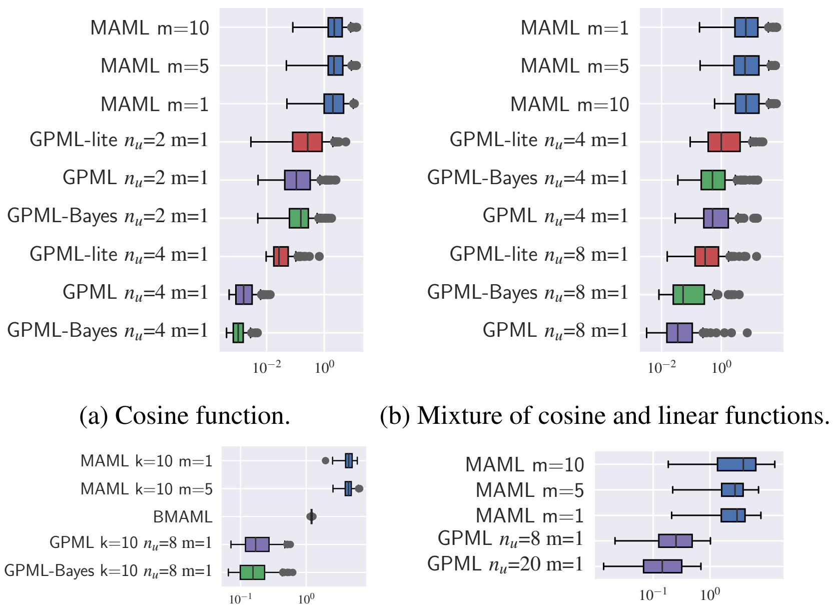
Learning to Learn with Gaussian Processes (Supplementary Material)

Quoc Phong Nguyen[1] **Bryan Kian Hsiang Low**[1] **Patrick Jaillet**[2]

[1]Department of Computer Science, National University of Singapore, Republic of Singapore
[2]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, USA

A META TESTING WITHOUT GRADIENT DESCENT (GD) ADAPTATION

There has been a growing interest in MAML variants that do not require the second-order gradient information such as first-order MAML Finn et al. [2017], Reptile Nichol et al. [2018] that simply ignore the second-order gradients; and implicit MAML (iMAML) Rajeswaran et al. [2019] that includes a proximal regularization to the loss function and employs the implicit differentiation to optimize the common initializer without the second-order gradient. The iMAML is an interesting solution that can be integrated into GPML as a future work. In this section, we focus on the meta testing to demonstrate the effectiveness of our methods in learning a task-dependent initializer without the GD adaptation.



(a) Cosine function. (b) Mixture of cosine and linear functions.

(c) Noisy sine function. (d) Light sensor dataset.

Figure 1: MSE of $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ for 5-shot regression (a,b) and 10-shot regression (c,d).

Fig. 1 shows MSE of $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ (i.e., meta testing without GD adaptation) in 4 experiments with cosine functions, a mixture of cosine and linear functions, noisy sine functions, and the light sensor dataset. We can observe that our methods outperform MAML in all these experiments and outperform BMAML in the noisy sine function experiment. GPML-lite does not perform as well as GPML and GPML-Bayes due to the inconsistent evaluation of the task kernel in the GP posterior belief. As the number of inducing tasks increases to capture the diversity of tasks, the performance of our methods improves.

B DISCUSSIONS ON THE PERFORMANCE AND INTERPRETABILITY OF GPML-BAYES

B.1 COSINE FUNCTION

The problem in experiments with cosine functions (Sec. 4.1) is 5-shot regression problem that learns a random cosine function, i.e., $y_t(x) = a\cos(x+b)$ where a and b are uniformly sampled in $[0.1, 5.0]$ and $[0, \pi]$, respectively Finn et al. [2017]. The input x is uniformly sampled in $\mathcal{D}_x = [-5, 5]$.

In this section, we explain the performance of GPML-Bayes in the above experiments with respect to the number of inducing tasks by visualizing the correlation between tasks through the distance between a task and inducing tasks. Since different parameters of the neural network have different GP hyperparameters, it is not easy to visualize the correlation between the initializers of different tasks via the task kernel. Thus, we use the MSE of a task w.r.t. another task as an estimate of the distance between the two tasks. In other words, the larger the MSE is, the further the distance between the two tasks is. In turn, it implies that the correlation between their initializers is probably small. Similar to sparse GP regression, the initializer of a task is correlated with that of an inducing task if their distance is small. In contrast, if the distances of a task to all inducing tasks are large, the initializer of the task is similar to the GP prior mean $\bar{\boldsymbol{\mu}}$. This implies that the inducing tasks should be representative of the task distribution if the number of inducing tasks is large.

