Programming by Feedback

Riad Akrour RIAD.AKROUR@LRI.FR
Marc Schoenauer MARC.SCHOENAUER@INRIA.FR
Jean-Christophe Souplet JCSOUPLET@LRI.FR
Michele Sebag MICHELE.SEBAG@LRI.FR
TAO, INRIA/CNRS/LRI, Université Paris-Sud, 91405 France

Abstract

This paper advocates a new ML-based programming framework, called Programming by Feedback (PF), which involves a sequence of interactions between the active computer and the user. The latter only provides preference judgments on pairs of solutions supplied by the active computer. The active computer involves two components: the learning component estimates the user's utility function and accounts for the user's (possibly limited) competence; the optimization component explores the search space and returns the most appropriate candidate solution. A proof of principle of the approach is proposed, showing that PF requires a handful of interactions in order to solve some discrete and continuous benchmark problems.

1. Introduction

There is emerging evidence that the art of programming could be revisited in the light of the current state of the art in machine learning and optimization. While the notion of formal specifications has been at the core of software sciences for over three decades, the relevance of ML-based approaches has been demonstrated in the domain of pattern recognition since the early 90s. In domains such as e.g. hard combinatorial problems, a new trend dubbed *Programming by Optimization* advocates algorithm portfolios endowed with a control layer such that *determining what works best in a given use context [could be] performed automatically, substituting human labor by computation* (Hoos, 2012). In the domain of e.g. image classification, it is suggested that the *state of the art can be improved by configuring existing techniques better rather than inventing new learning paradigms* (Snoek et al., 2012). All the above approaches rely on learning surrogate models, predicting the performance yielded by a portfolio algorithm or an algorithm configuration depending on the context, and using sequential model-based optimization (Lizotte, 2008) to determine the best configuration for the particular problem instance.

Another trend is investigated in the ML field, specifically aimed at information retrieval (Viappiani & Boutilier, 2010; Yue & Joachims, 2009; Shivaswamy & Joachims, 2012) and reinforcement learning (Wilson et al., 2012; Akrour et al., 2012; Knox et al., 2013; Jain et al., 2013) (more in section 2). This trend, thereafter generally referred to as *Interactive Learning and Optimization* (ILO), implements an iterative 2-step process, with the active computer providing some input to the user, and the user providing a response (preference judgment, modification or suggestion to the active computer) until getting a satisfactory solution. In all of the above, the user and the active computer cooperate and achieve some division of labor between the exploration and the exploitation parts of the search process.

In practice however, ILO faces critical difficulties during the beginning and the ending phases of a run. In the beginning – the bootstrapping phase – the input provided by the active computer is hardly relevant and the user is easily driven into considering that all active computer suggestions are equally irrelevant. In the ending phase inversely, very relevant suggestions are provided and the user might easily make judgment errors; according to e.g. the Plackett-Luce model (Luce, 1959), the probability of misranking two solutions exponentially increases as their difference in quality decreases. In both phases, the user might be deterred by getting insufficient returns, inducing him to behave in an inconsistent way, and getting even more inconsistent returns as a result.

It has long been suggested that the cooperation between intelligent partners is better supported by each partner having a model of the other one (see e.g. Lörincz et al., 2007). A step toward such a cooperation between the active com-

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

puter and the user is presented in this paper, referred to as *Programming by Feedback* (PF) framework. PF addresses the above ILO limitation by adaptively approximating the user's utility function and accounting for her inconsistencies. Note that the active computer cannot make any difference between the user's competence (her preferences are consistent with her goal) and the user's consistency (her preferences are consistent); at any rate, accounting for the user's possibly limited competence enables the active computer to better exploit the user's instant feedback.

The paper is organized as follows. After discussing related work (section 2), an overview of PF is presented in section 3. Section 4 provides a proof-of-concept of the approach, showing that PF requires a handful of interactions to solve state-of-art benchmark problems in simulation, and to achieve on-board programming of the Nao robot (Aldebaran, 2013). The paper concludes with perspectives for further research.

2. Related Work

Interactive Learning and Optimization has mostly been applied to information retrieval and reinforcement learning. Still, to our best knowledge the ILO frame was first investigated by Brochu et al. (2007; 2010) to achieve interactive optimization.

2.1. ILO for Interactive Optimization

Brochu et al. (2007) are interested in exploring the image synthesis hyper-parameter search space and finding as quickly as possible a suitable visual rendering. The user is placed in the loop due to the lack of any computable function characterizing satisfactory visual rendering[1]. The user utility function is learned as a Gaussian process using a binomial probit regression model, trained from the user's ranking of pairs of visual renderings. The optimization component proceeds by returning the best solution out of a finite sample of the search space, according to the expected improvement criterion over the current best solution. While the authors are content with a satisfactory solution, they also note that finding the optimal solution, e.g. using the expected global improvement criterion (Jones et al., 1998) with a branch-and-bound method, raises technical issues in large search spaces.

2.2. ILO for Information Retrieval

Two early ILO approaches focussing on information retrieval are (Yue & Joachims, 2009) and (Viappiani & Boutilier, 2010). Yue & Joachims (2009) assume the existence of a parametric utility function on the information retrieval search space, defined from parameter vector $\mathbf{w}^*$. Iteratively, the current estimate $\mathbf{w}_t$ is compared to a perturbation $\mathbf{w}'_t$ thereof. The feedback – whether $\mathbf{w}'_t$ improves on $\mathbf{w}_t$, i.e. derives a better utility function – is provided via the interleaving method (Radlinski et al., 2008), thus considering an ensemble of users. This feedback is interpreted as an estimate of the performance gradient at $\mathbf{w}_t$, with a sublinear convergence toward $\mathbf{w}^*$ within the so-called dueling bandit framework.

In (Viappiani & Boutilier, 2010), the goal is to determine a basket of items optimally suited to the user. The system iteratively provides the user with a choice query, that is a set of solutions, of which the user selects the one she prefers. The ranking constraints are used to learn a parametric utility function $\mathbf{w}_t$, linear on the search space $X = \mathbb{R}^D$. Within the Bayesian setting, the uncertainty about the utility function is expressed through a belief θ defining a distribution over the space of possible utility functions, equivalent to the unit sphere of $\mathbb{R}^D$. The optimal next query ideally maximizes the Expected Posterior Utility of selection (EPU); a much cheaper criterion, the Expected Utility of Selection (EUS) can however be used, at the expense of a bounded loss (Viappiani & Boutilier, 2010) (more in Section 3).

