A Sample-Efficient Black-Box Optimizer to Train Policies for Human-in-the-Loop Systems with User Preferences

Nitish Thatte[1], Helei Duan[2], and Hartmut Geyer[1]

***Abstract*—We present a new algorithm for optimizing control policies for human-in-the-loop systems based on qualitative preference feedback. This method is especially applicable to systems such as lower-limb prostheses and exoskeletons for which it is difficult to define an objective function, hard to identify a model, and costly to repeat hardware experiments. To solve these problems, we combine and extend an algorithm for learning from preferences and the Predictive Entropy Search Bayesian optimization method. The resulting algorithm, Predictive Entropy Search with Preferences (PES-P), solicits preferences between pairs of control parameter sets that optimally reduce the uncertainty in the distribution of objective function optima with the least number of experiments. We find that this algorithm outperforms the expected improvement method (EI), and random comparisons via Latin hypercubes (LH) in three simulation tests that range from optimizing randomly generated functions to tuning control parameters of linear systems and of a walking model. Furthermore, we find in a pilot study on the control of a robotic transfemoral prosthesis that PES-P finds good control parameters quickly and more consistently than EI or LH given real user preferences. The results suggest the proposed algorithm can help engineers optimize certain robotic systems more accurately, efficiently, and consistently.**

***Index Terms*—Learning and Adaptive Systems; Human Factors and Human-in-the-Loop; Optimization and Optimal Control**

I. Introduction

OPTIMIZING control policies for human-in-the-loop robotic systems, such as lower-limb prostheses and exoskeletons, is a challenging task due to two key issues. First, to optimize these systems it is currently necessary to define an objective function that includes and correctly assigns importance to all characteristics that determine system performance. For instance, consider an amputee trying to optimize the control parameters of her robotic leg prosthesis. The amputee could evaluate the prosthesis performance via an objective function that trades off important gait characteristics in order to guide the optimization. However, gait features, such as metabolic energy consumption, speed, and gait symmetry, require a high level of technical expertise and equipment to measure. Moreover, other aspects of gait may also be important but difficult to quantify, such as the amputee's comfort and sense of stability. Even if the amputee could measure all these characteristics, the objective function would still need to assign weights to each feature that reflect the amputee's individual needs.

To solve the problem of defining objective functions for robotic systems that human operators can directly control, researchers have proposed *learning from demonstration* (LfD) [1]. In this paradigm, we can either circumvent learning the objective function by directly learning a policy that matches the distribution of state-action pairs recorded during human demonstrations of the desired behavior [2, 3], or we can learn a reward function consistent with the demonstrator's actions and visited states and use it to derive an optimal control [4–6]. LfD methods are attractive because they allow non-experts to specify both the quantifiable and qualitative aspects of the desired robot behavior via the non-technical language of demonstration.

For robot behavior that people cannot demonstrate, such as the optimal behavior of an amputee's prosthesis, or the desired behavior of complex, dynamic robots, we can alternatively query human users for qualitative feedback in order to shape the robot policy. For example, the TAMER framework [7, 8] utilizes good/bad assessments of a robot's recent actions to optimize its policy. Pilarski et al. use this method to allow subjects to optimize the policy of an EMG-controlled prosthesis arm via their positive and negative feedback signals [9]. Another paradigm in qualitative feedback is to obtain *preference feedback* between two or more policies or sequences of actions, which may provide more nuanced feedback than absolute ratings. For example, Jain et al. and Akrour et al. propose methods that learn a user's trajectory scoring function based on his rankings of possible policies [10, 11]. Similarly, Wilson et al. provide a method to directly identify a user's preferred policy based on her preferences between pairs of demonstrated trajectories [12]. These prior works demonstrate that we can successfully use qualitative feedback, such as preferences, from non-expert users to program robot behavior, without prescribing an objective function.

A drawback of the aforementioned methods that learn from preference feedback is their reliance on simulators to predict system behavior. Human-in-the-loop systems, such as lower-limb prostheses and exoskeletons, are challenging to simulate accurately, making these methods difficult to apply. If the control is governed by a fixed set of parameters, as is often the case for these kinds of systems, we can

Manuscript received: Sep 8, 2016; Revised Dec 7, 2016; Accepted Jan 5, 2017.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This material is based upon work supported by the National Science Foundation under Grant No. 1527140.

[1]Nitish Thatte nitisht@cs.cmu.edu and Hartmut Geyer hgeyer@cs.cmu.edu are with The Robotics Institute and [2]Helei Duan hduan@cmu.edu is with Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213.

Digital Object Identifier (DOI): see top of this page.

instead employ model-free *black-box* optimization methods. These methods have gained traction in the related field of control optimization for dynamic locomotion, where it can be difficult to model the nonlinear, discontinuous dynamics of these systems. Specifically, many have applied stochastic or "evolutionary" optimization methods, which repeatedly sample and mix control parameters that perform well, to locomotion control problems [13].

The second issue an operator tasked with optimizing control policies for human-in-the-loop systems faces is the expense, in terms of time and effort, of repeatedly executing policies. Consequently, stochastic sampling approaches may be less applicable in this domain. To minimize the number of trials needed, researchers have proposed black-box *Bayesian Optimization* (BO) methods that model both the objective function and its uncertainty. In these methods, the uncertainty informs an acquisition function that speeds up the optimization by exploiting regions of the parameter space with believed high objective value while still exploring regions where the objective function is uncertain. For example, researchers have successfully employed BO methods to efficiently optimize the gait parameters of a robotic snake [14] and a dynamic bipedal robot [15].

This paper is motivated by the observation that prior research has not thoroughly explored solutions that address both the difficulty of defining objective functions and the expense of running repeated experiments for systems that are difficult to model and for which qualitative characteristics are important. We present a new optimization algorithm, Predictive Entropy Search with Preferences (PES-P), that addresses these issues. The algorithm uses preference queries between pairs of control parameters to avoid the a priori definition of features and to consider unquantifiable qualities of the desired behavior. The algorithm further incorporates black-box Bayesian optimization to ensure its preference queries gather information efficiently without relying on a system model.

In developing the algorithm, we make three main contributions. First, we adapt an acquisition function previously proposed for interval scale feedback to the preference feedback case. This acquisition function seeks a pair of parameters for which a preference will maximally reduce the entropy of the distribution of objective function optima. Second, we compare in simulation the performance of the proposed optimization method against the expected improvement method (EI) and uniform random sampling via Latin hypercubes (LH) for two classes of examples: optimizing randomly generated objective functions and tuning the control parameters of simulated dynamical systems. Finally, we compare the performance of the three methods for the task of optimizing the control parameters of a robotic prosthesis given real user feedback.