Fig. 2 shows $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ (the GP prior mean) and $f(\mathbf{x}, \boldsymbol{\theta}^*_{t_u})$ for $t_u \in \mathcal{T}_u$ (inducing tasks). Surprisingly, despite having a better performance, when $n_u = 4$, $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ does not show a periodic pattern (of a cosine function) in Fig. 2c, unlike that when $n_u = 2$ in Fig. 2a. However, by plotting the distribution of MSE between a random task and its closest inducing task (Fig. 3a), we observe that when $n_u = 4$, MSE values are distributed closer to 0 than those when $n_u = 2$. It means that most tasks are close to some inducing tasks when $n_u = 4$. So, their initializers are correlated with those of the inducing tasks. Therefore, when $n_u = 4$, $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ is not necessarily a periodic function for GPML-Bayes to obtain a good performance, as long as their inducing tasks are periodic functions (Fig. 2d). When $n_u = 2$, the number of inducing tasks is not large enough to capture different cosine tasks in the task distribution. Thus, there exist some tasks having large distances to all inducing tasks (the right tail of the dashed line in Fig. 3a), i.e., the initializers of these tasks are not correlated with those of any inducing tasks. It means they are similar to the GP prior mean $\bar{\boldsymbol{\mu}}$, which makes $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ similar to a cosine function (Fig. 2a).

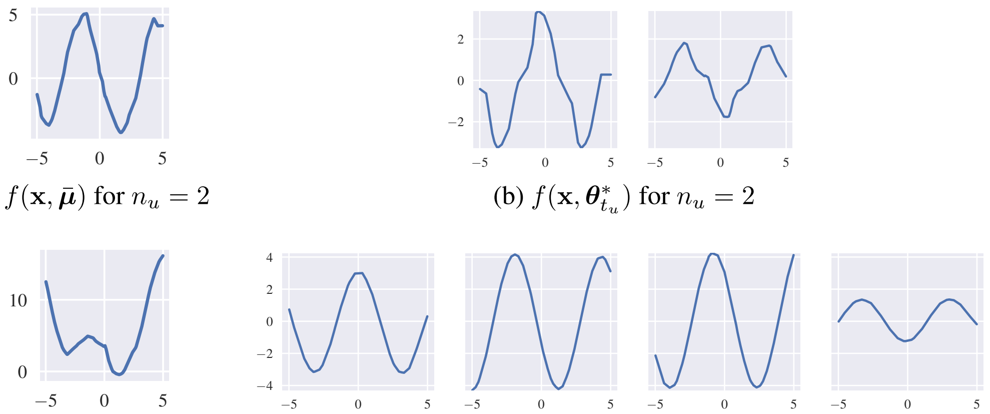
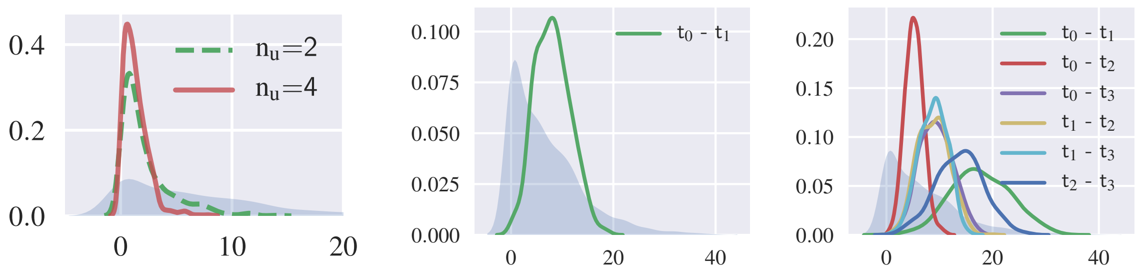


Figure 2: Plots of the GP prior mean and inducing tasks in the 5-shot regression on cosine tasks.