2.3. ILO for Reinforcement Learning

RL-ILO resumes the celebrated Inverse Reinforcement Learning (IRL) approach (Abbeel, 2008; Konidaris et al., 2010), except for one key difference. In IRL the active part is the expert's, demonstrating a few target behaviors. Quite the contrary, in ILO the behaviors are demonstrated by the agent and thereafter ranked by the expert, thereby relaxing the strong expertise requirement of RL and IRL: the expert is only required to comparatively assess the agent behaviors, as opposed to, demonstrate a (near) optimal behavior.

In (Fürnkranz et al., 2012) the motivation is to extend the RL scope beyond the use of numerical rewards. For instance in medical application domains, a numerical reward must be associated to events such as the patient death; while this reward value is admittedly fairly arbitrary, its impact on the optimal policy is hard to investigate, except by trial and error. The proposed alternative is to use preference learning in replacement of classification algorithms in roll out classification-based policy iteration (RCPI) (Dimitrakakis & Lagoudakis, 2008), thereby ordering the actions conditionally to a state and a policy. With same motivations, the TAMER framework (Knox et al., 2013) learns the reward function on (state action) pairs from the expert's feedback, using a credit assignment on the recent state-action pairs to account for the delays in human evaluation.

In (Wilson et al., 2012; Akrour et al., 2012), an active preference-based policy learning (PPL) scheme is pro-

[1] See also (Lin et al., 2010) in the domain of interactive text summarization.

posed, exploiting pairwise preferences among trajectories demonstrated by the agent. While PPL relaxes the strong expertise requirement of RL and IRL, it still assumes dedicated experts, taking the time to look at the agent demonstrations and reliably rank them. The expert burden is limited in (Wilson et al., 2012) by only considering short demonstrations, assuming that i) the initial states can be sampled after some prior distribution in order to enforce inspecting interesting regions of the behavioral space; ii) the user can compare sub-behaviors. In (Akrour et al., 2012), an active selection of the demonstrations is used, by optimizing an expected posterior utility criterion inspired from (Viappiani & Boutilier, 2010), reducing the number of demonstrations required to achieve a satisfactory behavior.

Both approaches suffer from different limitations. Wilson et al. (2012) assume that quite some expertise about the problem domain is available, through an informative prior about interesting initial states. Akrour et al. (2012) require the expert to look at long demonstrations, increasing the chances of ranking noise.

Another scheme called co-active policy learning (CPL) is presented by Jain et al. (2013). Like (Wilson et al., 2012; Akrour et al., 2012), CPL proceeds by asking the expert to iteratively rank pairs of demonstrations of the agent. The expert however plays a more active role in CPL than in PPL, either re-ranking the suggested trajectories or directly refining the agent demonstrations along the co-active mechanism (Shivaswamy & Joachims, 2012). Formally, the expert is in charge of the exploration part, while the learning agent does the exploitation part (extracting a model of the expert's preferences, a.k.a. utility function, and optimizing it to the extent permitted by computational constraints). CPL thus avoids the main bottleneck of interactive optimization, that is, getting trapped into a local optimum due to insufficient exploration, by delegating the exploration component to the human expert.

2.4. Discussion

The ILO schemes implemented in IRL, PPL and CPL achieve different trade-offs among two interdependent issues: the level of expertise required from the expert and the level of autonomy endowed to the agent. In decreasing expertise order, the expert is required: i) to associate a reward to any state in the state space (RL); ii) to demonstrate a solution behavior (IRL); iii) to correct and improve an agent demonstration (CPL); iv) to rank two demonstrations (PPL,CPL).

Meanwhile, in increasing autonomy order, the agent: i) computes the optimal policy based on the instant rewards (RL); ii) imitates *verbatim* the expert's demonstration (IRL); iii) imitates the expert's demonstrations with some generalization and variants (IRL); iv) learns the expert's utility function and produces a (near) optimal demonstration according to this function (CPL, IRL); v) learns the expert's utility function and produces a (near) optimal demonstration in the sense of the information gained about the utility function and its impact on the overall solution quality (expected posterior utility) (PPL).

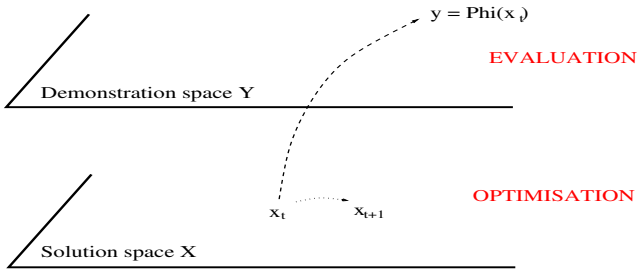


Figure 1. Programming by Feedback: Solution and demonstration search spaces

Note that the ILO goal significantly differs from that of active ranking: active ranking is interested in learning the true preference function, whereas ILO only aims at its optimum.

A third issue regards the modelling of the uncertainty and priors on the expert's utility function. In (Brochu et al., 2007), the utility function is learned as a GP, thus providing both a utility estimate and the confidence thereof. Interestingly however, Brochu et al. (2010) note that ILO being a "small data" problem, does not provide enough evidence to accurately tune the GP hyper-parameters. The process thereby suffers a loss of performance due to under-optimal utility estimate.

3. Programming by Feedback

This section presents the PF framework, introducing the notations and the overview of the algorithm, before describing the learning and optimization components.

3.1. Overview

Two search spaces are distinguished (Fig. 1): the search space $\mathcal{X}$ (analogous to the policy space in the RL context) is referred to as solution space. The evaluation space $\mathcal{Y}$ (analogous to the trajectory space in RL) is referred to as demonstration space. The user only sees a realization $\mathbf{y} = \Phi(\mathbf{x})$ of a program $\mathbf{x}$. In contrast with the direct policy learning setting (e.g. Deisenroth et al., 2013), no continuity assumption regarding the stochastic mapping Φ from the solution to the demonstration space can be assumed (clearly, small modifications of a program can result in significantly different behaviors).