II. Preliminaries

A. Learning from Preferences

To learn latent objective functions from preferences, we rely on the method developed by Chu and Ghahramani [16], briefly reviewed here. The method considers a training dataset D_n of n preferences between pairs of points, $\{x_1^{\mathrm{a}} \succ x_1^{\mathrm{b}}, \dots, x_k^{\mathrm{a}} \succ x_k^{\mathrm{b}}, \dots, x_n^{\mathrm{a}} \succ x_n^{\mathrm{b}}\}$. For instance, in a prosthesis tuning task, we would ask an amputee to walk with two control parameter sets $(x_k^{\mathrm{a}}, x_k^{\mathrm{b}})$ and provide a preference between them. D_n aggregates these responses over n repetitions of this task. From the dataset, the method finds a posterior distribution of latent objective functions $\boldsymbol{f}$, which in our case describe the user's overall assessment of controller performance,

$$\mathrm{P}(\boldsymbol{f}|D_n) = \frac{\mathrm{P}(D_n|\boldsymbol{f})\,\mathrm{P}(\boldsymbol{f})}{\mathrm{P}(D_n)}. \tag{1}$$

The method assumes that the prior distribution of objective functions is a zero-mean *Gaussian process* (GP), $\mathrm{P}(\boldsymbol{f}) = \mathcal{N}(0, \Sigma)$. (See [17] for a full description of GPs.) In eq. (1), $\mathrm{P}(D_n|\boldsymbol{f})$ is the overall likelihood of preferences in the dataset given specific objective function values. The likelihood model for preferences proposed by Chu and Ghahramani increases the certainty of a preference between x_k^{a} and x_k^{b} as the difference between $f(x_k^{\mathrm{a}})$ and $f(x_k^{\mathrm{b}})$ widens.

To obtain the posterior distribution $\mathrm{P}(\boldsymbol{f}|D_n)$ the method approximates eq. (1) with a Gaussian distribution. As a result, the predictive distribution (subscript p) of the objective function at test points, $\boldsymbol{f}_{\mathrm{t}}$, is also Gaussian, $\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}|D_n) = \mathcal{N}\left(\mu_{\mathrm{p}}, \Sigma_{\mathrm{p}}\right)$. Finally, the predictive distribution of a preference between two points x^{a} and x^{b} is

$$\mathrm{P}\left(x^{\mathrm{a}} \succ x^{\mathrm{b}}|D_n\right) = \int \mathrm{P}\left(x^{\mathrm{a}} \succ x^{\mathrm{b}}|\boldsymbol{f}_{\mathrm{t}}, D_n\right)\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}|D_n)\mathrm{d}\boldsymbol{f}_{\mathrm{t}} \tag{2}$$

$$= \Phi\left(\frac{\mu^{\mathrm{a}} - \mu^{\mathrm{b}}}{\sigma_{\mathrm{p}}}\right), \tag{3}$$

$$\sigma_{\mathrm{p}}^2 = 2\sigma^2 + \Sigma_{\mathrm{p}}^{\mathrm{aa}} + \Sigma_{\mathrm{p}}^{\mathrm{bb}} - \Sigma_{\mathrm{p}}^{\mathrm{ab}} - \Sigma_{\mathrm{p}}^{\mathrm{ba}}, \tag{4}$$

where $\Phi(\cdot)$ is the cumulative density function of the standard normal distribution.

Figure 1a provides an example of how the method estimates a ground-truth objective function shown in purple. The blue line and shaded area show the mean and standard deviation of the posterior distribution of objective functions, $\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}|D_n)$, after two preference queries between pairs of parameters (orange, higher is preferred over lower value). The queries have the effect of lifting the estimated objective function close to preferred points and pushing it down close to unpreferred points, approximating the true objective function over time.

B. Active Learning for Optimization

Learning from preferences describes how to find a distribution of objective functions given a dataset of comparisons. The question now becomes how to efficiently solicit preferences from the user. As our main goal is to find the optimal prosthesis control parameters x^*, we need not accurately model the user's objective function in all parameter regions. Instead, we should focus on regions where the objective might be high. Bayesian optimization addresses this problem with an acquisition function that helps to efficiently sample training data.

One such acquisition function is the expected improvement, which has been used both in the context of preference feedback [18] and interval scale feedback [19],

$$\mathrm{EI}(x) = (\mu^* - \mu(x))\Phi(d) + s(x)\phi(d), \tag{5}$$

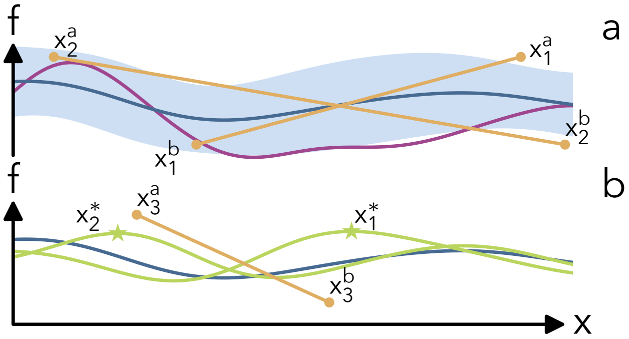


Fig. 1: Learning from preferences. (a) Mean and standard deviation of $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n)$ (blue) after two preferences queries (orange) from the true objective function (purple). (b) Mean of $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n)$ (blue) and means of $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n, x_m^*)$ (green) for two samples of x_m^*. PES-P queries a new comparison (orange) for which the preference is currently uncertain, but on average is certain after conditioning on all x_m^*.

where $d = (\mu^* - \mu(x))/s(x)$, μ^* is the mean of the current estimate of the optimum, and $\mu(x)$ and $s(x)$ are the mean and standard deviation of the objective of a new point x, respectively. As an alternative, for interval scale feedback, [20] and [21] proposed acquisition functions that seek to reduce the uncertainty in the distribution of objective function optima, measured in terms of the differential entropy. For example, the Predictive Entropy Search acquisition function [21] seeks a point x that is expected to reduce the entropy of the distribution of optima x^* after observing its value y,

$$\alpha_n(x) = \mathrm{H}[\mathrm{P}(x^*|D_n)] - \mathrm{E}_{\mathrm{P}(y|x,D_n)}[\mathrm{H}[\mathrm{P}(x^*|y,x,D_n)]], \quad (6)$$

where $\mathrm{H}[\mathrm{P}(x)] = -\int \mathrm{P}(x)\log \mathrm{P}(x)\mathrm{d}x$ is the differential entropy. The authors of these methods have shown they can outperform EI.

