Eager and Memory-Based Non-Parametric Stochastic Search Methods for Learning Control

Victor Barbaros[1], Herke van Hoof[1], Abbas Abdolmaleki[2] and David Meger[1]

Abstract— **Direct policy search has shown to be a successful method to optimize robot controller parameters. However, defining a good parametric form for the controller can be challenging for complex problems. Non-parametric methods provide a flexible alternative and are thus a promising tool in robot skill learning. In this paper, we investigate two non-parametric methods based on similar principles but utilizing differing computing schedules: an eager learner and a memory-based learner. We compare the methods experimentally on two different control problems. Furthermore, we define and evaluate a new 'hybrid' controller that combines the strong points of both of these methods.**

I. INTRODUCTION

Policy search has been shown to be a successful reinforcement learning approach for acquiring robot skills [1]. In particular, direct policy search methods attempt to find a good parameter vector for a controller without relying on an estimate of a value function. These abstract parameters are often mapped indirectly to low-level actions. For example, our parameters might be the gains of a PD controller.

In this paper, we will focus on episode-based policy search [1]–[5]. In this paradigm, a measure of quality called return is associated with each parameter vector. This return can, for example, be specified as the cumulative value of a reward function that defines the task. As episode-based policy search does not put any assumption on the structure of the return, it is a special case of stochastic search.

Stochastic search algorithms are gradient-free black-box optimizers that can be used to find a set of policy parameters which optimize robot performance directly. These algorithms can be applied with an arbitrary and unknown performance function and do not require gradients. The only requirement is that the performance of a particular control policy can be evaluated, for example, by executing the policy with a given parameter vector on the robot.

Stochastic search is of particular utility in *contextual* policy search, where a robot must perform well over a set of environments that are related by a context variable. For example, inserting a peg into a hole, where the hole position (the context vector) is provided in each case. The search distribution is then conditional on the current context. This conditional search distribution is then again optimized to generate the appropriate parameters for a low-level controller to perform well in the specific task context.

Stochastic search methods maintain a search distribution from which the parameter vectors to be evaluated are sampled. In contextual stochastic search, there can be a different search distribution for each context. Such methods thus need to represent a mapping from contexts to search distributions – this mapping represents the ability of the method to generalize across contextually-related environments. Simple examples are search distributions where the mean is a linear combination of context features, or a mixture of linear models [6]–[8]. However, if the mapping from contexts to good controller parameters is complex, a linear model is not powerful enough and designing good non-linear features can be a complex task even for domain experts.

Alternatively, the mapping from contexts to search distributions could be specified using a non-parametric technique. Non-parametric techniques do not have a fixed representation that is designed beforehand, but use a representation that is defined by the training data. Thus, non-parametric representations are highly flexible, tend be more precise in areas where many data points lie, and tend to automatically become more complex for larger training set sizes.

In this paper, we will investigate two different non-parametric methods that have recently been proposed for use in reinforcement learning. First, we will look at locally weighted methods [9]. This method falls in the category of *memory-based learning* where learning samples are stored and accessed each time a prediction is made. In particular, for every query a locally linear model is fit to nearby data points. In contrast to a globally linear model, the set of active data points is different for each test point, and, thus, the global shape can be highly non-linear.

On the other hand, we will investigate Gaussian processes, a kernel method [10]. A Gaussian process can be seen as a generalization of a linear model where we do not need to specify basis features, but instead compute inner products in some kernel space. Whereas locally linear methods tend to perform a relatively simple computation each time the model is queried, fitting a Gaussian process model requires intensive computation. However, this computation must be performed only once for a given training set. This up-front computation is the hallmark of an *eager* learning method.

Memory-based and eager models have different strengths and weaknesses in terms of performance and computation time. Our comparison will quantify these differences, and thus help determine when to select each approach. However, the primary contribution of this paper is a novel hybrid method that combines the representational power of the memory-based method with the efficient run-time complexity of the eager approach.

[1] School of Computer Science, McGill University, Montreal, Canada
[2] Universidade de Aveiro, Aveiro, Portugal
Correspondence: victor.barbaros@mail.mcgill.ca

II. RELATED WORK

In this section, we will give an overview of work on contextual stochastic search methods and non-parametric methods in policy search and reinforcement learning.