The true (unknown) user's utility function U is a linear

Algorithm 1 Programming by Feedback

Input: prior θ_0 on the utility function space $\mathbf{W}$
Init:
Generate $\mathbf{x}_0$ in $\mathcal{X}$
Demonstrate $\mathbf{y}_0 = \Phi(x_0)$; $\mathbf{y}_0^* = \mathbf{y}_0$
Archive $\mathcal{U}_t = \{\mathbf{y}_0\}$
repeat
  Optimize: Generate $\mathbf{x}_{t+1}$ from θ_t (Section 3.4
  Demonstrate $\mathbf{y}_{t+1} = \Phi(x_{t+1})$Discussion
  Receive the user's feedback $1_{\mathbf{y}_{t+1} \succ \mathbf{y}_t^*}$
  Update $\mathcal{U}_t = \mathcal{U}_t \cup \{\mathbf{y}_{t+1}, 1_{\mathbf{y}_{t+1} \succ \mathbf{y}_t^*}\}$
  Learn posterior θ_{t+1} from $\mathcal{U}_t$ (Section 3.2)
until User stops
Return: Generate $\mathbf{x}$ from θ

function on demonstration space $\mathcal{Y}$, parameterized as $\mathbf{w}^*$:

$$U(\mathbf{y}) = \langle \mathbf{w}^*, \mathbf{y} \rangle$$

Like ILO, PF is an iterative two-step process. At time step t, the active computer i) selects a solution $\mathbf{x}_t$ and demonstrates $\mathbf{y}_t = \Phi(\mathbf{x}_t)$; ii) receives the user's feedback, ranking $\mathbf{y}_t$ comparatively to the previous best demonstration[2] noted $\mathbf{y}_t^*$; iii) updates its model of the user's utility function.

3.2. Learning the Utility Function

Let $\mathbf{W}$ denote the space of normalized linear functions on $\mathcal{Y}$. Given a uniform prior on $\mathbf{W}$, the active computer learns by computing its posterior distribution θ_t from evidence $\mathcal{U}_t = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_t; (\mathbf{y}_{i_1} \succ \mathbf{y}_{i_2}), i = 1 \ldots t\}$ including all demonstrations and user's preferences up to the t-th time step. Given θ_t, the estimated utility $U(\theta_t, \mathbf{y})$ of a demonstration $\mathbf{y}$ is defined as:

$$U(\theta_t, \mathbf{y}) = \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} \rangle] \tag{1}$$

As already said, the utility function must account for the preference noise. Given two demonstrations $\mathbf{y}$ and $\mathbf{y}'$, and $\mathbf{w}^*$ denoting the true (unknown) utility of the user, the observed user's preference is usually modeled as a perturbation of his true preference $\langle \mathbf{w}^*, (\mathbf{y} - \mathbf{y}') \rangle$, considering a Gaussian perturbation (Chu & Ghahramani, 2005; Wilson et al., 2012) or following the Luce-Shepard model (Luce, 1959; Shepard, 1957). In both cases, the noise magnitude is calibrated using one hyper-parameter, respectively the standard deviation of the Gaussian perturbation or the temperature of the Luce-Shepard rule.

A ridge noise model is considered in PF, calibrated from a hyper-parameter δ ($\delta \in \mathbb{R}^+$). Given the preference margin $\mathrm{z} = \langle \mathbf{w}^*, (\mathbf{y} - \mathbf{y}') \rangle$, the observed preference is bound to coincide with the true preference if $|\mathrm{z}| > \delta$; otherwise the observed preference $\mathbf{y} \succ \mathbf{y}'$ is set to 1 with probability $\frac{\delta + \mathrm{Z}}{2\delta}$.

$$P(\mathbf{y} \succ \mathbf{y}' \mid \mathbf{w}^*, \delta) = \begin{cases} 0 & \text{if } \mathrm{z} < -\delta \\ 1 & \text{if } \mathrm{z} > \delta \\ \frac{\delta + \mathrm{Z}}{2\delta} & \text{otherwise} \end{cases} \tag{2}$$

The user's competence is thus modeled through parameter δ. A first option is to consider that the user's competence is characterized by a hidden but fixed δ^*. A second option is to consider that δ might vary along time. The drawback of the former option is that one abnormally large mistake can prevent the active computer from identifying an otherwise consistent utility function. Inversely, the latter option being more cautious could slow down the identification of the user's utility function. This latter option is however more appropriate to accommodate the cases where the task (or the user's understanding thereof, or his preferences) might change over time. Not only can the user change his mind; in the general case, one PF run might involve several users, responsible for the successive feedbacks.

Working hypothesis. For the sake of robustness, the noise parameter δ is assumed to be uniform on $[0, M]$.

Under these assumptions, the posterior distribution θ_t on the utility function space can be expressed in closed form.

Lemma 1 Given evidence $\mathcal{U}_t$, the posterior distribution on the utility function space reads:

$$\begin{array}{rcl} \theta_t(\mathbf{w}) & \propto & \prod_{i=1,t} P(\mathbf{y}_{i_1} \succ \mathbf{y}_{i_2} \mid \mathbf{w}) \\ & = & \prod_{i=1,t} \left(\frac{1}{2} + \frac{\mathrm{Z}_i(\mathbf{w})}{2M} \left(1 + \log \frac{M}{|\mathrm{Z}_i(\mathbf{w})|} \right) \right) \end{array} \tag{3}$$

where $\mathrm{z}_i(\mathbf{w})$ is set to $\langle \mathbf{w}, (\mathbf{y}_{i_1} - \mathbf{y}_{i_2}) \rangle$ if it is not greater (resp. lower) than M (resp. $-M$) in which case it takes the value M (resp. $-M$).

