ES-MAML: Simple Hessian-Free Meta Learning

Xingyou Song, Yuxiang Yang,[*] Krzysztof Choromanski
Google Brain
{xingyousong,yxyang,kchoro}@google.com

Aldo Pacchiano
UC Berkeley
pacchiano@berkeley.edu

Wenbo Gao,[†] Yunhao Tang[†]
Columbia University
{wg2279,yt2541}@columbia.edu

Abstract

We introduce *ES-MAML*, a new framework for solving the *model agnostic meta learning* (MAML) problem. Existing algorithms for MAML are based on policy gradients. We show how MAML can be solved using Evolution Strategy (ES) algorithms, which are conceptually simpler and easier to implement than policy gradient-based methods. Moreover, many techniques for improving the performance and exploration of ES methods become applicable to MAML. We show empirically that ES-MAML is competitive with existing methods and often yields better adaptation with fewer queries.

1 Introduction

We assume that the reader is familiar with *model agnostic meta learning* (MAML) [1, 2], an approach for few-shot learning on multiple tasks. To introduce our notation, we have a collection of tasks parameterized by $\mathcal{T}$, and each task has an associated *reward function* $f^{\mathcal{T}}$ and *adaptation operator* $U^{\mathcal{T}}$. The goal of MAML is to find a *meta-policy* θ^* such that adapting θ^* is on average successful across all tasks; that is, we aim to maximize $M(\theta) = \mathbb{E}_{\mathcal{T}} f^{\mathcal{T}}(U^{\mathcal{T}}(\theta))$. Existing algorithms for MAML are based on policy gradients. We propose an alternate paradigm: *evolution strategies* (ES) for MAML.

ES [3], or alternately *random search* [4], is a family of zero-order algorithms for optimizing potentially nonsmooth functions $f(\theta)$ based on the *Gaussian smoothing* $\widetilde{f}_\sigma(\theta) = \mathbb{E}_g[f(x + \sigma g)]$, where $g \sim \mathcal{N}(0, I)$. It can be shown that $\widetilde{f}_\sigma$ is differentiable, and its gradient can be expressed as

$$\nabla \widetilde{f}_\sigma(\theta) = \frac{1}{\sigma} \mathbb{E}_g[f(x + \sigma g) g] \tag{1}$$

which can be estimated by a simple Monte Carlo algorithm (Algorithm 1). This allows us to estimate stochastic gradients of $\widetilde{f}_\sigma(\theta)$ as an input to a stochastic first-order method.

One major disadvantage of policy gradient-based MAML (henceforth, PG-MAML) is the need to evaluate second derivatives of the reward. The standard adaptation operator $U^{\mathcal{T}}$ is a single gradient step, i.e $U^{\mathcal{T}}(\theta) = \theta + \alpha \nabla f^{\mathcal{T}}(\theta)$. As a result, the gradient $\nabla M(\theta)$ of the MAML function involves the Hessian $\nabla^2 f^{\mathcal{T}}(\theta)$, which is difficult to compute. Multiple authors [5, 6] have pointed out that the original implementation of MAML incorrectly estimates the Hessian. Improvements to the original MAML include ProMP [5], which introduces a new low-variance curvature (LVC) estimator for the Hessian, and T-MAML [6], which adds control variates to reduce the variance of the unbiased DiCE estimator [7]. However, these are not without their drawbacks: the proposed solutions are

[*]Work performed during the Google AI Residency Program. http://g.co/airesidency
[†]Work done while as intern at Google.

complicated, the variance of the Hessian estimate remains problematic, and LVC introduces an unknown estimator bias.

```
**ESGrad** $(f, \theta, n, \sigma)$
    inputs: function $f$, policy $\theta$, number of perturbations $n$, precision $\sigma$
    Sample $n$ i.i.d $N(0, I)$ vectors $g_1, \ldots, g_n$;
    return $\frac{1}{n\sigma} \sum_{i=1}^{n} f(\theta + \sigma g_i) g_i$;
```

Algorithm 1: Monte Carlo ES Gradient

In contrast, we will derive an algorithm for MAML based on ES which requires only zero-order evaluations, eliminating the need to explicitly compute gradients or Hessians. This has several advantages:

1. It enables us to use novel adaptation operators $U^{\mathcal{T}}$ which are either nonsmooth, or intractable to differentiate.
2. We can use deterministic policies instead of stochastic policies (required in PG). Empirical experiments from robotics suggest that the action randomization of stochastic policies can often be risky when adapting on real hardware. In particular, we consider *population search* operators [8] and *DPP ES* [9], which encourage exploratory behavior and are effective for tasks with sparse rewards.

   Empirically, ES-MAML also seems to be superior in training linear or compact policies than PG-MAML.
3. Policy gradients are computationally expensive and require careful hyperparameter tuning (see Alpha-MAML [10]), or annealing of learning rates (e.g [11]) to be effective.

Aside from PG-MAML, for which there is extensive literature, and our proposed ES-MAML, there are also biologically-inspired methods for meta-learning [12, 13]. We wish to emphasize that despite the similar names, our algorithm is fundamentally different from "Evolvability ES" [13]. Our algorithm solves the standard MAML problem, whereas [13] focuses on a loosely-related notion of "behavioral diversity", and our algorithm is derived from rigorous smoothings and estimations.

2 ES-MAML

Suppose that we can evaluate (or approximate) $f^{\mathcal{T}}$ and $U^{\mathcal{T}}$, but $f^{\mathcal{T}}, U^{\mathcal{T}}$ may be nonsmooth or their gradients may be intractable. We consider the Gaussian smoothing $\widetilde{M}$ of the MAML reward $\mathbb{E}_{\mathcal{T}} f^{\mathcal{T}}(U^{\mathcal{T}}(\theta))$, and optimize $\widetilde{M}$ using ES. The gradient $\nabla \widetilde{M}$ is given by

$$\mathbb{E}_g \mathbb{E}_{\mathcal{T}}[\frac{1}{\sigma} f^{\mathcal{T}}(U^{\mathcal{T}}(\theta + \sigma g))g] \tag{2}$$

and can be estimated by jointly sampling over (τ, g) and evaluating $f^{\mathcal{T}}(U^{\mathcal{T}}(\theta + \sigma g))$. This algorithm is specified in Algorithm 2, and we refer to it as *(zero-order) ES-MAML.*

```
Data: initial policy $\theta_0$, meta step size $\beta$
for $t = 0, 1, \ldots$ do
    Sample $n$ tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and $n$ i.i.d Gaussian perturbations $g_1, \ldots, g_n$;
    foreach $(\tau_i, g_i)$ do
        $v_i \leftarrow f^{\mathcal{T}_i}(U^{\mathcal{T}_i}(\theta_t + \sigma g_i))$
    end
    $\theta_{t+1} \leftarrow \theta_t + \frac{\beta}{\sigma n} \sum_{i=1}^{n} v_i g_i$
end
```