Figs. 3b and 3c demonstrate quantitatively the diversity of inducing tasks by showing the distribution of MSE between inducing tasks. To plot these distributions, MSE of pairs of inducing tasks is computed 100 times at different sets of 5 random inputs. The figure shows that the MSE values between these inducing tasks are relatively large compared with those between any random tasks in the task distribution (the shaded areas in Fig. 3). This observation agrees with Figs. 2b and 2d: the inducing tasks are cosine functions of different amplitudes and phases.

Furthermore, when $n_u = 2$, some tasks in the task distribution have initializers that are similar to the GP prior mean. We observe that the GP prior mean is a roughly cosine function of amplitude 5 (Fig. 2a) while the inducing tasks are cosine



(a) (b) $n_u = 2$ (c) $n_u = 4$

Figure 3: Plots of (a) distribution of MSE of a random task w.r.t. its closest inducing task, and (b-c) distributions of MSE between inducing tasks.

functions of smaller amplitudes (Fig. 2b). This is different from $n_u = 4$ where inducing tasks are capable of representing cosine functions of varying amplitudes and phases, i.e., the task distribution (Fig. 2d), so it is not necessary for the GP prior mean to be similar to a cosine function (Fig. 2c).

Fig. 4 shows $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ and $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$ trained with MAML ($m = 1, 5, 10$) and GPML-Bayes ($m = 1$ and $n_u = 2, 4$) for 5 random tasks in the task distribution. This is additional visualization of the predicted functions to those in Fig. 5. We can observe the superior predictive performance of GPML-Bayes with $n_u = 4$ for all 5 tasks of varying amplitudes and phases.

B.2 MIXTURE OF COSINE AND LINEAR FUNCTIONS

The problem in experiments with a mixture of cosine and linear functions (Section 4.1) is a 5-shot regression problem on a multimodal task distribution: a mixture of cosine and linear functions with equal probabilities. The cosine function is defined as above, while the linear function is defined as $y_t(x) = cx + d$ where $\mathcal{D}_x = [-5, 5]$, $c \in [-3, 3]$, and $d \in [-3, 3]$.

This section shows the behavior of GPML-Bayes in the above experiments. We have similar observations to those in the cosine function experiment (Sec. 4.1 and Appendix B.1) as below.

Although GPML-Bayes with $n_u = 8$ has a better performance than GPML-Bayes with $n_u = 4$, $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ trained with GPML-Bayes when $n_u = 8$ does not resemble any task in the task distribution (Fig. 5c). On the other hand, $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ trained with GPML-Bayes when $n_u = 4$ looks like a linear function (Fig. 5a). This phenomenon is because there are a number of tasks in the task distribution far from all inducing tasks when $n_u = 4$ (the right tail of the green line in Fig. 6a). For these tasks, their initializers are similar to the GP prior mean $\bar{\boldsymbol{\mu}}$, so the training makes $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ similar to a task in the task distribution (in this case, it is a linear function). On the other hand, when $n_u = 8$, all tasks are close to an inducing task (Fig. 6b), and all inducing tasks represent some tasks in the task distributions (i.e., linear and cosine functions in Fig. 5d). Thus, $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ can be different from all tasks in the task distribution, while GPML-Bayes still performs well.

Fig. 7 shows $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ and $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$ for 5 tasks in the task distribution. As MAML only learns a common initializer for all tasks, it is difficult to find 1 initializer that initializes well for both cosine and linear functions even when m increases to 10 as shown in Figs. 7a-c. On the other hand, the inducing tasks help GPML-Bayes to capture different tasks in the task distribution. Thus, GPML-Bayes shows a good performance, especially when $n_u = 8$ in Fig. 7e.

C NOISY SINE FUNCTION

In this section, tasks are random sine functions: $y_t(x) = A\sin(wx + b) + \epsilon$ where $A \in [0.1, 5.0]$ is the amplitude, $b \in [0.0, 2\pi]$ is the phase, $w \in [0.5, 2.0]$ is the frequency, and $\epsilon \sim \mathcal{N}(0, (0.01A)^2)$ is the noise. This experiment is adopted from Yoon et al. [2018]. The results are presented in Fig. 8a. The training and test sets are obtained by uniformly sampling $x \in [-5.0, 5.0]$. This task distribution is more challenging than the cosine task distribution in Sec. 4.1 due to the varying frequency and the noise. The regression model is a neural network of 3 hidden layers, each of which consists of 40 hidden neurons with ReLU activation functions. The BMAML implementation is available at the website of its authors: `https://github.com/jsikyoon/bmaml` Yoon et al. [2018]. It is noted that we use the MSE as the loss function for MAML and our methods to be consistent with other experiments in this paper, unlike BMAML maximizing the likelihood.

