APPEARED IN
Proc. IEEE Int. Conf. on Robotics & Automation, 2019

Factored Contextual Policy Search with Bayesian Optimization

Robert Pinsler*[1], Peter Karkus*[2], Andras Kupcsik[3,4], David Hsu[2] and Wee Sun Lee[2]

***Abstract*— Scarce data is a major challenge to scaling robot learning to truly complex tasks, as we need to generalize locally learned policies over different task contexts. Contextual policy search offers data-efficient learning and generalization by explicitly conditioning the policy on a parametric context space. In this paper, we further structure the contextual policy representation. We propose to factor contexts into two components: target contexts that describe the task objectives, e.g. target position for throwing a ball; and environment contexts that characterize the environment, e.g. initial position or mass of the ball. Our key observation is that experience can be directly generalized over target contexts. We show that this can be easily exploited in contextual policy search algorithms. In particular, we apply factorization to a Bayesian optimization approach to contextual policy search both in sampling-based and active learning settings. Our simulation results show faster learning and better generalization in various robotic domains. See our supplementary video: `https://youtu.be/MNTbBAOufDY`.**

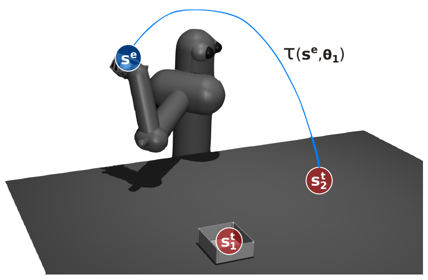


Fig. 1: Ball throwing task, where the robot is asked to hit target context $\boldsymbol{s}_1^t$ given initial position $\boldsymbol{s}^e$ of the ball. The robot chooses parameters $\boldsymbol{\theta}_1$ that generates a ball trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{s}^e, \boldsymbol{\theta}_1)$ landing at $\boldsymbol{s}_2^t$. Despite a low reward, knowing that $\boldsymbol{\tau}$ led to $\boldsymbol{s}_2^t$ is beneficial if the robot is asked again to throw near $\boldsymbol{s}_2^t$.

I. INTRODUCTION

Enabling robots to operate in truly complex domains requires learning policies from a small amount of data and generalizing learned policies over different tasks. Policy search methods with low-dimensional, parametric policy representations enable data-efficient learning of local policies [1]. Contextual policy search (CPS) [2], [3] further enables generalization over different task settings by structuring the policy. CPS uses an upper-level policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ to select parameters $\boldsymbol{\theta}$ of a lower-level policy given context $\boldsymbol{s}$, where the context $\boldsymbol{s}$ specifies the task. The goal is to learn a policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ that maximizes the expected reward $\mathbb{E}[\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}}]$.

We propose to further structure the contextual policy representation by introducing a factorization of the context space. In particular, we factorize a context vector $\boldsymbol{s}$ into two components: (1) *target contexts* $\boldsymbol{s}^t$ that specify task objectives, e.g. for a ball throwing task the target coordinates of the ball, and (2) *environment contexts* $\boldsymbol{s}^e$ that characterize the environment and the system dynamics, e.g. initial position of the ball. Formally, we assume that the expected reward is given by $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} = \int p(\boldsymbol{\tau}|\boldsymbol{s}^e, \boldsymbol{\theta}) R(\boldsymbol{s}^t, \boldsymbol{\tau}) d\boldsymbol{\tau}$, where $\boldsymbol{\tau}$ is a trajectory with unknown dynamics $p(\boldsymbol{\tau}|\boldsymbol{s}^e, \boldsymbol{\theta})$, and $R(\boldsymbol{s}^t, \boldsymbol{\tau})$ is the reward function. The key difference between $\boldsymbol{s}^t$ and $\boldsymbol{s}^e$ is that the dynamics only depend on $\boldsymbol{s}^e$. We can exploit this property and re-evaluate prior experience in light of a new target context, leading to improved data-efficiency and better generalization.

For example, assume a robot is learning to throw balls at different targets $\boldsymbol{s}^t$ (Fig. 1). The robot is asked to aim at $\boldsymbol{s}_1^t$. It chooses parameters $\boldsymbol{\theta}_1$, executes the throw, and observes a ball trajectory $\boldsymbol{\tau}_1$ that hits target $\boldsymbol{s}_2^t$, $\boldsymbol{s}_2^t \neq \boldsymbol{s}_1^t$. This yields a reward $R(\boldsymbol{s}_1^t, \boldsymbol{\tau}_1)$. Assume the robot is now asked to aim at target $\boldsymbol{s}_2^t$. Standard CPS methods try to generalize prior experience solely based on the upper-level policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$, e.g. by assuming that rewards obtained under similar contexts are correlated. Context factorization instead allows to treat the two context types differently. The target context $\boldsymbol{s}_2^t$ can be used to evaluate $R(\boldsymbol{s}_2^t, \boldsymbol{\tau}_1)$ directly, yielding the exact reward we would get for $\boldsymbol{\tau}_1$ when targeting $\boldsymbol{s}_2^t$. That is because a trajectory is independent from the target context. The same is not true for environment contexts, and thus we must rely on the upper-level policy to generalize over them.

We demonstrate the benefits of factorization by applying it to CPS approaches based on Bayesian optimization (BO) [4]; however, other CPS methods would be also possible. First, we consider a passive learning setting, where the context is given to the robot, and introduce a factored variant of BO for CPS (BO-CPS) [5], [6]. We then consider an active learning setting [7], where the robot can choose the context during learning. We introduce factored contexts to ACES [8], a CPS method based on entropy search [9]. For moderately low-dimensional search spaces, e.g. when learning pre-structured policies, such global optimization techniques achieve high data-efficiency by directly searching for the optimal parameters using a surrogate reward model.