Algorithm 2: Zero-Order ES-MAML (general adaptation operator $U^{\mathcal{T}}$)

```
Data: initial policy $\theta_0$, adaptation step size $\alpha$, meta step size $\beta$, number of queries $K$
for $t = 0, 1, \ldots$ do
    Sample $n$ tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and $n$ i.i.d Gaussian perturbations $g_1, \ldots, g_n$;
    foreach $(\mathcal{T}_i, g_i)$ do
        $d^{(i)} \leftarrow \text{ESGrad}(f^{\mathcal{T}_i}, \theta_t + \sigma g_i, K, \sigma)$;
        $\theta_t^{(i)} \leftarrow \theta_t + \sigma g_i + \alpha d^{(i)}$;
        $v_i \leftarrow f^{\mathcal{T}}(\theta_t^{(i)})$;
    end
    $\theta_{t+1} \leftarrow \theta_t + \frac{\beta}{n\sigma} \sum_{i=1}^{n} v_i g_i$;
end
```

Algorithm 3: Zero-Order ES-MAML

The adaptation operator we consider most often is the one-step task gradient. Since $f^{\mathcal{T}}$ is permitted to be nonsmooth in our setting, we use the adaptation operator $U^{\mathcal{T}}(\theta) = \theta + \alpha \nabla \widetilde{f}^{\mathcal{T}}(\theta)$; that is, the gradient taken is of the smoothing $\widetilde{f}^{\mathcal{T}}$. Expanding the definition of $\widetilde{M}$, the gradient of the smoothed MAML is then given by

$$\nabla \widetilde{J}_{\sigma}(\theta) = \frac{1}{\sigma} \mathbb{E}_{\substack{\mathcal{T}\sim\mathcal{P}(\mathcal{T}) \\ \mathbf{g}\sim\mathcal{N}(0,\mathbf{I})}} \left[f^{T}\left(\theta + \sigma\mathbf{g} + \frac{1}{\sigma}\mathbb{E}_{\mathbf{h}\sim\mathcal{N}(0,\mathbf{I})}[f^{T}(\theta+\sigma\mathbf{g}+\sigma\mathbf{h})\mathbf{h}]\right)\mathbf{g}\right]. \tag{3}$$

We describe an algorithm for this case in Algorithm 3, where the adaptation operator $U^{\mathcal{T}}$ is itself estimated using ES in the inner loop.

We can also derive an algorithm analogous to PG-MAML by applying a first-order method to the MAML reward $\mathbb{E}_{\mathcal{T}} \widetilde{f}^{\mathcal{T}}(\theta + \alpha\nabla\widetilde{f}^{\mathcal{T}}(\theta))$ directly. The gradient is then given by $\nabla M(\theta) = \mathbb{E}_{\mathcal{T}} \nabla \widetilde{f}^{\mathcal{T}}(\theta + \alpha\nabla\widetilde{f}^{\mathcal{T}}(\theta))(I + \alpha\nabla^2\widetilde{f}^{\mathcal{T}}(\theta))$, which corresponds to e.g. equation (3) in [6] when expressed in terms of policy gradients. Every term in this expression has a simple Monte Carlo estimator. We discuss this algorithm (Algorithm 5) in greater detail in Appendix A.1. This formulation can be viewed as the "**MAML of the smoothing**", compared to the "**smoothing of the MAML**" which is the basis for Algorithm 2. It is the additional smoothing present in equation 2 which eliminates the gradient of $U^{\mathcal{T}}$ (and hence, the second derivatives of $f^{\mathcal{T}}$). We find this to be a significant advantage of Algorithm 2 over more specialized algorithms such as Algorithm 5.

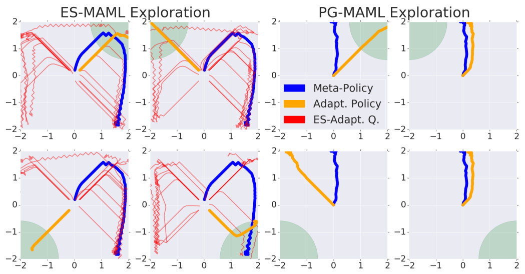
3 Experiments

We write K for the number of *queries* (rollouts) used by the adaptation operator; an algorithm is typically superior if it can find a strong meta-policy that can adapt with smaller K.

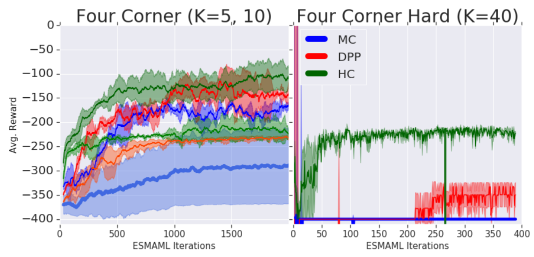
3.1 Exploration: Four Corners

The *four corners* benchmark was introduced in [5] to demonstrate the weaknesses of exploration in PG-MAML. An agent on a 2D square receives reward for moving towards a selected corner of the square, but only observes rewards once it is sufficiently close to the target corner, making the reward sparse. An effective exploration strategy for this set of tasks is for the meta-policy θ^* to travel in circular trajectories to observe which corner produces rewards; however, for a single policy to produce this exploration behavior is difficult.

Figure 1: (a) ES-MAML and PG-MAML exploration behavior. (b) Different exploration methods when K is limited ($K = 5$ plotted with lighter colors) or large penalties are added on wrong goals.



(a)



(b)

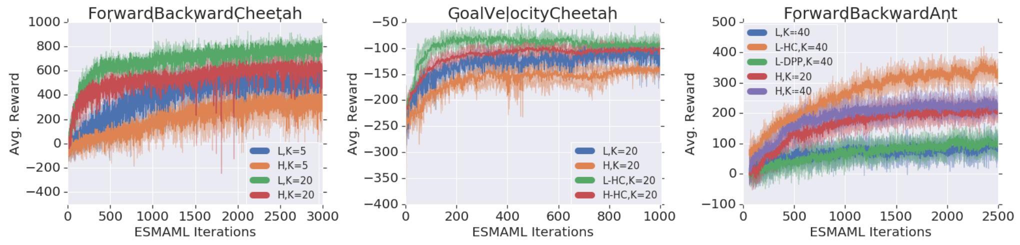
In Figure 1, we demonstrate the behavior of ES-MAML on the four corners problem. When $K = 20$, the same number of rollouts for adaptation as used in [5], the basic version of Algorithm 3 is able to correctly explore and adapt to the task by finding the target corner. Moreover, it does not require any modifications to encourage exploration, unlike PG-MAML. We further used $K = 10, 5$, which caused the performance to drop. For better performance in this low-information environment, we experimented with two different adaptation operators $U^{\mathcal{T}}$ in Algorithm 2. The *hill climbing* (HC) operator is a type of population search which randomly perturbs the best policy observed, queries the perturbed policy, and records the best observed policy. The *DPP-ES* operator [9] uses an enhanced

ES gradient which generates the perturbations g_i from DPP sampling [14, 15] instead of using i.i.d normal perturbations. ES-MAML was able to train with both operators, and we found hill climbing to be particularly successful for solving this task with limited information. This shows the utility of ES-MAML in handling nonsmooth operators such as HC.

