s,a) \log \frac{\pi(a|s)}{\tilde{\pi}(a|s)}\, ds\, da \\ &\approx -\frac{1}{N}\sum_i \frac{p(s_i,a_i)}{q(s_i,a_i)} \log \tilde{\pi}(a_i|s_i) + \text{const} \\ &= -\frac{1}{N}\sum_i \exp\left(\frac{Q(s_i,a_i) - V(s_i)}{\eta}\right) \log \tilde{\pi}(a_i|s_i), \end{aligned} \tag{14}$$

where we replaced the integral by samples, which have been generated by our sampling distribution $q(s,a)$. Note that Eq. (14) is equivalent to the weighted negative log-likelihood of policy $\tilde{\pi}$ given the state action pairs s_i and a_i with weighting

$$w_i = \exp\left(\frac{Q(s_i, a_i) - V(s_i)}{\eta}\right).$$

Hence, minimizing the expected KL is equivalent to a weighted maximum likelihood (ML) estimate of $\tilde{\pi}$. Note that, for the AC-REPS algorithm it is sufficient to have access to samples from $q(s, a)$, which can be obtained by performing rollouts with the current policy. In case of discrete action spaces, this can be improved by creating samples for each action for each observed state, regardless of the observed action. These samples have then to be weighted by $q(a|s)$.

Weighted ML estimates can be obtained in closed form for many types of distributions. We will use a Gaussian Process (GP) policy. In order to keep the computation tractable, we adapt the weighted formulation of a GP given by Kober et al. (2010) to the sparse Gaussian process case (Snelson and Ghahramani 2006).

Steps 3 and 4: Computing Trajectories and Requesting Preferences

The new, stochastic REPS policy is now used to generate N new trajectories as additional transition samples (step 3). For requesting new preferences (step 4), we are interested in trajectories assumed to dominate the currently maintained set of undominated trajectories. Considering that we assume the updated utility estimate to be more accurate, we take M out of the N trajectories, with the highest expected utility, based on the new utility function. Each trajectory is iteratively compared with current best ones, replacing them if a domination preference was encountered.

Experiments

In our experiments, the learner does not have access to the true reward signal but is given preference feedback based on the undiscounted sum of rewards. All reported results are averaged over 20 trials, except the Acrobot domain where we had to reduce to 10. We always collect 10 trajectories per iteration and use a discount factor of $\gamma = 0.98$. We request 1 preference per iteration. For the first iteration, we request 2 preferences, because it is required to have a good starting point but, as we can not yet compute a utility estimate, we need to sample randomly. We stayed clear of tuning this for each experiment because it would significantly increase the workload for the expert in a real world scenario. For the elliptic slice sampling, we utilize $100k$ samples each for burn-in and evaluation. The LSTD regularization factors, the RBF bandwith and number of basis function, are only tuned once per domain, to determine a reasonable setup. The ϵ bound of AC-REPS, the sigmoid shape parameter m and the variance of the prior σ are manually tuned on the preference-based tasks. The GPs hyperparameter are automatically tuned, using a CMA-ES (Hansen and Ostermeier 2001) after each iteration. The optimization target was to minimize the training set error, so that no additional data was required.

Results

In the following graphs, the solid lines define the median while the shaded areas and the dashed lines show the 25% and 75% percentile of the policy value.

”LP” are the results obtained by the linear programming approximation while ”Bayes ML” and ”Bayes Mean” are defining the results obtained by the Bayesian approach, via maximum likelihood and posterior evaluation. ”PF” denotes the results obtained by Akrour et al. (2014) and ”EPMC” are the results of Wirth and Frnkranz (2013a).

Gridworld. As a first testing domain, we use the Gridworld defined by Akrour et al. (2014). For allowing a direct comparison to the PF algorithm, a tabular policy is used instead of the Gaussian process. The environment is not difficult from an RL point of view, but interesting for a preference learning scenario as the rewards for different fields are similar, making it difficult to reconstruct the true reward function. The dimensionality of $\phi(s)$ is 25 in this domain.

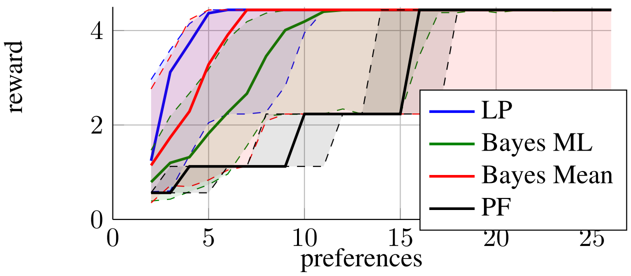

Figure 1: Gridworld task

The Gridworld results suggest that it is more beneficial to compute an exact solution for an approximation, than to use a more powerful but only sampled based Bayesian approach. Additionally, it can be seen that our approach is outperforming the algorithm by Akrour et al. (2014).

