# Model-Free Preference-based Reinforcement Learning

**Christian Wirth** cwirth@ke.tu-darmstadt.de
*Knowledge Engineering*
*Technische Universität Darmstadt, Germany*

**Gerhard Neumann** geri@robot-learning.de
*Computational Learning for Autonomous Systems*
*Technische Universität Darmstadt, Germany*

**Editor:** Alessandro Lazaric, Mohammad Ghavamzadeh, Remi Munos

## Abstract

Specifying a numeric reward function for reinforcement learning typically requires a lot of hand-tuning from an 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 for PBRL with non-parametric policies require a known or estimated model. 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. In a model-free setup, we need to use a random exploration strategy which is implemented by a stochastic policy. We show that, by controlling the greediness of the policy update, a comparable performance to commonly used directed exploration strategies, which require a model, can be achieved. Our new algorithm is based on the Relative Entropy Policy Search algorithm and can learn non-parametric continuous action policies from a small number of preferences.

## 1. Introduction

One major limitation of reinforcement learning is that a numeric reward function needs to be specified by the user. This reward function is often hard to design by hand and requires a lot of fine tuning. 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 et al., 2012; Wirth and Fürnkranz, 2013). Many PBRL approaches rely on a model of the system dynamics which they can use for directed exploration of the utility function. Yet, such a directed exploration is hard to perform if the model is unknown or the state-action space is continuous and possibly high dimensional.

In this paper, we learn a continuous action policy without requiring knowledge of the model or maintaining an explicit approximation of it. In contrast to traditional PBRL methods (Akrour et al., 2012; Wilson et al., 2012), our method is able utilize an online estimate of the model and can, nevertheless, achieve a comparable performance. We show that our RL formulation outperforms standard approximate policy iteration and that restrictions to the space of utility functions leads to further improvements.

### 1.1 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 $(S, A, \delta, \gamma)$. Given are *states* $S \subseteq \mathbb{R}^{D_S}$ and *actions* $A \subseteq \mathbb{R}^{D_A}$, represented by feature vectors $\boldsymbol{\phi}(s)$ and $\boldsymbol{\phi}(a)$. The combined feature space is denoted $\boldsymbol{\varphi}(s, a)$. The *state transition function* $\delta(s'|s, a)$ is assumed to be probabilistic. In contrast to the definition of Abbeel and Ng (2004), we utilize continuous state action spaces with dimensions $D_S \in \mathbb{N}$ and $D_A \in \mathbb{N}$. A *policy* $\pi(a|s)$ is a probabilistic function 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* $s_0$. The parameter $\gamma \in [0, 1)$ is the discount factor.

**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

$$\boldsymbol{\psi}(\boldsymbol{\tau}_i) = \sum_{t=0}^{|\boldsymbol{\tau}_i|} \gamma^t \boldsymbol{\varphi}(s, a)_{t,i}, \tag{1}$$

with $t$ as the time-step. The feature expectations

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

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

**The Preference Case.** As feedback, we assume trajectory preferences in the form $\boldsymbol{\tau}_i \succ \boldsymbol{\tau}_j$ where we define $\zeta$ as the set of all observed preferences. The preferred and dominated trajectory of the $k$-th preference is 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. This means, the problem is two-fold as we can not access preferences over all possible trajectories. On the one hand, we need 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 $\pi$ will realize trajectory $\boldsymbol{\tau}$. On the other hand, we also need to add new preferences to $\zeta$, concerning trajectories that are possibly dominating the current set of undominated trajectories.

**The Preference-Based Reinforcement Learning Cycle.** The main approach for solving the given problem is a four step loop of (1) approximating the experts utility function, (2) improving the policy, (3) collecting new trajectories and (4) requesting new preferences. The first step can be solved by viewing the preferences as constraints over the space of possible value functions. The resulting value function can be broken down into state-action utilities, comparable to rewards. 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. Details of this procedure and how we realise the elements of this loop are given in Sec. 2.

## 2. Online Preference Based Reinforcement Learning

For determining the experts utility function, we use Preference-based Inverse Reinforcement Learning (PBIRL) (see. 2.1). An approach derived from the ideas of Akrour et al. (2012, 2014), as they are among the most efficient PBRL algorithms currently known. But our approach differs in the definition of the optimization problem as well as how the utility function is used. The PBIRL algorithm can directly be 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. A new algorithm called Actor Critic Relative Entropy Policy Search (AC-REPS) allows us to directly control the exploration-exploitation trade-off by bounding the relative entropy between the old and the new policy, in contrast to indirectly controlled approaches like SoftMax or $\epsilon$-greedy. This is a well known strategy in policy search (Peters et al., 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. We sample a set of $N$ trajectories which are used for the next AC-REPS iteration and use a subset of $M << N$ trajectories to compare to the current set of undominated trajectories (see Sec. 1.1). The new estimate of the utility function is then fed back to the value function estimation algorithm, that produces a new Q-function for the policy update of AC-REPS.