3.2 Good Adaptation with Compact Architectures

One of the main benefits of ES is due to its ability to train compact linear, deterministic policies. We demonstrate this on several benchmark MAML problems in the HalfCheetah and Ant environments in Figure 2. In contrast, [16] observed that PG-MAML was successful with relatively large neural networks. We find that on the Forward-Backward and Goal-Velocity MAML benchmarks, ES-MAML is consistently able to train successful linear policies faster than deep networks. We also find again, for the Forward-Backward Ant problem, that ES-MAML with the new HC operator is the most performant. Using more compact policies also directly speeds up ES-MAML, since fewer perturbations are needed for gradient estimation.

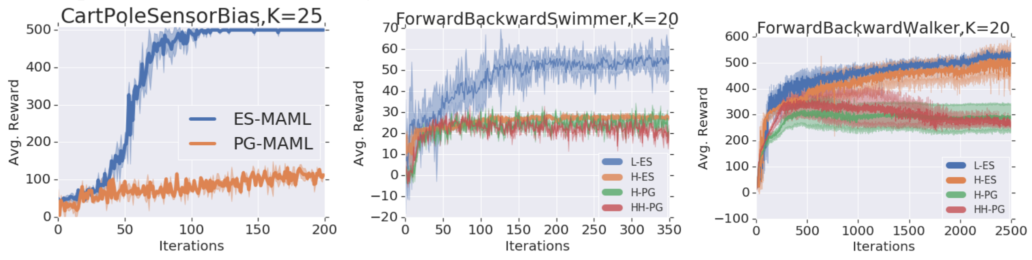
Figure 2: The Forward-Backward and Goal-Velocity MAML problems. We compare the performance for Linear (L) policies and policies with one hidden layer (H) for different numbers of queries K.



3.3 Exploration with Deterministic Policies

We find that deterministic policies often produce more stable behavior than the stochastic policies required for PG, where randomized actions in unstable environments can lead to catastrophic outcomes. In PG, this is often mitigated by reducing the entropy bonus, but this has an undesirable side effect of reducing exploration. In contrast, ES-MAML explores in parameter space, which mitigates this issue. To demonstrate this, we use the "Biased-Sensor CartPole" environment from [17]. This environment has unstable dynamics and sparse rewards, so it requires exploration but is also risky. We see in Figure 3 that ES-MAML is able to stably maintain the maximum reward (500). We also include results in Figure 3 from two other environments, Swimmer and Walker2d, for which it is known that PG is surprisingly unstable, and ES yields better training [4].

Figure 3: Stability comparisons of ES and PG on the Biased-Sensor CartPole and Swimmer, Walker2d environments. (L) denotes linear policies, (H) denotes policies with one hidden layer, and (HH) denotes policies with two hidden layers.



4 Conclusions and Further Discussion

We have presented a new algorithm for MAML based on ES algorithms. The ES approach avoids the problems of Hessian estimation which necessitated complicated alterations in PG-MAML [5, 6] and is easy to implement. Further extensions and analysis are shown in the Appendix.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1126–1135, 2017.

[2] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 9537–9548, 2018.

[3] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv:1712.06567*, 2017.

[4] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *Advances in Neural Information Processing Systems 31*, pages 1800–1809, 2018.

[5] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[6] Hao Liu, Richard Socher, and Caiming Xiong. Taming MAML: efficient unbiased meta-reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 4061–4071, 2019.

[7] Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. DiCE: The infinitely differentiable Monte Carlo estimator. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1529–1538, 2018.

[8] David Moriarty, Alan Schultz, and John Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[9] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, and Yunhao Tang. Structured monte carlo sampling for nonisotropic distributions via determinantal point processes. *arXiv:1905.12667*, 2019.

[10] Harkirat Singh Behl, Atilim Günes Baydin, and Philip H. S. Torr. Alpha MAML: adaptive model-agnostic meta-learning. *CoRR*, abs/1905.07435, 2019.

[11] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[12] Chrisantha Fernando, Jakub Sygnowski, Simon Osindero, Jane Wang, Tom Schaul, Denis Teplyashin, Pablo Sprechmann, Alexander Pritzel, and Andrei A. Rusu. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, pages 109–110, 2018.

[13] Alexander Gajewski, Jeff Clune, Kenneth O. Stanley, and Joel Lehman. Evolvability ES: scalable and direct optimization of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 107–115, 2019.

[14] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

[15] Christian Wachinger and Polina Golland. Sampling from determinantal point processes for scalable manifold learning. *Information Processing for Medical Imaging*, pages 687—-698, 2015.

[16] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[17] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Jie Tan, and Chelsea Finn. Norml: No-reward meta learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 323–331, 2019.

[18] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[19] Jose Blanchet, Donald Goldfarb, Garud Iyengar, Fengpei Li, and Chaoxu Zhou. Unbiased simulation for optimizing stochastic function compositions. *arXiv:1711.07564*, 2017.

[20] Mengdi Wang, Ji Liu, and Xingyuan Fang. Accelerating stochastic composition optimization. *Journal of Machine Learning Research*, 18:1–23, 2017.

[21] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[22] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 969–977, 2018.

[23] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, and Yunhao Tang. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. *NeurIPS 2019*, 2019.

[24] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Deepali Jain, Yuxiang Yang, Atil Iscen, Jasmine Hsu, and Vikas Sindhwani. Provably robust blackbox optimization for reinforcement learning. *accepted to CoRL 2019*, 2019.

