Guided Meta-Policy Search

Russell Mendonca [1] **Abhishek Gupta** [1] **Rosen Kralev** [1] **Pieter Abbeel** [1] **Sergey Levine** [1] **Chelsea Finn** [1]

Abstract

Reinforcement learning (RL) algorithms have demonstrated promising results on complex tasks, yet often require impractical numbers of samples because they learn from scratch. Meta-RL aims to address this challenge by leveraging experience from previous tasks in order to more quickly solve new tasks. However, in practice, these algorithms generally also require large amounts of on-policy experience during the *meta-training* process, making them impractical for use in many problems. To this end, we propose to learn a reinforcement learning procedure through imitation of expert policies that solve previously-seen tasks. This involves a nested optimization, with RL in the inner loop and supervised imitation learning in the outer loop. Because the outer loop imitation learning can be done with off-policy data, we can achieve significant gains in meta-learning sample efficiency. In this paper, we show how this general idea can be used both for meta-reinforcement learning and for learning fast RL procedures from multi-task demonstration data. The former results in an approach that can leverage policies learned for previous tasks without significant amounts of on-policy data during meta-training, whereas the latter is particularly useful in cases where demonstrations are easy for a person to provide. Across a number of continuous control meta-RL problems, we demonstrate significant improvements in meta-RL sample efficiency in comparison to prior work as well as the ability to scale to domains with visual observations.

1. Introduction

Meta-learning is a promising approach for using previous experience across a breadth of tasks to significantly accelerate learning of new tasks. Meta-reinforcement learning considers this problem specifically in the context of learning new behaviors through trial and error with only a few interactions with the environment by building on previous experience. Building effective meta-RL algorithms is critical towards building agents that are *flexible*, such as an agent being able to manipulate new objects in new ways without learning from scratch for each new object and goal. Being able to reuse prior experience in such a way is arguably a fundamental aspect of intelligence.

Enabling agents to adapt via meta-RL is particularly useful for acquiring behaviors in real-world situations with diverse and dynamic environments. However, despite recent advances (Duan et al., 2016; Finn et al., 2017a; Houthooft et al., 2018), current meta-RL methods are generally limited to much simpler domains, such as relatively low-dimensional continuous control tasks (Finn et al., 2017a; Sung et al., 2017) and navigation with discrete action commands (Duan et al., 2016; Mishra et al., 2018). Optimization stability and sample complexity are major challenges for the meta-training phase of these methods, with some recent techniques requiring 250 million transitions for meta-learning in tabular MDPs, which typically require a fraction of a second to solve in isolation (Duan et al., 2016).

We make the following observation in this work: while the goal of meta-reinforcement learning is to acquire fast and efficient reinforcement learning procedures, those procedures themselves do not need to be acquired through reinforcement learning directly. Instead, we can use a significantly more stable and efficient algorithm for providing supervision at the meta-level. In this work we show that a practical choice is to use supervised imitation learning. A meta-reinforcement learning algorithm can receive more direct supervision during *meta-training*, in the form of expert actions, while still optimizing for the ability to quickly learn tasks via reinforcement. Crucially, these expert policies can themselves be produced automatically by standard reinforcement learning methods, such that no additional assumptions on supervision are actually needed. They can also be acquired using very efficient off-policy reinforcement learning algorithms which are otherwise challenging to use with meta-reinforcement learning. When available, incorporating human-provided demonstrations can enable even more efficient meta-training, particularly in domains where demonstrations are easy to collect. At meta-test time, when faced with a new task, the method solves the same problem

[1]EECS Department, UC Berkeley, Berkeley, CA, USA. Correspondence to: Russell Mendonca <russellm@berkeley.edu>.

as conventional meta-reinforcement learning: acquiring the new skill using only reward signals.

Our main contribution is a meta-RL method that learns fast reinforcement learning via supervised imitation. As illustrated in Figure 1, we optimize for a set of parameters such that only one or a few gradient steps leads to a policy that matches the expert's actions. Since supervised imitation is stable and efficient, our approach can gracefully scale to visual control domains and high-dimensional convolutional networks. By using demonstrations during meta-training, there is less of a challenge with exploration in the meta-optimization, making it possible to effectively learn how to learn in sparse reward environments. While the combination of imitation and RL has been explored before (Peters & Schaal, 2006; Kober & Peters, 2009), the particular combination of imitation and RL in a meta-learning context has not been studied in prior work. As we show in our experiments, this combination is in fact extremely powerful: compared to meta-reinforcement learning, our method can meta-learn comparable adaptation skills with up to 10x fewer interaction episodes, making meta-RL much more viable for real-world learning. Our experiments also indicate that our method can be used to acquire reinforcement learning procedures that can learn from sparse rewards.

2. Related Work

Our work builds upon prior work on meta-learning (Schmidhuber, 1987; Bengio et al.; Thrun & Pratt, 2012), where the goal is to learn how to learn efficiently. We focus on the particular case of learning an efficient reinforcement learner, i.e., the meta-reinforcement learning setting (Schmidhuber, 1987; Duan et al., 2016; Wang et al., 2016; Finn et al., 2017a; Mishra et al., 2018; Gupta et al., 2018). Prior works have sought to solve this problem by optimizing for an efficient reinforcement learner through policy gradient and evolutionary optimization algorithms. These methods have represented the learner using a recurrent or recursive neural network (Duan et al., 2016; Wang et al., 2016; Mishra et al., 2018; Stadie et al., 2018), using gradient descent from a learned initialization (Finn et al., 2017a; Gupta et al., 2018; Rothfuss et al., 2018), using a learned critic that provides gradients to the policy (Sung et al., 2017; Houthooft et al., 2018), or using a planner and an adaptable model (Clavera et al., 2018; Sæmundsson et al., 2018). In contrast, our approach aims to leverage supervised learning for meta-optimization rather than relying on high-variance algorithms such as policy gradient or evolutionary strategies. We decouple the problem of obtaining expert trajectories for every task from the problem of learning a fast reinforcement learning algorithm. This allows us to obtain expert trajectories for every task using standard, efficient, and stable RL algorithms, and to utilize example demonstrations if available.

Our approach is also related to few-shot imitation learning (Duan et al., 2017; Finn et al., 2017b), in that we can leverage supervised learning for meta-optimization. However, in contrast to these methods, our approach leans a completely automatic meta-reinforcement learner, which can learn using only reward signals and does not require demonstrations for new tasks.

Meta-learning is closely related to multi-task learning (Caruana, 1998), where the goal is to master a fixed set of pre-defined goals or tasks (whereas meta-learning seeks to use experience from multiple tasks to quickly master new tasks). In this respect, our approach is related to multi-task RL methods that seek to distill policies for multiple tasks into a single policy, akin to guided policy search (Levine et al., 2016) and related approaches (Rusu et al., 2016; Parisotto et al., 2016; Teh et al., 2017; Omidshafiei et al., 2017; Ghosh et al., 2018). Like these prior works, we train a separate expert to provide trajectories for each condition or task, but unlike these approaches, we use these experts to train a reinforcement learner, rather than a single policy. This enables the policy to quickly *adapt* its behavior to new tasks rather than having to rely on contextualization or robustness to perform well on new tasks.