III. Methods

Our goal is to simultaneously address both the difficulty of defining objective functions when an expert cannot demonstrate the desired robot behavior and the expense of running experiments on hardware. To this end, we adapt the Predictive Entropy Search acquisition function (eq. (6)) to the preference learning case.

A. Acquisition Function

To obtain the optimal parameters x^* with the smallest number of preference queries, we solicit preferences that maximize the expected information gain about the distribution of objective function optima $\mathrm{P}(x^*|D_n)$. Adapting eq. (6) to preference feedback yields

$$\begin{aligned}\alpha_n\left(x^\mathrm{a}, x^\mathrm{b}\right) &= \mathrm{H}[\mathrm{P}(x^*|D_n)] \\ &\quad - \mathrm{E}_{\mathrm{P}(y|x^\mathrm{a},x^\mathrm{b},D_n)}\left[\mathrm{H}\left[\mathrm{P}\left(x^*|y, x^\mathrm{a}, x^\mathrm{b}, D_n\right)\right]\right], \quad (7)\end{aligned}$$

where y is a binary random variable that represents the preference between x^a and x^b. The first term in this function is the current entropy of objective function optima and the second term is the entropy of optima after observing the preference y. As we have not yet observed the preference, we take the second term in expectation over the two possible preference outcomes.

As discussed in [21], this acquisition function is intractable to compute. However, following the approach used for the original PES algorithm, we can rewrite eq. (7) in terms of the entropies of the predictive distribution of the preference between x^a and x^b,

$$\begin{aligned}\alpha_n\left(x^\mathrm{a}, x^\mathrm{b}\right) &= \mathrm{H}\left[\mathrm{P}\left(y|x^\mathrm{a}, x^\mathrm{b}, D_n\right)\right] \\ &\quad - \mathrm{E}_{\mathrm{P}(x^*|D_n)}\left[\mathrm{H}\left[\mathrm{P}\left(y|x^*, x^\mathrm{a}, x^\mathrm{b}, D_n\right)\right]\right] \quad (8)\\ &\approx \mathrm{H}\left[\mathrm{P}\left(y|x^\mathrm{a}, x^\mathrm{b}, D_n\right)\right] \\ &\quad - \frac{1}{M}\sum_{x_m^* \sim \mathrm{P}(x_m^*|D_n)}^{M} \mathrm{H}\left[\mathrm{P}\left(y|x_m^*, x^\mathrm{a}, x^\mathrm{b}, D_n\right)\right]. \quad (9)\end{aligned}$$

This reformulation improves computability for several reasons. First, the new acquisition function computes the entropies of probabilities of preferences, given by eq. (3). Second, instead of computing the entropy of $\mathrm{P}(x^*|D_n)$, we now take an expectation over this distribution, which we can perform by sampling M functions from $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n)$ and optimizing each one to get M samples of x^* (see Appendix for details). Finally, the second term no longer requires conditioning the GP on every pair of x^a and x^b considered during optimization of the acquisition function. Instead, we only have to condition the Gaussian process M times on (x_m^*, D_n).

For the experiments in section IV we choose $M = 12$, which allows us to construct and optimize $\alpha_n(x^\mathrm{a}, x^\mathrm{b})$ in about five seconds, which is fast enough for our prosthesis application. Although 12 samples of x^* is not enough to compute an accurate expectation over $\mathrm{P}(x^*|D_n)$, interpreting the algorithm as an example of active learning by disagreement may explain why it still works well. As shown in fig. 1b, optimizing the acquisition function chooses a pair x^a and x^b for which the preference is currently uncertain, but certain on average after conditioning on all x_m^*. The sampled x_m^* do not necessarily agree on which point is preferred; hence, after observing the preference, the algorithm can rule out x_m^* that made the model certain but wrong about the preference. This intuition is similar to that provided by [22] for Bayesian active learning by disagreement for GP classifiers.

B. Conditioning the Gaussian Process on x^*

The second term on the right side of eq. (9) requires us to compute the distribution of the preference given the location of the optimum,

$$\begin{aligned}&\mathrm{P}\left(y|x_m^*, x^\mathrm{a}, x^\mathrm{b}, D_n\right) = \\ &\quad\int \mathrm{P}\left(x^\mathrm{a} \succ x^\mathrm{b}|\boldsymbol{f}_\mathrm{t}, x_m^*, D_n\right)\mathrm{P}(\boldsymbol{f}_\mathrm{t}|x_m^*, D_n)\mathrm{d}\boldsymbol{f}_\mathrm{t}. \quad (10)\end{aligned}$$

It is not directly feasible to condition the predictive distribution on x^*, so instead we turn to approximating this condition with three constraints (see appendix for details):

Algorithm 1 Predictive Entropy Search with Preferences

```
 1: procedure PES-P
 2:     $D_n = \varnothing$
 3:     for $n \leftarrow 0$ to $N-1$ do                                  ▷ $N$ iterations
 4:         $F \leftarrow \{\boldsymbol{f}_m \sim \mathrm{P}(\boldsymbol{f}_{\mathrm{t}}|D_n)|m \in [1, M]\}$
 5:         $X^* \leftarrow \{\arg\max_x(\boldsymbol{f}_m)|\boldsymbol{f}_m \in F\}$
 6:         $(x^{\mathrm{a}}_{n+1}, x^{\mathrm{b}}_{n+1}) \leftarrow \arg\max_{(x^{\mathrm{a}},x^{\mathrm{b}})} \alpha_n(x^{\mathrm{a}}, x^{\mathrm{b}}; X^*)$
 7:         $y_{n+1} \leftarrow$ QueryUserPref$(x^{\mathrm{a}}_{n+1}, x^{\mathrm{b}}_{n+1})$
 8:         $D_{n+1} \leftarrow D_n \cup (x^{\mathrm{a}}_{n+1}, x^{\mathrm{b}}_{n+1}, y_{n+1})$
 9:     end for
10:     return $x^* \leftarrow \arg\max_x \mathrm{mode}(\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}(x)|D_N))$
11: end procedure

12: function $\alpha_n(x^{\mathrm{a}}, x^{\mathrm{b}}; X^*)$                        ▷ acquisition function
13:     $h \leftarrow \Big\{\mathrm{H}\Big[\mathrm{P}\Big(y|x^{\mathrm{a}}, x^{\mathrm{b}}, D_n, \mathrm{C1}, \mathrm{C2}, \mathrm{C3}\Big)\Big]\Big|x^*_m \in X^*\Big\}$
14:     return $\mathrm{H}\Big[\mathrm{P}\Big(y|x^{\mathrm{a}}, x^{\mathrm{b}}, D_n\Big)\Big] - \mathrm{mean}(h)$
15: end function
```

