Bayesian Model-Agnostic Meta-Learning

Taesup Kim[*‡2], **Jaesik Yoon**[*3], **Ousmane Dia**[1], **Sungwoong Kim**[4],
Yoshua Bengio[2,5], **Sungjin Ahn**[‡6]

[1]Element AI, [2]MILA Université de Montréal, [3]SAP, [4]Kakao Brain,
[5]CIFAR Senior Fellow, [6]Rutgers University

Abstract

Due to the inherent model uncertainty, learning to infer Bayesian posterior from a few-shot dataset is an important step towards robust meta-learning. In this paper, we propose a novel Bayesian model-agnostic meta-learning method. The proposed method combines efficient gradient-based meta-learning with nonparametric variational inference in a principled probabilistic framework. Unlike previous methods, during fast adaptation, the method is capable of learning complex uncertainty structure beyond a simple Gaussian approximation, and during meta-update, a novel Bayesian mechanism prevents meta-level overfitting. Remaining a gradient-based method, it is also the first Bayesian model-agnostic meta-learning method applicable to various tasks including reinforcement learning. Experiment results show the accuracy and robustness of the proposed method in sinusoidal regression, image classification, active learning, and reinforcement learning.

1 Introduction

Two-year-old children can infer a new category from only one instance (Smith & Slone, 2017). This is presumed to be because during early learning, a human brain develops foundational structures such as the "shape bias" in order to learn the learning procedure (Landau et al., 1988). This ability, also known as *learning to learn* or *meta-learning* (Biggs, 1985; Bengio et al., 1990), has recently obtained much attention in machine learning by formulating it as few-shot learning (Lake et al., 2015; Vinyals et al., 2016). Because, initiating the learning from scratch, a neural network can hardly learn anything meaningful from such a few data points, a learning algorithm should be able to extract the statistical regularity from past tasks to enable warm-start for subsequent tasks.

Learning a new task from a few examples inherently induces a significant amount of uncertainty. This is apparent when we train a complex model such as a neural network using only a few examples. It is also empirically supported by the fact that a challenge in existing few-shot learning algorithms is their tendency to overfit (Mishra et al., 2017). A robust meta-learning algorithm therefore must be able to systematically deal with such uncertainty in order to be applicable to critical problems such as healthcare and self-driving cars. Bayesian inference provides a principled way to address this issue. It brings us not only robustness to overfitting but also numerous benefits such as improved prediction accuracy by Bayesian ensembling (Balan et al., 2015), active learning (Gal et al., 2016), and principled/safe exploration in reinforcement learning (Houthooft et al., 2016). Therefore, developing a Bayesian few-shot learning method is an important step towards robust meta-learning.

Motivated by the above arguments, in this paper we propose a Bayesian meta-learning method, called Bayesian MAML. By introducing Bayesian methods for fast adaptation and meta-update, the proposed method learns to quickly obtain an approximate posterior of a given unseen task and thus provides the benefits of having access to uncertainty. Being an efficient and scalable gradient-based

[*]Equal contribution, Correspondence to sungjin.ahn@rutgers.edu, ‡Work also done while working at Element AI

meta-learner which encodes the meta-level statistical regularity in the initial model parameters, our method is the first Bayesian model-agnostic meta-learning method which is thus applicable to various tasks including reinforcement learning. Combining an efficient nonparametric variational inference method with gradient-based meta-learning in a principled probabilistic framework, it can learn complex uncertainty structures while remaining simple to implement.

The main contributions of the paper are as follows. We propose a novel Bayesian method for meta-learning. The proposed method is based on a novel Bayesian fast adaptation method and a new meta-update loss called the Chaser loss. To our knowledge, the Bayesian fast adaptation is the first in meta-learning that provides flexible capability to capture the complex uncertainty curvature of the task-posterior beyond a simple Gaussian approximation. Furthermore, unlike the previous methods, the Chaser loss prevents meta-level overfitting. In experiments, we show that our method is efficient, accurate, robust, and applicable to various problems: sinusoidal regression, image classification, reinforcement learning, and active learning.

2 Preliminaries

Consider a model $y = f_\theta(x)$ parameterized by and differentiable w.r.t. θ. Task τ is specified by a K-shot dataset $\mathcal{D}_\tau$ that consists of a small number of training examples, e.g., K pairs (x_k, y_k) per class for classification. We assume that tasks are sampled from a task distribution $\tau \sim p(\mathcal{T})$ such that the sampled tasks share the statistical regularity of the task distribution. A meta-learning algorithm leverages this regularity to improve the learning efficiency of subsequent tasks. The whole *dataset of tasks* is divided into training/validation/test *tasksets*, and the dataset of each task is further divided into task-training/task-validation/task-test *datasets*.

Model-Agnostic Meta Learning (MAML) proposed by Finn et al. (2017) is a gradient-based meta-learning framework. Because it works purely by gradient-based optimization without requiring additional parameters or model modification, it is simple and generally applicable to any model as long as the gradient can be estimated.

In Algorithm 1, we briefly review MAML. At each meta-train iteration t, it performs: (i) *Task-Sampling*: a mini-batch $\mathcal{T}_t$ of tasks is sampled from the task distribution $p(\mathcal{T})$. Each task $\tau \in \mathcal{T}_t$ provides task-train data $\mathcal{D}_\tau^{\text{trn}}$ and task-validation data $\mathcal{D}_\tau^{\text{val}}$. (ii) *Fast Adaptation* (or *Inner-Update*): the parameter for each task τ in $\mathcal{T}_t$ is updated by starting from the *current* generic initial model θ_0 and then performing n gradient descent steps on the task-train loss, an operation which we denote by $\text{GD}_n(\theta_0; \mathcal{D}_\tau^{\text{trn}}, \alpha)$ with α being a step size. (iii) *Meta-Update* (or *Outer-Update*): the generic initial parameter θ_0 is updated by gradient descent. The meta-loss is the summation of task-validation losses for all tasks in $\mathcal{T}_t$, i.e., $\sum \mathcal{L}(\theta_\tau; \mathcal{D}_\tau^{\text{val}})$ where the summation is over all $\tau \in \mathcal{T}_t$. At meta-test time, given an unseen test-task $\bar{\tau} \sim p(\mathcal{T})$, starting from the optimized initial model θ_0^*, we obtain a model $\theta_{\bar{\tau}}$ by taking a small number of inner-update steps using K-shot *task-training* data $\mathcal{D}_{\bar{\tau}}^{\text{trn}}$. Then, the learned model $\theta_{\bar{\tau}}$ is evaluated on the *task-test* dataset $\mathcal{D}_{\bar{\tau}}^{\text{tst}}$.

Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016) is a recently proposed nonparametric variational inference method. SVGD combines the strengths of MCMC and variational inference. Unlike traditional variational inference, SVGD does not confine the family of approximate distributions within tractable parametric distributions while still remaining a simple algorithm. Also, it converges faster than MCMC because its update rule is deterministic and leverages the gradient of the target distribution. Specifically, to obtain M samples from target distribution $p(\theta)$, SVGD maintains M instances of model parameters, called *particles*. We denote the particles by $\Theta = \{\theta^m\}_{m=1}^M$. At iteration t, each particle $\theta_t \in \Theta_t$ is updated by the following rule:

$$\theta_{t+1} \leftarrow \theta_t + \epsilon_t \phi(\theta_t) \;\; \text{where} \;\; \phi(\theta_t) = \frac{1}{M} \sum_{j=1}^{M} \left[k(\theta_t^j, \theta_t) \nabla_{\theta_t^j} \log p(\theta_t^j) + \nabla_{\theta_t^j} k(\theta_t^j, \theta_t) \right], \qquad (1)$$

where ϵ_t is step-size and $k(x, x')$ is a positive-definite kernel. We can see that a particle consults with other particles by asking their gradients, and thereby determines its own update direction. The importance of other particles is weighted according to the kernel distance, relying more on closer particles. The last term $\nabla_{\theta^j} k(\theta^j, \theta^m)$ enforces repulsive force between particles so that they do not collapse to a point. The resulting particles can be used to obtain the posterior predictive distribution $p(y|x, \mathcal{D}^\tau) = \int p(y|x, \theta) p(\theta|\mathcal{D}^\tau) \text{d}\theta \approx \frac{1}{M} \sum_m p(y|x, \theta^m)$ where $\theta^m \sim p(\theta|\mathcal{D}^\tau)$.

Algorithm 1 MAML

Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
for each task $\tau \in \mathcal{T}_t$ **do**
  $\theta_\tau \leftarrow \mathrm{GD}_n(\theta_0; \mathcal{D}_\tau^{\text{trn}}, \alpha)$
end for
$\theta_0 \leftarrow \theta_0 - \beta \nabla_{\theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}(\theta_\tau; \mathcal{D}_\tau^{\text{val}})$

Algorithm 2 Bayesian Fast Adaptation

Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
for each task $\tau \in \mathcal{T}_t$ **do**
  $\Theta_\tau(\Theta_0) \leftarrow \mathrm{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\text{trn}}, \alpha)$
end for
$\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}})$

A few properties of SVGD are particularly relevant to the proposed method: (i) when the number of particles M equals 1, SVGD becomes standard gradient ascent on the objective $\log p(\theta)$, (ii) under a certain condition, an SVGD-update increasingly reduces the distance between the approximate distribution defined by the particles and the target distribution, in the sense of Kullback-Leibler (KL) divergence (Liu & Wang, 2016), and finally (iii) it is straightforward to apply to reinforcement learning by using Stein Variational Policy Gradient (SVPG) (Liu et al., 2017).

3 Proposed Method

3.1 Bayesian Fast Adaptation

Our goal is to *learn to infer* by developing an efficient Bayesian gradient-based meta-learning method to efficiently obtain the task-posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ of a novel task. As our method is in the same class as MAML – in the sense that it encodes the meta-knowledge in the initial model by gradient-based optimization – we first consider the following probabilistic interpretation of MAML with one inner-update step,

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} p(\mathcal{D}_\tau^{\text{val}} \mid \theta'_\tau = \theta_0 + \alpha \nabla_{\theta_0} \log p(\mathcal{D}_\tau^{\text{trn}} \mid \theta_0)), \tag{2}$$

where $p(\mathcal{D}_\tau^{\text{val}}|\theta'_\tau) = \prod_{i=1}^{|\mathcal{D}_\tau^{\text{val}}|} p(y_i|x_i, \theta'_\tau)$, $\mathcal{D}_{\mathcal{T}}^{\text{trn}}$ denotes all task-train sets in training taskset $\mathcal{T}$, and $\mathcal{D}_{\mathcal{T}}^{\text{val}}$ has the same meaning but for task-validation sets. From the above, we can see that the inner-update step of MAML amounts to obtaining task model θ'_τ from which the likelihood of the task-validation set $\mathcal{D}_\tau^{\text{val}}$ is computed. The meta-update step is then to perform maximum likelihood estimation of this model w.r.t. the initial parameter θ_0. This probabilistic interpretation can be extended further to applying empirical Bayes to a hierarchical probabilistic model (Grant et al., 2018) as follows:

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} \left(\int p(\mathcal{D}_\tau^{\text{val}} \mid \theta_\tau) p(\theta_\tau \mid \mathcal{D}_\tau^{\text{trn}}, \theta_0) \mathrm{d}\theta_\tau \right). \tag{3}$$

We see that the probabilistic MAML model in Eq. (2) is a special case of Eq. (3) that approximates the task-train posterior $p(\theta_\tau|\theta_0, \mathcal{D}_\tau^{\text{trn}})$ by a point estimate θ'_τ. That is, $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}, \theta_0) = \delta_{\theta'_\tau}(\theta_\tau)$ where $\delta_y(x) = 1$ if $x = y$, and 0 otherwise. To model the task-train posterior which also becomes the prior of task-validation set, Grant et al. (2018) used an isotropic Gaussian distribution with a fixed variance.

"*Can we use a more flexible task-train posterior than a point estimate or a simple Gaussian distribution while maintaining the efficiency of gradient-based meta-learning*?" This is an important question because as discussed in Grant et al. (2018), the task-train posterior of a Bayesian neural network (BNN) trained with a few-shot dataset would have a significant amount of uncertainty which, according to the Bayesian central limit theorem (Le Cam, 1986; Ahn et al., 2012), cannot be well approximated by a Gaussian distribution.

Our first step for designing such an algorithm starts by noticing that SVGD performs deterministic updates and thus gradients can be backpropagated through the particles. This means that we now maintain M initial particles Θ_0 and by obtaining samples from the task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}, \Theta_0)$ using SVGD (which is now conditioned on Θ_0 instead of θ_0), we can optimize the following Monte Carlo approximation of Eq. (3) by computing the gradient of the meta-loss $\log p(\mathcal{D}_{\mathcal{T}}^{\text{val}}|\Theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}})$ w.r.t. Θ_0,

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \Theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) \approx \prod_{\tau \in \mathcal{T}} \left(\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}_\tau^{\text{val}} \mid \theta_\tau^m) \right) \quad \text{where} \quad \theta_\tau^m \sim p(\theta_\tau \mid \mathcal{D}_\tau^{\text{trn}}, \Theta_0). \tag{4}$$

Being updated by gradient descent, it hence remains an efficient meta-learning method while providing a more flexible way to capture the complex uncertainty structure of the task-train posterior than a point estimate or a simple Gaussian approximation.