Prior methods have also sought to use demonstrations to make standard reinforcement learning more efficient in the single-task setting (Peters & Schaal, 2006; Kober & Peters, 2009; Kormushev et al., 2010; Taylor et al., 2011; Brys et al., 2015; Subramanian et al., 2016; Hester et al., 2018; Sun et al., 2018; Rajeswaran et al., 2018; Nair et al., 2018; Kober et al., 2013; Silver et al., 2016). These methods aim to learn a policy from demonstrations and rewards, using demonstrations to make the RL problem easier. Our approach instead aims to leverage demonstrations to learn how to efficiently reinforcement learn new tasks without demonstrations, learning new tasks only through trial-and-error. The version of our algorithm where data is aggregated across iterations, is an extension of the DAgger algorithm (Ross et al., 2011) into the meta-learning setting, and this allows us to provide theoretical guarantees on algorithm performance.

3. Preliminaries

In this section, we introduce the meta-reinforcement learning problem and overview model-agnostic meta-learning (MAML) (Finn et al., 2017a), which we build on in our work.

We assume a distribution of tasks $\mathcal{T} \sim p(\mathcal{T})$, where meta-training tasks are drawn from p and meta-testing consists of learning held-out tasks sampled from p through trial-and-error, by leveraging what was learned during meta-training. Formally, each task $\mathcal{T} = \{r(\mathbf{s}_t, \mathbf{a}_t), q(\mathbf{s}_1), q(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\}$ consists of a reward function $r(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbb{R}$, an initial state

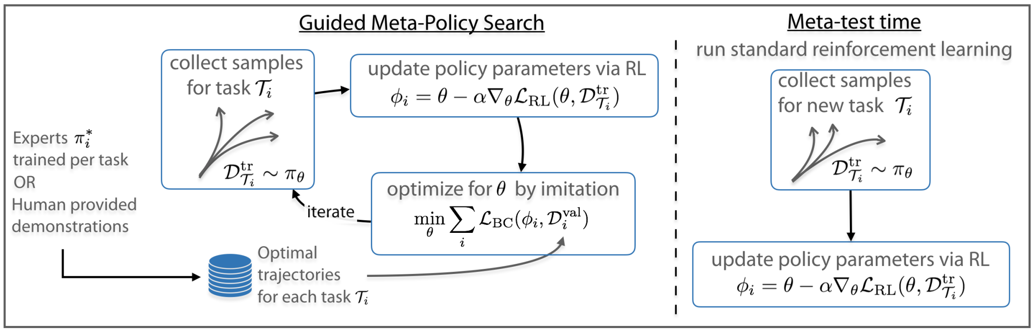

Figure 1. Overview of the guided meta-policy search algorithm: We learn a policy π_θ which is capable of fast adaptation to new tasks via reinforcement learning, by using reinforcement learning in the inner loop of optimization and supervised learning in the meta-optimization. This algorithm either trains per-task experts π_i^* or assumes that they are provided by human demonstrations, and then uses this for meta-optimization. Importantly, when faced with a new task we can simply perform standard reinforcement learning via policy gradient, and the policy will quickly adapt to new tasks because of the meta-training.

distribution $q(\mathbf{s}_1)$, and unknown dynamics $q(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The state space, action space, and horizon H are shared across tasks. Meta-learning methods learn to learn using experience from the meta-training tasks, and are evaluated on their ability to learn new meta-test tasks. MAML in particular performs meta-learning by optimizing for a deep network's initial parameter setting such that one or a few steps of gradient descent on a few datapoints leads to effective generalization. Then, after meta-training, the learned parameters are fine-tuned on data from a new task.

Concretely, consider a supervised learning problem with a loss function denoted as $\mathcal{L}(\theta, \mathcal{D})$, where θ denotes the model parameters and $\mathcal{D}$ denotes the labeled data. During meta-training, a task $\mathcal{T}$ is sampled, along with data from that task, which is randomly partitioned into two sets, $\mathcal{D}^{\text{tr}}$ and $\mathcal{D}^{\text{val}}$. MAML optimizes for a set of model parameters θ such that one or a few gradient steps on $\mathcal{D}^{\text{tr}}$ produces good performance on $\mathcal{D}^{\text{val}}$. Thus, using $\phi_\mathcal{T}$ to denote the updated parameters, the MAML objective is the following:

$$\min_\theta \sum_\mathcal{T} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}_\mathcal{T}), \mathcal{D}^{\text{val}}_\mathcal{T}) = \min_\theta \sum_\mathcal{T} \mathcal{L}(\phi_\mathcal{T}, \mathcal{D}^{\text{val}}_\mathcal{T}).$$

where α is a step size that can be set as a hyperparameter or learned. Moving forward, we will refer to the outer objective as the *meta-objective*. Subsequently, at meta-test time, K examples from a new, held-out task $\mathcal{T}_{\text{test}}$ are presented and we can run gradient descent starting from θ to infer model parameters for the new task:

$$\phi_{\mathcal{T}_{\text{test}}} = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}_{\mathcal{T}_{\text{test}}}).$$

The MAML algorithm can also be applied to the meta-reinforcement learning setting, where each dataset $\mathcal{D}_{\mathcal{T}_i}$ consists of trajectories of the form $\mathbf{s}_1, \mathbf{a}_1, ..., \mathbf{a}_{H-1}, \mathbf{s}_H$ and where the inner and outer loss function corresponds to the negative expected reward:

$$\mathcal{L}_{\text{RL}}(\phi, \mathcal{D}_{\mathcal{T}_i}) = -\frac{1}{|\mathcal{D}_{\mathcal{T}_i}|} \sum_{\mathbf{s}_t, \mathbf{a}_t \in \mathcal{D}_{\mathcal{T}_i}} r_i(\mathbf{s}_t, \mathbf{a}_t) \tag{1}$$

$$= -\mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\phi, q_{\mathcal{T}_i}} \left[\frac{1}{H} \sum_{t=1}^{H} r_i(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{2}$$

Policy gradients (Williams, 1992) were used to estimate the gradient of this loss function. Thus, the algorithm proceeded as follows: for each task $\mathcal{T}_i$, first collect samples $\mathcal{D}^{\text{tr}}_{\mathcal{T}_i}$ from the policy π_θ, then compute the updated parameters using the policy gradient evaluated on $\mathcal{D}^{\text{tr}}_{\mathcal{T}_i}$, then collect new samples $\mathcal{D}^{\text{val}}_{\mathcal{T}_i}$ via the updated policy parameters, and finally update the initial parameters θ by taking a gradient step on the meta-objective. In the next section, we will introduce a new approach to meta-reinforcement learning using ideas from the MAML algorithm, but introducing a more stable optimization for the meta-objective.

4. Guided Meta-Policy Search

Existing meta-RL algorithms generally perform meta-learning from scratch with on-policy methods. This typically requires a large number of samples during meta-training. What if we instead formulate meta-training as a data-driven process, where the agent had previously learned a variety tasks with standard multi-task reinforcement learning techniques, and now must use the data collected from those tasks for meta-training? Can we use this experience or these policies in meaningful way during meta-training? Our goal is to develop an approach that can use these previously learned skills to guide the meta-learning process. While we will still require on-policy data, we will require considerably less of it than what we would need without using this prior experience. Surprisingly, as we will show in our experiments, separating meta-training into two phases in this way – a phase that individually solves the meta-training tasks and a second phase that uses them for meta-learning – actually requires less total experience overall, as the individual tasks can be solved using highly-efficient off-policy reinforcement learning methods that actually require less experience taken together than a single meta-RL training phase. We can also improve sample efficiency during meta-training even further by incorporating explicit demonstrations.