A. Stochastic search methods

Several authors have considered optimizing policies for each target context independently and subsequently using regression methods to learn a mapping function to generalize the learned parameter vectors for seen contexts to a new, unseen context [11], [12]. A drawback of such methods is that optimising for each context and generalisation between contexts are two decoupled processes. Hence, learning the optimal policy for each initial context starts from scratch, which can be time consuming.

Learning for multiple tasks without restarting the learning process is known as contextual (multi-task) policy search [5], [8], [13]–[15]. In the area of contextual stochastic search algorithms, a multi-task learning capability was established for information-theoretic policy search algorithms [16]. Examples include the Contextual Relative Entropy Policy Search algorithm [5], Contextual Model Based Relative Entropy Stochastic Search (CMORE) [17] and Contextual Covariance Matrix Adaptation Evolutionary Strategy (C-CMAES) [18]. The general framework of such systems is shown in Fig. 1.

B. Non-parametric methods for learning control

The contextual policy search algorithms in the previous section learn a parametric mapping from context to parameters which is linear in some hand designed features. Therefore these algorithms require a careful feature selection to be able to find a high quality policy for the underlying task. In order to alleviate this problem, non-parametric methods have been proposed. Non-parametric methods do not rely on a fixed set of features, but have a representation that depends on the data set, and can grow in complexity as more data becomes available. Two main types of non-parametric methods include 'eager' learners and 'lazy' learners. Eager learners build a model from the data set, and then use the model to generalize to new data, performing most of the computations up front. Lazy (or memory-based) learners store the data, and then use the data to make predictions whenever a query is made, thus performing most of the computations during deployment.

There are two main sub-types of eager non-parametric learners: those methods that explicitly represent the policy, and those that only represent a value function. These objects of interest are usually represented using a linear combinations of kernels centered at the points in the data set. Methods that do not explicitly represent the policy include non-parametric value iteration methods [19], approximate dynamical programming methods [20]–[22], TD-learning methods [23], and least-squares methods [24]. For control, policy iteration schemes can be employed [25], [26]. The approximate linear programming algorithm [27] does not use kernels. Instead, this model-free method assumes the value function is Lipschitz continuous and assumes deterministic dynamics. These methods tend to use deterministic actions and so cannot easily be applied to robotics problems with continuous controls.

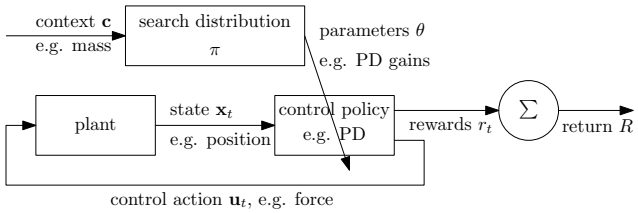

Fig. 1. General framework for contextual policy search. The target is to update the search distribution π so as to maximize the returns.

Policy-search methods, that represent the policy explicitly, are popular for control problems with continuous action spaces. One such method, specific to the stochastic search setting, is cost-regularized kernel regression [28]. For step-based decision making, non-parametric policy gradient methods [29]–[31], path-integral based methods [32] and methods based on a non-parametric Bayesian model [33] are available. The non-parametric relative entropy policy search algorithm [34] uses a non-parametric representation for the transition model, policy, and value function.

Lazy, or memory-based, non-parametric methods for reinforcement learning and policy search have a long history. Like eager methods, they can be subdivided based on what is represented. Some methods used non-parametric methods to represent the transition model, which was subsequently used to learn a policy [35], [36], solving, among others, the task of learning to catch with a simulated robot. Other methods represented the value function or state-action value function in this way [37]–[40]. Finally, a lazy non-parametric method was used to approximate the policy in the non-parametric contextual stochastic search method [15].

In this paper, we want to compare the advantages and disadvantages of memory-based and eager methods on simulated robotic tasks with continuous controls. Since contextual policy search algorithms based on the relative entropy policy search constraint [16] have proven effective in the past [15], we will specifically look at a memory-based method [15] and an eager method [34] based on these principles.

III. METHODS

As discussed in the previous section, in this paper we want to compare a lazy and an eager contextual policy search method for several simulated robot skill learning tasks. In this paper, we will use the following notation. The contexts $\mathbf{c} \in C$ are drawn from some distribution $\mu(\mathbf{c})$ unknown to the learner. The learner then chooses parameters $\theta \in \Theta$ according to the search distribution $\pi(\theta|\mathbf{c})$. Applying controller parameters θ in context $\mathbf{c}$ then yields a return $R_{\mathbf{c}}^{\theta}$, which the agent aims to optimize (this is typically a sum over per time-step rewards).