### 2.1 Preference-based Inverse Reinforcement Learning (PBIRL)

The problem of reconstructing a utility function 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 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, meaning 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 the PBIRL problem and, subsequently, modify the baseline algorithm to use a more compact representation of the utility function. We will show that this extension can decrease the number of required preferences substantially.

**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 $V^{\pi^*}$ should be higher than for any other policy $V^{\pi^i}$. Ng et.al. assume a reward function representation $r(s) = \boldsymbol{\phi}(s)^T \boldsymbol{w}$ that is linear in a given feature space. Due to this linearity, the resulting value $V^\pi$ is also linear in the *feature expectations* $\boldsymbol{\psi}(\pi)$, i.e.,

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

The constraints $V^{\pi^*} \geq V^{\pi^i}$ now translate into linear constraints on $\boldsymbol{w}$, i.e, $\boldsymbol{w}^T\boldsymbol{\psi}(\pi^*) \geq \boldsymbol{w}^T\boldsymbol{\psi}(\pi^i)$. Together with a set of boundary constraints for each element of the linear weight vector $\mathbf{w}$ and turning the value constraints into soft constraints with a penalty function $c$, the original IRL optimization problem is given by

$$\begin{aligned} &\max \sum_{i=1}^{k} c\Big(\boldsymbol{w}^T\big(\boldsymbol{\psi}(\pi^*) - \boldsymbol{\psi}(\pi_i)\big)\Big), \\ &s.t. |\boldsymbol{w}_j| \leq 1, i = 1, ..., d, \\ &c(x) = \begin{cases} x & \text{if } x > 0 \\ 2x & \text{else} \end{cases}. \end{aligned} \tag{5}$$

The result of the optimization is a new realization of $\boldsymbol{w}$ that specifies the estimated reward function Equation 4.

IRL can be easily extended to the PBIRL case. First, we don't compare policies but trajectories, i.e., instead of using feature expectations over policies we use feature averages over trajectories $\boldsymbol{\psi}(\boldsymbol{\tau})$. Additionally, we don't have access to optimal trajectories $\boldsymbol{\tau}^{\pi^*}$, but only to pairwise feedback $\boldsymbol{\tau}_i \succ \boldsymbol{\tau}_j$. Therefore, we have to rewrite the objective of the IRL problem given in Equation 5 as

$$\max \sum_{k=0}^{|\zeta|} c\Big(\boldsymbol{w}^T\big(\boldsymbol{\psi}(\boldsymbol{\tau}^{\succ_k}) - \boldsymbol{\psi}(\boldsymbol{\tau}^{\prec_k})\big)\Big). \tag{6}$$

The obtained reward or utility function $r(s) = \boldsymbol{w}^T\boldsymbol{\phi}(s)$ is now maximizing the difference between the value of the preferred trajectories to the dominated trajectories.

**Compact Representation of the Utility Function.** In difference to standard IRL, we only have a limited set of trajectories available for the preferences. Yet, the feature space can be high dimensional, especially in continuous domains, which would require a large amount of preferences acquired from the human expert. To minimize the amount of required preference, we use a more compact representation of the utility function which scales with the number of available preferences.

Such requirement translates in the following formulation for the weight vector $\boldsymbol{w}$,

$$\boldsymbol{w} = \sum_{i=0}^{n} \alpha_i \boldsymbol{\psi}(\boldsymbol{\tau}_i), \quad |\alpha_i| \leq 1, \tag{7}$$

i.e., the resulting weight vector must be a linear combination of the feature expectations realized by the observed trajectories, with $n$ as the number of trajectories. The advantage of this method is demonstrated empirically.

**Reweighting Old Preferences.** We are mainly interested in realizing the preferences for the currently undominated set of trajectories. Hence, we down weight preferences which have been generated in previous iterations. The newest preferences are always related to the undominated set and, hence, they are the most likely dominating preferences. We use the following weight for each preference

$$w(\boldsymbol{\tau}^{\succ_k}, \boldsymbol{\tau}^{\prec_k}) = d^{(I - \max(I(\boldsymbol{\tau}^{\succ_k}), I(\boldsymbol{\tau}^{\prec_k})))}, \tag{8}$$

with $I$ as the current iteration number, $I(\boldsymbol{\tau})$ as the iteration number when trajectory $\boldsymbol{\tau}$ was generated and $d$ as a user defined parameter. The current weighting function is based on a heuristic that works well in practice.