In this section, we describe our approach, analyze its theoretical properties, and discuss its practical implementation in multiple real world scenarios.

4.1. Guided Meta-Policy Search Algorithm

In the first phase of the algorithm, task learning, we learn policies for each of the meta-training tasks. While these policies solve the meta-training tasks, they do not accelerate learning of future meta-test tasks. In Section 4.3, we describe how these policies are trained. Instead of learning policies explicitly through reinforcement learning, we can also obtain expert demonstrations from a human demonstrator, which can be used equivalently with the same algorithm. In the second phase, meta-learning, we will learn to reinforcement learn using these policies as supervision at the meta-level. In particular, we train for a set of initial parameters θ such that only one or a few steps of gradient descent produces a policy that matches the policies learned in the first phase.

We will denote the optimal or near-optimal policies learned during the task-learning phase for each meta-training task $\mathcal{T}_i$ as $\{\pi_i^*\}$. We will refer to these individual policies as "experts," because after the first phase, they represent optimal or near-optimal solutions to each of the tasks. Our goal in the meta-learning phase is to optimize the same meta-objective as the MAML algorithm, $\mathcal{L}_{\text{RL}}(\phi_i, \mathcal{D}_i)$, where ϕ_i denotes the parameters of the policy adapted to task $\mathcal{T}_i$ via gradient descent. The inner policy optimization will remain the same as the policy-gradient MAML algorithm; however, we will optimize this meta-objective by leveraging the policies learned in the first phase. In particular, we will base the outer objective on supervised imitation, or behavior cloning (BC), of expert actions. The behavioral cloning loss function is:

$$\mathcal{L}_{\text{BC}}(\phi_i, \mathcal{D}_i) \triangleq -\sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}} \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t). \tag{3}$$

Gradients from supervised learning are lower variance, and hence more stable than reinforcement learning gradients (Norouzi et al., 2016). In Section 4.2, we will show that the proposed algorithm converges to the correct solution of the original RL meta-objective under some regularity assumptions.

The specific implementation of the second phase proceeds as follows: we first roll out each of the policies π_i^* to collect

Algorithm 1 Guided Meta-Policy Search

Require: Set of meta-training tasks $\{\mathcal{T}_i\}$
1: Use RL to acquire π_i^* for each meta-training task $\mathcal{T}_i$
2: Initialize $\mathcal{D}^* = \{\mathcal{D}_i^*\}$ with roll-outs from each π_i^*.
3: Randomly initialize θ
4: **while** not done **do**
5: Optimize meta-objective in Equation 4 w.r.t. θ using Algorithm 2 with aggregated data $\mathcal{D}^*$
6: **for** each meta-training task $\mathcal{T}_i$ **do**
7: Collect $\mathcal{D}_i^{\text{tr}}$ as K roll-outs from π_θ in task $\mathcal{T}_i$
8: Compute task-adapted parameters with gradient descent: $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{RL}}(\theta, \mathcal{D}_i^{\text{tr}})$
9: Collect roll-outs from π_{ϕ_i}, resulting in data $\{(\mathbf{s}_t, \mathbf{a}_t)\}$
10: Aggregate $\mathcal{D}_i^* \leftarrow \mathcal{D}_i^* \bigcup \{(\mathbf{s}_t, \pi_i^*(\mathbf{s}_t))\}$
11: **end for**
12: **end while**

a dataset of expert trajectories $\mathcal{D}_i^*$ for each of the meta-training tasks $\mathcal{T}_i$. Using this initial dataset, we update our policy according to the following meta-objective:

$$\min_\theta \sum_{\mathcal{T}_i} \sum_{\mathcal{D}_i^{\text{val}} \sim \mathcal{D}_i^*} \mathbb{E}_{\mathcal{D}_i^{\text{tr}} \sim \pi_\theta} \left[\mathcal{L}_{\text{BC}}(\theta - \alpha \nabla_\theta \mathcal{L}_{\text{RL}}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{val}}) \right]. \tag{4}$$

We discuss how this objective can efficiently be optimized in Section 4.3. The result of this optimization is a set of initial policy parameters θ that can adapt to a variety of tasks, to produce ϕ_i, in a way that comes close to the expert policy's actions. Note that, so far, we have not actually required querying the expert beyond access to the initial rollouts; hence, this first step of our method is applicable to problem domains where demonstrations are available in place of learned expert policies. However, in the case where we do have policies for the meta-training tasks, we can continue to improve. In particular, while supervised learning provides stable, low-variance gradients, behavior cloning objectives are known to be prone to compounding errors. In the single task imitation learning setting, this issue can be addressed by collecting additional data from the learned policy, and then labeling the visited states with optimal actions from the expert policy, as in the DAgger algorithm (Ross et al., 2011). We extend this idea to the meta-learning setting by alternating between data aggregation into a dataset $\mathcal{D}^*$ and the meta-policy optimization in Equation 4. Data aggregation entails (1) adapting the current policy parameters θ to each of the meta-training tasks to produce $\{\phi_i\}$, (2) rolling out the current adapted policies $\{\pi_{\phi_i}\}$ to produce states $\{\{\mathbf{s}_t\}_i\}$ for each task, (3) querying the experts to produce supervised data $\mathcal{D} = \{\{(\mathbf{s}_t, \pi_i^*(\mathbf{s}_t)\}_i\}$, and finally (4) aggregating this data with the existing supervised data $\mathcal{D}^* \leftarrow \mathcal{D}^* \bigcup \mathcal{D}$. This meta-training algorithm is summarized in Algorithm 1, and analyzed in Section 4.2.

The result of meta-training is initial policy parameters θ that can be adapted to new tasks. When provided with new tasks at meta-test time, we initialize with parameters θ and run the policy gradient algorithm, in exactly the same way as standard MAML (Finn et al., 2017a).

Our algorithm, which we call guided meta-policy search (GMPS), has appealing properties that arise from decomposing the meta-learning problem explicitly into the task learning phase and the meta-learning phase. This decomposition enables the use of previously learned policies or human-provided demonstrations. We find that it also leads to increased stability of training. Lastly, the decomposition makes it easy to leverage privileged information that may only be available during meta-training such as shaped rewards, task information, low-level state information such as the positions of objects. In particular, this privileged information can be provided to the initial policies as they are being learned and hidden from the meta-policy such that the meta-policy can be applied in test settings where such information is not available. This technique can make it quite straight-forward to learn vision-based policies, for example, as the bulk of learning can be done without vision, while visual features are learned with supervised learning in the second phase. Our method also inherits appealing properties from policy gradient MAML, such as the ability to continue to learn as more and more experience is collected, in contrast to recurrent neural networks that cannot be easily fine-tuned on new tasks.

In the next section, we will also show that, although our proposed method optimizes a behavior cloning loss in the outer loop, it can still be shown to maximize the task reward for the post-update policy.