There are 25 random tasks drawn from the task distribution at each optimization iteration. The number of data points k is set to 10. The number of particles in BMAML is 10.

We observe that this is a challenging few-shot regression problem as both MAML (with $m = 1, 5$) and BMAML cannot achieve MSE values less than 1.0 (Fig. 8a). When we plot $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^{\circ}))$ of MAML, it cannot learn a periodic function, but it learns a constant function of 0 even with $m = 5$ GD updates (Figs. 9a-b). On the other hand, GPML-Bayes is able to achieve small MSE values (Fig. 8a). Fig. 9c shows that GPML-Bayes fits the sine functions with different amplitudes, phases, and frequencies.

D REGRESSION WITH HIGH-DIMENSIONAL INPUTS

Since our methods view a task as a vector of outputs indexed by their corresponding inputs (Sec. 3.2), our methods are scalable to problems with high-dimensional inputs. We perform the 10-shot regression on the MNIST dataset of 28×28 gray-scale images of written digits (784 dimensions). The output is a polynomial: $ax^3 + bx^2 + cx + d$ where $a \in [0, 0.01]$, $b \in [0, 0.1]$, $c \in [0, 0.5]$, $d \in [0, 1]$, and $x \in [0, 1, \ldots, 9]$ is the digit in an image (x is not provided to the algorithms). The meta training set includes 60000 images and meta test set includes 10000 images. In Fig. 8b, GPML and GPML-Bayes with only 6 inducing inputs outperforms MAML. It shows that GPML with even a small number of inducing tasks outperforms MAML in this high-dimensional problem.

E NEURAL NETWORK MODELS

In the cosine task (Sec. 4.1), the mixture of cosine and linear tasks (Sec. 4.1), and the light sensor dataset experiments (Sec. 4.2), the neural network models consist of 2 hidden layers of size 40 each with *rectified linear unit* (ReLU) activation functions. The output layer consists of 1 neuron with a linear activation function (since the output dimension in these experiments is 1).

In the noisy sine function experiment (Appendix C), the neural network consists of 3 hidden layers of size 40 each with ReLU activation functions. The output layer consists of 1 neuron with a linear activation function.

In the MNIST experiment (Appendix D), the neural network consists of 3 hidden layers of size 40, 40, and 20 with ReLU activation functions. The output layer consists of 1 neuron with a linear activation function.

In the experiment with the sushi dataset (Sec. 4.2), the neural network consists of 2 hidden layers of size 40 and 20 with ReLU activation functions. To constrain the output of the model to the 5 rating scores, the output layer consists of 5 neurons with the softmax activation function. Unlike previous experiments that use MSE loss, we use the cross-entropy loss in this experiment. To evaluate the task kernel, we convert the output vector of 5 probabilities from the softmax layer to a numerical value by taking the expectation of the output score, i.e.,

$$k(t, t_u|\mathcal{X}_t) \triangleq \sigma_s^2 \exp\left(-\sum_{\mathbf{x}\in\mathcal{X}_t}\left(y_t(\mathbf{x}) - \bar{y}_{\theta^*_{t_u}}(\mathbf{x})\right)^2 / \left(2|\mathcal{X}_t|\, l_{\mathbf{x}}^2\right)\right)$$

where $\bar{y}_{\theta^*_{t_u}}(\mathbf{x}) \triangleq \sum_{i=1}^{5} i \times f\left(\mathbf{x}, \theta^*_{t_u}\right)[i]$, and $f\left(\mathbf{x}, \theta^*_{t_u}\right)[i]$ is the probability of the i-th rating score, i.e., the probability vector output of the softmax layer.

It is noted that this experiment is different from image recognition experiments as the output (ratings) is an ordinal variable which has a meaningful distance between its values. Although our preliminary results show that our method works with nominal/categorical outputs (e.g., image recognition), we believe the result can be significantly improved with a new design of the task kernel that can capture the distance between tasks. Hence, it is left as a future research direction.