So far we assumed that we can re-evaluate the reward function $R(\boldsymbol{s}^t, \boldsymbol{\tau})$ for arbitrary target contexts. This assump-

[1]University of Cambridge, UK [2]National University of Singapore, Singapore [3]Bosch Center for AI, Germany [4]Idiap Research Institute, Switzerland.

* denotes equal contribution. PK conceived the idea and performed initial experiments. RP extended the approach and evaluation, and wrote most of the manuscript. Correspondence to `rp586@cam.ac.uk`.

RP receives funding from iCASE RG89603 supported by Nokia. PK, DH and WSL acknowledge funding from Singapore Ministry of Education AcRF grant MOE2016-T2-2-068. PK is supported by the NUS Graduate School for Integrative Sciences and Engineering Scholarship.

tion is reasonable in real robot applications, where rewards typically encode objectives defined by the system designer. However, if the agent only has access to samples from the reward function, we can still exploit factored contexts by re-evaluating the current trajectory w.r.t. the achieved outcome. This approach can be seen as an extension of hindsight experience replay (HER) [10], a recently proposed data augmentation technique for goal-oriented RL algorithms.

We analyze the proposed methods first on a toy task. We then validate the benefits of context factorization on three simulated robotics environments from the OpenAI Gym [11], where we employ dynamic movement primitives [12] to efficiently generate trajectories. We show that context factorization is easy to implement, can be broadly applied to CPS problems, and consistently improves data-efficiency and generalization for various robotic tasks.

II. Related Work

There are several CPS approaches that generalize over a context space. One group of work first learns different local policies and then uses supervised learning to interpolate policy parameters over contexts [13], [14]. These methods are suitable for problems where local policies are available or easy to learn, but they are inefficient otherwise. The second group of work jointly learns local policies and generalizes over the context space [15], [16], [2]. These approaches were applied to a variety of real-world tasks, including playing table tennis [16], darts [16] and hockey [2]. Although all tasks involve target contexts, generalization over contexts solely relies on correlation. Similarly, CPS approaches based on BO [6], [8], [17], [18] learn a probabilistic reward model that generalizes over the context space through correlation. In this paper, we extend two BO approaches with factorization, namely BO-CPS [5], [6] and ACES [8].

To the best of our knowledge, there is no prior work that explicitly factors the context space. Similar ideas are implicitly used by Kober et al. [16] who learn a contextual policy for discrete targets while performing a higher-level task. While they map experience gained in one context to another, they do so for estimating discrete outcome probabilities and not for improving the policy. GP-REPS [19] iteratively learns a transition model of the system using a Gaussian process (GP) [20], which is then used to generate trajectories offline for updating the policy. The authors consider generating additional samples for artificial contexts, but they do not define an explicit factorization.

The idea of replacing the goal of a trajectory has recently been explored in HER [10], which increases data-efficiency in goal-based RL tasks with sparse rewards. The key idea is to augment the dataset with additional experience by replacing the original target context of a rollout to be the achieved outcome. Instead of replacing the target context after each rollout, we replace the target context of all previous episodes before each rollout and re-evaluate the entire dataset. Our approach additionally generalizes over environment contexts that are typical in CPS problems. If we have only access to sample rewards, we show how context factorization can be used to extend HER to CPS.

III. Background

A. Bayesian Optimization for Contextual Policy Search

In a CPS problem, the agent observes a *context* $\boldsymbol{s} \sim \gamma(\boldsymbol{s})$ before each episode, where the context specifies the task setting and $\gamma(\boldsymbol{s})$ is a distribution over contexts. To solve the task, the agent maintains an upper-level policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over parameters $\boldsymbol{\theta}$ of a lower-level policy, e.g. a dynamic movement primitive [12]. Executing the lower-level policy with parameters $\boldsymbol{\theta}$ generates a trajectory $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{s},\boldsymbol{\theta})$ that yields reward $R(\boldsymbol{s},\boldsymbol{\tau})$. The goal of the agent is to learn an upper-level policy that maximizes the expected reward,

$$\mathbb{E}[\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}}] = \iiint \gamma(\boldsymbol{s})\pi(\boldsymbol{\theta}|\boldsymbol{s})p(\boldsymbol{\tau}|\boldsymbol{s},\boldsymbol{\theta})R(\boldsymbol{s},\boldsymbol{\tau})d\boldsymbol{\tau}d\boldsymbol{\theta}d\boldsymbol{s}.$$

BO-CPS [5], [6] frames CPS as a BO problem. BO is a global search method for optimizing real-valued functions, assuming only access to noisy sample evaluations. Starting from a prior belief about the objective, BO employs an acquisition function to guide the sampling procedure. In BO-CPS, a probabilistic reward model $p(R|\mathcal{D},\boldsymbol{s},\boldsymbol{\theta})$ is learned from N data samples $\mathcal{D} = \{\boldsymbol{s}_i,\boldsymbol{\theta}_i,\boldsymbol{R}_i\}_{i=1}^{N}$, which allows to evaluate potential parameters $\boldsymbol{\theta}$ for a query context $\boldsymbol{s}_q$. BO-CPS commits to a GP prior [20] with predictive posterior $p(R|\mathcal{D},\boldsymbol{s}_q,\boldsymbol{\theta}) = \mathcal{N}(\mu_{\boldsymbol{s}_q,\boldsymbol{\theta}},\sigma^2_{\boldsymbol{s}_q,\boldsymbol{\theta}})$, and uses the GP-UCB acquisition function [21],

$$\text{GP-UCB}(\boldsymbol{s}_q,\boldsymbol{\theta}) = \mu_{\boldsymbol{s}_q,\boldsymbol{\theta}} + \kappa\sigma_{\boldsymbol{s}_q,\boldsymbol{\theta}}, \tag{1}$$

where κ trades off exploration and exploitation. The policy parameters $\boldsymbol{\theta}$ are selected by the upper-level policy π, which optimizes the acquisition function given the query context,

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}_q) = \delta\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*|\boldsymbol{s}_q\right), \tag{2}$$

where $\boldsymbol{\theta}^*|\boldsymbol{s}_q = \arg\max_{\boldsymbol{\theta}} \text{GP-UCB}(\boldsymbol{s}_q,\boldsymbol{\theta})$, and $\delta(\cdot)$ is the Dirac delta function. The algorithm is summarized in Alg. 1.