Without additional constraints, based on a limited data-set choosing a search distribution that maximizes expected return is likely to result in overly greedy updates, that can

cause premature divergence to a suboptimal solution or divergence of the policy. Therefore, we will consider methods based on the relative entropy policy search principle. In relative entropy policy search (REPS), the relative entropy or Kullback-Leibler (KL) divergence between subsequent joint distributions of contexts and parameters (in the stochastic search setting[1]) is constrained. This constraint prevents premature converge or divergence due to overly greedy policy updates [16].

Using this notation, the goal of the agent then is to choose a new distribution $\pi(\theta|\mathbf{c})\mu(\mathbf{c})$ under several constraints that will be explained below. This goal can be formalized as the contextual REPS optimization problem [7]:

$$\max_{\pi,\mu} J(\pi,\mu) = \iint_{\Theta\times C} R^{\theta}_{\mathbf{c}} \pi(\theta|\mathbf{c})\mu(\mathbf{c}) d\theta d\mathbf{c}, \quad (1)$$

$$\text{s.t.} \quad \iint_{\Theta\times C} \pi(\theta|\mathbf{c})\mu(\mathbf{c}) d\theta d\mathbf{c} = 1, \quad (2)$$

$$\iint_{\Theta\times C} \pi(\theta|\mathbf{c})\mu(\mathbf{c})\phi(\mathbf{c}) d\theta d\mathbf{c} = \phi^*, \quad (3)$$

$$\mathrm{KL}(\pi(\theta|\mathbf{c})\mu(\mathbf{c})||q(\mathbf{c},\theta)) \leq \varepsilon. \quad (4)$$

Solutions to this problem aim to find the search distribution π that maximizes the average reward (Eq. 1). Constraints ensure that the joint distribution over contexts and parameters is a probability distribution (Eq. 2) and that the chosen context distribution is approximately equal to the observed context distribution (as approximated by the matching of features $\phi(\mathbf{c})$ to the observed feature average ϕ^*). Finally, to avoid overly greedy optimization, the Kullback-Leibler (KL) divergence from a reference distribution q to the new distribution over contexts and parameters is constrained to some small value ε (Eq. 4). The reference distribution is usually chosen to be the joint distribution from the previous iteration.

Writing down the Lagrangian and solving for the primal parameter $p(\mathbf{c},\theta) = \pi(\theta|\mathbf{c})\mu(\mathbf{c})$ yields

$$\pi(\theta|\mathbf{c}) \propto q(\mathbf{c},\theta) \exp\left(\frac{R^{\theta}_{\mathbf{c}} - \mathbf{v}^T\phi(\mathbf{c})}{\eta}\right), \quad (5)$$

where $\mathbf{v}$ and η are Lagrangian multipliers for constraints (3) and (4), respectively [16]. We solve for these multipliers by minimizing the dual function

$$g(\eta,\mathbf{v}) = \eta\varepsilon + \eta \log\left(\sum_{i=1}^{n} \frac{1}{n} \exp\left(\frac{R^{\theta_i}_{\mathbf{c}_i} - \mathbf{v}^T\phi(\mathbf{c}_i)}{\eta}\right)\right), \quad (6)$$

where integrals in (Eq. 1-4), that express expected values, are approximated using samples $(\mathbf{c}_i,\theta_i)$ from reference distribution $q(\mathbf{c},\theta)$. A detailed derivation of these equation is given in [7], [16].

In general, returns $R^{\theta}_{\mathbf{c}}$ are only known for context-parameter pairs that have been evaluated already. Thus, the policy in Eq. (5) is only defined for sampled context-parameter pairs. To be able to select actions in new contexts, a generalizing policy $\tilde{\pi}$ is fit using supervised learning techniques. Samples from q are used as training data, with the re-weighting terms $d_i = \exp((R^{\theta_i}_{\mathbf{c}_i} - v^T\phi(\mathbf{c}_i))/\eta)$ as importance weights.

[1]In the step-based setting, the joint distribution of states and actions is considered instead.