Algorithm 2 describes an implementation of the above model. Specifically, at iteration t, for each task τ in a sampled mini-batch $\mathcal{T}_t$, the particles initialized to Θ_0 are updated for n steps by applying the SVGD updater, denoted by $\text{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$ – the target distribution (the $p(\theta_t^j)$ in Eq. (1) is set to the task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto p(\mathcal{D}_\tau^{\text{trn}}|\theta_\tau)p(\theta_\tau)$[1]. This results in task-wise particles Θ_τ for each task $\tau \in \mathcal{T}_t$. Then, for the meta-update, we can use the following meta-loss, $\log p(\mathcal{D}_{\mathcal{T}_t}^{\text{val}}|\Theta_0, \mathcal{D}_{\mathcal{T}_t}^{\text{trn}})$

$$\approx \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}}) \;\text{ where }\; \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}}) = \log \left[\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}_\tau^{\text{val}}|\theta_\tau^m) \right], \tag{5}$$

Here, we use $\Theta_\tau(\Theta_0)$ to explicitly denote that Θ_τ is a function of Θ_0. Note that, by the above model, all the initial particles in Θ_0 are *jointly* updated in such a way as to find the best joint-formation among them. From this optimized initial particles, the task-posterior of a new task can be obtained quickly, i.e., by taking a small number of update steps, and efficiently, i.e, with a small number of samples. We call this Bayesian Fast Adaptation (BFA). The method can be considered a Bayesian ensemble in which, unlike non-Bayesian ensemble methods, the particles interact with each other to find the best formation representing the task-train posterior. Because SVGD with a single particle, i.e., $M = 1$, is equal to gradient ascent, Algorithm 2 reduces to MAML when $M = 1$.

Although the above algorithm brings the power of Bayesian inference to fast adaptation, it can be numerically unstable due to the product of the task-validation likelihood terms. More importantly, for meta-update it is not performing Bayesian inference. Instead, it looks for the initial prior Θ_0 such that SVGD-updates lead to minimizing the empirical loss on task-validation sets. Therefore, like other meta-learning methods, the BFA model can still suffer from overfitting despite the fact that we use a flexible Bayesian inference in the inner update. The reason is somewhat apparent. Because we perform only a small number of inner-updates while the meta-update is based on empirical risk minimization, the initial model Θ_0 can be overfitted to the task-validation sets when we use highly complex models like deep neural networks. Therefore, to become a fully robust meta-learning approach, it is desired for the method to retain the uncertainty during the meta-update as well while remaining an efficient gradient-based method.

3.2 Bayesian Meta-Learning with Chaser Loss

Motivated by the above observation, we propose a novel meta-loss. For this, we start by defining the loss as the dissimilarity between approximate task-train posterior $p_\tau^n \equiv p_n(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}; \Theta_0)$ and true task-posterior $p_\tau^\infty \equiv p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}})$. Note that p_τ^n is obtained by taking n fast-adaptation steps from the initial model. Assuming that we can obtain samples Θ_τ^n and Θ_τ^∞ respectively from these two distributions, the new meta-learning objective can be written as

$$\arg\min_{\Theta_0} \sum_\tau d_p(p_\tau^n \parallel p_\tau^\infty) \approx \arg\min_{\Theta_0} \sum_\tau d_s(\Theta_\tau^n(\Theta_0) \parallel \Theta_\tau^\infty). \tag{6}$$

Here, $d_p(p\|q)$ is a dissimilarity between two distributions p and q, and $d_s(s_1\|s_2)$ a distance between two sample sets. We then want to minimize this distance using gradient w.r.t. Θ_0. This is to find optimized Θ_0 from which the task-train posterior can be obtained quickly and closely to the true task-posterior. However, this is intractable because we neither have access to the true posterior p_τ^∞ nor its samples Θ_τ^∞.

To this end, we approximate Θ_τ^∞ by Θ_τ^{n+s}. This is done by (i) obtaining Θ_τ^n from $p_n(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}; \Theta_0)$ and then (ii) taking s additional SVGD steps with the updated target distribution $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}})$, i.e., augmented with additional observation $\mathcal{D}_\tau^{\text{val}}$. Although it is valid in theory not to augment the leader with the validation set, to help fast convergence we take advantage of it like other meta-learning methods. Note that, because SVGD-updates provide increasingly better approximations of the target

[1] In our experiments, we put hyperprior on the variance of the prior (mean is set to 0). Thus, the posterior of hyperparameter is automatically learned also by SVGD, i.e., the particle vectors include the prior parameters.

Algorithm 3 Bayesian Meta-Learning with Chaser Loss (BMAML)

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Compute chaser $\Theta_\tau^n(\Theta_0) = \text{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\text{trn}}, \alpha)$
    Compute leader $\Theta_\tau^{n+s}(\Theta_0) = \text{SVGD}_s(\Theta_\tau^n(\Theta_0); \mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}}, \alpha)$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_\tau^n(\Theta_0) \parallel \text{stopgrad}(\Theta_\tau^{n+s}(\Theta_0)))$
end for

distribution as s increases, the *leader* Θ_τ^{n+s} becomes closer to the target distribution Θ_τ^∞ than the *chaser* Θ_τ^n. This gives us the following meta-loss:

$$\mathcal{L}_{\text{BMAML}}(\Theta_0) = \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_\tau^n \parallel \Theta_\tau^{n+s}) = \sum_{\tau \in \mathcal{T}_t} \sum_{m=1}^{M} \|\theta_\tau^{n,m} - \theta_\tau^{n+s,m}\|_2^2. \tag{7}$$

Here, to compute the distance between the two sample sets, we make a one-to-one mapping between the leader particles and the chaser particles and compute the Euclidean distance between the paired particles. Note that we do not back-propagate through the leader particles because we use them as targets that the chaser particles follow. A more sophisticated method like maximum mean discrepancy (Borgwardt et al., 2006) can also be used here. In our experiments, setting n and s to a small number like $n = s = 1$ worked well.

Minimizing the above loss w.r.t. Θ_0 places Θ_0 in a region where the chaser Θ_τ^n can efficiently *chase* the leader Θ_τ^{n+s} in n SVGD-update steps starting from Θ_0. Thus, we call this meta-loss the *Chaser* loss. Because the leader converges to the posterior distribution instead of doing empirical risk minimization, it retains a proper level of uncertainty and thus prevents from meta-level overfitting. In Algorithm 3, we describe the algorithm for supervised learning. One limitation of the method is that, like other ensemble methods, it needs to maintain M model instances. Because this could sometimes be an issue when training a large model, in the Experiment section we introduce a way to share parameters among the particles.

4 Related Works

There have been many studies in the past that formulate meta-learning and learning-to-learn from a probabilistic modeling perspective (Tenenbaum, 1999; Fe-Fei et al., 2003; Lawrence & Platt, 2004; Daumé III, 2009). Since then, the remarkable advances in deep neural networks (Krizhevsky et al., 2012; Goodfellow et al., 2016) and the introduction of new few-shot learning datasets (Lake et al., 2015; Ravi & Larochelle, 2017), have rekindled the interest in this problem from the perspective of deep networks for few-shot learning (Santoro et al., 2016; Vinyals et al., 2016; Snell et al., 2017; Duan et al., 2016; Finn et al., 2017; Mishra et al., 2017). Among these, Finn et al. (2017) proposed MAML that formulates meta-learning as gradient-based optimization.