Proof: For a given $\mathcal{U}_t$ and $\mathbf{w}^*$, for any y, y' with $z = \langle \mathbf{w}, (\mathbf{y} - \mathbf{y}') \rangle$, and using the ridge model (2), it comes

$$\begin{array}{l} \int_0^M P(\mathbf{y} \succ \mathbf{y}' \mid \mathbf{w}^*, \delta) d\delta \\ = \int_0^{|z|} P(\mathbf{y} \succ \mathbf{y}' \mid \mathbf{w}^*, \delta) d\delta + \int_{|z|}^M P(\mathbf{y} \succ \mathbf{y}' \mid \mathbf{w}^*, \delta) d\delta \\ = \int_0^{|z|} 1_{z>0} d\delta + \int_{|z|}^M (\frac{1}{2} + \frac{z}{2\delta}) d\delta \\ = |z| \cdot 1_{z>0} + \frac{1}{2}(M - |z|) + \frac{z}{2}(\log(M) - \log(|z|)) \\ = \frac{1}{2}(M + z) + \frac{z}{2} \log(\frac{M}{|z|}) \end{array}$$

hence the result, normalizing total mass from M to 1. ■

Comparatively to the Gaussian or the Luce-Shepard noise models, the ridge noise model enables a straightforward in-

[2] Two settings are considered, referred to as oblivious and non-oblivious. In the oblivious setting, the user is presented with two demonstrations in each time step, and ranks them comparatively to each other. In the non-oblivious setting, the user is presented with a single demonstration in each iteration, which he ranks comparatively to (his memory of) the best former demonstration. The non-oblivious setting has been considered in the experiments as it is less demanding for the user.

terpretation and calibration: the upper bound M is homogeneous to a utility, as there is no noise when the preference margin is higher than M.

3.3. Optimization in the Demonstration Space

Conditionally to $\mathcal{U}_t$, the expected utility of selection of a pair of demonstrations $\mathbf{y}, \mathbf{y}'$ to be demonstrated to the user is defined as follows, where θ_t^+ (respectively θ_t^-) denotes the posterior distribution on $\mathcal{Y}$ based on evidence $\mathcal{U}_t \cup \{(\mathbf{y} \succ \mathbf{y}')\}$ (resp $\mathcal{U}_t \cup \{(\mathbf{y} \prec \mathbf{y}')\}$):

$$\begin{aligned} EUS(\mathbf{y}, \mathbf{y}') &= \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle > 0] \,.\, U(\theta_t^+, \mathbf{y}) \\ &+ \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle < 0] \,.\, U(\theta_t^-, \mathbf{y}') \end{aligned} \quad (4)$$

Viappiani & Boutilier (2010) advocate the use of the expected posterior utility, defined as the expected utility of the best behavior $\mathbf{y}^*$ (respectively $\mathbf{y}'^*$) conditionally to the user preferring $\mathbf{y}$ to $\mathbf{y}'$ (resp. $\mathbf{y}'$ to $\mathbf{y}$):

$$\begin{aligned} EPU(\mathbf{y}, \mathbf{y}') &= \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle > 0] \,.\, max_{\mathbf{y}} U(\theta^+, \mathbf{y}) \\ &+ \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle < 0] \,.\, max_{\mathbf{y}} U(\theta^-, \mathbf{y}) \\ &= \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle > 0] \,.\, U(\theta^+, \mathbf{y}^*) \\ &+ \mathbb{E}_{\theta_t}[\langle \mathbf{w}, \mathbf{y} - \mathbf{y}' \rangle < 0] \,.\, U(\theta^-, \mathbf{y}'^*) \end{aligned} \quad (5)$$

By construction $EUS(\mathbf{y}, \mathbf{y}') \leq EPU(\mathbf{y}, \mathbf{y}')$; and $EPU(\mathbf{y}, \mathbf{y}') \leq EUS(\mathbf{y}^*, \mathbf{y}'^*)$ (Viappiani & Boutilier, 2010). The optimization of the highly computationally demanding EPU criterion (since computing EPU(**y**,**y**') requires solving two optimization problems) can therefore be replaced with the optimization of the EUS criterion:

$$\max\{EUS(\mathbf{y}, \mathbf{y}')\} = \max\{EPU(\mathbf{y}, \mathbf{y}')\}$$

The proposed noise model has an impact on both the EUS and the EPU criteria. Let us first bound its impact on the EUS criterion. In the following, the *noiseless, NL* (respectively, *noisy, N*) subscript indicates that $\mathbf{y}$ is preferred to $\mathbf{y}'$ for every utility $\mathbf{w}$ such that $\langle \mathbf{w}, \mathbf{y} \rangle > \langle \mathbf{w}, \mathbf{y}' \rangle$ (resp. with probability defined by Eq. 2). Note that in both cases, the expectation is taken w.r.t. the posterior distribution on the $\mathbf{W}$ utility space given by Eq. 3, thus accounting for the user's noise. Then,

Lemma 2 For any pair of demonstrations $\mathbf{y}, \mathbf{y}'$, and for any archive $\mathcal{U}_t$, the difference between their expected utility of selection under the noisy and noiseless models is bounded as follows:

$$EUS^{NL}(\mathbf{y}, \mathbf{y}') - L \leq EUS^{N}(\mathbf{y}, \mathbf{y}') \leq EUS^{NL}(\mathbf{y}, \mathbf{y}')$$

With $L = \frac{M}{2\lambda}(1 - \frac{1+ln\lambda}{\lambda})$, $\lambda = e^{-\frac{1}{2} - W_{-1}(-\frac{1}{2}e^{-\frac{1}{2}})}$ and W_{-1} is the lower branch of the Lambert W function. Approximatively, $L \approx \frac{M}{19.6433}$.

Proof: from Lemma 1 and (Viappiani & Boutilier, 2010).

Note that the lower bound is tight; it corresponds to the adverse case where the preference margin is intermediate: sufficiently low to entail a high chance of error and sufficiently high to entail a significant loss of utility. The impact of the noise model on the EPU criterion is finally bounded as follows. Let $EPU_t^{*,N}$ (respectively $EUS_t^{*,N}$) denote the optimum of the noisy EPU (resp. EUS) criterion conditionally to $\mathcal{U}_t$; let likewise $EUS_t^{*,NL}$ denote the optimum of the noiseless EUS. Then:

Proposition 1

$$EUS_t^{*,NL} - L \leq EPU_t^{*,N} \leq EUS_t^{*,NL} + L$$

Proof: One first shows using Lemma 1 and adapting the proof sketch in (Viappiani & Boutilier, 2010), that

$$EUS_t^{*,N} \leq EPU_t^{*,N} \leq EUS_t^{*,N} + L$$

The result then follows from Lemma 2.
PF thus proceeds by optimizing the noiseless EUS with a bounded loss of performance compared to the noisy EPU.