C1: First we impose that x^* is a local maximum by ensuring that the gradient of $f(x^*)$ is zero and its Hessian is negative definite. We further simplify the Hessian constraint to only require that the Hessian's off-diagonal elements are zero and its diagonal elements are less than zero. We implement the gradient and off-diagonal constraints by conditioning the prior, $\mathrm{P}(\boldsymbol{f})$, on derivative observations as outlined in [23]. To constrain the diagonal elements of the Hessian, we amend the likelihood term in eq. (1) by adding terms that penalize Hessians with positive diagonal elements.

C2: Second, we try to ensure that x^* is also a global maximum by enforcing that $f(x^*)$ is greater than the function values of all training points sampled so far. We impose this constraint by adding more preference relations into the likelihood term in eq. (1) between x^* and all training points.

C3: Finally, to further ensure that $f(x^*)$ is a global maximum, we require that it is also larger than the function values of the two new test points, $f(x^{\mathrm{a}})$ and $f(x^{\mathrm{b}})$. Whereas C2 ensures $f(x^*)$ exceeds function values in areas explored so far, C3 ensures that $f(x^*)$ also exceeds function values in unexplored regions. We approximate this constraint analytically by conditioning on the single constraint $f(x^*) > (f(x^{\mathrm{a}}) + f(x^{\mathrm{b}}))/2$ using the method detailed in [24].

C. Algorithm Summary

With constraints C1 to C3, at each iteration we can efficiently compute the acquisition function, eq. (9). We summarize the resulting Predictive Entropy Search with Preferences (PES-P) algorithm as follows (algorithm 1): At each iteration n, first, the algorithm samples M objective functions from the current distribution, $\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}|D_n)$, and optimizes each one to generate M samples of x^* (lines 4 and 5). Next, using the set of sampled optimums X^*, we maximize the acquisition function to obtain the next two points to present to the user x^{a}_{n+1} and x^{b}_{n+1} (lines 6 and 12–15). Note: we can precompute the effect of C1 and C2 before evaluating $\alpha_n(x^{\mathrm{a}}, x^{\mathrm{b}})$ as these two constraints do not depend on x^{a}_{n+1} and x^{b}_{n+1}. On the other hand, C3 depends directly on x^{a}_{n+1} and x^{b}_{n+1} and therefore is computed within the acquisition function for every pair of points considered during the optimization of $\alpha_n(x^{\mathrm{a}}, x^{\mathrm{b}})$. We then query the user to obtain her preference y_{n+1} between these two points and add it to the dataset of preferences (lines 7 and 8). In our prosthesis optimization case, this involves having the amputee walk with both control parameters in succession and provide a preference between them. Finally, at the end of the N iterations of the algorithm, we return the optimum x^* of the most likely function, $\mathrm{mode}(\mathrm{P}(\boldsymbol{f}_{\mathrm{t}}(x)|D_N))$, which is equal to the posterior mean function in the Gaussian process case (line 10). While it may be more correct to return $\mathrm{mode}(\mathrm{P}(x^*|D_N))$, we do not do this as the PES algorithm seeks to avoid approximating this distribution.

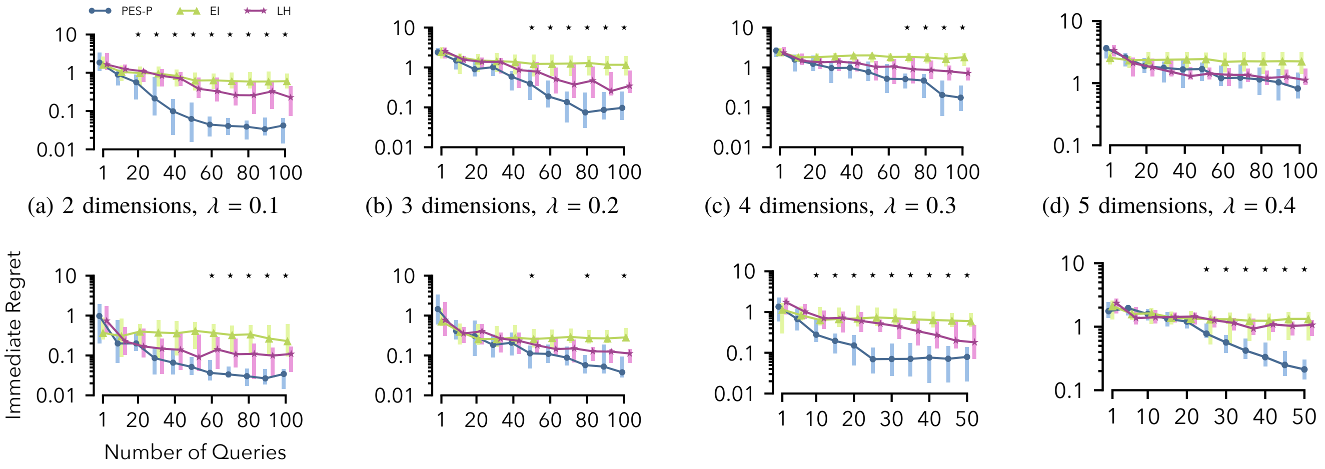
IV. Results

We test the ability of PES-P to solve optimization problems in four cases with increasing realism from the optimization of randomly generated objective functions drawn from a GP, to the tuning of feedback gains of random linear systems and a neuromuscular walking model, to the optimization of control parameters for a powered transfemoral prosthesis given real user feedback. In all four cases, we compare the performance of the proposed algorithm to the expected improvement criterion (EI) (eq. (5)) and random sampling via Latin hypercubes (LH)[1] [25]. For the three simulated cases, we show results over 20 trials and measure performance in terms of the immediate regret, defined as $IR = |f(\tilde{x}^*_n) - f(x^*)|$, versus the number iterations. Here, $f(\tilde{x}^*_n)$ is the objective value of the current estimate of the optimum at this iteration, $f(x^*)$ is the value of the true optimum, and an iteration consists of a single preference query between two points. Additionally, we also check the statistical significance of the reduction in IR obtained by PES-P compared to both EI and LH via one-sided Mann-Whitney U tests ($p < 0.05$).