B. Active Contextual Entropy Search

Active contextual entropy search (ACES) [8] is an extension of entropy search (ES) [9] to the active CPS setting, where both the parameters $\boldsymbol{\theta}$ and the context $\boldsymbol{s}$ are chosen by the agent before an episode. ACES maintains a conditional probability distribution $p(\boldsymbol{\theta}^*|\mathcal{D},\boldsymbol{s}) = p(\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f(\boldsymbol{s},\boldsymbol{\theta})|\mathcal{D},\boldsymbol{s})$, expressing the belief about $\boldsymbol{\theta}$ being optimal in context $\boldsymbol{s}$. The most informative query point is chosen by maximizing the expected information gain integrated over the context space,

$$\text{ACES}(\boldsymbol{s}_q,\boldsymbol{\theta}_q) = \sum\nolimits_{c=1}^{C} G^{\boldsymbol{s}_c}(\boldsymbol{s}_q,\boldsymbol{\theta}_q), \tag{3}$$

where $\{\boldsymbol{s}_c\}_{c=1}^{C}$ is a set of randomly chosen representer points. The expected information gain in context $\boldsymbol{s}_c$ after performing a hypothetical rollout with $(\boldsymbol{s}_q,\boldsymbol{\theta}_q)$ is given by

$$G^{\boldsymbol{s}_c}(\boldsymbol{s}_q,\boldsymbol{\theta}_q) = H[p(\boldsymbol{\theta}^*|\mathcal{D},\boldsymbol{s}_c)] - \mathbb{E}\big[H[p(\boldsymbol{\theta}^*|\mathcal{D}^+,\boldsymbol{s}_c)]\big], \tag{4}$$

where the expectation is taken over $p(R|\mathcal{D},\boldsymbol{s}_q,\boldsymbol{\theta}_q,\boldsymbol{s}_c)$, and $\mathcal{D}^+ = \mathcal{D} \cup \{\boldsymbol{s}_q,\boldsymbol{\theta}_q,R\}$ is an updated dataset that contains the

Algorithm 1 BO-CPS [6]

```
repeat
    Observe $s_q \sim \gamma(s)$

    Learn reward model $p(R|\mathcal{D}, s, \theta)$ from $\mathcal{D}$ (Eq. 6)
    Select $\theta_q \sim \pi(\theta|s_q)$ (Eq. 1, 2) using reward model
    Execute rollout $\tau \sim p(\tau|s_q, \theta_q)$ with the robot
    Add $(s_q, \theta_q, R(s_q, \tau))$ to $\mathcal{D}$
until Policy $\pi$ converges
```

Algorithm 2 BO-FCPS (ours)

```
repeat
    Observe $s_q \sim \gamma(s)$, where $s_q = (s^t_q, s^e_q)$
    Construct dataset $\mathcal{D}_q$ from $\mathcal{D}$ (Eq. 5)
    Learn reward model $p(R^t_q|\mathcal{D}_q, s^e_q, \theta)$ from $\mathcal{D}_q$ (Eq. 6)
    Select $\theta_q \sim \pi(\theta|s_q)$ (Eq. 1, 2) using reward model
    Execute rollout $\tau \sim p(\tau|s^e_q, \theta_q)$ with the robot
    Add $(s^e_q, \theta_q, \tau)$ to $\mathcal{D}$
until Policy $\pi$ converges
```

hypothetical query point. In practice, Eq. 3 requires further approximations, which are explained in the original work [9], [8]. The algorithm is summarized in Alg. 3.

C. Dynamic Movement Primitives

Dynamic movement primitives (DMPs) are often used as lower-level policies in robot learning tasks. A DMP [12] is a spring-damper system whose hyper-parameters can be flexibly adapted while retaining the general shape of the movement. These include the final position $\boldsymbol{y}_f$, final velocity $\dot{\boldsymbol{y}}_f$ and temporal scaling $\boldsymbol{\tau}$. The motion is further modulated by a non-linear forcing function $f_{\boldsymbol{w}}(z) = \boldsymbol{w}^\top \Phi(z)$ with basis functions $\Phi(z)$ parameterized by phase variable z. The parameters $\boldsymbol{w}$ determine the shape of the movement and can be obtained by imitation learning. Each generated DMP trajectory is followed by the control policy of the robot, which implements a low-level feedback controller.

IV. FACTORED CONTEXTUAL POLICY SEARCH

In this section, we introduce context factorization and show how it can be integrated into CPS algorithms.

We propose to factorize a context vector $\boldsymbol{s}$ into two types of contexts, $\boldsymbol{s} = (\boldsymbol{s}^t, \boldsymbol{s}^e)$:

- target contexts $\boldsymbol{s}^t$ which specify the task objective, and
- environment contexts $\boldsymbol{s}^e$ which characterize the environment and the system dynamics.