F INTERPRETABILITY IN EXPERIEMENTS WITH SUSHI DATASET

To highlight the interpretability of GPML with the inducing tasks vs. MAML, we plot the proportion of different sushi groups that have the highest (i.e., 5) and the lowest (i.e., 1) ratings for 5 inducing tasks learned using GPML and the initializer learned using MAML in Fig. 10. The proportion is estimated from the prediction of these inducing tasks and MAML in 500 random tasks in the task distribution. From the initializer learned using MAML (Fig. 10a), we can interpret that aomono, akami, and shrimp/crab are the 3 highest-rated groups. However, there is no information about the lowest-rated groups. On the other hand, we can observe some groups with the lowest rating from the 5 inducing tasks of GPML (Figs. 10b-f). In particular, anomono is rated lowest in Fig. 10b and highest in Figs. 10e and 10f. It is difficult to capture this contradiction among different tasks using only 1 common initializer in MAML. Furthermore, since a task (ratings from a person) is close to an inducing task (Fig. 8c), we can interpret that there are some people who like some sushi in the akami group while disliking some sushi in the clam/shell group from Fig. 10c.

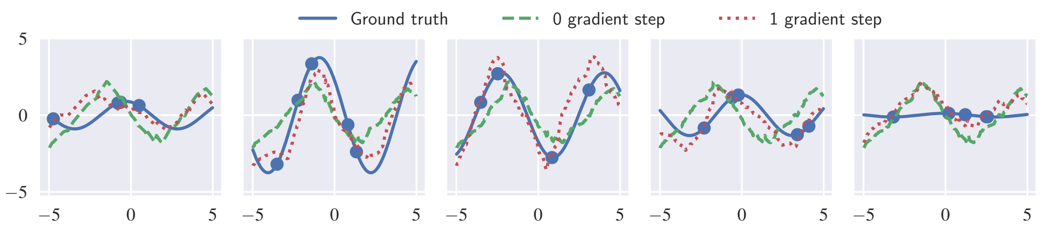
References

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, pages 1126–1135, 2017.

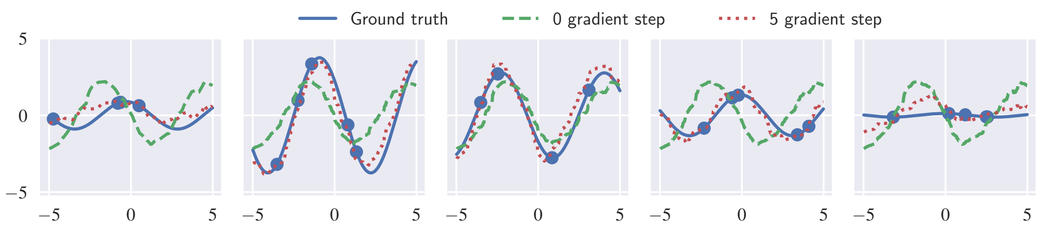
A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Proc. NeurIPS*, pages 113–124, 2019.

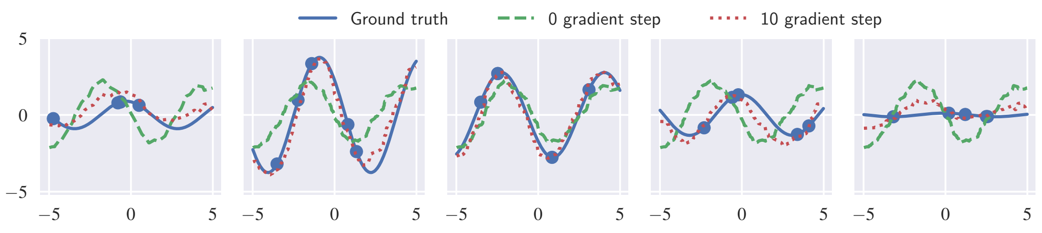
J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *Proc. NeurIPS*, pages 7332–7342, 2018.