Grant et al. (2018) reinterpreted MAML as a hierarchical Bayesian model, and proposed a way to perform an implicit posterior inference. However, unlike our proposed model, the posterior on validation set is approximated by local Laplace approximation and used a relatively complex 2^{nd}-order optimization using K-FAC (Martens & Grosse, 2015). The fast adaptation is also approximated by a simple isotropic Gaussian with fixed variance. As pointed by Grant et al. (2018), this approximation would not work well for skewed distributions, which is likely to be the case of BNNs trained on a few-shot dataset. The authors also pointed that their method is limited in that the predictive distribution over new data-points is approximated by a point estimate. Our method resolves these limitations. Although it can be expensive when training many large networks, we mitigate this cost by parameter sharing among the particles. In addition, Bauer et al. (2017) also proposed Gaussian approximation of the task-posterior and a scheme of splitting the feature network and the classifier which is similar to what we used for the image classification task. Lacoste et al. (2017) proposed learning a distribution of stochastic input noise while fixing the BNN model parameter.

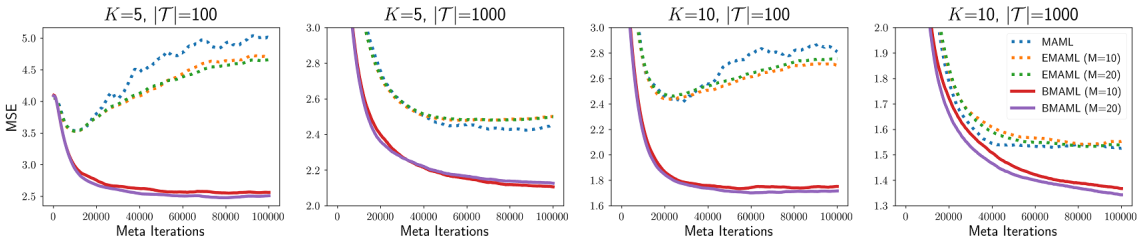


Figure 1: Sinusoidal regression experimental results (meta-testing performance) by varying the number of examples (K-shot) given for each task and the number of tasks $|\mathcal{T}|$ used for meta-training.

5 Experiments

We evaluated our proposed model (BMAML) in various few-shot learning tasks: sinusoidal regression, image classification, active learning, and reinforcement learning. Because our method is a Bayesian ensemble, as a baseline model we used an ensemble of independent MAML models (EMAML) from which we can easily recover regular MAML by setting the number of particles to 1. In all our experiments, we configured BMAML and EMAML to have the same network architecture and used the RBF kernel. The experiments are designed in such a way to see the effects of uncertainty in various ways such as accuracy, robustness, and efficient exploration.

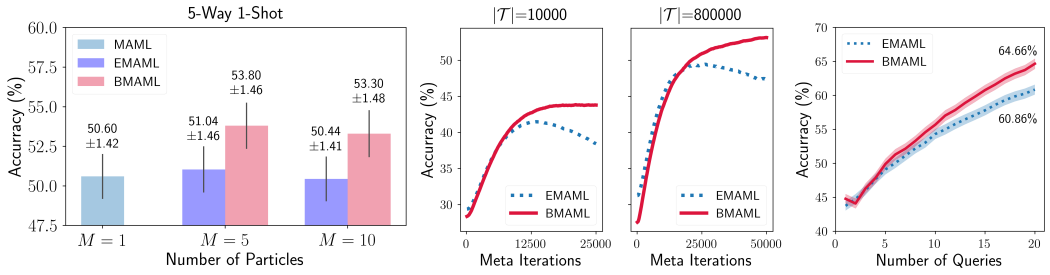
Regression: The population of the tasks is defined by a sinusoidal function $y = A\sin(wx + b) + \epsilon$ which is parameterized by amplitude A, frequency w, and phase b, and observation noise ϵ. To sample a task, we sample the parameters uniformly randomly $A \in [0.1, 5.0]$, $b \in [0.0, 2\pi]$, $w \in [0.5, 2.0]$ and add observation noise from $\epsilon \sim \mathcal{N}(0, (0.01A)^2)$. The K-shot dataset is obtained by sampling x from $[-5.0, 5.0]$ and then by computing its corresponding y with noise ϵ. Note that, because of the highly varying frequency and observation noise, this is a more challenging setting containing more uncertainty than the setting used in Finn et al. (2017). For the regression model, we used a neural network with 3 layers each of which consists of 40 hidden units.

In Fig. 1, we show the mean squared error (MSE) performance on the test tasks. To see the effect of the degree of uncertainty, we controlled the number of training tasks $|\mathcal{T}|$ to 100 and 1000, and the number of observation shots K to 5 and 10. The lower number of training tasks and observation shots is expected to induce a larger degree of uncertainty. We observe, as we claimed, that both MAML (which is EMAML with $M = 1$) and EMAML overfit severely in the settings with high uncertainty although EMAML with multiple particles seems to be slightly better than MAML. BMAML with the same number of particles provides significantly better robustness and accuracy for all settings. Also, having more particles tends to improve further.

Classification: To evaluate the proposed method on a more complex model, we test the performance on the *mini*Imagenet classification task (Vinyals et al., 2016) involving task adaptation of 5-way classification with 1 shot. The dataset consists of 60,000 color images of 84×84 dimension. The images consist of total 100 classes and each of the classes contains 600 examples. The entire classes are split into 64, 12, and 24 classes for meta-train, meta-validation, and meta-test, respectively. We generated the tasks following the same procedure as in Finn et al. (2017).

In order to reduce the space and time complexity of the ensemble models (i.e., BMAML and EMAML) in this large network setting, we used the following parameter sharing scheme among the particles, similarly to Bauer et al. (2017). We split the network architecture into the feature extractor layers and the classifier. The feature extractor is a convolutional network with 5 hidden layers with 64 filters. The classifier is a single-layer fully-connected network with softmax output. The output of the feature extractor which has 256 dimensions is input to the classifier. We share the feature extractor across all the particles while each particle has its own classifier. Therefore, the space complexity of the network is $\mathcal{O}(|\theta_{\text{feature}}| + M|\theta_{\text{classifier}}|)$. Both the classifier and feature extractor are updated during meta-update, but for inner-update only the classifier is updated. The baseline models are updated in the same manner. We describe more details of the setting in Appendix A.2.