Formally, we assume that the reward function is given by $R(\boldsymbol{s}^t, \boldsymbol{\tau})$[1], where the trajectory $\boldsymbol{\tau}$ is generated by unknown system dynamics, $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{s}^e, \boldsymbol{\theta})$. Importantly, while the dynamics function depends on environment contexts, it does not depend on target contexts. This means that we can exchange the target context of a rollout without altering its trajectory, allowing to re-evaluate a rollout under different target contexts. For example, in our ball throwing task (Fig. 1) we can re-evaluate a previously observed trajectory pretending we were aiming at a different target. We cannot do the same with environment contexts, e.g. the initial ball pose, because a different initial pose would result in a different trajectory. In the following, we exploit factored contexts to reduce the data requirements of CPS algorithms. To what extend factorization can be exploited depends on the knowledge of the reward function $R(\boldsymbol{s}^t, \boldsymbol{\tau})$. First, we assume that the reward function is fully known or that it can be evaluated for arbitrary targets $\boldsymbol{s}^t$. This allows to construct highly data-efficient algorithms, as we demonstrate on a passive (Section IV-A) and active (Section IV-B) CPS algorithm. In Section IV-C, we drop the assumption of a known reward function and propose an extension of hindsight experience replay [10] to the CPS setting using context factorization.

A. Bayesian Optimization for Factored CPS

Context factorization can be easily incorporated into BO-CPS. The resulting algorithm, Bayesian optimization for factored contextual policy search (BO-FCPS), is shown in Alg. 2. It maintains a dataset $\mathcal{D} = \{\boldsymbol{s}^e_i, \boldsymbol{\theta}_i, \boldsymbol{\tau}_i\}_{i=1}^N$[2] that can be used to re-evaluate past experiences for a new query context $\boldsymbol{s}_q = (\boldsymbol{s}^t_q, \boldsymbol{s}^e_q)$. Given reward function $R(\boldsymbol{s}^t, \boldsymbol{\tau})$, we construct a query-specific dataset,

$$\mathcal{D}_q = \{\boldsymbol{s}^e_i, \boldsymbol{\theta}_i, R(\boldsymbol{s}^t_q, \boldsymbol{\tau}_i)\}_{i=1}^N, \tag{5}$$

for learning a specialized reward model,

$$p(R^t_q|\mathcal{D}_q, \boldsymbol{s}^e_q, \boldsymbol{\theta}) = \mathcal{N}(R^t_q|\mu_{\boldsymbol{s}_q,\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{s}_q,\boldsymbol{\theta}}), \tag{6}$$

before each rollout. This model is specific to the current target context $\boldsymbol{s}^t_q$. Jointly, the set of all possible reward models $\{p(R^t_{q'}|\mathcal{D}_{q'}, \boldsymbol{s}^e, \boldsymbol{\theta})\}$ w.r.t. arbitrary targets $\boldsymbol{s}^t_{q'}$ generalizes directly over the target context space. Thus, each reward model only needs to generalize over environment contexts $\boldsymbol{s}^e$ and policy parameters $\boldsymbol{\theta}$, leading to a reduced input space compared to the original reward model. This has the added benefit of a smaller search space during optimization. The parameters $\boldsymbol{\theta}$ for context $\boldsymbol{s}_q$ are found by optimizing the acquisition function given the target-specific reward model (Eq. 1, 2). We employ the DIRECT [22] algorithm for optimization, followed by L-BFGS [23] to refine the result.

We can formally compare BO-FCPS and BO-CPS if we assume that both approaches share the same dataset $\mathcal{D} = \{\boldsymbol{s}_i, \boldsymbol{\theta}_i, \boldsymbol{\tau}_i, \boldsymbol{R}(\boldsymbol{s}^e_i, \boldsymbol{\tau}_i)\}_{i=1}^N$ and the same GP hyper-parameters. In this case, the reward model of BO-FCPS evaluated at a particular target context $\boldsymbol{s}^t_q$ is always at least as accurate as the one learned by BO-CPS. To see this, recall that BO-FCPS differs from BO-CPS in two ways: (1) BO-FCPS re-computes the rewards for $\boldsymbol{s}^t_q$, and (2) BO-FCPS

[1]In general, the reward function may also depend on $\boldsymbol{s}^e$ and $\boldsymbol{\theta}$. We omit this dependence for improved readability; the principle remains the same.

[2]Instead of storing the entire trajectory, in practice we may only record its sufficient statistics for computing the reward, i.e. an outcome $\boldsymbol{o} = \phi(\boldsymbol{\tau})$.

Algorithm 3 ACES [8]

```
repeat
    Sample representer points $\{\boldsymbol{s}_c\}_{c=1}^C$ (Section III-B)

    Learn reward model $p(R|\mathcal{D}, \boldsymbol{s}, \boldsymbol{\theta})$ from $\mathcal{D}$ (Eq. 6)
    Select $(\boldsymbol{s}_q, \boldsymbol{\theta}_q) = \arg\max_{\boldsymbol{s},\boldsymbol{\theta}} \text{ACES}(\boldsymbol{s}, \boldsymbol{\theta})$ (Eq. 3)
    Execute rollout $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{s}_q, \boldsymbol{\theta}_q)$ with the robot
    Add $(\boldsymbol{s}_q, \boldsymbol{\theta}_q, R(\boldsymbol{s}_q, \boldsymbol{\tau}))$ to $\mathcal{D}$
until Policy $\pi$ converges
```