4.2. Convergence Analysis

We can prove that GMPS with data aggregation, described above, obtains near-optimal cumulative reward when supplied with near-optimal experts. Our proof follows a similar approach to prior work that analyzes the convergence of imitation algorithms with aggregation (Ross et al., 2011; Kahn et al., 2016), but extends these results into the meta-learning setting. More specifically, we can prove the following theorem, given a task distribution $p(\mathcal{T})$ and horizon H.

Theorem 4.1 *For* GMPS*, assuming reward-to-go bounded by δ, and training error bounded by $\epsilon_{\theta*}$, we can show that* $\mathbb{E}_{i\sim p(\mathcal{T})}[\mathbb{E}_{\pi_{\theta+\nabla_\theta \mathbb{E}_{\pi_\theta}[R_i]}}[\sum_{t=1}^{H} r_i(\mathbf{s}_t, \mathbf{a}_t)]] \geq \mathbb{E}_{i\sim p(\mathcal{T})}[\mathbb{E}_{\pi_i^*}[\sum_{t=1}^{H} r_i(\mathbf{s}_t, \mathbf{a}_t)]] - \delta\sqrt{\epsilon_{\theta*}}O(H)$*, where π_i^* are per-task expert policies.*

The proof of this theorem requires us to assume that the inner policy update in Equation 4 can bring the learned policy to within a bounded error of each expert, which amounts to an assumption on the *universality* of gradient-based meta-learning (Finn & Levine, 2018). The theorem amounts to saying that, GMPS can achieve an expected reward that is within a bounded error of the optimal reward (i.e., the reward of the individual experts), and the error is linear in H and $\sqrt{\epsilon_{\theta*}}$.

The analysis holds for GMPS when each iteration generates samples by adapting the current meta-trained policy to each training task. However, we find in practice that the off-policy version where data is simply drawn from per task experts π_i^* is quite stable, with lower sample complexity and we use this in our experimental evaluation. We have omitted the proof of this theorem from the main text for brevity. For a complete theoretical analysis, and the full proof of Theorem 4.1, please refer to Appendix A.

4.3. Algorithm Implementation

Now that we have provided theoretical motivation for our approach, we flesh out the algorithm described above in Section 4.1 and how it is implemented in practice.

4.3.1. Expert Policy Optimization

The first phase of our algorithm entails learning policies for each of the meta-training tasks. The simplest approach is to simply learn a separate policy for each task from scratch. This can already improve over standard meta-RL, since we can employ efficient off-policy reinforcement learning algorithms that are faster than current meta-RL methods, which are typically on-policy (Finn et al., 2017a; Duan et al., 2016). We can improve the efficiency of this approach by employing a *contextual* policy to represent the experts, which simultaneously uses data from all of the tasks. We can express such a policy as $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \omega)$, where ω represents the task context. This context can be any piece of information that uniquely identifies the task, such as a goal position or even the task index. Crucially, the context only needs to be known during *meta-training* – the end result of our algorithm, after the second phase, still uses raw task rewards without knowledge of the context at meta-test time. In our experiments, we employ this approach, together with soft-actor critic (SAC) (Haarnoja et al., 2018), an efficient off-policy RL method.

In settings where it is easy and intuitive for a human to provide demonstrations, we can also use demonstration rollouts and omit the latter data aggregation iterations, requiring on-policy data for only the inner loop. Finally, we can incorporate extra information during meta-training that is unavailable at meta-test time, such as knowledge of the state or better shaped rewards. The former has been explored before in the context of single-task RL (Levine et al., 2016; Pinto et al., 2017), while the latter has been studied for methods that learn exploration strategies for sparse rewards using dense rewards for meta-training (Gupta et al., 2018).

Algorithm 2 Optimization of Meta Objective

Require: Set of meta-training tasks $\{\mathcal{T}_i\}$
Require: Aggregated dataset $\mathcal{D}^*$, consisting of $\mathcal{D}_i^*$ for each task $\mathcal{T}_i$
Require: α, β: step size hyperparameters
Require: θ initial parameters

```
while not done do
    Sample task T_i ~ {T_i} {or minibatch of tasks}
    Sample K roll-outs D_i^tr = {(s_1, a_1, ...s_H)} with π_θ in T_i
    θ_init ← θ
    for n = 1...N_BC do
        Evaluate ∇_θ L_RL(θ, D_i^tr) according to Equation 5 with
        importance weights π_θ(a_t|s_t) / π_θinit(a_t|s_t)
        Compute adapted parameters with gradient descent:
        φ_i = θ − α∇_θ L_RL(θ, D_i^tr)
        Sample expertly-labeled trajectories D_i^val ~ D_i^*
        Update θ ← θ − β∇_θ L_BC(φ_i, D_i^val).
    end for
end while
```

4.3.2. Optimization Algorithm

In order to *efficiently* optimize the meta-objective in Equation 4, we adopt an approach similar to MAML. At each meta-iteration and for each task $\mathcal{T}_i$, we first draw samples $\mathcal{D}_{\mathcal{T}_i}^{\text{tr}}$ from the policy π_θ, then compute the updated policy parameters $\phi_{\mathcal{T}_i}$ using the $\mathcal{D}_{\mathcal{T}_i}^{\text{tr}}$, then we update θ to optimize $\mathcal{L}_{\text{BC}}$, averaging over all tasks in the minibatch. This requires sampling from π_θ, so for efficient learning, we should minimize the number of meta-iterations.

We can still take multiple gradient steps on the behavior cloning meta-objective in each meta-iteration, since this objective does not require on-policy samples. However, after the first gradient step on the meta-objective modifies the pre-update parameters θ, we need to recompute the adapted parameters ϕ_i starting from θ, and we would like to do so *without* collecting new data from π_θ. To achieve this, we use an importance-weighted policy gradient, with importance weights $\frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{\text{init}}}(\mathbf{a}_t|\mathbf{s}_t)}$, where θ_{init} denotes the policy parameters at the start of the meta-iteration (the parameters under which the data was collected).

At the start of a meta-iteration, we sample trajectories τ from the current policy with parameters denoted as $\theta = \theta_{\text{init}}$. Then, we take many off-policy gradient steps on θ. Each off-policy gradient step involves recomputing the updated parameters ϕ_i using importance sampling:

$$\phi_i = \theta + \alpha \mathbb{E}_{\tau\sim\pi_\theta}\left[\frac{\pi_\theta(\tau)}{\pi_{\theta_{\text{init}}}(\tau)} \nabla_\theta \log \pi_\theta(\tau) A_i(\tau)\right] \tag{5}$$

where A_i is the advantage function. Then, the off-policy gradient step is computed and applied using the updated parameters using the behavioral cloning objective in Equation 3:

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\text{BC}}(\phi_i, \mathcal{D}_i^{\text{val}}). \tag{6}$$

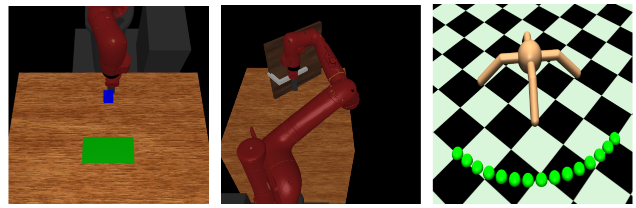

Figure 2. Illustration of a pushing task (left), door opening task (center) and a legged locomotion task (right) in our experimental evaluation. The goal location for the pushing task is sampled from the region indicated by the green rectangle. The target angle for the door opening task is sampled from 0 to 60 degrees, and the goal for the ant is sampled from a quadrant of a circle as shown by the green markers.