[25] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

A.1 First-Order ES-MAML

A.1.1 Algorithm

Suppose that we *first* apply Gaussian smoothing to the task rewards and then form the MAML problem, so we have $J(\theta) = \mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})} \widetilde{f}^T(U(\theta, T))$. The function J is then itself differentiable, and we can directly apply first-order methods to it. The classical case where $U(\theta, T) = \theta + \alpha\nabla\widetilde{f}^T(\theta)$ yields the gradient

$$\nabla J(\theta) = \mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})} \nabla\widetilde{f}^T(\theta + \alpha\nabla\widetilde{f}^T(\theta))(\mathbf{I} + \alpha\nabla^2\widetilde{f}^T(\theta)). \tag{4}$$

This is analogous to formulas obtained in e.g [6] for the policy gradient MAML. We can then approximate this gradient as an input to stochastic first-order methods. An example with standard SGD is shown in Algorithm 5.

```
ESHess $(f, \theta, n, \sigma)$
  inputs : function $f$, policy $\theta$, number of
           perturbations $n$, precision $\sigma$
  Sample i.i.d $\mathcal{N}(0, \mathbf{I})$ vectors $\mathbf{g}_1, \ldots, \mathbf{g}_n$;
  $v \leftarrow \frac{1}{n}\sum_{i=1}^n f(\theta + \sigma\mathbf{g}_i)$;
  $\mathbf{H}^0 \leftarrow \frac{1}{n}\sum_{i=1}^n f(\theta + \sigma\mathbf{g}_i)\mathbf{g}_i\mathbf{g}_i^T$;
  return $\frac{1}{\sigma^2}(\mathbf{H}^0 - v \cdot \mathbf{I})$;
```

Algorithm 4: Monte Carlo ES Hessian

```
Data: initial policy $\theta_0$, adaptation step size $\alpha$,
      meta step size $\beta$, number of queries $K$
for $t = 0, 1, \ldots$ do
    Sample $n$ tasks $T_1, \ldots, T_n$;
    foreach $T_i$ do
        $\mathbf{d}_1^{(i)} \leftarrow \mathrm{ESGRAD}(f^{T_i}, \theta_t, K, \sigma)$;
        $\mathbf{H}^{(i)} \leftarrow \mathrm{ESHESS}(f^{T_i}, \theta_t, K, \sigma)$;
        $\theta_t^{(i)} \leftarrow \theta_t + \alpha \cdot \mathbf{d}_i$;
        $\mathbf{d}_2^{(i)} \leftarrow \mathrm{ESGRAD}(f^{T_i}, \theta_t^{(i)}, K, \sigma)$;
    end
    $\theta_{t+1} \leftarrow \theta_t + \frac{\beta}{n}\sum_{i=1}^n(\mathbf{I} + \alpha\mathbf{H}^{(i)})\mathbf{d}_2^{(i)}$;
end
```

Algorithm 5: First Order ES-MAML

A central problem, as discussed in [5, 6] is the estimation of $\nabla^2\widetilde{f}^T(\theta)$. However, a simple expression exists for this object in the ES setting; it can be shown that

$$\nabla^2\widetilde{f}^T(\theta) = \frac{1}{\sigma^2}(\mathbb{E}_{\mathbf{h}\sim\mathcal{N}(0,\mathbf{I})}[f^T(\theta + \sigma\mathbf{h})\mathbf{h}\mathbf{h}^T] - \widetilde{f}^T(\theta)\mathbf{I}]. \tag{5}$$

Note that for the vector $\mathbf{h}$, $\mathbf{h}^T$ is the transpose (and unrelated to tasks T). A basic MC estimator is shown in Algorithm 4. Given an independent estimator for $\nabla\widetilde{f}^T(\theta + \alpha\nabla\widetilde{f}^T(\theta))$, we can then take the product to obtain an estimator for ∇J.

A.1.2 Experiments with First-Order ES-MAML

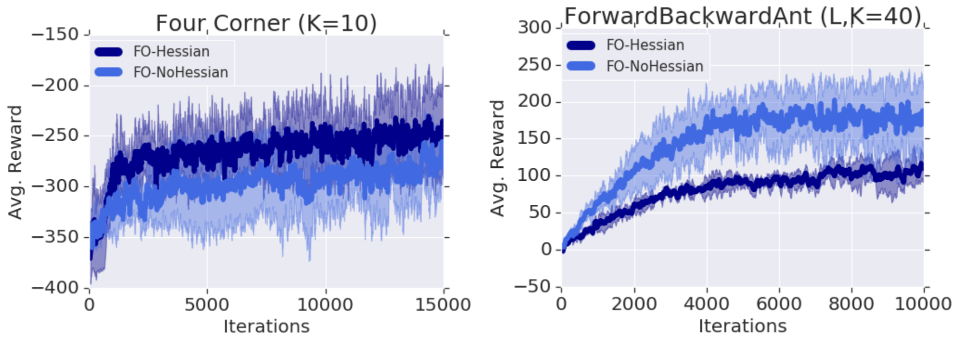
Unlike zero-order ES-MAML (Algorithm 3), the first-order ES-MAML explicitly builds an approximation of the Hessian of f^T. Given the literature on PG-MAML, we expect that estimating the Hessian $\nabla^2\widetilde{f}^T(\theta)$ with Algorithm 4 without any control variates may have high variance. We compare two variants of first-order ES-MAML:

1. The full version (FO-Hessian) specified in Algorithm 5.
2. The 'first-order approximation' (FO-NoHessian) which ignores the term $\mathbf{I} + \alpha\nabla^2\widetilde{f}^T(\theta)$ and approximates the MAML gradient as $\mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})}\nabla\widetilde{f}^T(\theta + \alpha\nabla\widetilde{f}^T(\theta))$. This is equivalent to setting $\mathbf{H}^{(i)} = 0$ in line 5 of Algorithm 5.

The results on the four corner exploration problem (Section 3.1) and the Forward-Backward Ant, using Linear policies, are shown in Figure A1. On Forward-Backward Ant, FO-NoHessian actually outperformed FO-Hessian, so the inclusion of the Hessian term actually slowed convergence. On the four corners task, both FO-Hessian and FO-NoHessian have large error bars, and FO-Hessian slightly outperforms FO-NoHessian.

There is conflicting evidence as to whether the same phenomenon occurs with PG-MAML; [1, §5.2] found that on *supervised learning* MAML, omitting Hessian terms is competitive but slightly worse

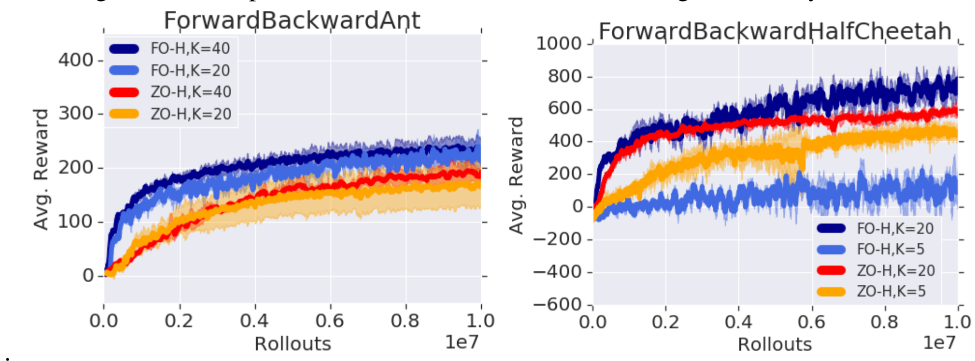
Figure A1: Comparisons between the FO-Hessian and FO-NoHessian variants of Algorithm 5.



than the full PG-MAML, and does not report comparisons with and without the Hessian on RL MAML. [5, 6] argue for the importance of the second-order terms in proper credit assignment, but use heavily modified estimators (LVC, control variates; see Algorithm 3) in their experiments, so the performance is not directly comparable to the 'naive' estimator in Algorithm 4. Our interpretation is that Algorithm 4 has high variance, making the Hessian estimates inaccurate, which can slow training on relatively 'easier' tasks like Forward-Backward walking but possibly increase the exploration on four corners.