We can see from Fig. 2 (a) that for both $M = 5$ and $M = 10$ BMAML provides more accurate predictions than EMAML. However, the performance of both BMAML and EMAML with 10



(a) **(b)** **(c)**

Figure 2: Experimental results in *mini*Imagenet dataset: (a) few-shot image classification using different number of particles, (b) using different number of tasks for meta-training, and (c) active learning setting.

particles is slightly lower than having 5 particles[2]. Because a similar instability is also observed in the SVPG paper (Liu & Wang, 2016), we presume that one possible reason is the instability of SVGD such as sensitivity to kernel function parameters. To increase the inherent uncertainty further, in Fig. 2 (b), we reduced the number of training tasks $|\mathcal{T}|$ from 800K to 10K. We see that BMAML provides robust predictions even for such a small number of training tasks while EMAML overfits easily.

Active Learning: In addition to the ensembled prediction accuracy, we can also evaluate the effectiveness of the measured uncertainty by applying it to active learning. To demonstrate, we use the *mini*Imagenet classification task. To do this, given an unseen task τ at test time, we first run a fast adaptation from the meta-trained initial particles Θ_0^* to obtain Θ_τ of the task-train posterior $p(\theta_\tau|\mathcal{D}_\tau; \Theta_0^*)$. For this we used 5-way 1-shot labeled dataset. Then, from a pool of unlabeled data $\mathcal{X}_\tau = \{x_1, \ldots, x_{20}\}$, we choose an item x^* that has the maximum predictive entropy $\arg\max_{x \in \mathcal{X}_\tau} \mathbb{H}[y|x, D_\tau] = -\sum_{y'} p(y'|x, D_\tau) \log p(y'|x, D_\tau)$. The chosen item x^* is then removed from $\mathcal{X}_\tau$ and added to $\mathcal{D}_\tau$ along with its label. We repeat this process until we consume all the data in $\mathcal{X}_\tau$. We set M to 5. As we can see from Fig. 2 (c), active learning using the Bayesian fast adaptation provides consistently better results than EMAML. Particularly, the performance gap increases as more examples are added. This shows that the examples picked by BMAML so as to reduce the uncertainty, provides proper discriminative information by capturing a reasonable approximation of the task-posterior. We presume that the performance degradation observed in the early stage might be due to the class imbalance induced by choosing examples without considering the class balance.

Reinforcement Learning: SVPG is a simple way to apply SVGD to policy optimization. Liu et al. (2017) showed that the maximum entropy policy optimization can be recast to Bayesian inference. In this framework, the particle update rule (a particle is now parameters of a policy) is simply to replace the target distribution $\log p(\theta)$ in Eq. (1) with the objective of the maximum entropy policy optimization, i.e., $\mathbb{E}_{q(\theta)}[J(\theta)] + \eta\mathbb{H}[q]]$ where $q(\theta)$ is a distribution of policies, $J(\theta)$ is the expected return of policy θ, and η is a parameter for exploration control. Deploying multiple agents (particles) with a principled Bayesian exploration mechanism, SVPG encourages generating diverse policy behaviours while being easy to parallelize.

We test and compare the models on the same MuJoCo continuous control tasks (Todorov et al., 2012) as are used in Finn et al. (2017). In the goal velocity task, the agent receives higher rewards as its current velocity approaches the goal velocity of the task. In the goal direction task, the reward is the magnitude of the velocity in either the forward or backward direction. We tested these tasks for two simulated robots, the ant and the cheetah. The goal velocity is sampled uniformly at random from $[0.0, 2.0]$ for the cheetah and from $[0.0, 3.0]$ for the ant. As the goal velocity and the goal direction change per task, a meta learner is required to learn a given unseen task after trying K episodes. We implemented the policy network with two hidden-layers each with 100 ReLU units. We tested the number of particles for $M \in \{1, 5, 10\}$ with $M = 1$ only for non-ensembled MAML. We describe more details of the experiment setting in Appendix C.1.

[2]We found a similar instability in the relationship between the number of particles and the prediction accuracy from the original implementation by the authors of the SVGD paper.

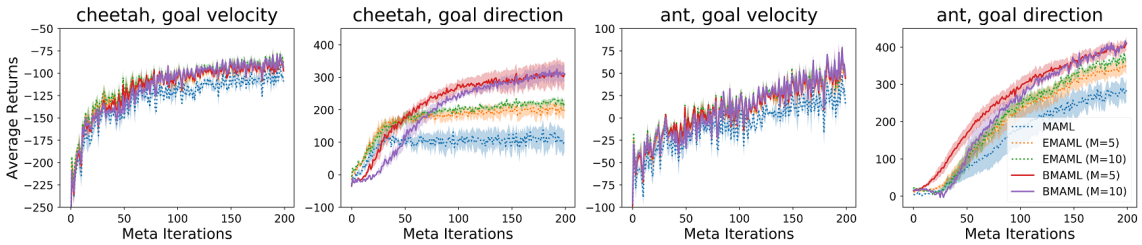


Figure 3: Locomotion comparison results of SVPG-TRPO and VPG-TRPO

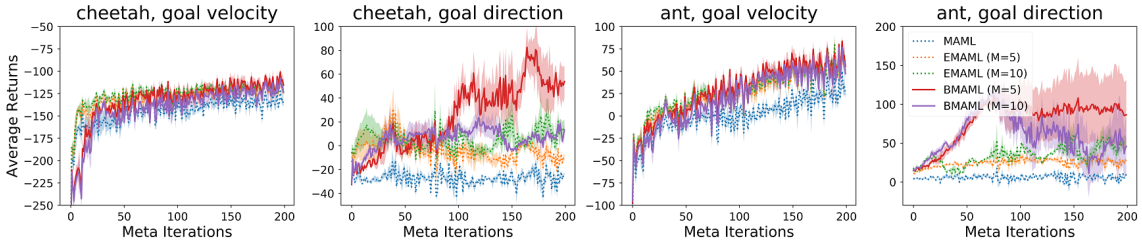


Figure 4: Locomotion comparison results of SVPG-Chaser and VPG-Reptile

For meta-update, MAML uses TRPO (Schulman et al., 2015) which is designed with a special purpose to apply for reinforcement learning and uses a rather expensive 2^{nd}-order optimization. However, the meta-update by the chaser loss is general-purpose and based on 1^{st}-order optimization[3]. Thus, for a fair comparison, we consider the following two experiment designs. First, in order to evaluate the performance of the inner updates using Bayesian fast adaptation, we compare the standard MAML, which uses vanilla policy gradient (REINFORCE, Williams (1992)) for inner-updates and TRPO for meta-updates, with the Bayesian fast adaptation with TRPO meta-update. We label the former as VPG-TRPO and the later as SVPG-TRPO. Second, we compare SVPG-Chaser with VPG-Reptile. Because, similarly to the chaser loss, Reptile (Nichol et al., 2018) performs 1^{st}-order gradient optimization based on the distance in the model parameter space, this provides us a fair baseline to evaluate the chaser loss in RL. The VPG-TRPO and VPG-Reptile are implemented with independent multiple agents. We tested the comparing methods for $M = [1, 5, 10]$. More details of the experimental setting is provided in Appendix C.3.