This optimization algorithm is summarized in Algorithm 2.

5. Experimental Evaluation

We evaluate GMPS separately as a meta-reinforcement algorithm, and for learning fast RL procedures from multi-task demonstration data. We consider the following questions:

As a meta-RL algorithm, (1) can GMPS meta-learn more efficiently than prior meta-RL methods?

For learning from demonstrations, (2) does using imitation learning in the outer loop of optimization enable us to overcome challenges in exploration, and learn from sparse rewards? (3) using GMPS, can we effectively meta-learn vision-based policies that can quickly adapt to new tasks?

5.1. Experimental Setup

To help us answer these questions, we evaluate GMPS in a number of simulated continuous control domains visualized in Figure 2.

Sawyer Manipulation Tasks. The tasks involving the 7-DoF sawyer arm are performed with 3D position control of a parallel jaw gripper (four DoF total, including open/close). The sawyer environments include:

- Pushing, full state: The tasks involve pushing a block with a fixed initial position to a target location sampled from a 20 cm $\times$ 10 cm region (This region is indicated by the green area in the illustration of the pushing task in Figure 2). The target location within this region is not observed and must be implicitly inferred through trial-and-error. The 'full state' observations include the 3D position of the end effector and of the block.
- Pushing, vision: Same as above, except the policy receives an RGB image instead of the block position.
- Door opening: The task distribution involves opening a door to a target angle sampled uniformly from 0 to 60 degrees. The target angle is not present in the observations, and must be implicitly inferred through trial-and-error.

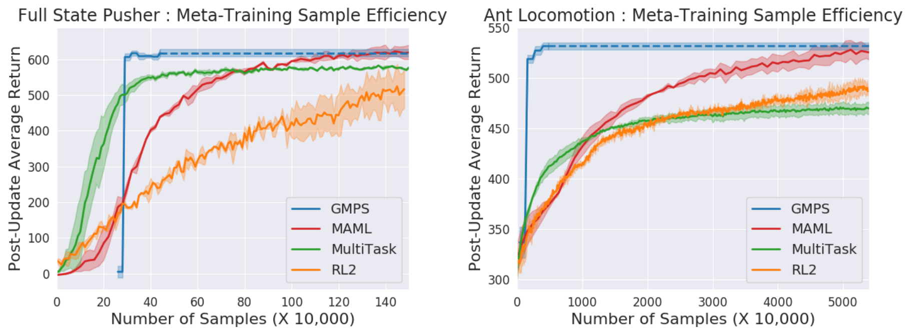

Figure 3. Meta-training sample efficiency comparison on full state pushing (left) and dense reward ant locomotion (right). All methods reach approximately the same asymptotic performance, but GMPS is able to achieve the performance with significant gains in sample efficiency.

The 'full state' observations include the 3D end effector position of the arm, the state of the gripper, and the current position and angle of the door.

Quadrupedal Legged Locomotion. This environment uses the quadruped (ant) environment introduced by Schulman et al. (2016) and implemented in OpenAI gym (Brockman et al., 2017). The task distribution comprises goal positions sampled uniformly from the edge of a circle with radius 2 m, between 0 and 90 degrees. We have a dense-reward version version of the environment to evaluate GMPS as a meta-RL agorithm, and a sparse-reward version where we evaluate GMPS with demonstrations.

Further details such as the reward functions for all environments, network architectures, and hyperparameters swept over are in the appendix. Videos of our results are available online[1].

5.2. Meta-Reinforcement Learning

We first evaluate the sample efficiency of GMPS as a meta-RL algorithm, measuring performance as a function of the total number of samples used during meta-training. We compare to the policy gradient version of model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017a), a state-of-the-art meta-learning algorithm that uses vanilla policy gradient in the inner loop and TRPO in the outer loop. We also compare to RL^2 (Duan et al., 2016), and to a single policy that is trained across all meta-training tasks (we refer to this comparision as MultiTask).

At meta-training time, we assume access to the task context (information that completely specifies the task, namely the target location for the pushing and locomotion experiments). We train a contextual policy conditioned on the target position with soft actor-critic (SAC) to obtain expert trajectories which are used by GMPS. The samples used to train this expert policy with SAC are included in our evaluation. At meta-test time, when adapting to new validation tasks, we *only* have access to the reward, which hence necessitates meta-learning without providing the task contexts to the policy. As a result, note that, in principle, the task context could also be represented as a one-hot vector that indicates the task identity, since the meta-RL algorithm does not rely on it for effective generalization to new tasks.

From the meta-learning curves in Figure 3, we see about 4x improvement for sawyer object pushing and about 12x improvement for legged locomotion in terms of the number of samples required. Hence, the combination of (1) an off-policy RL algorithm such as SAC for obtaining per-task experts, and (2) the ability to take multiple off-policy supervised gradient steps w.r.t. the experts in the outer loop, enables us to obtain significant overall sample efficiency gains as compared to on-policy meta-RL algorithm such as MAML. These sample efficiency gains are important since they bring us significantly closer to having a meta-reinforcement learning algorithm which can be run on physical robots with practical time scales and sample complexity.

5.3. Meta-Learning from Demonstrations

For challenging tasks involving sparse rewards and image observations, access to demonstrations can greatly help with learning reinforcement learners. GMPS allows us to incorporate this extra supervision much more easily than prior methods.

We compare against MAML and MultiTask as in the previous section. When evaluating on tasks requiring exploration, such as sparse-reward tasks, we additionally compare against model agnostic exploration with structured noise (MAESN) for meta-training (Gupta et al., 2018), which is designed with sparse reward tasks in mind. Finally, we compare GMPS to a single policy is trained with imitation learning across all meta-training tasks using the provided demonstrations (we refer to this comparison as MultiTask Imitation) for adaptation to new validation tasks via fine-

[1] The website is at https://sites.google.com/berkeley.edu/guided-metapolicy-search

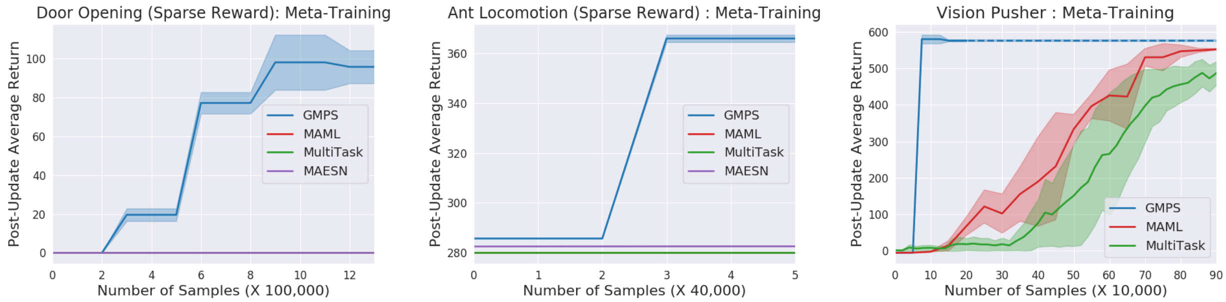

Figure 4. Meta-training comparisons for sparse reward door opening (left), sparse reward ant locomotion (middle) and vision pusher (right). Our method is able to learn when only sparse rewards are available for adaptation, whereas prior methods struggle. For vision-based tasks, we find that GMPS is able to effectively leverage the demonstrations to quickly and stably learn to adapt.