Bicycle. The second task is the bicycle balance task (Lagoudakis and Parr 2003), where we use continuous states and actions with a GP policy, with 400 RBF centers. We trained on episodes with 50 time-steps for 25 iterations. It should be noted, this setup is not directly comparable to the work of Akrour et al. (2014), because we do not require a generative model and use a Gaussian process policy instead of a neural network. In this domain (Fig. 2) are the LP and the Mean nearly on par, as the best approach, showing that complex Bayesian approaches are probably not required.

SwingUp. The third domain is the inverted pendulum swing-up task using continuous states and actions with a GP policy. The dimensionality of $\phi(s)$ is 700. Episodes have 60 time-steps, trained over 40 iterations. In Figure 3, we can see that LP is the fastest to converge to a plateau of approximately -1900, which all approaches are struggling to overcome. Over time, all approaches are converging, with the ML approach as the quickest. The LP approach is not among

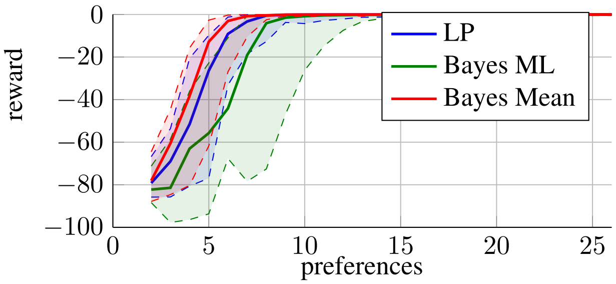

Figure 2: Bicycle task, reward evaluation

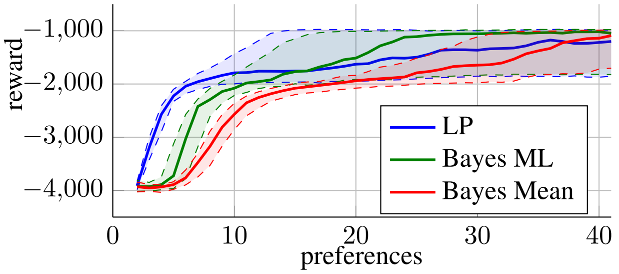

Figure 3: SwingUp task

the best in this domain, but still outperforms the Mean approach.

Acrobot. The last domain is the acrobot task (Sutton and Barto 1998) using continuous states and actions with a GP policy. The dimensionality of $\phi(s)$ is 300. Preferences are given by comparing the number of steps required to reach the terminal as in Wirth and Frnkranz (2013a). Episodes are created with 500 time-steps, trained over 15 iterations. This task is particularly difficult from a preference-learning point of view, because under a random policy only few trajectories will reach the terminal state within 500 steps. Therefore, we ensured that all runs encountered a preference within the first 3 iterations. As shown in Figure 4, we only computed the results for the LP approach, because of time constraints. The median policy value found after 15 iterations is -211.3. A substantial improvement over the -385.2 achieved by EPMC under the same setup.

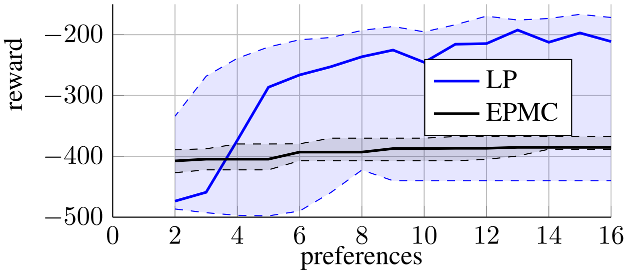

Figure 4: Acrobot task

To conclude, the LP approach is performing well in all domains. The Bayes Mean is able to outperform the LP in the Bicycle task, but converges worse in the SwingUp task.

Runtime. The Gurobi Solver[2] solved all linear programs in less than 0.1 sec. The elliptic slice sampler required $80 - 90$ sec. for sampling in the Gridworld domain and $100 - 150$ sec. in the three other domain. Even considering that it is probably possible to reduce the amount of samples with more tuning, a substantial difference will remain. It should be noted, the GP calculation dominates the required CPU time.