### 2.2 Actor Critic Relative Entropy Policy Search

In order to perform a policy improvement step, we have to deal with the following requirements. We do not want to assume a known (or approximated) model of the MDP as this assumption is unrealistic 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 (Rasmussen and Williams, 2005). Finally, we do not want to use directed exploration strategies of the utility space as such strategies typically scale poorly. Hence, 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 et al., 2010), we will call our algorithm Actor Critic Relative Entropy Policy Search (AC-REPS). Our algorithm consists of three steps. First, we estimate the Q-function using the current estimate of the utility function. Subsequently, we compute a new policy that maximizes the Q-values while staying close to the old policy. As in REPS, the policy update can only be performed in closed form for a finite set of samples, where we obtain a desired probability for each of these samples. Finally, we need to fit a new policy to these weighted set of samples.

**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. Moreover, we generate new on-policy actions for all successor states in our data-set, i.e., $a_i' \sim \pi(\cdot|s)$. Given this 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 (Boyan, 1999) algorithm.

**Actor Critic REPS.** The policy update of actor critic REPS is inspired by the episodic REPS algorithm (Daniel et al., 2012). We want to find a policy $\pi(a|s)$ that optimizes the expected Q-value, but 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 in the given state sample distribution $\mu(s)$, i.e., $p(s) = \mu(s), \forall s$. These set of constraints is implemented by matching feature averages of the distributions $p(s)$ and $\mu(s)$ (Daniel et al., 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

$$\begin{aligned} &\arg\max_p \int p(s,a)Q(s,a)dsda, \\ &\quad \text{s.t. } \mathrm{KL}(p(s,a)||q(s,a)) \leq \epsilon, \\ &\qquad \int p(s)\boldsymbol{\phi}(s)ds = \hat{\boldsymbol{\phi}}, \quad \int p(s,a)dsda = 1, \end{aligned} \tag{9}$$

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

$$p(s)\pi(a|s) \propto q(s,a) \exp\left(\frac{Q(s,a) - V(s)}{\eta}\right), \tag{10}$$

where $V(s) = \boldsymbol{v}^T \boldsymbol{\phi}(s)$ is a state dependent baseline. The parameters $\eta$ 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

$$\begin{aligned} g(\eta, \boldsymbol{v}) =& \epsilon\eta + \boldsymbol{v}^T \hat{\boldsymbol{\phi}} + \\ & \eta \log \sum_i \frac{1}{N} \exp\left(\frac{Q(s_i, a_i) - \boldsymbol{v}^T \boldsymbol{\phi}(s_i)}{\eta}\right), \end{aligned} \tag{11}$$

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

The optimization problem is similar to the one of the episodic REPS algorithm given in Daniel et al. (2012) For details of the derivation of the given equations, we refer to the papers given above and a survey paper (Deisenroth et al., 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 $p(s_i, a_i) = p(s_i)\pi(a_i|s_i)$ with Equation (10). 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 $\pi$ and $\tilde{\pi}$, i.e.,

$$\begin{aligned} &\mathbb{E}_s\left[\mathrm{KL}\left(\pi(a|s)||\tilde{\pi}(a|s)\right)\right] = \int p(s,a) \log \frac{\pi(a|s)}{\tilde{\pi}(a|s)} dsda \\ &\approx -\frac{1}{N} \sum_i \frac{p(s_i, a_i)}{q(s_i, a_i)} \log \tilde{\pi}(a|s) + \text{const} \\ &= -\frac{1}{N} \sum_i \exp\left(\frac{Q(s_i, a_i) - V(s_i)}{\eta}\right) \log \tilde{\pi}(a|s), \end{aligned} \tag{12}$$

where we replaced the integral by samples, which have been generated by our sampling distribution $q(s,a)$. Note that Equation (12) 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.

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 in Kober et al. (2010) to the sparse Gaussian process case given in Snelson and Ghahramani (2006).

### 2.3 Requesting preferences

The new, stochastic REPS policy is now used to compute $N$ new trajectories as additional transition samples. For requesting new preferences, we are interested in trajectories assumed to dominate the currently maintained set of undominated trajectories (see Sec. 1.1). 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. This means the, we apply the new utility function to the feature averages (Sec. 1.1) of the newly sampled trajectories and select by the outcome. Each trajectory is iteratively compared with current best ones, replacing them if a domination preference was encountered.

## 3. Experiments

In our experiments, the learner does not have access to the true reward but is given a preference feedback, based on the undiscounted sum of rewards. All experiments are averaged over 30 trials, where we collect 10 trajectories per iteration and use a discount factor of $\gamma = 0.98$. All hyperparameters are optimized.

### 3.1 Domains

**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. 5 new preferences are requested in each of the 10 iterations. The environment is not difficult from an RL point of view, but interesting for a preference learning scenario. This is due to the fact, that it is not trivial to reconstruct the true reward function, because the rewards for different fields are similar.