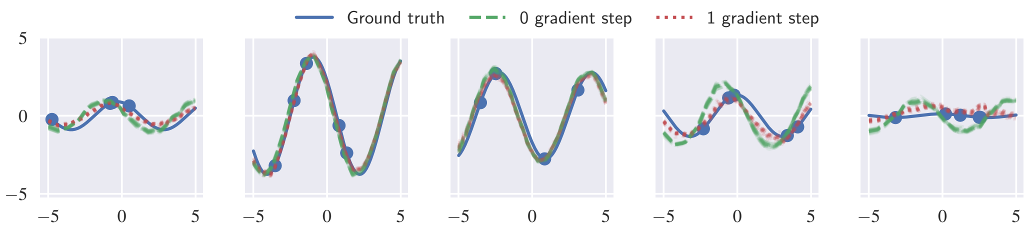
(a) MAML $m = 1$



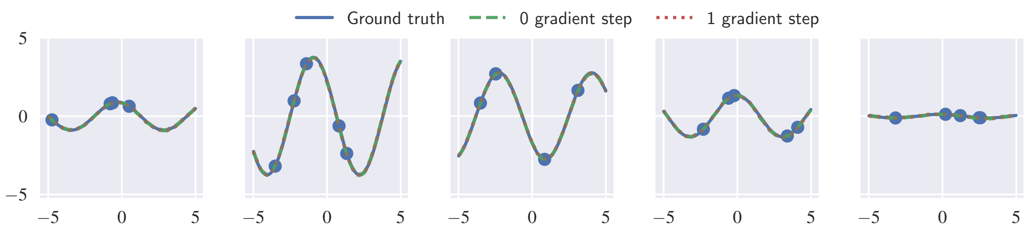
(b) MAML $m = 5$



(c) MAML $m = 10$

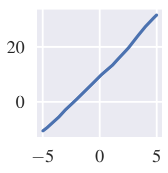


(d) GPML-Bayes $m = 1$ $n_u = 2$

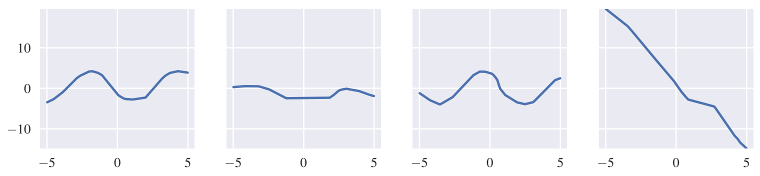


(e) GPML-Bayes $m = 1$ $n_u = 4$

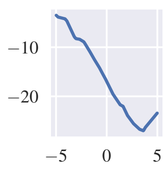
Figure 4: Cosine function: Plots of $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ as dashed green lines and $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$ as dotted red lines. Training data are plotted as blue dots.



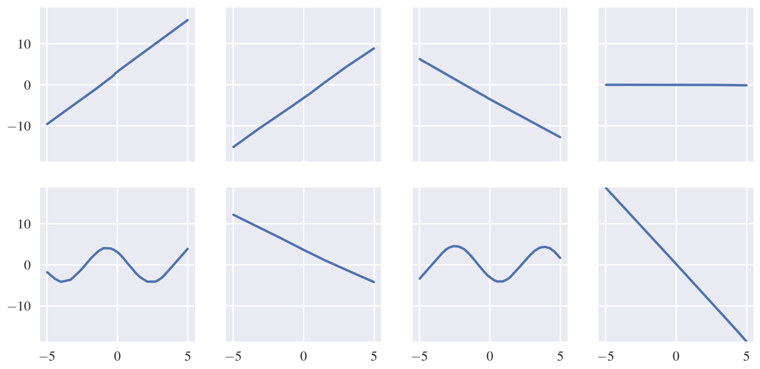
(a) $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ for $n_u = 4$



(b) $f(\mathbf{x}, \boldsymbol{\theta}^*_{t_u})$ for $n_u = 4$

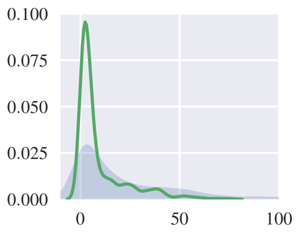


(c) $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ for $n_u = 8$

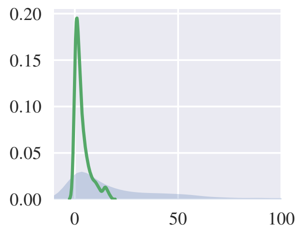


(d) $f(\mathbf{x}, \boldsymbol{\theta}^*_{t_u})$ for $n_u = 8$

Figure 5: Mixture of cosine and linear functions: Plot of GP prior mean $f(\mathbf{x}, \bar{\boldsymbol{\mu}})$ and the inducing tasks $f(\mathbf{x}, \boldsymbol{\theta}^*_{t_u})$ trained with GPML-Bayes for 5-shot regression.