Conclusion

We have demonstrated that it is possible to use PBRL in an online manner, even in a non-parametric, model-free setting with continuous state action spaces. Moreover, our results show that complex directed exploration might be unnecessary. Random exploration is sufficient if the exploration/exploitation trade-off can be controlled efficiently. Our results compare favorably to the state-of-the-art which requires a generative model or an individual model learning phase. Future work will focus on coupling the exploration parameter ϵ with the confidence of the utility estimate to allow for more aggressive updates. Furthermore, the Bayesian PBIRL algorithm will be analysed with noisy preferences.

Acknowledgments

This work was supported by the German Research Foundation (DFG). This project has received additional funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No #645582 (RoMaNS). We also like to thank Eyke Hüllermeier and Robert Busa-Fekete for interesting discussions and Riad Akrour for supporting us in the comparison to his work.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine learning (ICML-04)*. ACM.

Akrour, R.; Schoenauer, M.; Sebag, M.; and Souplet, J.-C. 2014. Programming by Feedback. In *Proceedings of the 31nd International Conference on Machine Learning (ICML-14)*, number 32, 1503–1511. JMLR.org.

Akrour, R.; Schoenauer, M.; and Sebag, M. 2012. 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*, 116–131. Springer.

Boyan, J. 1999. Least-Squares Temporal Difference Learning. In *In Proceedings of the 16-th International Conference on Machine Learning (ICML-99)*, 49–56. Morgan Kaufmann.

Daniel, C.; Neumann, G.; and Peters, J. 2012. Hierarchical relative entropy policy search. In *Proceedings of the*

[2] http://www.gurobi.com/

15-th International Conference on Artificial Intelligence and Statistics (AISTATS-12), 273–281. JMLR.org.

Deisenroth, M. P.; Neumann, G.; and Peters, J. 2013. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics* 388–403.

Geramifard, A.; Walsh, T. J.; Tellex, S.; Roy, N.; and How, J. P. 2013. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Foundations and Trends in Machine Learning* 6(4):375–451.

Hansen, N., and Ostermeier, A. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2):159–195.

Hoffman, M.; Lazaric, A.; Ghavamzadeh, M.; and Munos, R. 2012. Regularized least squares temporal difference learning with nested l2 and l1 penalization. In Sanner, S., and Hutter, M., eds., *Recent Advances in Reinforcement Learning*, volume 7188 of *LNCS*. Springer. 102–114.

Kober, J.; Mulling, K.; Kroemer, O.; Lampert, C. H.; Schlkopf, B.; and Peters, J. 2010. Movement templates for learning of hitting and batting. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-10)*, 853–858.

Kupcsik, A.; Deisenroth, M.; Peters, J.; and Neumann, G. 2013. Data-Efficient Generalization of Robot Skills with Contextual Policy Search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-13)*.

Lagoudakis, M., and Parr, R. 2003. Least-Squares Policy Iteration. *Journal of Machine Learning Research (JMLR)* 4:1107–1149.

Murray, I.; Adams, R. P.; and MacKay, D. J. C. 2010. Elliptical slice sampling. *JMLR: W&CP* 9:541–548.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In Langley, P., ed., *Proceedings of the 17-th International Conference on Machine Learning (ICML-00)*, 663–670. Stanford, CA: Morgan Kaufmann.

Peters, J.; Mülling, K.; and Altun, Y. 2010. Relative Entropy Policy Search. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Puterman, M. L. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 2nd edition.

Rasmussen, C. E., and Williams, C. K. I. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Snelson, E., and Ghahramani, Z. 2006. Sparse gaussian processes using pseudo-inputs. In Weiss, Y.; Schlkopf, B.; and Platt, J. C., eds., *Advances in Neural Information Processing Systems 18 (NIPS-06)*. MIT Press. 1257–1264.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Boston, MA: MIT Press.

Wilson, A.; Fern, A.; and Tadepalli, P. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Advances in Neural Information Processing Systems 25 (NIPS-12)*. Curran Associates. 1142–1150.

Wirth, C., and Frnkranz, J. 2013a. EPMC: Every visit preference Monte Carlo for reinforcement learning. In *Proceedings of the 5th Asian Conference on Machine Learning, (ACML-13)*, volume 29 of *JMLR Proceedings*, 483–497. JMLR.org.

Wirth, C., and Frnkranz, J. 2013b. Preference-based reinforcement learning: A preliminary survey. In *Proceedings of the ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*.