Algorithm 1 Eager non-parametric stochastic search

for iteration i = 1, ..., iterations do
- **weighting step**
- Input data set $\left[\mathbf{c}^{1:k}, \theta^{1:k}, R^{1:k}\right]$
- Optimize the dual function g for η and α
- Compute the global sample weights d^k
- **policy fitting step**
- Fit global non-parametric policy $\tilde{\pi}(\theta|\mathbf{c})$
- **for episode j = 1, ..., samples (episodes) do**
  - Input query context $\mathbf{c}_j$
  - Sample and execute parameters from $\tilde{\pi}(\theta|\mathbf{c}_j)$

A. Eager non-parametric stochastic search

The standard version of the REPS algorithm as described above relies on a good set of features ϕ. The non-parametric REPS (NP-REPS) algorithm proposed in [34] instead proposes using a data-driven representation. This algorithm can be straightforwardly adapted from the sequential setting to the stochastic search setting[2].

The resulting eager non-parametric stochastic search algorithm solves Eqs. 1–4 for the entire data set. The context matching constraint in Eq. 3 is replaced by a non-parametric variant, which can be interpreted as defining one feature for each point in the data set as a kernel centered on that point. Likewise, the approximating policy $\tilde{\pi}$ is represented non-parametrically by a Gaussian process (GP).

Thus, most computations for eager non-parametric REPS happen off-line. When the model is queried with a new context, the GP policy only needs to be evaluated at that context. The eager non-parametric stochastic search algorithm summarized in Alg. 1.

B. Memory based non-parametric stochastic search

Recently a non-parametric stochastic search method [15], called local Covariance Estimation with Controlled Entropy Reduction (local CECER) was introduced. For every query context, local CECER defines a optimization problem based only on nearby data points. The re-weighting terms then take both the closeness and the quality of data points into account. Subsequently it fits a local Gaussian distribution which is used to generate a new parameter θ^* for the given context. The approximation does not need to be valid globally, so a simple linear model can be used.

At each iteration, for every query context $\mathbf{c}^*$ a locality weighting w^k is computed for each sample, with respect

[2]The algorithm is adapted by considering contexts and parameters instead of states and actions, and by constraining the context distribution to be the observed distribution (as in Eq. 3) rather than constraining it to respect the transition dynamics in the sequential setting.

Algorithm 2 Memory-based non-parametric stochastic search

for iteration i = 1, ..., iterations do
  for episode j = 1, ..., samples (episodes) do
    weighting step
    Input data set $\left[\mathbf{c}^{1:k}, \theta^{1:k}, R^{1:k}\right]$, query context $\mathbf{c}_j$
    Compute locality weightings w_j^k
    Optimize local dual function g_j for η and v
    Compute the local sample weights d_j^k
    policy fitting step
    Fit local generalizing policy $\tilde{\pi}_j(\theta|\mathbf{c}_j)$
    Sample and execute parameters from $\tilde{\pi}_j(\theta|\mathbf{c}_j)$

Essential steps	Memory-based search	Eager search
Weighting	Weighting is performed online for each sample, thus time-consuming	Weighting is done once, offline, for the entire iteration
Policy covariance	Policy can adapt to local changes of the covariance in the sample	Policy covariance determined by density of data, less flexible
Policy fitting	Locally linear policy can be fit efficiently	Fitting complex policy requires expensive and failure-prone optimization

TABLE I
LOCAL CECER VS GLOBAL NP-REPS

to the query context $\mathbf{c}^*$. The locality weighting is further used to compute the re-weighting d^k that allows us to define a local Gaussian search distribution $\pi_*(\theta|\mathbf{c})$. Then, using the search distribution $\pi_*(\theta|\mathbf{c})$ we generate new pairs $(\theta^*, \mathbf{c}^*)$, the quality of which are evaluated through the objective function $R(\theta^*, \mathbf{c}^*)$. Finally, the dataset is updated with the new sample $\{\mathbf{c}^*, \theta^*, \sum^*, \mathbf{R}^*\}$. This memory-based non-parametric stochastic search algorithm is summarized in Alg. 2. The main difference to the eager method, is that in this case the weighting step and the policy fitting step are performed *inside* the inner loop.

C. A novel hybrid policy search method

Eager and memory-based non-parametric search each have their own advantages and disadvantages. For example, the memory-based algorithm has to perform expensive calculations for every query, and thus tends to be slower at run-time.

On the other hand, the local character of the memory-based algorithm allows it to adapt to local characteristics of the gathered data, while the global eager search algorithm sometimes has difficulty finding global hyper-parameters that fit the data in all parts of the state space (as noted in [34]). These respective advantages are summarized in Table I.