A. Optimizing Randomly Generated Objective Functions

To avoid inducing bias by hand-engineering test functions, we first evaluate the algorithm on random synthetic objective functions. We generate objective functions on the domain $x \in [-1, 1]^D$ by sampling a vector of 500 function values from a GP prior with a quadratic mean, $\mu(x) = -x^{\mathrm{T}}x$, and isometric squared exponential covariance $\mathrm{k}(x_i, x_j) = \exp\left(\frac{-1}{2\lambda} x_i^T x_j\right)$. We use a quadratic mean function to bias the function distribution away from those that have their optimum on a boundary of the domain, as these functions are easier to optimize. We continue to generate the rest of the function as it is optimized by conditioning the GP on the 500 seed values and all function values sampled during the optimization. We assume the mean of the final function distribution is the true objective function. To simulate more realistic situations, we provide the algorithms with noisy preferences by corrupting sampled function values with Gaussian noise ($\sigma^2 = 0.1$).

[1] LH sampling divides the parameter space into $(2N)^D$ hypercubes, where D is the dimensionality of the space. $2N$ samples are placed such that each hypercube has at most one sample and there is at most one filled hypercube along any row of hypercubes when viewed along any direction. This method ensures that the samples are roughly uniformly distributed in the entire space. At each iteration we choose two of these samples to query users.



(a) 2 dimensions, $\lambda = 0.1$ (b) 3 dimensions, $\lambda = 0.2$ (c) 4 dimensions, $\lambda = 0.3$ (d) 5 dimensions, $\lambda = 0.4$

(e) LQR 3 dim (f) LQR 4 dim (g) Biped Walking 2 dim (h) Biped Walking 3 dim

Fig. 2: Performance of predictive entropy search with preferences (PES-P), expected improvement (EI), and Latin hypercube random sampling (LH) for optimizing random objective functions sampled from a GP (a-d), and tuning feedback control parameters of random linear systems (e-f) and a biped walking model (g-h). Shown are the median and interquartile range over 20 trials of the immediate regret (IR) against the number of preference queries. Black stars indicate iterations for which PES-P achieves statistically significant stochastic reductions in IR compared to both EI and LH according to one-sided Mann-Whitney U tests ($p < 0.05$).

Figures 2a to 2d show the immediate regret for two to five dimensional problems with λ, the length scale of the kernel, scaling from 0.1 to 0.4 as the dimensionality of the problem increases. On two to four dimensional problems, PES-P outperforms EI and LH by achieving statistically significant reductions in IR. However, as the dimensionality increases, it takes more iterations for this advantage to become apparent. In the five dimensional case, there is no significant difference between PES-P and LH, likely due to $M = 12$ samples of x_m^* being insufficient and the difficulty of accurately sampling x_m^* in higher dimensions.

B. *Tuning Controllers for Random Linear Systems*

Next, we test the ability of PES-P to optimize simple control systems by optimizing the feedback gains K for D-dimensional single-input linear systems $\dot{\xi} = A\xi + Bu$ with feedback $u = K\xi$. We sample the elements of the A matrix from the standard normal distribution while $B = [0_{1\times(D-1)}, 1]^{\mathrm{T}}$. We assume a quadratic instantaneous cost resulting in the objective function

$$f(K) = -\int_0^{t_f} \xi_K^{\mathrm{T}}(t)(Q + K^T RK)\xi_K(t)dt, \tag{11}$$

where $\xi_K(t)$ is the evolution of the state under the control policy K and a fixed initial condition ξ_0, $Q = I_{D\times D}$ and $R = 1$. To obtain a finite search domain, we find the stable range of parameters by varying the elements of the true optimal control parameters K^* one at a time while keeping other elements constant. We scale and shift this region to map to the domain $[-1, 1]^D$. Finally, we use the Automatic Relevance Determination Gaussian Kernel and optimize the hyperparameters at each iteration by maximizing the posterior probability of the hyperparameters under a gamma hyperprior [16, 17]. In order to apply a consistent noisy preference model ($\sigma^2 = 0.1$) across all sampled systems, we transform all objective values by first mapping them through $-\log(-f(K))$ and then shifting and scaling the values by the mean and range of the values of 10^D randomly sampled controllers.

Figures 2e and 2f show the resulting optimization performance on three and four dimensional systems. In the 3 dimensional case, PES-P achieves a lower median IR than LH after 30 iterations. This difference becomes significant after 60 iterations. In the 4 dimensional case, PES-P significantly outperforms LH after 50 iterations, but the significance of this improvement is sporadic as the iterations continue. A possible reason for the reduced performance difference between PES-P and LH in the LQR problem as compared to the random objective function problems is the existence of hard-to-optimize flat regions in the LQR objective functions. This suggests that PES-P may be more well suited for problems that have clear optimum.

C. *Tuning Control Parameters of a Walking Model*

In the third case, we test the ability of PES-P to optimize the feedback gains for a neuromuscular model of walking [26], a system with a complex non-linear controller addressing the specific application domain of human locomotion. We perform two and three dimensional optimizations, in which we tune the feedback gains for a subset of the model's muscle actuators. We use the negative cost of transport plus the distance walked over a 20 second time span as the objective function. As in the previous linear systems example, we obtain noisy preferences between parameters and optimize the hyperparameters at every iteration.

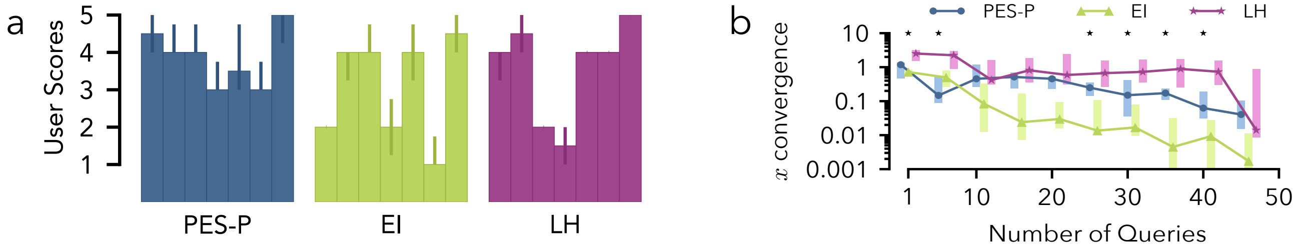


Fig. 3: Optimization of prosthesis control with user preferences. (a) Median and interquartile range of user scores achieved by PES-P, EI and LH after 50 iterations (total of 42 scores per algorithm: seven users times six scorings). (b) Median and interquartile range of convergence achieved by the three algorithms as measured by the Euclidean distance between the current and final estimates of the optimum. Black stars indicate iterations for which PES-P achieves statistically significant stochastic improvements in convergence compared LH according to one-sided Mann-Whitney U tests ($p < 0.10$). PES-P and LH achieve the same median score of 4 across all users but PES-P converges faster and more consistently. EI converges fastest but to a lower median score of 3.