We also compare FO-NoHessian against Algorithm 3 on Forward-Backward HalfCheetah and Ant in Figure A2. In this experiment, the two methods ran on servers with different number of workers available, so we measure the score by the total number of rollouts. We found that FO-NoHessian was slightly faster than Algorithm 3 when measured by rollouts on Ant, but FO-NoHessian had notably poor performance when the number of queries was low ($K = 5$) on HalfCheetah, and failed to reach similar scores as the others even after running for many more rollouts.

Figure A2: Comparisons between FO-NoHessian and Algorithm 3, by rollouts



A.2 Handling Estimator Bias

Since the adapted policy $U(\theta, T)$ generally cannot be evaluated exactly, we cannot easily obtain unbiased estimates of $f^T(U(\theta, T))$. This problem arises for both PG-MAML and ES-MAML.

We consider PG-MAML first as an example. In PG-MAML, the adaptation operator is $U(\theta, T) = \theta + \alpha \nabla_\theta \mathbb{E}_{\tau \sim \mathcal{P}_\mathcal{T}(\tau|\theta)}[R(\tau)]$. In general, we can only obtain an estimate of $\nabla_\theta \mathbb{E}_{\tau \sim \mathcal{P}_\mathcal{T}(\tau|\theta)}[R(\tau)]$ and not its exact value. However, the MAML gradient is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{T} \sim \mathcal{P}(\mathcal{T})}[\mathbb{E}_{r' \sim \mathcal{P}_\mathcal{T}(\tau'|\theta')}[\nabla_{\theta'} \log \mathcal{P}_\mathcal{T}(\tau'|\theta') R(\tau') \nabla_\theta U(\theta, T)]] \tag{6}$$

which requires exact sampling from the adapted trajectories $\tau' \sim \mathcal{P}_\mathcal{T}(\tau'|U(\theta, T))$. Since this is a nonlinear function of $U(\theta, T)$, we cannot obtain unbiased estimates of $\nabla J(\theta)$ by sampling τ' generated by an *estimate* of $U(\theta, T)$.

In the case of ES-MAML, the adaptation operator is $U(\theta, T) = \theta + \alpha \nabla \widetilde{f}(\theta, T) = \mathbb{E}_{\mathbf{h}} u(\theta, T; \mathbf{h})$ for $\mathbf{h} \sim \mathcal{N}(0, I)$, where $u(\theta, T; \mathbf{h}) = \theta + \frac{\alpha}{\sigma} f^T(\theta + \sigma \mathbf{h})\mathbf{h}$. Clearly, $f^T(u(\theta, T; \mathbf{h}))$ is not an unbiased estimator of $f^T(U(\theta, T))$.

We may question whether using an unbiased estimator of $f^T(U(\theta, T))$ is likely to improve performance. One natural strategy is to reformulate the objective function so as to make the desired estimator unbiased. This happens to be the case for the algorithm E-MAML [18], which treats the adaptation operator as an explicit function of K sampled trajectories and "moves the expectation outside". That is, we now have an adaptation operator $U(\theta, T; \tau_1, \ldots, \tau_K)$, and the objective function becomes

$$\mathbb{E}_T[\mathbb{E}_{\tau_1,\ldots,\tau_k \sim \mathcal{P}_{\mathcal{T}}(\tau|\theta)} f^T(U(\theta, T; \tau_1, \ldots, \tau_K))] \tag{7}$$

An unbiased estimator for the E-MAML gradient can be obtained by sampling only from $\tau \sim \mathcal{P}_{\mathcal{T}}(\tau|\theta)$ [18]. However, it has been argued that by doing so, E-MAML does not properly assign credit to the pre-adaptation policy [5]. Thus, this particular mathematical strategy seems to be disadvantageous for RL.

The problem of finding estimators for function-of-expectations $f(\mathbb{E}X)$ is difficult and while general unbiased estimation methods exist [19], they are often complicated and suffer from high variance. In the context of MAML, ProMP compares the *low variance curvature* (LVC) estimator [5], which is biased, against the unbiased DiCE estimator [7], for the Hessian term in the MAML gradient, and found that the lower variance of LVC produced better performance than DiCE. Alternatively, control variates can be used to reduce the variance of the DiCE estimator, which is the approach followed in [6].

In the ES framework, the problem can also be formulated to avoid exactly evaluating $U(\cdot, T)$, and hence circumvents the question of estimator bias. We observe an interesting connection between MAML and the *stochastic composition* problem. Let us define $u_{\mathbf{h}}(\theta, T) = u(\theta, T; \mathbf{h})$ and $f^T_{\mathbf{g}}(\theta) = f^T(\theta + \sigma \mathbf{g})$. For a given task T, the MAML reward is given by

$$\widetilde{f}^T(U(\theta, T)) = \widetilde{f}^T[\mathbb{E}_{\mathbf{h}} u_{\mathbf{h}}(\theta, T)] = \mathbb{E}_{\mathbf{g}} f^T_{\mathbf{g}}(\mathbb{E}_{\mathbf{h}} u_{\mathbf{h}}(\theta, T)). \tag{8}$$

This is a two-layer nested stochastic composition problem with outer function $\widetilde{f}^T = \mathbb{E}_{\mathbf{g}} f^T_{\mathbf{g}}$ and inner function $U(\cdot, T) = \mathbb{E}_{\mathbf{h}} u_{\mathbf{h}}(\cdot, T)$. An accelerated algorithm (ASC-PG) was developed in [20]] for this class of problems. While neither $f^T_{\mathbf{g}}$ nor $u_{\mathbf{h}}(\cdot, T)$ is smooth, which is assumed in [20], we can verify that the crucial content of the assumptions hold:

1. $\mathbb{E}_{\mathbf{h}} u_{\mathbf{h}}(\theta, T) = U(\theta, T)$
2. We can define two functions
$$\zeta^T_{\mathbf{g}}(\theta) = \frac{1}{\sigma} f^T_{\mathbf{g}}(\theta)\mathbf{g}, \quad \xi^T_{\mathbf{h}}(\theta) = \mathbf{I} + \frac{\alpha}{\sigma^2}(f^T_{\mathbf{h}}(\theta)\mathbf{h}\mathbf{h}^T - f^T_{\mathbf{h}}(\theta)\mathbf{I})$$
such that for any θ_1, θ_2,
$$\mathbb{E}_{\mathbf{g},\mathbf{h}}[\xi^T_{\mathbf{h}}(\theta_1)\zeta^T_{\mathbf{g}}(\theta_2)] = JU(\theta_1, T)\nabla \widetilde{f}^T(\theta_2)$$
where JU denotes the Jacobian of $U(\cdot, T)$, and $\mathbf{g}, \mathbf{h}$ are independent vectors sampled from $\mathcal{N}(0, \mathbf{I})$. This follows immediately from equation 1 and equation 5.