Since we determined through preliminary experiments that the re-weighting step is the slowest part of the algorithm, we propose a novel 'hybrid' stochastic search method that combines the off-line calculation of the re-weighting terms from eager stochastic search with the flexible and quick locally linear policy fitting from memory-based stochastic search.

Algorithm 3 Hybrid non-parametric stochastic search

for iteration i = 1, ..., iterations do
  weighting step
  Input data set $\left[\mathbf{c}^{1:k}, \theta^{1:k}, R^{1:k}\right]$
  Optimize the dual function g for η and α
  Compute the global sample weights d^k
  for episode k = 1, ...,samples (episodes) do
    policy fitting step
    Input query context $\mathbf{c}_j$
    Fit local generalizing policy $\tilde{\pi}_j(\theta|\mathbf{c}_j)$
    Sample and execute parameters from $\tilde{\pi}_j(\theta|\mathbf{c}_j)$

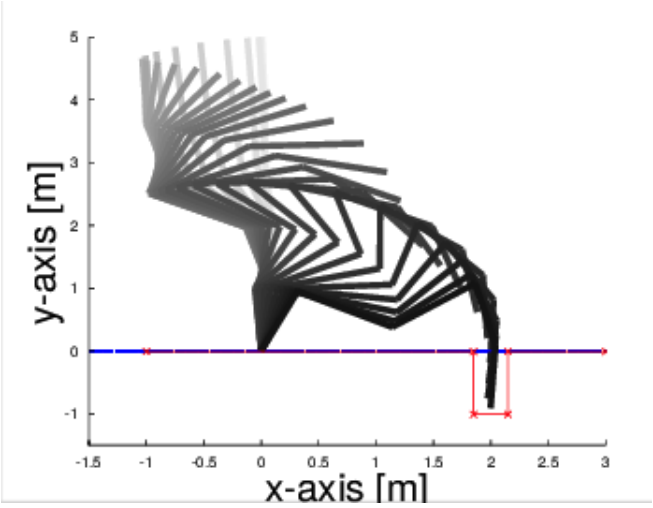

Fig. 2. The main setup for the Hole Reaching task experiment and optimal policy applied. Source - "Non-parametric stochastic policy search" [15].

The hybrid non-parametric stochastic search algorithm should thus inherit the positive aspects of the memory-based and eager algorithms. The hybrid algorithm is summarized in Alg. 3. The main difference to previous methods is that, in this case, the weighting step is performed *outside* the inner loop and the policy fitting step is performed *inside* the inner loop.

IV. EXPERIMENTS

A series of contextual control tasks were utilized to compare our methods. In this section, we describe the two simulated control domains that we considered: the hole reaching task and the cart-pole balancing task.

A. The hole-reaching task

First, we consider a planar Hole Reaching Task which has been proposed in [15]. The Hole Reaching task involves a planar robotic arm consisting of 5 links of unit length. The robot, represented as a decoupled linear dynamical system, needs to reach the bottom of a hole in the task space without colliding with the walls of the given hole.

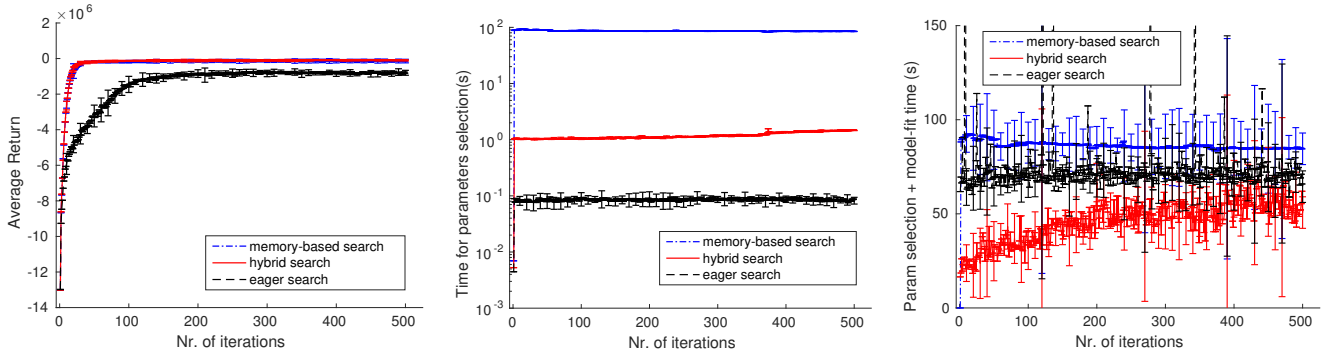

Fig. 3. Left: Average return for the three learners on the Hole Reaching task. Center: Time performance of the three learners on the Hole Reaching task for parameter selection. Right: Time performance of the three learners on the Hole Reaching task on both parameter selection and model fitting. Error bars show four standard deviations calculated over 5 independent learning trials.