Figures 2g and 2h show the performance of PES-P, EI, and LH. In this example, PES-P achieves a significant reduction in IR in just 10 iterations in the 2-dimensional case and in 25 iterations in the 3 dimensional case. Furthermore, in the 3D case the PES-P's median solution is approximately 10 times better than those found by EI or LH.

D. Tuning a Transfemoral Prosthesis from User Preferences

In the last test case, we applied the three algorithms to optimize the control parameters for a powered transfemoral prosthesis given real user preferences. Specifically, a neuromuscular model similar to the one used in section IV-C controls the prosthesis and we optimize the strengths of three virtual knee muscles of this control [26].

We performed this test in a pilot study with seven healthy users. They walked on a treadmill and wore the powered prosthesis with a modified knee brace (compare [26]). We allowed all users an hour-long session to acclimate to the device, during which they experienced a variety of controller conditions. On a second day, we optimized the prosthesis parameters using the three algorithms (PES-P, EI, and LH) in a random order, for 50 iterations each. During an iteration, the users walked with two parameter settings chosen by the algorithm (each for 10 seconds) and then indicated which setting they preferred. After completing the three optimizations, the users walked with the optimum parameters identified by each algorithm (in a random order) for fifteen seconds and then rated each optimum on a 1 (bad) to 5 (good) scale. We repeated the scoring procedure six times to cover all possible orderings of the three optima.

Figure 3 summarizes the results from the optimizations with user preferences. PES-P and LH achieved median user scores of 4 while EI achieved a median score of 3 (Fig. 3a). In addition, PES-P, LH, and EI achieved mean scores of 4.0, 3.5, and 3.1, respectively (not shown). The gap between the mean and median scores for LH implies that LH does not achieve high scores as consistently as PES-P. A second observation is that PES-P converged faster than LH to the optimum as measured by the distance between its current and final estimates, $\|\tilde{x}_n^* - \tilde{x}_N^*\|$ (Fig. 3b). Meanwhile, EI tended to converge fastest, but to lower scoring parameters on average (Fig. 3a).

V. Discussion and Conclusion

We presented a new optimization algorithm (PES-P) that extends Predictive Entropy Search to preference feedback. The algorithm addresses two key problems frequently encountered in system optimization. First, it circumvents the often difficult process of parameterizing and learning an objective function by directly querying users for preferences between pairs of parameters. Second, the algorithm minimizes the required number of hardware experiments by employing Bayesian optimization techniques that ensure the queries maximize the information gained about the location of the optimum. Moreover, unlike previous approaches for preference learning on robotic systems [10, 12], PES-P does not require a model of the system.

Our experiments show that the proposed algorithm outperforms baseline algorithms. In most of the simulation experiments PES-P found optima that achieved higher objective values than those found by the expected improvement method (EI) or by random comparisons via Latin hypercubes (LH) (fig. 2). In the prosthesis experiment, PES-P outperformed EI and achieved final scores similar to LH with faster convergence (fig. 3). These results suggest the proposed algorithm can help engineers optimize some types of human-in-the-loop robotic systems more accurately, efficiently, and consistently.

The reason why PES-P outperformed EI is likely due to the former's explicit consideration of how the limited, noisy information obtained from a preference query will affect the knowledge about potential objective function optima. The acquisition function (eq. (7)) recognizes that preferences become more uncertain the closer two sample points are to each other. EI, on the other hand, does not reason about noisy preferences and, instead, still assumes it can sample values (eq. (5)). Consequently, EI ignores the distance between sample points, which often leads to a greedy strategy that solicits preferences between adjacent points. While this strategy can resemble gradient ascent with convergence to local optima in a noise-free optimization, it often failed in our simulated and real experiments characterized by noisy observations. Note, however, that such limitations were not observed by Brochu and colleagues [18], who successfully used EI with preferences to optimize parameters for a graphics application, possibly because the associated visual task produced less noisy responses than did our simulations or prosthesis walking task.

Several modifications could improve the PES-P algorithm. First, using a non-zero prior mean function governed by a set of hyperparameters could embed specific knowledge about the problem to speed up optimization. To improve efficiency in this way, [27] details an approach for learning hyperpriors that could be integrated with PES-P. Second, integrating more varied user feedback may also help improve the algorithm. For example, "I don't know" responses could imply that the function values at two points are similar, absolute good and bad ratings could encourage the algorithm to more quickly explore promising control polices and avoid bad ones, and derivative observations could indicate the user prefers more or less of a parameter. With these two changes, the algorithm may be able to tackle higher dimensional problems. Finally, including time as a dimension in the GP could account for user adaptation to the robotic system.

Acknowledgement

We would like to thank the reviewers for their many helpful comments that substantially improved the manuscript.

Appendix

To obtain X^* (line 5, algorithm 1), we sample M functions from the posterior by approximating $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n)$ using Bayesian linear regression with Fourier features (as outlined in [21]) and sampling M feature weight vectors. As the Fourier features have analytic derivatives, we can optimize each linear function using a second order method with multiple restarts.

We approximate conditioning the predictive distribution on x^* via three constraints:

C1 x^* is a local maximum. $\nabla f|_{x^*} = 0$ and the Hessian of the objective function is negative definite by imposing $\mathrm{diag}(\nabla\nabla f|_{x^*}) < 0$ and $\mathrm{upper}(\nabla\nabla f|_{x^*}) = 0$. We group $\nabla f|_{x^*} = 0$ and $\mathrm{upper}(\nabla\nabla f|_{x^*}) = 0$ into constraint C1.1 and $\mathrm{diag}(\nabla\nabla f|_{x^*}) < 0$ into constraint C1.2.