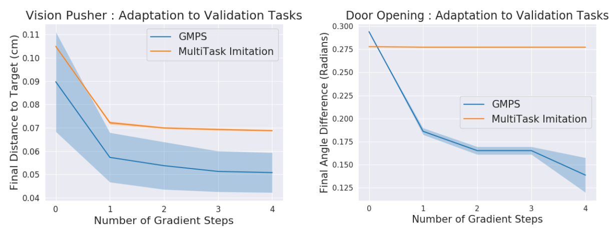

Figure 5. Comparison between GMPS and fine-tuning a policy pretrained with multi-task imitation learning on held-out validation tasks for vision pusher (left) and door opening (right). For pushing, we plot the distance between the block and the target position at the last time-step (in cm), while for door opening we plot the ℓ_1 distance between the door angle at the last time-step and the target angle (in radians). The error-bars correspond to standard error across different seeds.

tuning.

For both the vision and sparse reward experiments, the position of the goal location is not provided as input to the policy (just as for the sample efficiency experiments) - the meta-learning algorithm must discover an effective strategy for inferring the goal from the reward.

5.3.1. Sparse Reward Tasks

One of the potential benefits of learning to learn from demonstrations is that exploration challenges are substantially reduced for the meta-optimizer, since the demonstrations provide detailed guidance on how the task should be performed. We hypothesize that in typical meta-RL, lack of easily available reward signal in sparse reward tasks makes meta-optimization very challenging, while using demonstrations makes this optimization significantly easier.

To test this hypothesis, we experiment with learning to reinforcement learn from sparse reward signals in two different domains: door opening and sparse legged locomotion, as described in Section 5.1. As seen from Figure 4, unlike meta-RL methods such as MAESN (Gupta et al., 2018) and MAML (Finn et al., 2017a), we find that GMPS is able to successfully find a good solution in sparse reward settings and learn to explore. This benefit is largely due to the fact that we can tackle the exploration problem better with demonstrations than requiring meta-reinforcement learning from scratch.

We observe that GMPS is able to adapt to validation tasks more successfully than a policy pre-trained with MultiTask imitation (see Figure 5). This shows that the GMPS training procedure actually optimizes for quick adaptation to new tasks, and uses the gradient update for task inference in order to improve over vanilla imitation learning. On the other hand, the policy pre-trained with imitation learning on the demonstrations does not effectively transfer to the new validation tasks via fine-tuning, since it is not trained for adaptability.

5.3.2. Vision Based Tasks

Deep reinforcement learning methods have the potential to acquire policies that produce actions based simply on visual input, without access to the underlying state of the system (Lange et al., 2012; Mnih et al., 2015; Finn & Levine, 2017). However, vision based policies that can quickly adapt to new tasks using meta-reinforcement learning have proven to be challenging because of the difficulty of optimizing the meta-objective with policy gradient algorithms. These algorithms often have extremely high variance which makes learning from visual inputs challenging. On the other hand, visual imitation learning algorithms and RL algorithms that leverage supervised learning have been far more successful (Levine et al., 2016; Bojarski et al., 2016; Giusti et al., 2016; Zhang & Cho, 2017), which can largely be attributed to the stability of supervised learning as compared with reinforcement learning.

We evaluate GMPS with visual observations under the assumption that we have access to visual demonstrations for the tasks being meta-trained on. Given these demonstrations, we directly train vision-based policies using GMPS with RL in the inner loop and imitation in the outer loop. To best leverage the added stability provided by imitation learning, we meta-optimize the entire policy (both fully connected and convolutional layers), but we only adapt the fully connected layers in the inner loop. This enables us to get the benefits of fast adaptation while retaining the stability of meta-imitation.

As we see in Figure 4, learning vision based policies with GMPS is more stable and achieves higher reward than using meta-learning algorithms such as MAML. Additionally, we find that both GMPS and MAML are able to achieve better performance than a single policy trained with reinforcement learning across all the training tasks, indicating that the policy is indeed adapting to different goal positions rather than learning behavior that averages across the tasks. Also in Figure 5, we find that GMPS does better than MultiTask Imitation for adaptation to validation tasks, just as in the sparse reward case.

6. Discussion and Future Work

In this work, we presented a meta-reinforcement learning algorithm that learns efficient reinforcement learning procedures via supervised imitation. This enables a substantially more efficient meta-training phase that incorporates expert-provided demonstrations to drastically accelerate the acquisition of reinforcement learning procedures and priors. We believe that our method addresses a major limitation in meta-reinforcement learning: although meta-reinforcement learning algorithms can effectively acquire adaptation procedures that can learn new tasks at meta-test time with just a few samples, they are typically extremely expensive in terms of sample count during meta-training, limiting their applicability to real-world problems. By accelerating meta-training via demonstrations, we can enable sample-efficient learning *both* at meta-training time and meta-test time. Given the efficiency and stability of supervised imitation, we expect our method to be readily applicable to domains with high-dimensional observations, such as images. Further, given the number of samples needed in our experiments, our approach is likely efficient enough to be practical to run on physical robotic systems, learning fast reinforcement learning procedures in the real world. Investigating applications of our approach to real-world reinforcement learning is an exciting direction for future work.

References

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2017.

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., and Nowé, A. Reinforcement learning from demonstration through shaping. In *IJCAI*, 2015.

Caruana, R. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

Clavera, I., Nagabandi, A., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 2018.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *NIPS*, pp. 1087–1098, 2017.

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Finn, C. and Levine, S. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *International Conference on Learning Representations (ICLR)*, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017b.

Ghosh, D., Singh, A., Rajeswaran, A., Kumar, V., and Levine, S. Divide-and-conquer reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5307–5316, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., et al. Deep q-learning from demonstrations. *AAAI*, 2018.

Houthooft, R., Chen, R. Y., Isola, P., Stadie, B. C., Wolski, F., Ho, J., and Abbeel, P. Evolved policy gradients. *arXiv preprint arXiv:1802.04821*, 2018.

Kahn, G., Zhang, T., Levine, S., and Abbeel, P. PLATO: policy learning using adaptive trajectory optimization. *CoRR*, abs/1603.00622, 2016.

Kober, J. and Peters, J. R. Policy search for motor primitives in robotics. In *Neural Information Processing Systems (NIPS)*, 2009.

Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013.

Kormushev, P., Calinon, S., and Caldwell, D. G. Robot motor skill coordination with em-based reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, 2010.

Lange, S., Riedmiller, M., and Voigtländer, A. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2012.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 17(39):1–40, 2016.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *International Conference on Learning Representations (ICLR)*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. *International Conference on Robotics and Automation (ICRA)*, 2018.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. Divergences, surrogate loss functions and experimental design. In *NIPS*, 2005.