Algorithm 4 FACES (ours)

```
repeat
    Sample representer points $\{\boldsymbol{s}_c\}_{c=1}^C$, $\boldsymbol{s}_c = (\boldsymbol{s}_c^t, \boldsymbol{s}_c^e)$
    Construct datasets $\{\mathcal{D}_c\}_{c=1}^C$ from $\mathcal{D}$ (Eq. 5)
    Learn reward models $\{\text{GP}_c\}_{c=1}^C$ from $\{\mathcal{D}_c\}_{c=1}^C$ (Eq. 6)
    Select $(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q) = \arg\max_{\boldsymbol{s}^e,\boldsymbol{\theta}} \text{FACES}(\boldsymbol{s}^e, \boldsymbol{\theta})$ (Eq. 7)
    Execute rollout $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{s}_q^e, \boldsymbol{\theta}_q)$ with the robot
    Add $(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q, \boldsymbol{\tau})$ to $\mathcal{D}$
until Policy $\pi$ converges
```

does not consider the rewards at other target contexts $\boldsymbol{s}_{q'}^t \neq \boldsymbol{s}_q^t$. Re-computing the rewards is trivially beneficial because BO-FCPS knows the true reward for target context $\boldsymbol{s}_q^t$ given a trajectory $\boldsymbol{\tau}$, whereas BO-CPS needs to infer the reward from correlations between target contexts. Disregarding rewards at other target contexts $\boldsymbol{s}_{q'}^t \neq \boldsymbol{s}_q^t$ does not degrade the predictive performance of our model either. That is because for a given context-parameter pair the only source of uncertainty w.r.t. their expected reward is in the execution of the trajectory $\boldsymbol{\tau}$ and its effect on the environment. Since the trajectory does not depend on the target context, evaluating the same trajectory under different target contexts does not reveal more information about the trajectory itself.

Note that while a better reward model at a given target context leads to better greedy performance, e.g. during offline evaluation, it does not necessarily imply higher cumulative rewards during learning. Furthermore, in practice both the dataset $\mathcal{D}$ and the GP hyper-parameters would be different. Empirically, BO-FCPS does achieve both higher online and offline performance, as we show in Section V. We defer a more extensive theoretical analysis to future work.

B. Factored Active Contextual Entropy Search

In the active learning setting, the agent chooses both the policy parameters $\boldsymbol{\theta}$ and the context $\boldsymbol{s}$. Applying GP-UCB is problematic as it would not take the varying difficulty of tasks into account [7]. Instead, we follow ACES [8] and use an ES-based acquisition function, which aims to choose the most informative query points for global optimization [9].

We integrate factored contexts into ACES as follows. In each iteration, we map previous experience to all representer points $\{\boldsymbol{s}_c\}_{c=1}^C$ in the context space, i.e. we construct C different datasets $\{\mathcal{D}_c\}_{c=1}^C$ as in Eq. 5. From these datasets, we construct a set of GP models $\{\text{GP}_c\}_{c=1}^C$ that we use to evaluate the ACES acquisition function. In particular, we employ the corresponding $\text{GP}_c : p(R_c^t|\mathcal{D}_c, \boldsymbol{s}_c^e, \boldsymbol{\theta})$ when the expected information gain $G^{\boldsymbol{s}}(\boldsymbol{s}_q, \boldsymbol{\theta}_q)$ after a hypothetical query $(\boldsymbol{s}_q, \boldsymbol{\theta}_q)$ is evaluated for $\boldsymbol{s} = \boldsymbol{s}_c$. Similar to BO-FCPS, we therefore directly use the target-specific GPs instead of relying on the correlation between target contexts.

Note that the choice on the target-type query context $\boldsymbol{s}_q^t$ is actually indifferent if we ignore rewards during training, and thus we only need to select $(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q)$ by maximizing

$$\text{FACES}(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q) = \sum\nolimits_{c=1}^{C} G^{\boldsymbol{s}_c}(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q). \tag{7}$$

We call the resulting algorithm factored active contextual entropy search (FACES). The algorithm is shown in Alg. 4.

C. Hindsight Experience Replay for Factored CPS

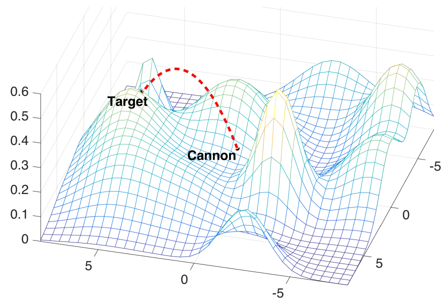
So far we have assumed that the agent has access to the reward function $R(\boldsymbol{s}^t, \boldsymbol{\tau})$. If we cannot query the reward function at arbitrary points, it is still possible to leverage factored contexts. In particular, we replace the current target context $\boldsymbol{s}_q^t$ after a rollout by the achieved target $\boldsymbol{s}_{\boldsymbol{\tau}}^t$ and evaluate it again, yielding reward $R(\boldsymbol{s}_{\boldsymbol{\tau}}^t, \boldsymbol{\tau})$. Thus, the only requirement is to be able to obtain the sample reward $R(\boldsymbol{s}_{\boldsymbol{\tau}}^t, \boldsymbol{\tau})$ in addition to the actual reward $R(\boldsymbol{s}_q^t, \boldsymbol{\tau})$. The additional data point $(\boldsymbol{s}_q^e, \boldsymbol{\theta}_q, R(\boldsymbol{s}_{\boldsymbol{\tau}}^t, \boldsymbol{\tau}))$ can then be added to the training dataset $\mathcal{D}$, and standard CPS methods such as BO-CPS, cost-regularized kernel regression [16] or contextual relative entropy policy search (C-REPS) [3], [2] can be used without further modifications. Such an approach can be seen as an extension of HER to the CPS setting. We believe we are the first ones that explicitly make this connection.