The ASC-PG algorithm does not immediately extend to the full MAML problem, as upon taking an outer expectation over T, the MAML reward $J(\theta) = \mathbb{E}_T \mathbb{E}_{\mathbf{g}} f^T_{\mathbf{g}}(\mathbb{E}_{\mathbf{h}} u_{\mathbf{h}}(\theta, T))$ is no longer a stochastic composition of the required form. In particular, there are conceptual difficulties when the number of tasks in $\mathcal{T}$ is infinite. However, it can be used to solve the MAML problem for each task within a consensus framework, such as consensus ADMM [21].

A.3 Extensions of ES

In this section, we discuss several general techniques for improving the basic ES gradient estimator (Algorithm 1). These can be applied both to the ES gradient of the meta-training (the 'outer loop' of Algorithm 3), and more interestingly, to the adaptation operator itself. That is, given $U(\theta, T) = \theta + \alpha \nabla \widetilde{f}^T_\sigma(\theta)$, we replace the estimation of U by ESGRAD on line 4 of Algorithm 3 with an improved estimator of $\nabla \widetilde{f}^T_\sigma(\theta)$, which even may depend on data collected during the meta-training stage. Many techniques exist for reducing the variance of the estimator such as Quasi Monte Carlo sampling [22]. Aside from variance reduction, there are also methods with special properties.

A.3.1 Active subspaces

Active Subspaces is a method for finding a low-dimensional subspace where the contribution of the gradient is maximized. Conceptually, the goal is to find and update on-the-fly a low-rank subspace $\mathcal{L}$ so that the projection $\nabla f^T(\theta)_{\mathcal{L}}$ of $\nabla f^T(\theta)$ into $\mathcal{L}$ is maximized and apply $\nabla f^T(\theta)_{\mathcal{L}}$ instead of $\nabla f^T(\theta)$. This should be done in such a way that $\nabla f^T(\theta)$ does not need to be computed explicitly. Optimizing in lower-dimensional subspaces might be computationally more efficient and can be thought of as an example of guided ES methods, where the algorithm is guided how to explore space in the anisotropic way, leveraging its knowledge about function optimization landscape that it gained in the previous steps of optimization. In the context of RL, the active subspace method ASEBO [23] was successfully applied to speed up policy training algorithms. This strategy can be made data-dependent also in the MAML context, by learning an optimal subspace using data from the meta-training stage, and sampling from that subspace in the adaptation step.

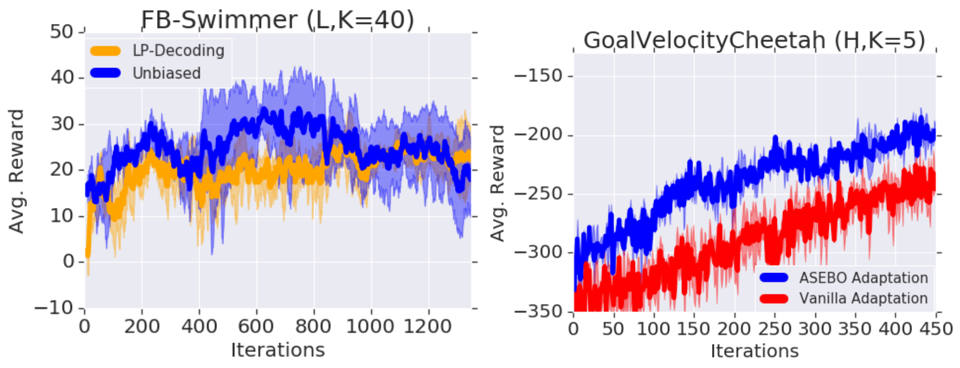
A.3.2 Regression-Based Optimization

Regression-Based Optimization (RBO) is an alternative method of gradient estimation. From Taylor series expansion we have $f(\theta + \mathbf{d}) - f(\theta) = \nabla f(\theta)^T \mathbf{d} + O(\|\mathbf{d}\|^2)$. By taking multiple finite difference expressions $f(\theta + \mathbf{d}) - f(\theta)$ for different $\mathbf{d}$, we can recover the gradient by solving a regularized regression problem. The regularization has an additional advantage - it was shown that the gradient can be recovered even if a substantial fraction of the rewards $f(\theta + \mathbf{d})$ are corrupted [24]. Strictly speaking, this is not based on the Gaussian smoothing as in ES, but is another method for estimating gradients using only zero-th order evaluations.

A.3.3 Experiments

We present a preliminary experiment with RBO and ASEBO gradient adaptation in Figure A3. To be precise, the algorithms used are identical to Algorithm 3 except that in line 4, $\mathbf{d}^{(i)} \leftarrow \text{ESGRAD}$ is replaced by $\mathbf{d}^{(i)} \leftarrow \text{RBO}$ (yielding RBO-MAML) and $\mathbf{d}^{(i)} \leftarrow \text{ASEBO}$ (yielding ASEBO-MAML) respectively.

Figure A3: RBO-MAML and ASEBO-MAML compared to ES-MAML.



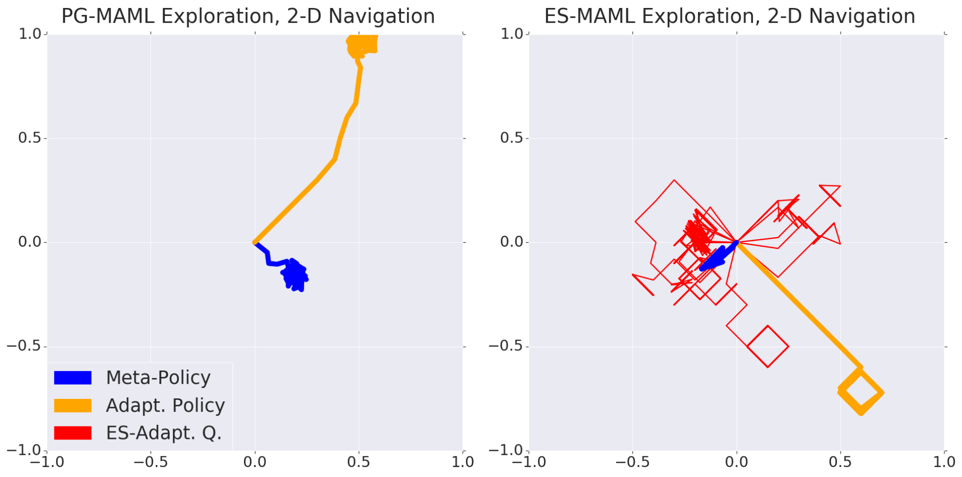
On the left plot, we test for noise robustness on the Forward-Backward Swimmer MAML task, comparing standard ES-MAML (Algorithm 3) to RBO-MAML. To simulate noisy data, we randomly corrupt 25% of the queries $f^T(\theta + \sigma g)$ used to estimate the adaptation operator $U(\theta, T)$ with an enormous additive noise. This is the same type of corruption used in [24]. Interestingly, RBO does not appear to be more robust against noise than the standard MC estimator, which suggests that the original ES-MAML has some inherent robustness to noise.