As shown in Fig. 3 and Fig. 4, we can see that overall, BMAML shows superior performance to EMAML. In particular, BMAML performs significantly and consistently better than EMAML in the case of using TRPO meta-updater. In addition, we can see that BMAML performs much better than EMAML for the goal direction tasks. We presume that this is because in the goal direction task, there is no goal velocity and thus a higher reward can always be obtained by searching for a better policy. This, therefore, demonstrates that BMAML can learn a better exploration policy than EMAML. In contrast, in the goal velocity task, exploration becomes less effective because it is not desired once a policy reaches the given goal velocity. This thus explains the results on the goal velocity task in which BMAML provides slightly better performance than EMAML. For some experiments, we also see that having more particles do not necessarily provides further improvements. As in the case of classification, we hypothesize that one of the reasons could be due to the instability of SVGD. In Appendix C.4, we also provide the results on 2D Navigation task, where we observe similar superiority of BMAML to EMAML.

6 Discussions

In this section, we discuss some of the issues underlying the design of the proposed method.

BMAML is tied to SVGD? In principle, it could actually be more generally applicable to any inference algorithm that can provide *differentiable samples*. Gradient-based MCMC methods like HMC (Neal et al., 2011) or SGLD (Welling & Teh, 2011) are such methods. We however chose SVGD specifically

[3]When considering the inner update together, TRPO, Chaser and Reptile are $3^{\text{rd}}/2^{\text{nd}}/1^{\text{st}}$-order, respectively.

for BMAML because jointly updating the particles altogether is more efficient for capturing the distribution quickly by a small number of update steps. In contrast, MCMC would require to wait for much more iterations until the chain mixes enough and a long backpropagation steps through the chain.

Parameter space v.s. prediction space? We defined the chaser loss by the distance in the model parameter space although it is also possible to define it in the prediction distance, i.e., by prediction error. We chose the parameter space because (1) we can save computation for the forward-pass for predictions, and (2) it empirically showed better performance for RL and similar performance for other tasks. The advantages of working in the parameter space is also discussed in Nichol et al. (2018).

Do the small number of SVGD steps converge to the posterior? In our small-data-big-network setting, a large area of a true task-posterior will be meaningless for other tasks. Thus, it is not desired to fully capture the task-posterior but instead we need to find an area which will be broadly useful for many tasks. This is the goal of hierarchical Bayes which our method approximate by finding such area and putting Θ_0 there. In theory, the task-posterior can be fully captured with infinite number of particles and update-steps, and thus dilute the initialization effect. In practice, the full coverage would, however, not be achievable (and not desired) because SVGD or MCMC would have difficulties in covering all areas of the complex multimodal task-posterior like that of a neural network.

7 Conclusion

Motivated by the hierarchical probabilistic modeling perspective to gradient-based meta-learning, we proposed a Bayesian gradient-based meta learning method. To do this, we combined the Stein Variational Gradient Descent with gradient-based meta learning in a probabilistic framework, and proposed the Bayesian Fast Adaptation and the Chaser loss for meta-update. As it remains a model-agnostic model, in experiments, we evaluated the method in various types of learning tasks including supervised learning, active learning, and reinforcement learning, and showed its superior performance in prediction accuracy, robustness to overfitting, and efficient exploration.

As a Bayesian ensemble method, along with its advantages, the proposed method also inherits the generic shortcomings of ensemble methods, particularly the space/time complexity proportional to the number of particles. Although we showed that our parameter sharing scheme is effective to mitigate this issue, it would still be interesting to improve the efficiency further in this direction. In addition, because the performance of SVGD can be sensitive to the parameters of the kernel function, incorporating the fast-adaptation of the kernel parameter into a part of meta-learning would also be an interesting future direction.

Acknowledgments

JY thanks SAP and Kakao Brain for their support. TK thanks NSERC, MILA and Kakao Brain for their support. YB thanks CIFAR, NSERC, IBM, Google, Facebook and Microsoft for their support. SA, Element AI Fellow, thanks Nicolas Chapados, Pedro Oliveira Pinheiro, Alexandre Lacoste, Negar Rostamzadeh for helpful discussions and feedback.

References

Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Anoop Korattikara Balan, Vivek Rathod, Kevin Murphy, and Max Welling. Bayesian dark knowledge. *CoRR*, abs/1506.04416, 2015. URL `http://arxiv.org/abs/1506.04416`.

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartłomiej Świątkowski, Bernhard Schölkopf, and Richard E Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle, 1990.

John B Biggs. The role of metalearning in study processes. *British journal of educational psychology*, 55(3):185–212, 1985.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Hal Daumé III. Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 135–142. AUAI Press, 2009.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1134–1141. IEEE, 2003.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Bayesian Deep Learning workshop, NIPS*, 2016.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Barbara Landau, Linda B Smith, and Susan S Jones. The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321, 1988.

Neil D Lawrence and John C Platt. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 65. ACM, 2004.

L.M. Le Cam. *Asymptotic methods in statistical decision theory*. Springer, 1986.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2378–2386, 2016.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

A. Nichol, J. Achiam, and J. Schulman. On First-Order Meta-Learning Algorithms. *ArXiv e-prints*, March 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Linda B Smith and Lauren K Slone. A developmental approach to machine learning? *Frontiers in psychology*, 8:2124, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.

Joshua Brett Tenenbaum. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Appendix A Supervised Learning

A.1 Regression

For regression, we used 10 tasks for each meta-batch and the meta-validation dataset $\mathcal{D}_\tau^{\text{val}}$ is set to have the same size of the meta-training dataset $\mathcal{D}_\tau^{\text{trn}}$ ($|\mathcal{D}_\tau^{\text{trn}}| = |\mathcal{D}_\tau^{\text{val}}| = K$). During training, the number of steps n for chaser is set to $n = 1$ and also the number of steps s for leader is set to $s = 1$. We used different step sizes α for computing chaser and leader, 0.01 and 0.001, respectively. This allows the leader to stay nearby the chaser but toward the target posterior and stabilized the training. The models were trained with using different size of training dataset $|\mathcal{T}|$, the number of tasks observable during training, and we trained the model over 10000 epochs for $|\mathcal{T}| = 100$ and 1000 epochs for $|\mathcal{T}| = 1000$. In Fig. 6, we show the qualitative results on randomly sampled sinusoid task and we used 5 update steps. The task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ decomposes into the train data likelihood and parameter prior as $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto p(\mathcal{D}_\tau^{\text{trn}}|\theta_\tau)p(\theta_\tau)$ and this is formulated as:

$$p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto \prod_{(x,y)\in\mathcal{D}_\tau^{\text{trn}}} \mathcal{N}(y|f_W(x), \gamma^{-1}) \prod_{w\in W} \mathcal{N}(w|0, \lambda^{-1})\text{Gamma}(\gamma|a, b)\text{Gamma}(\lambda|a', b')$$

where θ_τ consists of network parameters W and scaling parameters γ, λ. In all experiments, we set Gamma distribution hyper-parameters as $a = 2.0, b = 0.2$ and $a' = 2.0, b' = 2.0$. During meta-update with chaser-loss, we used Adam optimizer (Kingma & Ba, 2014) with learning rate $\beta = 0.001$.