Norouzi, M., Bengio, S., Chen, Z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. Reward augmented maximum likelihood for neural structured prediction. *CoRR*, abs/1609.00150, 2016.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent rl under partial observability. *International Conference on Machine Learning (ICML)*, 2017.

Parisotto, E., Ba, J. L., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2016.

Peters, J. and Schaal, S. Policy gradient methods for robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, 2006.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Pollard, D. Asymptopia: an exposition of statistical asymptotic theory. 2000.

Rajeswaran, A., Kumar, V., Gupta, A., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems*, 2018.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *CoRR*, abs/1810.06784, 2018.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *International Conference on Learning Representations (ICLR)*, 2016.

Sæmundsson, S., Hofmann, K., and Deisenroth, M. P. Meta reinforcement learning with latent variable gaussian processes. *CoRR*, abs/1803.07551, 2018.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Subramanian, K., Isbell Jr, C. L., and Thomaz, A. L. Exploration from demonstration for interactive reinforcement learning. In *International Conference on Autonomous Agents & Multiagent Systems*, 2016.

Sun, W., Bagnell, J. A., and Boots, B. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *International Conference on Learning Representations (ICLR)*, 2018.

Sung, F., Zhang, L., Xiang, T., Hospedales, T., and Yang, Y. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

Taylor, M. E., Suay, H. B., and Chernova, S. Integrating reinforcement learning with human demonstrations of varying ability. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2017.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 1992.

Zhang, J. and Cho, K. Query-efficient imitation learning for end-to-end simulated driving. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

A. Theoretical Analysis

We can perform a theoretical analysis of algorithm performance in a manner extremely similar to (Kahn et al., 2016). We perform an analysis for an on-policy version of GMPS in this section, although we find that using the off-policy version with expert trajectories only is much more sample efficient, while remaining stable.

Given a policy π, let us denote d_π^t as the state distribution at time t when executing policy π from time 1 to $t-1$. We can define the cost function for a particular task i as $c_i(\mathbf{s}_t, \mathbf{a}_t) = -r_i(\mathbf{s}_t, \mathbf{a}_t)$ as a function of state $\mathbf{s}_t$ and action $\mathbf{a}_t$, with $c_i(\mathbf{s}_t, \mathbf{a}_t) \in [0, 1]$ without loss of generality. We will prove the bound using the notation of cost first, and subsequently express the same in terms of rewards.

Let us define $\pi_\theta + \nabla_i \pi_\theta = \pi_{\theta + \nabla_\theta \mathbb{E}_{\pi_\theta}[R_i]}$ as a shorthand for the policy which is obtained after the inner loop update of meta-learning for task i, with return R_i during meta-optimization. This will be used throughout the proof to represent a one-step update on a task indexed by i, essentially corresponding to policy gradient in the inner loop. We define the performance of a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ over time horizon H, for a particular task i as:

$$J^i(\pi) = \sum_{t=1}^{H} \mathbb{E}_{\mathbf{s}_t \sim d_{\pi_\theta}^t}[\mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[c_i(\mathbf{s}_t, \mathbf{a}_t)]].$$

This can be similarly extended to meta-updated policies as

$$J^i(\pi_\theta + \nabla_i \pi_\theta) = \sum_{t=1}^{H} \mathbb{E}_{\mathbf{s}_t \sim d_{\pi_\theta + \nabla_i \pi_\theta}^t}[\mathbb{E}_{\mathbf{a}_t \sim \pi_\theta + \nabla_i \pi_\theta}[c_i(\mathbf{s}_t, \mathbf{a}_t)]].$$

Let us define $J_t^i(\pi, \tilde{\pi})$ as the expected cost for task i when executing π for t time steps, and then executing $\tilde{\pi}$ for the remaining $H-t$ time steps, and let us similarly define $Q_t^i(\mathbf{s}, \pi, \tilde{\pi})$ as the cost of executing π for one time step, and then executing $\tilde{\pi}$ for $t-1$ time steps.

We will assume the cost-to-go difference between the learned policy and the optimal policy for task i is bounded: $Q_t^i(\mathbf{s}, \pi_\theta, \pi^*) - Q_t^i(\mathbf{s}, \pi^*, \pi^*) \leq \delta, \forall i$. This can be ensured by assuming universality of meta-learning (Finn & Levine, 2018).

When collecting data in order to perform the supervised learning in the outer loop of meta optimization, we can either directly use the 1-step updated policy $\pi_\theta + \nabla_i \pi_\theta$ for each task i, or we can use a mixture policy $\pi_j^i = \beta_j \pi_i^* + (1 - \beta_j)(\pi_\theta + \nabla_i \pi_\theta)$, where j denotes the current iteration of meta-training. This is very similar to the mixture policy suggested in the DAgger algorithm (Ross et al., 2011). In fact, directly using the 1-step updated policy $\pi_\theta + \nabla_i \pi_\theta$ is equivalent to using the mixture policy with $\beta_j = 0, \forall j$. However, to simplify the derivation, we will assume that we always use $\pi_\theta + \nabla_i \pi_\theta$ to collect data, but we can generalize this result to full mixture policies, which would allow us to use more expert data initially and then transition to using on-policy data.

When optimizing the supervised learning objective in the outer loop of meta-optimization to obtain the meta-learned policy initialization π_θ, we assume the supervised learning objective function error is bounded by a constant $D_{\text{KL}}(\pi_\theta + \nabla_i \pi_\theta || \pi_i^*) \leq \epsilon_{\theta*}$ for all tasks i and all per-task expert policies π_i^*. This bound essentially corresponds to assuming that the meta-learner attains bounded training error, which follows from the universality property proven in (Finn & Levine, 2018).

Let $l_i(\mathbf{s}, \pi_\theta + \nabla_i \pi_\theta, \pi_i^*)$ denote the expected 0-1 loss of $\pi_\theta + \nabla_i \pi_\theta$ with respect to π_i^* in state $\mathbf{s}$: $\mathbb{E}_{\mathbf{a}_\theta \sim (\pi_\theta + \nabla_i \pi_\theta)(\mathbf{a}|\mathbf{s}), \mathbf{a}^* \sim \pi_i^*(\mathbf{a}|\mathbf{s})}[\mathbf{1}[\mathbf{a}_\theta \neq \mathbf{a}^*]]$. From prior work, we know that the total variation divergence is an upper bound on the 0-1 loss (Nguyen et al., 2005) and KL-divergence is an upper bound on the total variation divergence (Pollard, 2000).