3.4. Optimization in the Solution Space

At time step t, PF tackles the optimization problem defined on $\mathcal{X}$ as:

$$\text{Find argmax } \mathcal{F}_t(\mathbf{x}) = \mathbb{E}_{\Phi}[EUS_t^{NL}(\Phi(\mathbf{x}), \mathbf{y}_t^*)] \quad (6)$$

with EUS_t^{NL} the expected utility of selection under θ_t (where the user's noise is taken into account when updating the posterior θ_t) and $\mathbf{y}_t^*$ the best-so-far demonstration. As discussed in Section 3.1, this non-oblivious setting is meant to decrease the computational cost and the user's cognitive fatigue. While the performance loss compared to the oblivious setting is limited[3], its bounding is left for further work (section 5).

A main issue is that Eq. 6 defines a noisy black-box optimization problem: On the one hand, $\mathcal{F}_t(\mathbf{x})$ is defined as an expectation; on the other hand, as Φ is not expressed in closed form in the general case, the expectation needs be approximated by an empirical average over demonstrations drawn from $\Phi(\mathbf{x})$.

Two steps are taken to handle this optimization problem. Firstly, $\mathcal{F}_t$ is replaced by a lower bound thereof. The expected utility of selection of the average demonstration $\mathbb{E}[\Phi(\mathbf{x})]$, noted $\bar{\mathbf{y}}$ in the following, is a lower bound of the utility of selection expectation on $\Phi(\mathbf{x})$, due to the convexity of the max operator on $\mathbf{W}$ and the Jensen inequality:

$$\mathbb{E}_{\Phi}[EUS^{NL}(\Phi(\mathbf{x}), \mathbf{y}_t^*)] \geq EUS^{NL}(\bar{\mathbf{y}}, \mathbf{y}_t^*)$$

[3] Due to the sub-modularity of EUS^{NL}, selecting $\mathbf{y}_0$ with maximal expected utility and $\mathbf{y}_1 = \text{argmax} EUS^{NL}(\mathbf{y}_0, \mathbf{y})$ yields a bounded loss compared to the maximization of $EUS^{NL}(\mathbf{y}, \mathbf{y}')$; and by construction $\mathbf{y}_t^*$ has a high expected utility.

Secondly, a sequence of solutions with increasing $EUS_t^{NL}(\bar{\mathbf{y}}, \mathbf{y}_t^*)$ is built as follows. Let $\mathbf{x}_1$ be the solution with optimal expected utility according to some $\mathbf{w}_0$ drawn according to θ_t:

$$\mathbf{x}_1 = \text{ argmax } \{\mathcal{G}_1(\mathbf{x}) = \langle \mathbf{w}_0, \bar{\mathbf{y}} \rangle\}$$

For $i \geq 1$, let $\bar{\mathbf{y}}_i$ denote the average demonstration of $\mathbf{x}_i$, and θ_i denote the posterior on $\mathbf{W}$ from $\mathcal{U}_t \cup \{(\bar{\mathbf{y}}_i \succ \mathbf{y}_t^*)\}$; the $i+1$-th optimization problem is defined as:

$$\mathbf{x}_{i+1} = \text{ argmax } \{\mathcal{G}_{i+1}(\mathbf{x}) = \langle \mathbb{E}_{\theta_i}[\mathbf{w}], \bar{\mathbf{y}} \rangle\}$$

Proposition 2
The expected utility of selection of $(\bar{\mathbf{y}}_i, \mathbf{y}_t^*)$ monotonically increases with i:

$$EUS_t^{NL}(\bar{\mathbf{y}}_i, \mathbf{y}_t^*) \leq EUS_t^{NL}(\bar{\mathbf{y}}_{i+1}, \mathbf{y}_t^*)$$

Proof: By construction of $\mathbf{x}_{i+1}$,

$$\int_{\mathbf{W}, \bar{\mathbf{y}}_i \succ \mathbf{y}_t^*} \langle \mathbf{w}, \bar{\mathbf{y}}_i \rangle \mathrm{d}\theta_t(\mathbf{w}) \leq \int_{\mathbf{W}, \bar{\mathbf{y}}_i \succ \mathbf{y}_t^*} \langle \mathbf{w}, \bar{\mathbf{y}}_{i+1} \rangle \mathrm{d}\theta_t(\mathbf{w})$$

Hence (omitting domain $\mathbf{W}$ for clarity)

$$\begin{array}{l} EUS_t^{NL}(\bar{\mathbf{y}}_i; \mathbf{y}_t^*) \\ = \int \max(\langle \mathbf{w}, \bar{\mathbf{y}}_i \rangle, \langle \mathbf{w}, \mathbf{y}_t^* \rangle) \mathrm{d}\theta_t(\mathbf{w}) \\ = \int_{\bar{\mathbf{y}}_i \succ \mathbf{y}_t^*} \langle \mathbf{w}, \bar{\mathbf{y}}_i \rangle \mathrm{d}\theta_t(\mathbf{w}) + \int_{\bar{\mathbf{y}}_i \prec \mathbf{y}_t^*} \langle \mathbf{w}, \mathbf{y}_t^* \rangle \mathrm{d}\theta_t(\mathbf{w}) \\ \leq \int_{\bar{\mathbf{y}}_i \succ \mathbf{y}_t^*} \langle \mathbf{w}, \bar{\mathbf{y}}_{i+1} \rangle \mathrm{d}\theta_t(\mathbf{w}) + \int_{\bar{\mathbf{y}}_i \prec \mathbf{y}_t^*} \langle \mathbf{w}, \mathbf{y}_t^* \rangle \mathrm{d}\theta_t(\mathbf{w}) \\ \leq \int \max(\langle \mathbf{w}, \bar{\mathbf{y}}_{i+1} \rangle, \langle \mathbf{w}, \mathbf{y}_t^* \rangle) \mathrm{d}\theta_t(\mathbf{w}) \\ = EUS_t^{NL}(\mathbf{y}_{i+1}, \mathbf{y}_t^*) \end{array} \quad \blacksquare$$

As this sequence of optimization problems leads to a local optimum of EUS_t^{NL}, multiple restarts are used with different samples $\mathbf{w}_0$ and the best solution is retained.