On the right plot, we compare ASEBO-MAML to ES-MAML on the Goal-Velocity HalfCheetah task in the low-K setting. We found that when measured in iterations, ASEBO-MAML outperforms ES-MAML. However, ASEBO requires additional linear algebra operations and thus uses significantly more wall-clock time (not shown in plot) per iteration, so if measured by real time, then ES-MAML was more effective.

A.4 Navigation-2D Exploration Task

Navigation-2D [1] is a classic environment where the agent must explore to adapt to the task. The agent is represented by a point on a 2D square, and at each time step, receives reward equal to its distance from a given target point on the square. Note that unlike the four corners and six circles tasks, the reward for Navigation-2D is dense. We visualize the differing exploration strategies learned by PG-MAML and ES-MAML in Figure A4. Notice that PG-MAML makes many tiny movements in multiple directions to 'triangulate' the target location using the differences in reward for different state-action pairs. On the other hand, ES-MAML learns a meta-policy such that each perturbation of the meta-policy causes the agent to move in a different direction (represented by red paths), so it can determine the target location from the total rewards of each path.

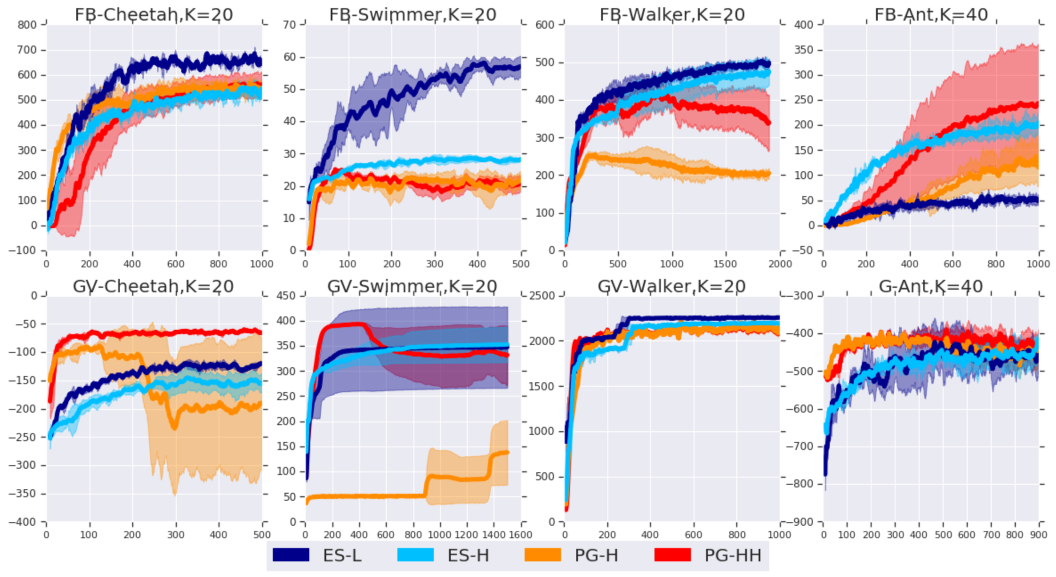
Figure A4: Comparing the exploration behavior of PG-MAML and ES-MAML on the Navigation-2D task. We use $K = 20$ queries for each algorithm.



A.5 PG-MAML RL Benchmarks

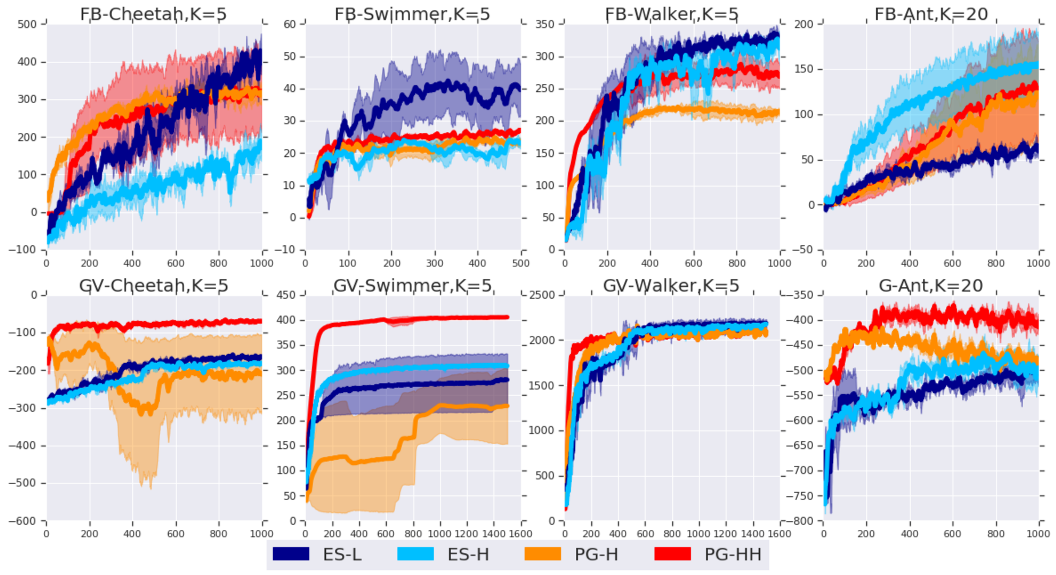
In Figure A5, we compare ES-MAML and PG-MAML on the Forward-Backward and Goal-Velocity tasks for HalfCheetah, Swimmer, Walker2d, and Ant, using the same values of K that were used in the original experiments of [1].

Figure A5: Comparisons between ES-MAML and PG-MAML using the queries K from [1].



A.5.1 Low-K Benchmarks

Figure A6: Low K comparisons between ES-MAML and PG-MAML.



For real-world applications, we may be constrained to use fewer queries K than has typically been demonstrated in previous MAML works. Hence, it is of interest to compare how ES-MAML compares to PG-MAML for adapting with very low K.