V. EXPERIMENTS AND RESULTS

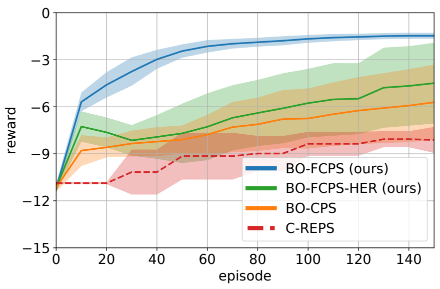
We perform experiments to answer the following questions: (a) does context factorization lead to more data-efficient learning for passive and active BO-based CPS algorithms; (b) how does the choice of acquisition function influence the performance; (c) does context factorization improve generalization; (d) is our method effective in more complex robotic domains? We address questions (a)-(c) through experiments on a toy cannon task, and (d) on three simulated tasks from the OpenAI Gym [11]. For the Gym tasks, we employ an extension [24] of the DMP framework [12] to efficiently generate goal-directed trajectories.

A. Toy Cannon Task

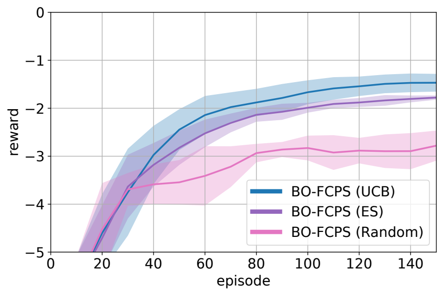
The toy cannon task is a popular domain for evaluating CPS algorithms [25], [6], [8]. As shown in Fig. 2a, a cannon is placed in the center of a 3D coordinate system and has to shoot at targets on the ground in the range of $[-11, 11] \times [-11, 11]$m. The contextual policy maps from 2D targets $\boldsymbol{s}^t \in \mathcal{R}^2$ to 3D launch parameters $\boldsymbol{\theta} \in \mathcal{R}^3$: horizontal orientation $\alpha \in [0, 2\pi]$, vertical angle $\beta \in [0.01, \pi/2 - 0.2]$ and speed $v \in [0.1, 5]$ ms. The reward function is given by $R(\boldsymbol{s}^t, \boldsymbol{\tau}) = -\|\boldsymbol{s}^t - \boldsymbol{s}_{\boldsymbol{\tau}}^t\| - 0.05v^2$, where $\boldsymbol{s}_{\boldsymbol{\tau}}^t$ is the achieved hitting location of a trajectory $\boldsymbol{\tau}$. To increase the difficulty of the problem, we add Gaussian noise ($\sigma_n = 1°$) to the desired



(a) Toy cannon setup



(b) Comparison of different algorithms

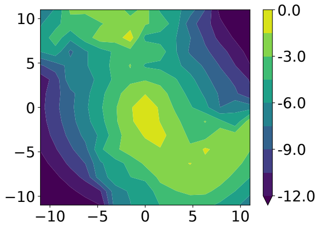


(c) Comparison of different variants

Fig. 2: **(a)** Visualization of the toy cannon task. **(b)-(c)** Offline performance evaluated on a fixed set of contexts from a 15×15 grid. Results are averaged over 10 randomly generated environments. Shaded areas denote one standard deviation.

	$t = 50$	$t = 100$	$t = 150$
C-REPS	-496 (± 17)	-955 (± 56)	-1357 (± 62)
BO-CPS	-461 (± 28)	-843 (± 70)	-1148 (± 151)
BO-FCPS-HER (ours)	-447 (± 24)	-809 (± 71)	-1111 (± 140)
BO-FCPS (ours)	**-303** (± **34**)	**-414** (± **44**)	**-499** (± **56**)

TABLE I: Online learning performance on the toy cannon task averaged over 10 random seeds. We report mean cumulative rewards obtained during the first t iterations.


(a) BO-CPS

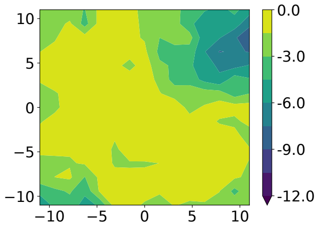

(b) BO-FCPS (ours)

Fig. 3: Achieved rewards in target context space after 150 episodes, where no contexts were sampled from the upper-right and lower-left corner during training.

launch angle during training and randomly place hills in the environment. The learning agent is unaware of the hills and the target contexts carry no information on the elevation.

First, we compare both our factored BO approach (BO-FCPS) and our factored HER-style BO approach (BO-FCPS-HER) to standard BO-CPS. Each algorithm uses the GP-UCB acquisition function. We employ a zero-mean GP prior, $p(\boldsymbol{f}) \sim \mathrm{GP}(\mathbf{0}, k(\boldsymbol{x}, \boldsymbol{x}'))$, with squared-exponential kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \boldsymbol{\Lambda}^{-1}(\boldsymbol{x} - \boldsymbol{x}')\right)$, where $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{\theta})$, and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\ell})$ contains positive length-scale parameters $\boldsymbol{\ell}$. The GP hyper-parameters are optimized by maximizing the marginal likelihood. We also compare to C-REPS, which performs local policy updates instead of global optimization. We use a linear Gaussian policy with squared context features that is updated every 30 episodes subject to the relative entropy bound $\epsilon = 0.5$.