Overall, the elementary optimization component in PF is concerned with optimizing $\langle \bar{\mathbf{w}}, \bar{\mathbf{y}} \rangle$ on $\mathcal{X}$. Different implementations of this optimization component are considered depending on $\mathcal{X}$ (next Section). As $\bar{\mathbf{w}}$ is the average utility function under distribution θ_t, this requires to integrate over $\mathbf{W}$ once, thus with tractable cost. In the experiments, less than 10 iterations are needed to find a local optimum, and 10 multiple restarts were considered.

4. Experimental results

A proof of concept of the PF framework is presented and discussed comparatively to (Wilson et al., 2012) on well studied benchmark RL problems, on discrete and continuous solution spaces $\mathcal{X}$, with a generative model (the bicycle problem) or without (the cartpole, the gridworld, the Nao robot). The computational time is less than 1 minute per run on a 2.4Ghz Intel processor for all problems except the Nao problem (10 mns).

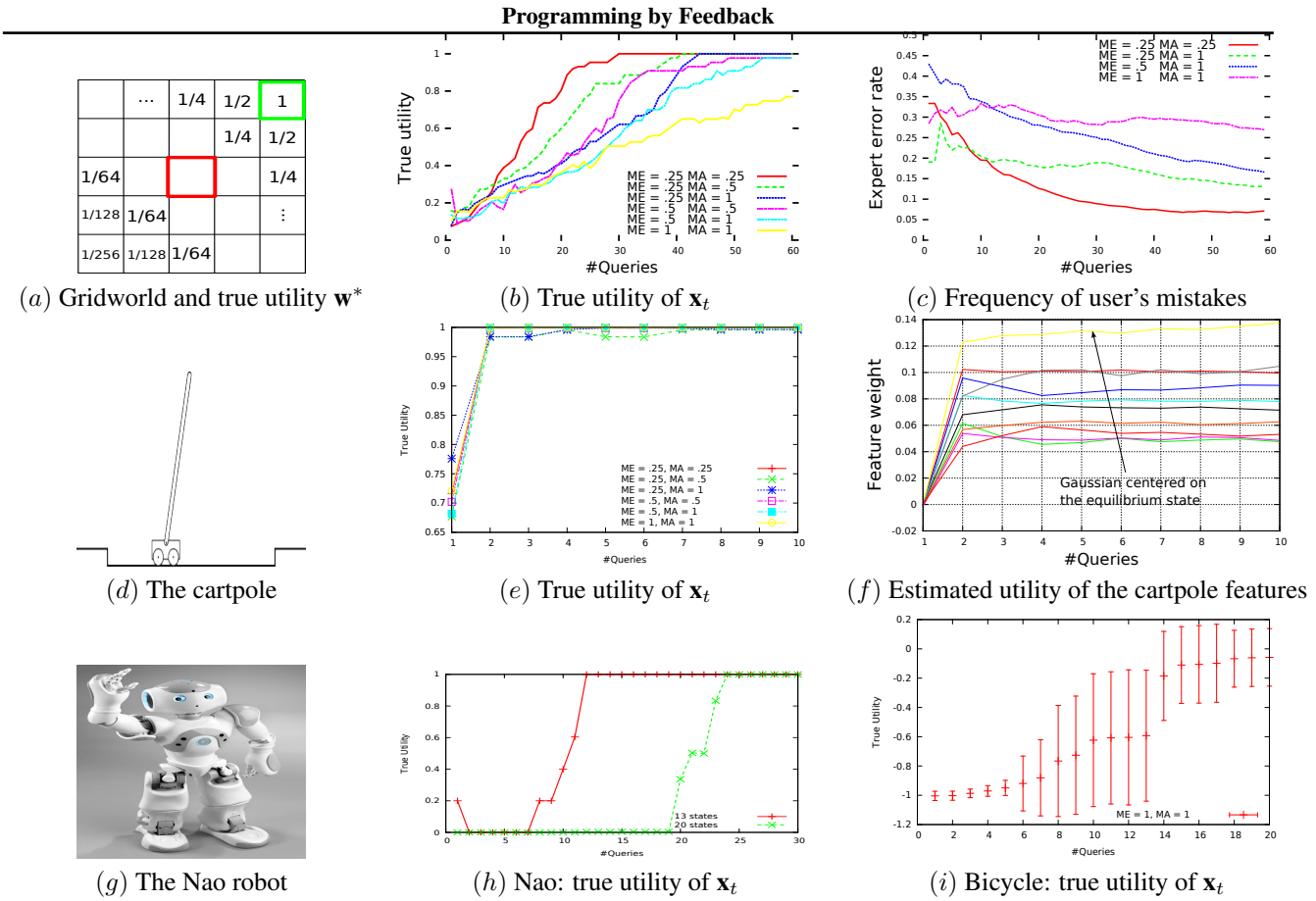
4.1. Discrete Case, no Generative Model

A stochastic **grid world** problem is considered, with 25 states and 5 actions (up, down, left, right or stay motionless). The transition model involves a 50% probability of staying motionless (100% if the selected action would send the agent in the wall). It is estimated from 1,000 random triplets. The reward function (true utility $\mathbf{w}^*$) is shown in Fig. 2.(a). The core optimization component (Section 3.4) implements a vanilla policy iteration algorithm, with $\gamma = .95$. Time horizon is set to $H = 300$. Results are averaged over 21 runs.