C2 x^* is preferred to current training points, $f(x^*) > f(x^\mathrm{a}_k)$ and $f(x^*) > f(x^\mathrm{b}_k)$, $\forall k \in [1, n]$.

C3 x^* is preferred to new training points, $f(x^*) > f(x^\mathrm{a}_{n+1})$ and $f(x^*) > f(x^\mathrm{b}_{n+1})$.

We precompute the effects of contraints C1 and C2 before evaluation of $\alpha_n(x^\mathrm{a}, x^\mathrm{b})$. To impose C1 and C2, we first divide their components into two groups: $\boldsymbol{c} = [\nabla f|^\mathrm{T}_{x^*}, \mathrm{upper}(\nabla\nabla f|_{x^*})^\mathrm{T}]^\mathrm{T}$ and $\boldsymbol{f}' = [\boldsymbol{f}^\mathrm{T}, \mathrm{diag}(\nabla\nabla f|_{x^*})^\mathrm{T}, f(x^*)]^\mathrm{T}$. Note C1.1 $\Longrightarrow$ $\boldsymbol{c} = 0$. We write the predictive distribution of the objective function at test points $\boldsymbol{f}_\mathrm{t}$ given constraints C1 and C2 as

$$\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n, \mathrm{C1}, \mathrm{C2}) = \int \mathrm{P}(\boldsymbol{f}_\mathrm{t}|\boldsymbol{f}', \mathrm{C1.1})\,\mathrm{P}(\boldsymbol{f}'|D_n, \mathrm{C1}, \mathrm{C2})\mathrm{d}\boldsymbol{f}'. \tag{12}$$

We use Bayes rule to evaluate the second term in the integral, $\mathrm{P}(\boldsymbol{f}'|D_n, \mathrm{C1}, \mathrm{C2}) = \frac{\mathrm{P}\left(D_n, \mathrm{C1.2}, \mathrm{C2}|\boldsymbol{f}'\right)\mathrm{P}\left(\boldsymbol{f}'|\mathrm{C1.1}\right)}{\mathrm{P}(D_n, \mathrm{C1.2}, \mathrm{C2}|\mathrm{C1.1})}$. We form the prior term $\mathrm{P}(\boldsymbol{f}'|\mathrm{C1.1})$ by conditioning the joint distribution, $\mathrm{P}(\boldsymbol{c}, \boldsymbol{f}')$ on C1.1 given by $\boldsymbol{c} = 0$. $\mathrm{P}(\boldsymbol{f}'|\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{f}'|\Sigma^\mathrm{T}_{\mathrm{cf}'}\Sigma^{-1}_\mathrm{cc}\boldsymbol{c}, \Sigma_{\mathrm{f}'\mathrm{f}'} - \Sigma^\mathrm{T}_{\mathrm{cf}'}\Sigma^{-1}_\mathrm{cc}\Sigma_{\mathrm{cf}'}\right)$ implies $\mathrm{P}(\boldsymbol{f}'|\boldsymbol{c} = 0) = \mathcal{N}(\boldsymbol{f}'|0, \Sigma_{\mathrm{f}'|\mathrm{c}})$.

We implement the likelihood term by adding extra factors to the likelihood in eq. (1) that impose soft constraints representing C1.2 and C2. For C1.2 we use the penalty term $\mathrm{P}([\nabla\nabla f|_{x^*}]_{dd} < 0|\nabla\nabla f|_{x^*}) = \Phi(-[\nabla\nabla f|_{x^*}]_{dd}/\sigma_\mathrm{h})$ and for C2 we add more preference relations between x^* and all training points.

$$\begin{aligned}
&\mathrm{P}(D_n, \mathrm{C1.2}, \mathrm{C2}, |\boldsymbol{f}') \\
&\quad= \prod_{k=1}^{n} \mathrm{P}\left(x^\mathrm{a}_k \succ x^\mathrm{b}_k | f(x^\mathrm{a}_k), f(x^\mathrm{b}_k)\right) \\
&\quad\times \prod_{d=1}^{D} \mathrm{P}([\nabla\nabla f|_{x^*}]_{dd} < 0|[\nabla\nabla f|_{x^*}]_{dd}) \\
&\quad\times \prod_{k=1}^{n} \mathrm{P}\left(x^* \succ x^\mathrm{a}_k | f(x^*), f(x^\mathrm{a}_k)\right) \\
&\quad\times \prod_{k=1}^{n} \mathrm{P}\left(x^* \succ x^\mathrm{b}_k | f(x^*), f(x^\mathrm{b}_k)\right) \\
&\quad= \prod_{k=1}^{n} \Phi(q_k) \prod_{d=1}^{D} \Phi(q^\mathrm{h}_d) \prod_{k=1}^{n} \Phi(q^{\mathrm{a}*}_k) \prod_{k=1}^{n} \Phi(q^{\mathrm{b}*}_k)
\end{aligned} \tag{13}$$

Where $q^h_d = \frac{-[\nabla\nabla f|_{x^*}]_{dd}}{\sigma_h}$, $q^{\mathrm{a}*}_k = \frac{f(x^*) - f(x^\mathrm{a}_k)}{\sqrt{2}\sigma}$ and $q^{\mathrm{b}*}_k = \frac{f(x^*) - f(x^\mathrm{b}_k)}{\sqrt{2}\sigma}$. We use Laplace's approximation to approximate $\mathrm{P}(\boldsymbol{f}'|D_n, \mathrm{C1}, \mathrm{C2})$ as Gaussian,

$$\mathrm{P}(\boldsymbol{f}'|D_n, \mathrm{C1}, \mathrm{C2}) \approx \mathcal{N}\left(\boldsymbol{f}'|\boldsymbol{f}'_\mathrm{MAP}, \left(\Sigma^{-1}_{\mathrm{f}'|\mathrm{c}} + \Lambda_{f'_\mathrm{MAP}}\right)^{-1}\right), \tag{14}$$

where $\boldsymbol{f}'_\mathrm{MAP} = \arg\min_{\boldsymbol{f}'} -\log \mathrm{P}(\boldsymbol{f}'|D_n, \mathrm{C1}, \mathrm{C2})$ and $\Lambda_{\mathrm{f}'_\mathrm{MAP}}$ is the Hessian of $-\log \mathrm{P}(D_n, \mathrm{C1.2}, \mathrm{C2}|\boldsymbol{f}')$ evaluated at $\boldsymbol{f}'_\mathrm{MAP}$.