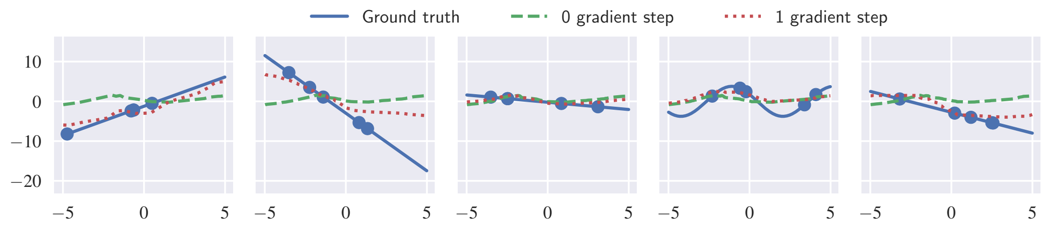


(a) $n_u = 4$

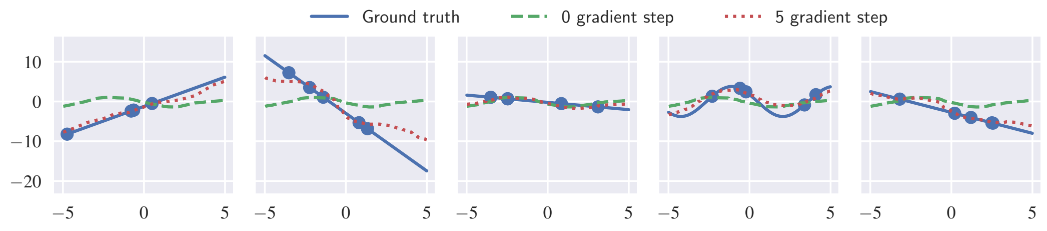


(b) $n_u = 8$

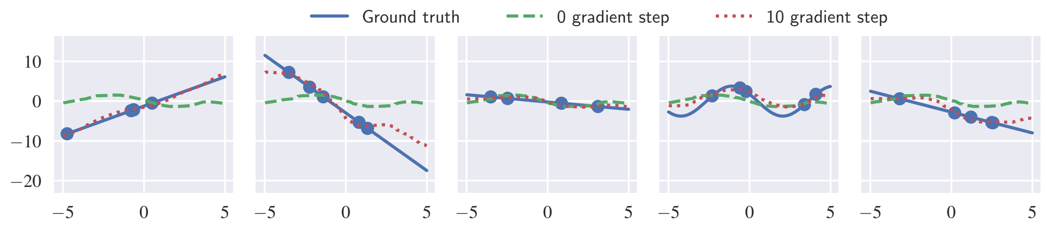
Figure 6: Mixture of cosine and linear functions: the distribution of MSE of a random task w.r.t. its closest inducing task. The shaded area shows the distribution of MSE between random tasks in the task distribution.