A first goal of experiment is to analyze the sensitivity of the PF framework w.r.t. the hyper-parameters of the noise model. The user's feedback is emulated using hyper-parameter M_E (the higher M_E, the less competent the user); M_A is the hyper-parameter of the user's noise model estimated by the active computer (the higher M_A, the more the active computer underestimates the user's competence), with M_E and M_A ranging in $\{1, .5, .25\}$ s.t. $M_A \geq M_E$. The performance indicator is the true utility of policy $\mathbf{x}_t$ in each PF time step (unknown to the active computer). Fig. 2.(b) shows that the performance primarily depends on the user's competence M_E, and secondarily on the active computer estimate M_A of the user's competence. However, Fig. 2.(c) shows that the number of the emulated user's mistakes most surprisingly increases as the active computer underestimates the user's competence (high M_A), irrespective of the user's true competence M_E. This unexpected finding is explained as the error rate does not only depend on the user's competence but also on the relevance of the demonstrations he is provided with. For $M_A = .25$, the active computer learns faster, thus submitting more relevant demonstrations to the user, thus priming a virtuous educational process. This also explains the fast decrease of the error rate for $M_E = .25, M_A = .25$ which seems to follow a different regime (empirically faster than linear) than $M_E = .25, M_A = 1$. Fig 2.(c) also supports the claim that PF most critical stage is the initial one, where the demonstrated trajectories are of low utility, making them harder to compare and increasing the probability of user's mistakes. The target behavior is reached in 30 PF interactions in a 25-state space. This relatively slow convergence is blamed on the poor representation of the demonstrations, associating to a demonstration the time spent in each state and thus poorly reflecting the adjacency structure of the state space.

A second experiment (discrete space, no simulator) trains the **Nao robot**, a 58 cm high humanoid robot (Aldebaran, 2013) to reach a target state (e.g., raise hand, see Fig. 2.(g)). The transition matrix is estimated from 1,000 random (s, a, s') triplets. The trajectory length is 10; the initial state is fixed.

The action space includes 3 actions. Fig. 2.(h) (average



(a) Gridworld and true utility $\mathbf{w}^*$ (b) True utility of $\mathbf{x}_t$ (c) Frequency of user's mistakes

(d) The cartpole (e) True utility of $\mathbf{x}_t$ (f) Estimated utility of the cartpole features

(g) The Nao robot (h) Nao: true utility of $\mathbf{x}_t$ (i) Bicycle: true utility of $\mathbf{x}_t$

Figure 2. Programming by Feedback. Top row: the gridworld problem, Influence of the noise hyper-parameters M_A and M_E (averaged over 21 runs). Medium row: the cartpole problem (averaged over 48 runs). Bottom row: The Nao robot (left, averaged over 5 runs); The bicycle (right, averaged over 21 runs). All plots are enlarged in the supplementary material.

results on 5 runs) shows that 10 PF interactions are required to reach a target state in a 13-state space (the shortest demonstration reaching the target state is 5-action long), versus 24 interactions for a 20-state space (the shortest demonstration reaching the target state is 10-action long).

4.2. Continuous Case, no Generative Model

The **cartpole problem** is concerned with balancing a pole fixed to a movable cart (Fig. 2.(d)). Same setting as in (Lagoudakis & Parr, 2003) is used, with state space $\mathbb{R}^2$ (the angle and angular velocity of the pendulum) and 3 actions; the demonstration space $\mathcal{Y}$ is $\mathbb{R}^9$, where each feature corresponds to a Gaussian $N(\mu, \Sigma)$ in the $\mathbb{R}^2$ state space. The transition model is estimated from 33,000 (s, a, s') triplets. The user's feedback is emulated by considering that the best demonstration is the longest one, subject to the noise model with hyper-parameter M_E. Each demonstration $\mathbf{y}$ is represented in $\mathbb{R}^9$ by computing for each feature the discounted sum of $\sum_s \gamma^s p(u(s)|N(\mu, \Sigma))$, where $u(s)$ is the cartpole state at time s. This discounted sum is computed using LSTD (with reward $r(u) = p(u|N(\mu, \Sigma))$. The PF core optimization component (Section 3.4) implements LSPI (Lagoudakis & Parr, 2003).

The demonstration length is 3,000. The true utility is the fraction of the demonstration where the cartpole is in equilibrium.

The results show that only two PF interactions are required on average to solve the cartpole problem, irrespective of the noise model hyper-parameters M_A and M_E (Fig. 2.(e)). These results favorably compare to Wilson et al. (2012), where ca 15 queries are required to maintain the cartpole in equilibrium more than 1,400 time steps. Fig. 2.(f), displaying the estimated utility vector in parallel coordinates, shows that the feature closest to the equilibrium position gets the highest weight.

4.3. Continuous Case, with Generative Model

The **bicycle problem** is concerned with riding the bicycle and preventing it from falling down. The generative model is the simulator used by Lagoudakis & Parr (2003). The state space is $\mathbb{R}^4$ (the angle and angular velocity of the bi-

cycle, the angle and angular velocity of the handlebars); the action space is $\mathbb{R}^2$ (the torque applied to the handlebars and the displacement of the rider on the saddle). The solution space $\mathcal{X}$ is set to $\mathbb{R}^{210}$ (weight vector of a 1-layer feedforward NN with 4 input, 29 hidden neurons and 2 output).

The maximum demonstration length is 30,000 time steps; however controllers tend to either fall down before 300 time steps, or to maintain the bicycle until the end of the trajectory. The true utility is 1 - the squared angle of the bicycle, averaged on the whole demonstration.

The goal of the experiments is to investigate the scalability of the approach w.r.t. the dimensionality of the solution space. The PF core optimization component (Section 3.4) implements the CMA-ES black-box optimization algorithm[4] (Hansen & Ostermeier, 2001).

The results show that 15 PF interactions are required on average to solve the bicycle problem for the low noise setting ($M_E = M_A = 1$), and that the results gracefully degrade as the noise increases (Fig. 2.(i)). These results favorably compare to those in (Wilson et al., 2012), where circa 20 queries are required to reach the equilibrium although the setting only involves 5 discrete actions. The improvement is explained as (Wilson et al., 2012) operates in the policy parameter space whereas PF operates on the trajectory utility space, thus reducing[5] the number of parameters by a factor 5.

5. Conclusion and Perspectives

The main contribution of the paper is to propose a new programming paradigm, where the active computer programs itself using a few binary feedback from the human user. A proof of concept of the validity of the PF approach is provided on the cart-pole, bicycle and grid-world problems. Compared to (Wilson et al., 2012; Knox et al., 2013), little prior knowledge is assumed: the user's feedback evaluates long behavioral sequences, whereas short "interesting" sequences are considered in (Wilson et al., 2012), and whereas the user's feedback is related to the last few state-action pairs in (Knox et al., 2013). The experiment on the Nao robot confirms the feasibility of teaching simple behaviors in a matter of minutes.