One possible concern is that low K might harm ES in particular because it uses only the cumulative rewards; if for example $K = 5$, then the ES adaptation gradient can make use of only 5 values. In comparison, PG-MAML uses $K \cdot H$ state-action pairs, so for $K = 5, H = 200$, PG-MAML still has 1000 pieces of information available.

However, we find experimentally that the standard ES-MAML (Algorithm 3) remains competitive with PG-MAML even in the low-K setting. In Figure A6, we compare ES-MAML and PG-MAML on the Forward-Backward and Goal-Velocity tasks across four environments (HalfCheetah, Swimmer, Walker2d, Ant) and two model architectures. While PG-MAML can generally outperform ES-MAML on the Goal-Velocity task, ES-MAML is similar or better on the Forward-Backward task. Moreover, we observed that for low K, PG-MAML can be highly unstable (note the wide error bars), with some trajectories failing catastrophically, whereas ES-MAML is relatively stable. This is an important consideration in real applications, where the risk of catastrophic failure is undesirable.

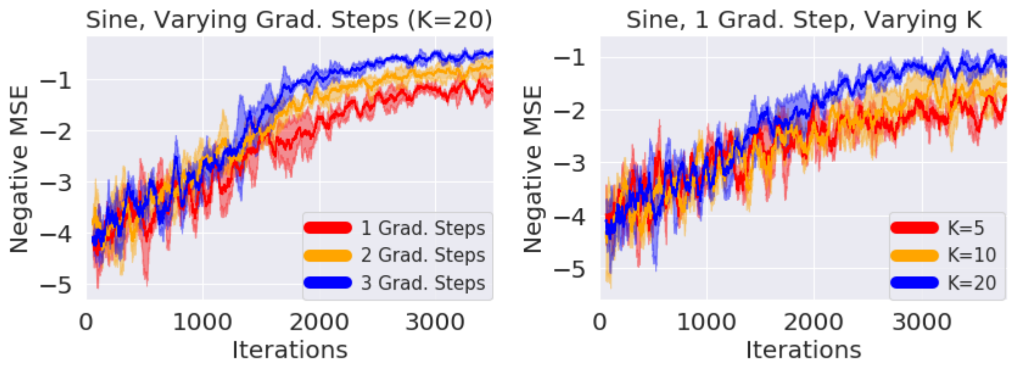
A.6 Regression and Supervised Learning

MAML has also been applied to supervised learning. We demonstrate ES-MAML on *sine regression* [1], where the task is to fit a sine curve f with unknown amplitude and phase given a set of K pairs $(x_i, f(x_i))$. The meta-policy must be able to learn that all of tasks have a common periodic nature, so that it can correctly adapt to an unknown sine curve outside of the points x_i.

For regression, the loss is the mean-squared error (MSE) between the adapted policy $\pi_\theta(x)$ and the true curve $f(x)$. Given data samples $\{(x_i, f(x_i)\}_{i=1}^{K}$, the empirical loss is $L(\theta) = \frac{1}{K}\sum_{i=1}^{K}(f(x_i) - \pi_\theta(x_i))^2$. Note that unlike in reinforcement learning, we *can* exactly compute $\nabla L(\theta)$; for deep networks, this is by automatic differentiation. Thus, we opt to use Tensorflow to compute the adaptation operator $U(\theta, T)$ in Algorithm 3. This is in accordance with the general principle that when gradients are available, it is more efficient to use the gradient than to approximate it by a zero-order method [25].

We show several results in Figure A7. The adaptation step size is $\alpha = 0.01$, which is the same as in [1]. For comparison, [1] reports that PG-MAML can obtain a loss of ≈ 0.5 after one adaptation step with $K = 5$, though it is not specified how many iterations the meta-policy was trained for. ES-MAML approaches the same level of performance, though the number of training iterations required is higher than for the RL tasks, and surprisingly high for what appears to be a simpler problem. This is likely again a reflection of the fact that for problems such as regression where the gradients are available, it is more efficient to use gradients.

Figure A7: The MSE of the adapted policy, for varying number of gradient steps and query number K. Runs are averaged across 3 seeds.



As an aside, this leads to a related question of the correct interpretation of the query number K in the supervised setting. There is a distinction between obtaining a data sample $(x_i, f(x_i))$, and doing a computation (such as a gradient) using that sample. If the main bottleneck is collecting the data $\{(x_i, f(x_i)\}$, then we may be satisfied with any algorithm that performs any number of operations on the data, as long as it uses only K samples. On the other hand, in the (on-policy) RL setting, samples cannot typically be 're-used' to the same extent, because rollouts τ sampled with a given policy π_θ follow an unknown distribution $\mathcal{P}(\tau|\theta)$ which reduces their usefulness away from θ. Thus, the corresponding notion to rollouts in the SL setting would be the number of backpropagations (for PG-MAML) or perturbations (for ES-MAML), but clearly these have different relative costs than doing simulations in RL.

A.7 Hyperparameters and Setups

A.7.1 Environments

Unless otherwise explicitly stated, we default to $K = 20$ and horizon = 200 for all RL experiments. We also use the standard reward normalization in [4], and use a global state normalization (i.e. the same mean, standard deviation normalization values for MDP states are shared across workers).

For the Ant environments (Goal-Position Ant, Forward-Backward Ant), there are significant differences in weighting on the auxiliary rewards such as control costs, contact costs, and survival rewards across different previous work (e.g. those costs are downweighted in [1] whereas the coefficients are vanilla Gym weightings in [6]). These auxiliary rewards can lead to local minima, such as the agent staying stationary to collect the survival bonus which may be confused with movement progress when presenting a training curve. To make sure the agent is explicitly performing the required task, we opted to remove such costs in our work and only present the main goal-distance cost and forward-movement reward respectively.

For the other environments, we used default weightings and rewards, since they do not change across previous works.

A.7.2 ES-MAML Hyperparameters

Let N be the number of possible distinct tasks possible. We sample tasks without replacement, which is important if $N \ll 5$, as each worker performs adaptations on all possible tasks.

For standard ES-MAML (Algorithm 3), we used the following settings.

Setting	Value
(Total Workers, # Perturbations, # Current Evals)	(300, 150, 150)
(Train Set Size, Task Batch Size, Test Set Size)	(50,5,5) or (N,N,N)
Number of rollouts per parameter	1
Number of Perturbations per worker	1
Outer-Loop Precision Parameter	0.1
Adaptation Precision Parameter	0.1
Outer-Loop Step Size	0.01
Adaptation Step Size (α)	0.05
Hidden Layer Width	32
ES Estimation Type	Forward-FD
Reward Normalization	True
State Normalization	True

For ES-MAML and PG-MAML, we took 3 seeded runs, using the default TRPO hyperparameters found in [6].