A.2 Classification

All models and experiments on the *mini*Imagenet classification task are trained with using the same network architecture and 16 tasks are used for each meta-batch during training. Each task is defined by randomly selected 5 classes with one instance of each class to adapt the model and it is evaluated on unseen instances within the selected 5 classes. We used the meta-validation dataset $\mathcal{D}_\tau^{\text{val}}$ containing one example per each class for the 5-way 1-shot setting. This reduced the computational cost and also improved the performance of all models. During training, the number of steps for chaser and leader both are set to 1 ($n = s = 1$). The chaser and leader used step size $\alpha = 0.01$ and $\alpha = 0.005$, respectively. The meta-update was done by using Adam optimizer ($\beta = 0.0005$). The models were trained with using different size of training dataset $|\mathcal{T}|$ and we trained the model with $|\mathcal{T}| = 800000$ and 1 epoch. With $|\mathcal{T}| = 10000$, the model was trained over 40 epochs. The task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ for classification is slightly different to the regression task due to using softmax for the data likelihood.

$$p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto \prod_{(x,y)\in\mathcal{D}_\tau^{\text{trn}}} p(y|f_W(x)) \prod_{w\in W} \mathcal{N}(w|0, \lambda^{-1})\text{Gamma}(\lambda|a, b)$$

where the hyper-parameters for Gamma distribution were set as $a = 2.0, b = 0.2$ or $a = 1.0, b = 0.1$ in our experiments.

Appendix B Active Learning

Algorithm 4 Active Learning on Image Classification

1: Sample a few-shot labeled dataset $\mathcal{D}_\tau$ and a pool of unlabeled dataset $\mathcal{X}_\tau$ of task τ
2: Initialize $\Theta_\tau \leftarrow \Theta_0^*$
3: Update $\Theta_\tau \leftarrow \text{SVGD}_n(\Theta_\tau; \mathcal{D}_\tau, \alpha)$
4: **while** $\mathcal{X}_\tau$ is not empty **do**
5: Select $x' \leftarrow \arg\max_{x\in\mathcal{X}_\tau} \mathbb{H}[y|x, \Theta_\tau]$ and remove x' from $\mathcal{X}_\tau$
6: Request y' of x'
7: Update $\mathcal{D}_\tau \leftarrow \mathcal{D}_\tau \cup \{(x', y')\}$
8: Update $\Theta_\tau \leftarrow \text{SVGD}_n(\Theta_\tau; \mathcal{D}_\tau, \alpha)$
9: **end while**

Appendix C Reinforcement Learning

C.1 Locomotion

The locomotion experiments require two simulated robots, a planar cheetah and 3D quadruped ones (called as ant), and two individual goals, to run in a particular direction or at a particular velocity. For the ant goal velocity, a positive bonus reward at each timestep is added to prevent the ant from ending the episode. In those experiments, the timestep in each episodes is 200, the number of episode per each inner update, K is 10 except the ant goal direction task, in which 40 episodes for each inner update is used, because of task complexity. The number of tasks per each meta update is 20, and the models are trained for up to 200 meta iterations.

C.2 Used Methods

We evaluate our proposed method on two cases, SVPG-TRPO vs VPG-TRPO and SVPG-Chaser vs VPG-Reptile. We describe the methods in this subsection, except VPG-TRPO, because this is MAML when $M = 1$.

C.2.1 SVPG-TRPO

This method is to use SVPG as inner update and TRPO as meta update, which is following to a simple Bayesian meta-learning manner. In K-shot reinforcement learning on this method, K episodes from each policy particles and task τ (total number of episode is KM), and the corresponding rewards are used for task learning on the task. This method gets the above data ($\mathcal{D}_\tau^{trn}$) from Θ_0, and updates the parameters Θ_0 to Θ_τ^n with $\mathcal{D}_\tau^{trn}$ and SVPG. After getting few-shot learned parameters (Θ_τ^n), our method get new data ($\mathcal{D}_\tau^{val}$) from Θ_τ^n. After all the materials for meta learning have been collected, our method finds the meta loss with few-shot learned particles and task-validation set, $\mathcal{D}_\tau^{val}$. On meta-learning, TRPO (Schulman et al., 2015) is used as MAML (Finn et al., 2017) for validating inner Bayesian learning performance. The overall algorithm is described in Algorithm 5.

Algorithm 5 Simple Bayesian Meta-Learning for Reinforcement Learning

Initialize Θ_0
for $t = 0, \ldots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{SVPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
    Sample trajectories $\mathcal{D}_\tau^{\text{val}}$ with Θ_τ^n in τ
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_\tau^{meta}(\Theta_\tau^n; \mathcal{D}_\tau^{\text{val}})$
end for

C.2.2 SVPG-Chaser

This method is to use SVPG as inner-update and chaser loss for meta-update to maintain uncertainty. Different to supervised learning, this method updates leader particles just with $\mathcal{D}_\tau^{\text{val}}$ in policy gradient update manner. Same chaser loss to supervised learning ones is consistently applied to evaluating the chaser loss extensibility. The chaser loss in RL changes the meta update from a policy gradient problem to a problem similar to imitation learning. Unlike conventional imitation learning with given expert agent, this method keeps the uncertainty provided by the SVPG by following one more updated agent, and ultimately ensures that the chaser agent is close to the expert. Compared to Algorithm 5, this method adds updating the leader and changes the method of meta update like Algorithm 6.

C.2.3 VPG-Reptile

Reptile (Nichol et al., 2018) solved meta learning problem by using only 1^{st}-order derivatives, and used meta update in parameter space. We design a version of meta loss similar to Reptile to verify

Algorithm 6 Bayesian Meta-Learning for Reinforcement Learning with Chaser Loss

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{SVPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
    Sample trajectories $\mathcal{D}_\tau^{\text{val}}$ with Θ_τ^n in τ
    Compute leader $\Theta_\tau^{n+s} = \text{SVPG}(\Theta_\tau^n; \mathcal{D}_\tau^{\text{val}})$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} D(\Theta_\tau^n || \text{stopgrad}(\Theta_\tau^{n+s}))$
end for

the performance of chaser loss in RL problem. This method computes the chaser Θ_τ^n using $\mathcal{D}_\tau^{\text{trn}}$ and then calculates the euclidean distance between this parameter and the global parameter as a meta loss (to prevent the gradient from being calculated through the chaser parameter to maintain the 1^{st}-order derivatives). The overall algorithm is described in Algorithm 7.