A key feature of the PF approach is to enable the active computer to account for, and deal with, the unavoidable user's mistakes. A most interesting lesson learned from the experiments regards the intricate interplay between the active computer and the user, and specifically, how the active computer opinion of the user's competence impacts the actual user's consistency. A cumulative (dis)advantage phenomenon is observed. On the one hand, a pessimistic competence estimate leads the active computer to present the user with poorly informative queries, *thereby increasing the probability for the user to make errors* and be inconsistent everything else being equal. On the other hand, when the active computer trusts a competent user, its skills steadily improve while the user's error rate stays low. Overall, the importance of a good educational start is witnessed: poor initial demonstrations lead to a poor utility model, leading itself to poorly informative queries.

Further experiments focusing on continuous action and state spaces will be conducted to investigate the limitations of the PF framework, examining for instance whether the Nao can learn to grasp a soft plastic glass; the challenge is whether the user's feedback together with the Nao camera can palliate the lack of touch sensors (this will be a challenge for the user, too).

The PF framework opens several avenues for further research. The instant optimization criterion (section 3.4), which can be viewed as the "intrinsic motivation" of the active computer, will be extended to take into account the variance of the demonstration utility, either as a constraint, or in a multi-objective perspective. The performance loss between the noisy and noiseless expected utilities of selection will be theoretically and empirically studied, establishing bounds and directly tackling EUS^N using black-box optimization. Along the same line, the oblivious setting will be investigated and compared to the non-oblivious one.

Another perspective inspired from (Wilson et al., 2012) is to identify sub-behaviors in the active computer behavioral sequences; the motivation is that a negative user's feedback is in general related to one or a few sub-behaviors in a possibly long behavioral sequence. This identification might allow the active computer to ask direct questions to the user (e.g., what is it that you don't like, my dancing or my running?), expectedly speeding up the active computer progress. The identification of the behavioral subsequences responsible for the user's feedback can be formalized in terms of multiple instance ranking (Bergeron et al., 2008), considering a long behavior as a set of sub-behaviors where a few sub-behaviors are responsible for the user's preferences. Furthermore, the multiple instance setting will mitigate the influence of the initial conditions of the demonstrations on the PF performances.

[4] In preliminary experiments, LSPI failed to deliver a decent controller when using the estimated utility function. This failure is blamed on the fact that the Q-value is learned using a mean-square error criterion, whereas the LSPI efficiency depends on the L_∞ error (see discussion in (Munos, 2003)).

[5] A more compact policy representation could have been used, considering e.g. the weights of a neural network instead of the Q value function. However such a compact representation tends to yield a highly non-smooth optimization landscape.

References

Abbeel, P. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Stanford University, 2008.

Akrour, R., Schoenauer, M., and Sebag, M. April: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 7524, pp. 116–131. Springer LNCS, 2012.

Aldebaran, Ltd. Discover NAO, 2013. URL http://www.aldebaran-robotics.com/en/.

Bergeron, C., Zaretzki, J., Breneman, C. M., and Bennett, K. P. Multiple instance ranking. In *ICML*, pp. 48–55, 2008.

Brochu, E., de Freitas, N., and Ghosh, A. Active preference learning with discrete choice data. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *NIPS*, 2007.

Brochu, E., Brochu, T., and de Freitas, N. A Bayesian interactive optimization approach to procedural animation design. In Popovic, Z. and Otaduy, M. A. (eds.), *Symposium on Computer Animation*, pp. 103–112. Eurographics Association, 2010.

Chu, W. and Ghahramani, Z. Preference learning with Gaussian processes. In *ICML*, pp. 137–144, 2005.

Deisenroth, M. P., Neumann, G., and Peters, J. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

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

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

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

Hoos, H. H. Programming by optimization. *Commun. ACM*, 55(2):70–80, 2012.

Jain, A., Joachims, T., and Saxena, A. Learning trajectory preferences for manipulators via iterative improvement. In *NIPS*, 2013.

Jones, D.R., Schonlau, M., and Welch, W.J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Knox, W. B., Stone, P., and Breazeal, C. Training a robot via human feedback: A case study. In *Int. Conf. on Social Robotics*, volume 8239 of *LNCS*, pp. 460–470. Springer, 2013.

Konidaris, G., Kuindersma, S., Barto, A., and Grupen, R. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *NIPS 23*, pp. 1162–1170, 2010.

Lagoudakis, M. and Parr, R. Least-squares policy iteration. *J. Machine Learning Research*, 4:1107–1149, 2003.

Lin, J., Madnani, N., and Dorr, B. Putting the user in the loop: Interactive maximal marginal relevance for query-focused summarization. In *NAACL*, pp. 305–308. ACL, 2010.

Lizotte, D. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

Lörincz, A., Gyenes, V., Kiszlinger, M., and Szita, I. Mind model seems necessary for the emergence of communication. *Neural Information Processing - Letters and Reviews*, 11(4-6):109–121, 2007.

Luce, R. D. *Individual choice behavior*. John Wiley, New York, 1959.

Munos, R. Error bounds for approximate policy iteration. In *ICML*, pp. 560–567. AAAI Press, 2003.

Radlinski, F., Kurup, M., and Joachims, T. How does clickthrough data reflect retrieval quality? In J. G. Shanahan et al. (ed.), *CIKM*, pp. 43–52. ACM Int. Conf. Proc. Series, 2008.

Shepard, R. N. Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika*, 22:325345, 1957.

Shivaswamy, P. and Joachims, T. Online structured prediction via coactive learning. In *ICML*, 2012.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pp. 2960–2968, 2012.

Viappiani, P. and Boutilier, C. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*, pp. 2352–2360, 2010.

Wilson, A., Fern, A., and Tadepalli, P. A Bayesian approach for policy learning from trajectory preference queries. In *NIPS*, pp. 1142–1150, 2012.

Yue, Y. and Joachims, T. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, number 382:151 in ACM Int. Conf. Proc., 2009.