We used a dynamic movement primitive (DMP) [41] as low-level controller, with 30 parameters (5 basis functions per dimension and 1 goal position per dimension). As in all of our tasks, we give this task a contextual nature and require our method to solve a wide range of problem variants, indexed by context. The Hole Reaching Task is defined using a three dimensional context: hole position, hole width, and hole depth. Therefore, our agents, once trained, are able to perform reaching into a wide range of hole geometries. All three parameters of the hole are varied uniformly withing a given range: the width is between 0.1m and 0.4m, the center is between 0.5m and 2.5m, and the depth is between 0.5m and 1.5m, as shown in Fig. 2

We used the reward function described in [15], which is defined by a quadratic cost term for each of the following: the desired final point, the high accelerations, and the collision with the environment.

B. The cart-pole balancing task

We also test the cart-pole environment in the OpenAI Gym suite of standard benchmarks [42]. In the Cart-pole balancing task, a cart with a pole connected by a unactuated joint is controlled by discrete actions that apply a force to the cart to the right or to left, respectively. For a successful completion of the task the movements should be applied such that the pole is kept close to the vertical axis. The action can be applied for a maximum number of time-steps and the learning episode is interrupted prematurely if the pole falls below a given threshold. The reward function outputs an integer equal to the number of time-steps successfully keeping the pole balanced. In our settings the maximum number of time-steps is set to 500.

At each episode the cart is initialized at the origin of the x-axis and can be moved up to -2.5 and +2.5 respectively. When the pendulum gets more than 12 degrees from the upright position, the episode is terminated. The standard cart has a mass of 1.0 kg and the pole has a length of 0.5 m. We modified the default cart-pole by varying the cart's mass and pole's length uniformly within a range of 20% of their original values, defining these variables as the context. This variation in the context values implies that from episode to episode the policy needs to adapt to heavier or lighter cart as well as to longer or shorter pole.

The primary performance criterion for each of the above experiments is average reward vs number of roll-outs attempted. Each roll-out allows our method to obtain the true cost of a context-parameter pair and good performance involves learning as quickly as possible from this data. We additionally show two timing graphs for each experiment. First we display the time spent on parameter selection (weighting step), and second the overall computational time (the sum of the weighting step and the policy update step) for each iteration. Note that the y-axis of the plots for the parameter optimization time is in logarithmic scale. We measure these durations separately, as in practical applications we might be more willing to wait off-line for fitting the model, than to wait for the parameters to be selected during operation. Each of our results will be analyzed in the following section.

V. RESULTS

For the first task, the hole reaching task, we generated 3000 initial samples for the first iteration and 600 samples, i.e. 20%, subsequently at each new iteration, keeping always the last 3000 samples in memory. For each metric, we plot the average reward and four times the standard deviation over 5 independent learning trials (starting learning from scratch each time). From the graph of the average reward, Fig. 3 (left), over the first 500 iterations we can see that hybrid REPS and memory-base NPSS have a very similar performance, while the eager REPS method converges slower to the optimal solution and does not reach it by the end of the 500 iterations. The plot of the average time (in seconds) for parameter selection per iteration on a logarithmic scale, Fig. 3 (center), shows the advantage of performing the weighting step offline - the eager REPS method has the shortest selection time, closely followed by the hybrid method, both clearly surpassing the memory-based NPSS. A similar pattern is observed when plotting the time taken for the parameter selection phase and policy fitting step together, Fig. 3 (right), where eager and hybrid REPS exhibit similar behavior. Both of these methods share eager characteristics, and require less time than the memory-based NPSS. In Fig.