The offline performance of each algorithm is shown in Fig. 2b. BO-FCPS requires only 60 episodes to find a good policy, a considerable improvement over standard BO-CPS. BO-FCPS-HER improves on BO-CPS as well, although the variance is much larger. This is because the GP hyper-

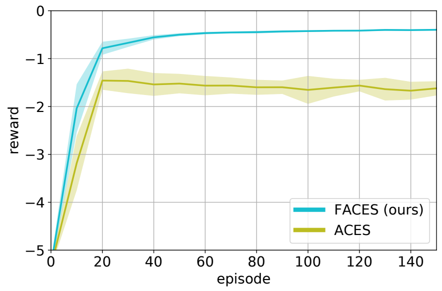


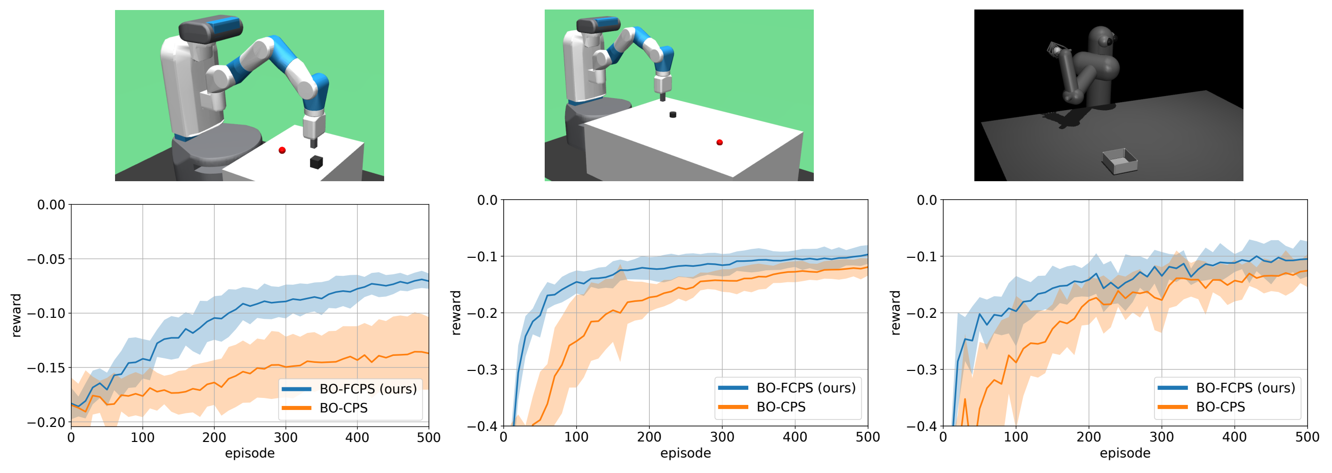
Fig. 4: Learning curve for active learning setting averaged over 10 randomly generated environments, evaluated offline based on contexts placed uniformly on an 8×8 grid. Shaded areas denote one standard deviation.

parameter optimization sometimes got stuck in a local minimum, overfitting to the context variables while ignoring the influence of the parameters $\boldsymbol{\theta}$. We hypothesize that a full Bayesian treatment would mitigate this issue. C-REPS is not competitive on this low-dimensional task since the policy adapts too slowly. The above findings are confirmed by the online performances, which are summarized in Table I.

In Fig. 2c, we evaluate the dependence of BO-FCPS on the acquisition function. We compare three variants: BO-FCPS with GP-UCB (UCB), entropy search (ES) and a random acquisition function. Using GP-UCB over ES leads to slightly faster learning as ES tends to explore too much. Likewise, random exploration is not sufficient for data-efficient learning. We therefore focus on GP-UCB.

Next, we demonstrate why evaluating previous rollouts for the given target context leads to improved generalization. We only present contexts $\boldsymbol{s}^t \in [-11, 0] \times [0, 11] \ \cup \ [0, 11] \times [-11, 0]$ during training, while the agent has to generalize to the entire context space during evaluation. As depicted in Fig. 3, BO-FCPS generalizes much better to unseen contexts due to a much more accurate reward model at locations where it has already shot to. When evaluated in previously unseen contexts, the mean rewards achieved by BO-FCPS were higher by 4.0; while the improvement was 3.05 in contexts that were sampled during training.

Finally, we consider an active learning setting, where the agent observes an additional context variable $\mathbb{I} \in [0, 1]$ that indicates whether the learning agent should shoot or not. If



(a) FetchPush-v1 (b) FetchSlide-v1 (c) Thrower-v2

Fig. 5: Learning curves for the OpenAI Gym tasks. Policies are evaluated after every 10 rollouts, in 25 fixed contexts sampled once before the experiment. Results are averaged over 10 random seeds. Shaded areas denote one standard deviation.

$\mathbb{I} \leq 0.1$, the agent receives reward $R(\boldsymbol{s}^t, \boldsymbol{\tau}) = -\|\boldsymbol{s}^t - \boldsymbol{s}^t_{\boldsymbol{\tau}}\| - 0.05v^2$ as before, and an action penalty $-\|\boldsymbol{\theta}\|$ otherwise. The agent should therefore actively select $\mathbb{I} \leq 0.1$, for which the reward function is much harder to learn. We compare FACES to ACES, and use 200 representer points to approximate the acquisition functions. The results are shown in Fig. 4. Similar to the passive case, factorization is greatly beneficial: FACES achieves much faster learning than ACES.

B. *Simulated Robotic Tasks*

Finally, we apply BO-FCPS to three distinct robotics tasks from the OpenAI Gym [11], [26], namely:

- **FetchPush-v1**: Push a box to a goal position.
- **FetchSlide-v1**: Slide a puck to a goal position that is out of reach of the robot.
- **Thrower-v2**: Throw a ball to a goal position.

The BO-FCPS algorithm is used to select the DMP parameters for performing each task according to the current context. We deviate from the original task specification of Thrower-v2 by replacing the joint-space controller in favor of task-space control to reduce the dimensionality of the problem. The same controller is employed in the Fetch environments. Moreover, we use the final distance between the object and the target context as the reward function in each environment. For more details, please refer to [11], [26].