**Bicycle.** The second task is the bicycle balance task (Lagoudakis and Parr, 2003), where we use continuous states and actions with a GP policy. We trained on episodes of length 50 and evaluated for 3000 steps with 15 iterations. It should be noted, this setup is not directly comparable to Akrour et al. (2014), because we do not require a generative model. and use a Gaussian process policy (Sec. 2.2).

### 3.2 Results

In the following graphs, the thick lines define the median while the shaded areas and the thin lines show the 25% and 75% percentile of the policy value. The set of gridworld experiments shows the advantage of REPS (Sec. 2.2) and the modifications to the PBIRL problem, we introduced in Sec. 2.1. Figure 1 shows the amount of preferences required to reach a certain level of optimality. The graph shows the result for the described version, our algorithm without the modifications to the base PBIRL algorithm (see Sec. 2.1), using Soft-Max exploration instead of REPS and the results of Akrour et al. (2014).

It can be seen that most trials have converged within 30 requested preferences, comparably to the results achieved by Akrour et al. (2014). However, we use an online estimate of the model, instead of an initial estimate. Moreover, this result demonstrates that the rather complex, directed exploration method used by Akrour et al. (2014) is not needed to solve the preference case efficiently.

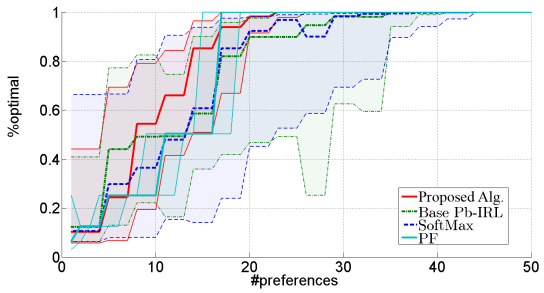
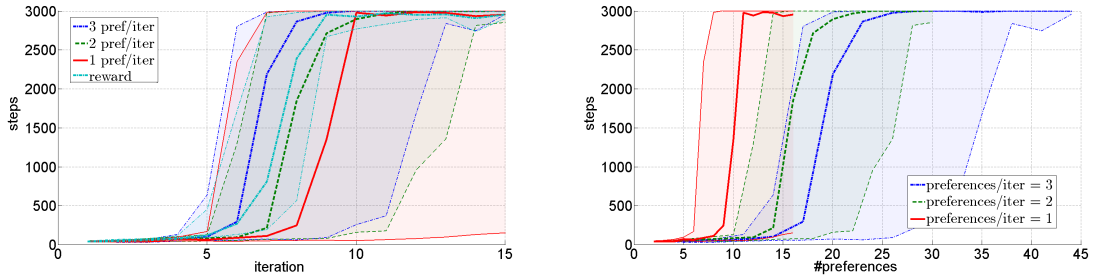



Figure 1: Gridworld results

With the soft-max policy, controlling the exploration-exploitation trade-off is much harder, and, hence, we can see a worse performance. A similar poor performance can be observed for the case with an unmodified PBIRL algorithm, as the quality of the estimated utility function degrades.




(a) Bicycle task, per iteration

(b) Bicycle task, per preference

On the bicycle task, we compare in Figure 2a the reward case to different amounts of preferences requests per iteration. Utilizing enough preferences per iteration enables convergence comparable to the reward case, where numeric, per step reward is available. The required amount of iterations and transition samples increases when lowering the amount of preferences per iteration.

When we only consider the amount of preferences requested, as in Figure 2b, it can be seen that a low amount of preferences per iteration with steady policy improvements works best. The difference in graph length is a result of the fixed iteration count.

## 4. Conclusion

We have demonstrated that is 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 are comparable to the state-if-the-art which requires a generative model or an individual model learning phase. Future work will focus on coupling the exploration parameter $\epsilon$ with the confidence of the utility estimate to allow for more aggressive updates. Furthermore, the PBIRL algorithm can be improved by utilizing entropy-based or bayesian formulations.

## References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine learning (ICML-04)*, page 1. ACM, 2004.

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)*, pages 116–131. Springer, 2012.

R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by Feedback. In *International Conference on Machine Learning*, Pékin, China, June 2014.

J. Boyan. Least-Squares Temporal Difference Learning. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 49–56, 1999.

C. Daniel, G. Neumann, and J. Peters. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

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

J. Kober, K. Mülling, O. Kroemer, C. H. Lampert, B. Schölkopf, and J. Peters. Movement Templates for Learning of Hitting and Batting. In *International Conference on Robotics and Automation (ICRA)*, 2010.

M. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, December 2003. ISSN 1532-4435.

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

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

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

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

E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-Inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264. MIT press, 2006.

A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS-12)*, pages 1142–1150. Curran Associates, 2012.

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