Algorithm 7 VPG-Reptile

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{VPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} D(\Theta_0 || \text{stopgrad}(\Theta_\tau^n))$
end for

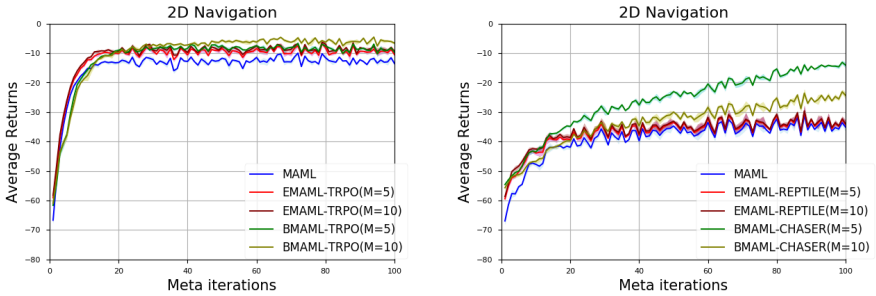
C.3 Experimental Details

Inner update learning rate and the number of inner update are set as 0.1 and 1 for all experiments, which are locomotion (ant/cheetah goal velocity and ant/cheetah goal direction) and 2D-Navigation experiments. Meta update learning rate is set as 0.1 for ant goal direction and 0.01 for other experiments. η, the parameter that controls the strength of exploration in SVPG is set as 0.1 for ant velocity experiment with SVPG-Chaser, ant goal direction and 2D Navigation with SVPG-Chaser, and 1.0 for other experiments. Each plots are based on an mean and a standard deviation from three different random seed. The subsumed results are plotted with the average reward of maximum task rewards in during of particles.

C.4 Additional Experiment Results

C.4.1 2D Navigation

We also compare the models on the toy experiment designed in previous work (Finn et al., 2017), 2D Navigation. This experiment is a set of tasks where agent must move to different goal positions in 2D, which is randomly set for each task within a unit square. The observation is the current 2D position, and actions correspond to velocity clipped to be in the range [-0.1, 0.1]. The reward is the negative squared distance between the goal and the current position, and episodes terminate when the agent is within 0.01 of the goal or at the timestep = 100. We used 10 episodes per each inner update ($K = 10$) and 20 tasks per each meta update. The models are trained for up to 100 meta iterations. The policy network has two hidden layers each with 100 ReLU units. We tested the number of particles for $M \in \{1, 5, 10\}$ with $M = 1$ only for non-ensembled MAML. Same to above locomotion experiments, we compare the models as SVPG-TRPO vs VPG-TRPO and SVPG-Chaser vs VPG-Reptile. As shown in Fig. 5, BMAML showed better performance than EMAML on both comparisons.



(a) **(b)**

Figure 5: 2D Navigation: (a) and (b) are results of SVPG-TRPO vs VPG-TRPO and SVPG-Chaser vs VPG-Reptile with three different random seed, respectively.

(a) MAML (M=1) **(b)** EMAML (M=20) **(c)** BMAML (M=20)

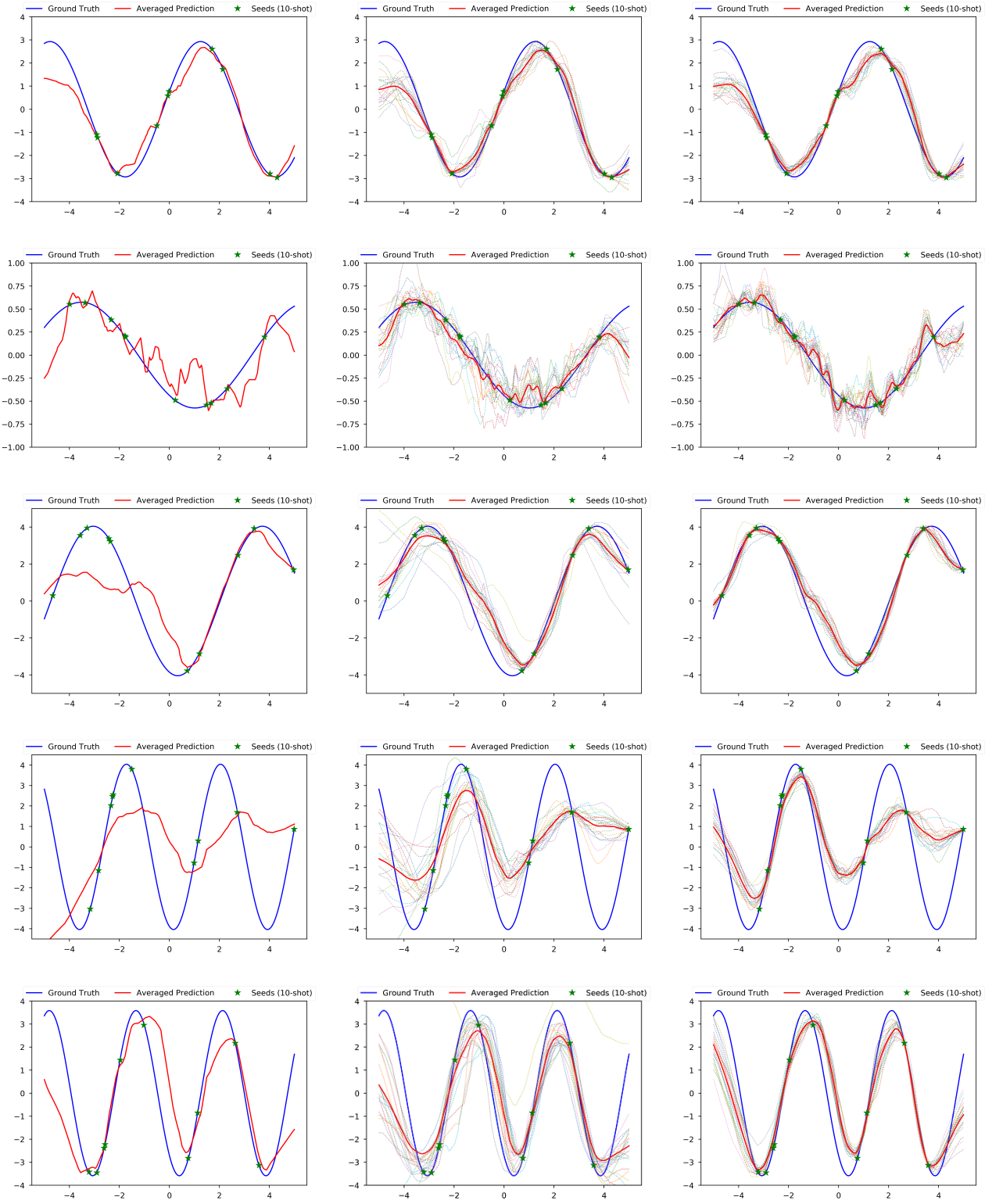


Figure 6: Regression qualitative examples: randomly sampled tasks with 10 examples (10-shot) and 10 gradient updates for adaptation