For FetchPush and FetchSlide, both the initial object position $\boldsymbol{s}^e \in \mathcal{R}^2$ and the desired goal position $\boldsymbol{s}^t \in \mathcal{R}^2$ are varied. The lower-level policy consists of two 3-dimensional task-space DMPs that are sequenced together. The first DMP is used to bring the robot arm into position to manipulate the object, where the trajectory is modulated by 25 basis functions per dimension, and the shape parameters $\boldsymbol{w}$ are learned by imitation. The second DMP is used to execute the actual movement (i.e. pushing, sliding), starting from the final position of the first DMP. The upper-level policy adapts the approach angle α of the first DMP w.r.t. the object, yielding a goal position $\boldsymbol{y}_1$ that is a fixed distance away from the object, and the goal position $\boldsymbol{y}_2$ of the second DMP. We use $\alpha \in [0, \pi], \boldsymbol{y}_2 \in [0, 0.4] \times [-0.4, 0.4]$ for FetchSlide and $\alpha \in [0, 2\pi], \boldsymbol{y}_2 \in [-0.2, 0.2]^2$ for FetchPush.

For the Thrower environment, only the desired goal position $\boldsymbol{s}_t \in \mathcal{R}^2$ is varied. The lower-level policy is a single 3-dimensional task-space DMP with 25 basis functions, where the shape parameters are learned by imitation. The upper-level policy selects the goal position $\boldsymbol{y} \in [-0.5, 0.5] \times [1, 1.5] \times [-0.5, 0.5]$ and goal velocity $\dot{\boldsymbol{y}} \in [0, 1]^3$ of the DMP, resulting in a 6-dimensional parameter vector $\boldsymbol{\theta}$.

Results are shown in Fig. 5. Our proposed approach consistently outperforms standard BO-CPS, suggesting that our earlier findings on the toy cannon task apply to more complex simulated robotic domains as well.

VI. Discussion and Future Work

We introduced context factorization and integrated it into passive and active learning approaches for CPS with BO. The improvement we can expect from factorization depends on the characteristics of the task. It is most effective if a large part of the learning challenge is about generalizing across contexts, as opposed to learning a good policy for a single context. In general, the larger the space of target contexts over environment contexts and policy parameters, the more factorization is expected to be beneficial.

In this paper, we focused on BO for CPS. One shortcoming of BO is that it does not scale well to high-dimensional problems. Future work may address scalability and explore alternative acquisition functions such as predictive entropy search [27]. Since context factorization is not specific to BO, it could also be applied to other CPS algorithms, e.g. C-REPS [2] or contextual CMA-ES [28]. A simple approach would be to populate the dataset with additional re-evaluated samples, similar to HER. Finally, we demonstrated the benefits of factorization in extensive simulated experiments. In the future, we plan to demonstrate the approach on a real robot system as well.

References

[1] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[2] A. G. Kupcsik, M. P. Deisenroth, J. Peters, G. Neumann, *et al.*, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1401–1407, 2013.

[3] C. Daniel, G. Neumann, and J. Peters, "Hierarchical relative entropy policy search," in *Artificial Intelligence and Statistics*, pp. 273–281, 2012.

[4] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[5] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization," in *Advances in Neural Information Processing Systems*, pp. 2447–2455, 2011.

[6] J. H. Metzen, A. Fabisch, and J. Hansen, "Bayesian optimization for contextual policy search," in *Proceedings of the Second Machine Learning in Planning and Control of Robot Motion Workshop*, 2015.

[7] A. Fabisch and J. H. Metzen, "Active contextual policy search," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3371–3399, 2014.

[8] J. H. Metzen, "Active contextual entropy search," *arXiv preprint arXiv:1511.04211*, 2015.

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

[10] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[12] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems*, pp. 1547–1554, 2003.

[13] B. C. Da Silva, G. Konidaris, and A. G. Barto, "Learning parameterized skills," in *Proceedings of the International Coference on International Conference on Machine Learning*, pp. 1443–1450, 2012.

[14] J. H. Metzen, A. Fabisch, L. Senger, J. de Gea Fernández, and E. A. Kirchner, "Towards learning of generic skills for robotic manipulation," *KI-Künstliche Intelligenz*, vol. 28, no. 1, pp. 15–20, 2014.

[15] J. Peters and S. Schaal, "Applying the episodic natural actor-critic architecture to motor primitive learning.," in *European Symposium on Artificial Neural Networks*, pp. 295–300, 2007.

[16] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[17] R. Pinsler, R. Akrour, T. Osa, J. Peters, and G. Neumann, "Sample and feedback efficient hierarchical reinforcement learning from human preferences," in *IEEE International Conference on Robotics and Automation*, 2018.

[18] B. Yang, G. Wang, R. Calandra, D. Contreras, S. Levine, and K. Pister, "Learning flexible and reusable locomotion primitives for a microrobot," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1904–1911, 2018.

[19] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann, "Model-based contextual policy search for data-efficient generalization of robot skills," *Artificial Intelligence*, vol. 247, pp. 415–439, 2017.

[20] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[21] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the International Conference on International Conference on Machine Learning*, pp. 1015–1022, 2010.

[22] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[23] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[24] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *IEEE International Conference on Robotics and Automation*, pp. 853–858, IEEE, 2010.

[25] B. Da Silva, G. Konidaris, and A. Barto, "Active learning of parameterized skills," in *Proceedings of the International Conference on Machine Learning*, pp. 1737–1745, 2014.

[26] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[27] 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*, pp. 918–926, 2014.

[28] A. Abdolmaleki, B. Price, N. Lau, L. P. Reis, and G. Neumann, "Contextual covariance matrix adaptation evolutionary strategies," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1378–1385, 2017.