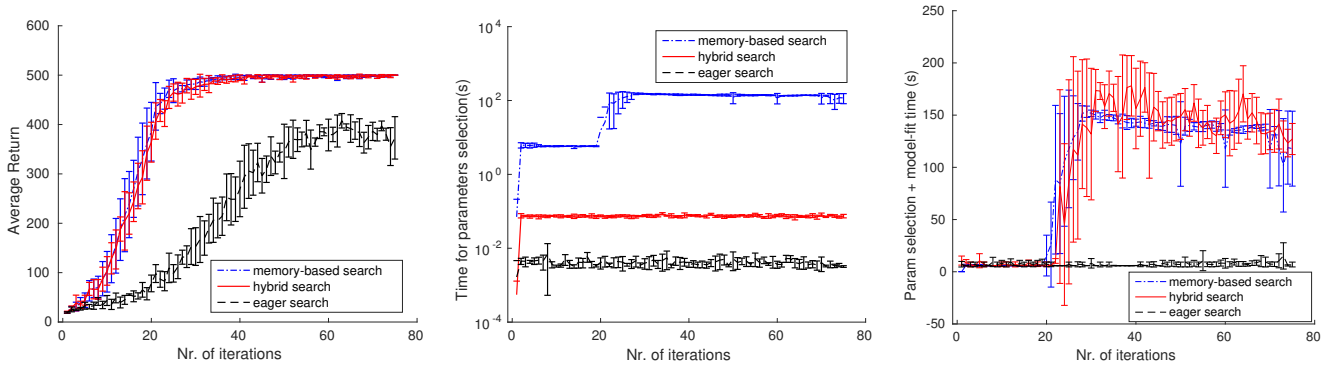

Fig. 4. Left: Learning performance of the three learners on the CartPole balancing task with 2-dimensional context. Center: Time performance of the three learners on parameter selection for the CartPole balancing task with 2-dimensional context. The computation time of the hybrid REPS is close to the one for the global REPS. Right: Time performance of the three learners on the CartPole balancing task with 2-dimensional context for both parameter selection and model fitting. Time complexity of the hybrid REPS is close to the one for the global REPS. Error bars show two standard deviations calculated over 5 independent learning trials.

3 (right) we also see occasional spikes for the global REPS method during the model fitting phase. We presume this is caused by the difficulty in finding good model parameters for certain data sets.

For the experiment with two-dimensional context of the cart-pole balancing task we generate 300 samples for the first iteration and 60 samples, i.e. 20%, subsequently at each new iteration, always keeping the last 300 samples. For each metric, we plot the average reward and two times the standard deviation over 5 independent learning trials (starting learning from scratch each time). The average return as a function of the number of training rollouts is shown in Fig. 4 (left). This figure shows that the hybrid REPS and memory-based NPSS yield similar learning curves while the global REPS needs more iterations before starting to find an optimal policy. The plot of the average time (in seconds) taken by each algorithm to select parameters in each iteration in logarithmic scale, Fig. 4 (center), again displays a better performance for eager REPS, closely followed by the hybrid method. Both methods clearly surpass memory-based NPSS. Here it is worth mentioning the sudden increase in the time taken for both the memory-based NPSS and the hybrid REPS to find optimal parameters – this could be caused by the fact that the optimization becomes harder as all samples perform relatively well, rather than having some samples that do very poorly and are easily discarded.

VI. CONCLUSIONS

This paper has compared two non-parametric stochastic search methods for contextual control learning that were derived from recent work. An eager method computes a global mapping of contexts to controller parameters offline and then performs rapid but inflexible computations at run-time. In contrast, a memory-based method requires little training time and performs well with flexible, local operations, but spends significant time on selecting parameters at run-time. Our novel hybrid algorithm is both efficient and high-performing – the best of both worlds.

Contextual policy search is a highly relevant task for robotics, where the environment is continually changing and we can not afford a long re-learning process. For example, contextual information can be extracted through visual perception, allowing natural generalization of behaviors from sensed changes in the environment. Our hybrid computational model will have particular benefit for visual contexts as the balance between efficiency and performance is paramount and we plan to investigate this extension in future work.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN), and the discovery grants program.

REFERENCES

[1] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, 2013. [Online]. Available: http://www.ias.tu-darmstadt.de/uploads/Site/EditPublication/PolicySearchReview.pdf

[2] T. Rückstieß, M. Felder, and J. Schmidhuber, "State-dependent Exploration for Policy Gradient Methods," in *Proceedings of the European Conference on Machine Learning (ECML)*, 2008.

[3] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber, "Efficient Natural Evolution Strategies," in *Proceedings of the Annual conference on Genetic and evolutionary computation (GECCO)*, 2009. [Online]. Available: http://doi.acm.org/10.1145/1569901.1569976

[4] F. Stulp and O. Sigaud, "Path Integral Policy Improvement with Covariance Matrix Adaptation," in *International Conference on Machine Learning (ICML)*, 2012.