We compute the first term in eq. (12), $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|\boldsymbol{f}', \mathrm{C1.1})$ by conditioning the joint distribution $\mathrm{P}(\boldsymbol{c}, \boldsymbol{f}', \boldsymbol{f}_\mathrm{t})$ on $\boldsymbol{f}'$ and $\boldsymbol{c} = 0$,

$$\begin{aligned}
\mathrm{P}(\boldsymbol{f}_\mathrm{t}|\boldsymbol{f}', \boldsymbol{c} = 0) = \mathcal{N}\Big(&\boldsymbol{f}_\mathrm{t}|\left(\Sigma^\mathrm{T}_\mathrm{ct}B + \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}D\right)\boldsymbol{f}', \\
&\Sigma_\mathrm{tt} - \begin{bmatrix}\Sigma^\mathrm{T}_\mathrm{ct} & \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}\end{bmatrix}\begin{bmatrix}A & B \\ C & D\end{bmatrix}\begin{bmatrix}\Sigma_\mathrm{ct} \\ \Sigma_{\mathrm{f}'\mathrm{t}}\end{bmatrix}\Big),
\end{aligned} \tag{15}$$

where, $\begin{bmatrix}A & B \\ C & D\end{bmatrix} = \begin{bmatrix}\Sigma_\mathrm{cc} & \Sigma_{\mathrm{cf}'} \\ \Sigma^\mathrm{T}_{\mathrm{cf}'} & \Sigma_{\mathrm{f}'\mathrm{f}'}\end{bmatrix}^{-1}$. We can substitute eq. (15) and eq. (14) into eq. (12) to yield the predictive distribution subject to constraints $C1$ and $C2$.

$$\begin{aligned}
&\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n, \mathrm{C1}, \mathrm{C2}) = \mathcal{N}\Big(\boldsymbol{f}_\mathrm{t}|(\Sigma^\mathrm{T}_\mathrm{ct}B + \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}D)\boldsymbol{f}'_\mathrm{MAP}, \\
&\quad\Sigma_\mathrm{tt} - \begin{bmatrix}\Sigma^\mathrm{T}_\mathrm{ct} & \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}\end{bmatrix}\begin{bmatrix}A & B \\ C & D\end{bmatrix}\begin{bmatrix}\Sigma_\mathrm{ct} \\ \Sigma_{\mathrm{f}'\mathrm{t}}\end{bmatrix} \\
&\quad+\left(\Sigma^\mathrm{T}_\mathrm{ct}B + \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}D\right)\left(\Sigma^{-1}_{\mathrm{f}'|\mathrm{c}} + \Lambda_{\mathrm{f}'_\mathrm{MAP}}\right)^{-1}\left(\Sigma^\mathrm{T}_\mathrm{ct}B + \Sigma^\mathrm{T}_{\mathrm{f}'\mathrm{t}}D\right)^\mathrm{T}\Big).
\end{aligned} \tag{16}$$

We obtain $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n, \mathrm{C1}, \mathrm{C2}, \mathrm{C3})$ by analytically conditioning eq. (16) on the single inequality $f(x^*_m) > (f(x^\mathrm{a}) + f(x^\mathrm{b}))/2$ using the method detailed in [24]. Finally, using eq. (10) we can compute the predictive distributions of preferences given the locations of x^*_m.

To optimize $\alpha_n(x^\mathrm{a}, x^\mathrm{b})$ (line 7, algorithm 1) we construct its gradient by evaluating $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n)$ and $\mathrm{P}(\boldsymbol{f}_\mathrm{t}|D_n, \mathrm{C1}, \mathrm{C2}, \mathrm{C3})$ at test points x^a and x^b as well as points offset by $\delta_x = \pm 0.001$ along each dimension. We then optimize $\alpha_n(x^\mathrm{a}, x^\mathrm{b})$ via gradient ascent.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[3] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[4] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, 2000, pp. 663–670.

[5] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 729–736.

[6] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Human behavior modeling with maximum entropy inverse optimal control." in *AAAI Spring Symposium: Human Behavior Modeling*, 2009, p. 92.

[7] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*. ACM, 2009, pp. 9–16.

[8] W. B. Knox, P. Stone, and C. Breazeal, "Training a robot via human feedback: A case study," in *Social Robotics*. Springer, 2013, pp. 460–470.

[9] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, 2011, pp. 1–7.

[10] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in neural information processing systems*, 2013, pp. 575–583.

[11] R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet, "Programming by feedback," in *International Conference on Machine Learning*, no. 32. JMLR. org, 2014, pp. 1503–1511.

[12] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," in *Advances in neural information processing systems*, 2012, pp. 1133–1141.

[13] D. Gong, J. Yan, and G. Zuo, "A review of gait optimization based on evolutionary computation," *Applied Computational Intelligence and Soft Computing*, vol. 2010, 2010.

[14] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1069–1074.

[15] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian gait optimization for bipedal locomotion," in *International Conference on Learning and Intelligent Optimization*. Springer, 2014, pp. 274–290.

[16] W. Chu and Z. Ghahramani, "Preference learning with gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 137–144.

[17] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 2006, vol. 2, no. 3.

[18] E. Brochu, N. D. Freitas, and A. Ghosh, "Active preference learning with discrete choice data," in *Advances in neural information processing systems*, 2008, pp. 409–416.

[19] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[20] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1809–1837, 2012.

[21] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," in *Advances in Neural Information Processing Systems*, 2014, pp. 918–926.

[22] N. Houlsby, F. Huszar, Z. Ghahramani, and J. M. Hernández-lobato, "Collaborative gaussian processes for preference learning," in *Advances in Neural Information Processing Systems*, 2012, pp. 2096–2104.

[23] E. Solak, R. Murray Smith, W. Leithead, D. Leith, and C. Rasmussen, "Derivative observations in gaussian process models of dynamic systems," *Advances in Neural Information Processing Systems*, pp. 1057–1064, 2003.

[24] L. Xu and X. R. Li, "Estimation and filtering of gaussian variables with linear inequality constraints," in *Information Fusion (FUSION), 2010 13th Conference on*. IEEE, 2010, pp. 1–6.

[25] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[26] N. Thatte and H. Geyer, "Toward balance recovery with leg prostheses using neuromuscular model control," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 5, pp. 904–913, 2016.

[27] E. Brochu, T. Brochu, and N. de Freitas, "A bayesian interactive optimization approach to procedural animation design," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2010, pp. 103–112.