Therefore, the 0-1 loss can be upper bounded, for all $\mathbf{s}$ drawn from $\pi_\theta + \nabla_i \pi_\theta$:

$$\begin{aligned} l_i(\mathbf{s}, \pi_\theta + \nabla_i \pi_\theta, \pi_i^*) &= \leq D_{\text{TV}}(\pi_\theta + \nabla_i \pi_\theta || \pi_i^*) \\ &\leq \sqrt{D_{\text{KL}}(\pi_\theta + \nabla_i \pi_\theta || \pi_i^*)} \\ &\leq \sqrt{\epsilon_{\theta*}}. \end{aligned}$$

This allows us to bound the meta-learned policy performance using the following theorem:

Theorem A.1 *Let the cost-to-go $Q_t^i(\mathbf{s}, \pi_\theta + \nabla_i \pi_\theta, \pi_i^*) - Q_t^i(\mathbf{s}, \pi_i^*, \pi_i^*) \leq \delta$ for all $t \in \{1, ..., T\}, i \sim p(\mathcal{T})$. Then in* GMPS, *$J(\pi_\theta + \nabla_i \pi_\theta) \leq J(\pi_i^*) + \delta\sqrt{\epsilon_{\theta*}} O(H)$, and by extension $\mathbb{E}_{i\sim tasks}[J(\pi_\theta + \nabla_i \pi_\theta)] \leq \mathbb{E}_{i\sim tasks}[J(\pi_i^*)] + \delta\sqrt{\epsilon_{\theta*}} O(H)$*

Proof:

$$
\begin{aligned}
J^i(\pi_\theta + \nabla_i \pi_\theta) &= J^i(\pi_i^*) + \sum_{t=0}^{T-1} J_{t+1}^i(\pi_\theta + \nabla_i \pi_\theta, \pi_i^*) - J_t^i(\pi_\theta + \nabla_i \pi_\theta, \pi_i^*) \\
&= J^i(\pi_i^*) + \sum_{t=1}^{H} \mathbb{E}_{\mathbf{s}\sim d^t_{\pi_\theta + \nabla_i \pi_\theta}} [Q_t^i(\mathbf{s}, \pi_\theta + \nabla_i \pi_\theta, \pi_i^*) - Q_t^i(\mathbf{s}, \pi_i^*, \pi_i^*)] \\
&\leq J^i(\pi_i^*) + \delta \sum_{t=1}^{H} \mathbb{E}_{\mathbf{s}\sim d^t_{\pi_\theta + \nabla_i \pi_\theta}} [l_i(\mathbf{s}, \pi_\theta + \nabla_i \pi_\theta, \pi_i^*)] && \text{(7a)} \\
&\leq J^i(\pi_i^*) + \delta \sum_{t=1}^{H} \sqrt{\epsilon_{\theta*}} && \text{(7b)} \\
&= J^i(\pi_i^*) + \delta T \sqrt{\epsilon_{\theta*}}
\end{aligned}
$$

Equation 7a follows from the fact that the expected 0-1 loss of $\pi_\theta + \nabla_i \pi_\theta$ with respect to π_i^* is the probability that $\pi_\theta + \nabla_i \pi_\theta$ and π_i^* pick different actions in $\mathbf{s}$; when they choose different actions, the cost-to-go increases by $\leq \delta$. Equation 7b follows from the upper bound on the 0-1 loss.

Now that we have the proof for a particular i, we can simply take expectation with respect to i sampled from the distribution of tasks to get the full result.

Proof:

$$
\begin{aligned}
&J^i(\pi_\theta + \nabla_i \pi_\theta) \leq J^i(\pi_i^*) + \delta T \sqrt{\epsilon_{\theta*}} \\
&\implies E_{i\sim p(\text{tasks})}[J^i(\pi_\theta + \nabla_i \pi_\theta)] \leq E_{i\sim p(\text{tasks})}[J^i(\pi_i^*)] + \delta T \sqrt{\epsilon_{\theta*}} && \text{(8a)}
\end{aligned}
$$

Now in order to convert back to the version using rewards instead of costs, we can simply negate the bound, thereby giving us the original theorem 4.1, which states:

$$
\mathbb{E}_{i\sim p(\mathcal{T})}[\mathbb{E}_{\pi_{\theta + \nabla_\theta \mathbb{E}_{\pi_\theta}[R_i]}}[\sum_{t=1}^{T} r_i(\mathbf{s}_t, \mathbf{a}_t)]] \geq \mathbb{E}_{i\sim p(\mathcal{T})}[\mathbb{E}_{\pi_i^*}[\sum_{t=1}^{H} r_i(\mathbf{s}_t, \mathbf{a}_t)]] - \delta\sqrt{\epsilon_{\theta*}} O(H)
$$

.

B. Reward Functions

Below are the reward functions used for each of our experiments.

- Sawyer Pushing (for both full state and vision observations)

$$
R = -\|x_{obj} - x_{pusher}\|_2 + 100 \mid c - \|x_{goal} - x_{pusher}\|_2 \mid
$$

  where c is the initial distance between the object and the goal (a constant).

- Door Opening

$$
R = \begin{cases} \mid 10x \mid & x \leq x^* \\ \mid 10(x^* - (x - x^*)) \mid & x > x^* \end{cases}
$$

  where x is the current door angle, and x^* is the target door angle

- Legged Locomotion (dense reward)

$$R = -||x - x^*||_1 + 4.0$$

where x is the location of centre of mass of the ant, x^* is the goal location.

- Legged Locomotion (sparse reward)

$$R = \begin{cases} -||x - x^*||_1 + 4.0 & \|x - x^*\|_2 \leq 0.8 \\ -m + 4.0 & \|x - x^*\|_2 > 0.8 \end{cases}$$

where x is the location of center of mass of the ant, x^* is the goal location, and m is the initial ℓ_1 distance between x and x^* (a constant).

C. Architectures

- State-based Experiments

  Used a neural network with two hidden layers of 100 units with ReLU nonlinearities each for GMPS, MAML, multi-task learning, and MAESN. As shown in prior work (Finn et al., 2017b), adding a bias transformation variable helps improve performance for MAML, so we ran experiments including this variation. [The bias transformation variable is simply a variable appended to the observation, before being passed into the policy. This variable is also adapted with gradient descent in the inner loop]. The learning rate for the fast adaptation step (α) is also meta-learned.

- Vision-based Experiments

  The image is passed through a convolutional neural network, followed by a spatial soft-argmax (Levine et al., 2016), followed by a fully connected network block. The 3D end-effector position is appended to the result of the spatial soft-argmax, which is then passed through a fully connected neural network block. The convolution block is specified as follows: 16 filters of size 5 with stride 3, followed by 16 filters of size 3 with stride 3 , followed by 16 filters of size 3 with stride 1. The fully-connected block is as follows: 2 hidden layers of 100 units each. All hidden layers use ReLU nonlinearities.

D. HyperParameters

The following are the hyper-parameter sweeps for each of the methods [run for each of the experimental domains] , run over 3 seeds.

1. GMPS
   (a) Number of trajectories sampled per task. : [20 , 50]
   (b) Number of tasks for meta-learning: [10 , 20]
   (c) Initial value for fast adaptation learning rate: [0.5, 0.1]
   (d) Variables included for fast adaptation: [all parameters, only bias transform variable]
   (e) Dimension of bias transform variable: [2, 4]
   (f) Number of imitation steps in between sampling new data from the pre-update policy: [1 , 200, 500, 1000, 2000]
2. MAML

   Hyper-parameter sweeps (a) - (d) from GMPS
3. MAESN

   Hyper-parameter sweeps (a) - (c) from GMPS
   (a) Dimension of latent variable: [2,4]
4. MultiTask
   (a) Batch size: [10000, 50000]

(b) Learning rate: [0.01, 0.02]

5. Contextual SAC [which is used to learn experts that are then used for GMPS]
    (a) Reward scale: [10, 50, 100] (constant which scales the reward)
    (b) Number of gradient steps taken for each batch of collected data: [1, 5, 10]