[5] 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, special Issue on AI and Robotics.

[6] A. Abdolmaleki, N. Lau, L. Paulo Reis, and G. Neumann, "Contextual stochastic search," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 2016, pp. 29–30.

[7] C. Daniel, G. Neumann, O. Kroemer, and J. Peters, "Hierarchical relative entropy policy search," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3190–3239, 2016.

[8] S. Ha and C. Liu, "Evolutionary optimization for parameterized whole-body dynamic motor skills," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[9] C. G. Atkeson, "Using locally weighted regression for robot learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1991, pp. 958–963.

[10] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[11] B. Da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," *International Conference on Machine Learning (ICML)*, 2012.

[12] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[13] J. Kober, E. Oztop, and J. Peters, "Reinforcement Learning to adjust Robot Movements to New Situations," in *Proceedings of Robotics: Science and Systems (R:SS)*, 2010.

[14] V. Tangkaratt, H. van Hoof, S. Parisi, G. Neumann, J. Peters, and M. Sugiyama, "Policy search with high-dimensional context variables," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[15] A. Abdolmaleki, N. Lau, L. P. Reis, and G. Neumann, "Non-parametric contextual stochastic search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*, 2016, pp. 2643–2648.

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

[17] A. Abdolmaleki, R. Lioutikov, J. Peters, N. Lua, L. Reis, and G. Neumann, "Model Based Relative Entropy Stochastic Search," in *Advances in Neural Information Processing Systems (NIPS), MIT Press*, 2015.

[18] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)." *Evolutionary Computation*, 2003.

[19] S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton, "Modelling transition dynamics in MDPs with RKHS embeddings," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2012, pp. 535–542.

[20] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[21] G. Taylor and R. Parr, "Kernelized value function approximation for reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 1017–1024.

[22] O. Kroemer and J. Peters, "A non-parametric approach to dynamic programming," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 1719–1727.

[23] Y. Engel, S. Mannor, and R. Meir, "Bayes meets Bellman: The Gaussian process approach to temporal difference learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 20, 2003, pp. 154–161.

[24] T. Jung and D. Polani, "Kernelizing LSPE(λ)," in *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 338–345.

[25] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[26] C. E. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning." in *Advances in Neural Information Processing Systems (NIPS)*, vol. 4, 2003, p. 1.

[27] J. Pazis and R. Parr, "Non-parametric approximate linear programming for MDPs." in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2011, pp. 459–464.

[28] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 2650–2655.

[29] G. Lever and R. Stafford, "Modelling policies in MDPs in reproducing kernel Hilbert space," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, 2015, pp. 590–598.

[30] J. Bagnell and J. Schneider, "Policy search in reproducing kernel Hilbert space," CMU, Tech. Rep. RI-TR-03-45, 2003.

[31] N. Vien, P. Englert, and M. Toussaint, "Policy search in reproducing kernel Hilbert space," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[32] K. Rawlik, M. Toussaint, and S. Vijayakumar, "Path integral control by reproducing kernel Hilbert space embedding," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[33] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2011, pp. 465–472.

[34] H. Van Hoof, G. Neumann, and J. Peters, "Non-parametric policy search with limited information loss," *Journal of Machine Learning Research*, vol. 18, no. 73, pp. 1–46, 2017.

[35] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 75–113, Feb 1997.

[36] D. W. Aha and S. L. Salzberg, "Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks," in *Selecting Models from Data: Artificial Intelligence and Statistics IV*, P. Cheeseman and R. W. Oldford, Eds. New York, NY: Springer New York, 1994, pp. 321–328.

[37] P. Tadepalli and D. Ok, "Scaling up average reward reinforcement learning by approximating the domain models and the value function," in *International Conference on Machine Learning*, 1996, pp. 471–479.

[38] L. C. Baird and A. H. Klopf, "Reinforcement learning with high-dimensional continuous actions," Wright Laboratory, Wright-Patterson Air Force Base, Tech. Rep. WL-TR-93-1147, 1993.

[39] J. Peng, "Efficient memory-based dynamic programming," in *Proceedings of the International Conference on Machine Learning (ICML)*, 1995, pp. 438–446.

[40] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2000, pp. 903–910.

[41] A. Ijspeert and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in Neural Information Processing Systems, (NIPS)*, 06 2003.

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