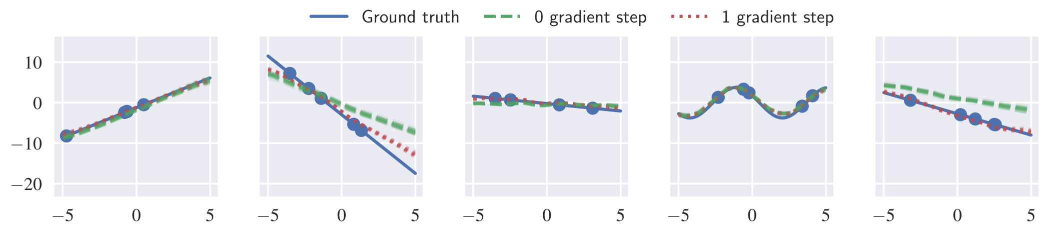
(a) MAML $m = 1$



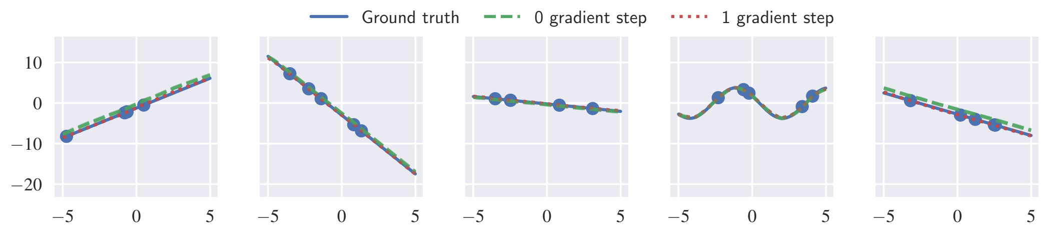
(b) MAML $m = 5$



(c) MAML $m = 10$

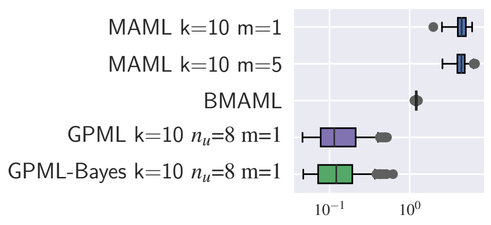


(d) GPML-Bayes $m = 1$ $n_u = 4$

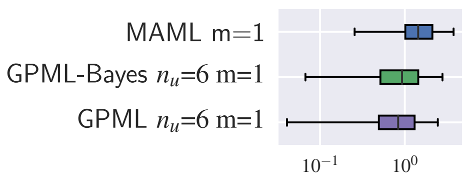


(e) GPML-Bayes $m = 1$ $n_u = 8$

Figure 7: Mixture of cosine and linear functions: Plots of $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ as dashed green lines and $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$ as dotted red lines. Training data are plotted as blue dots.

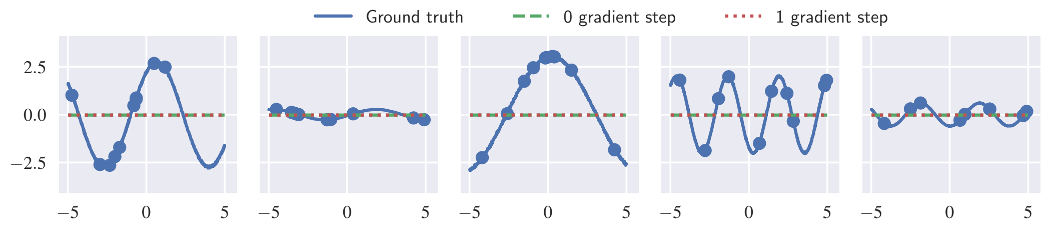


(a) Noisy sine task.

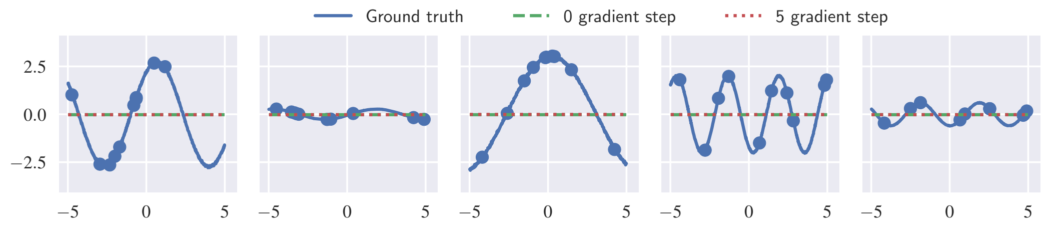


(b) MNIST.

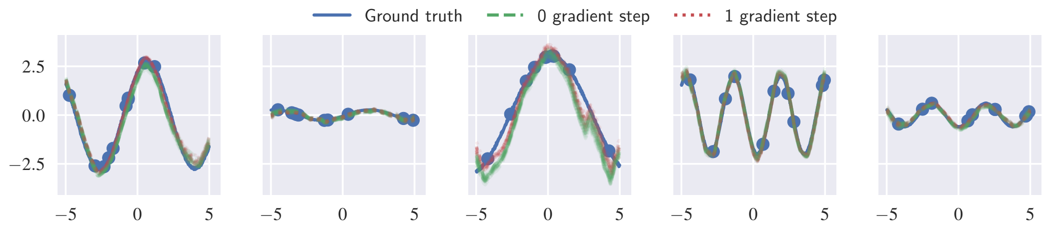
Figure 8: Boxplots of the MSE of $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$.



(a) MAML $m = 1$



(b) MAML $m = 5$



(c) GPML-Bayes $m = 1$ $n_u = 8$

Figure 9: Noisy sine function: Plots of $f(\mathbf{x}, \boldsymbol{\theta}_t^\circ)$ as dashed green lines and $f(\mathbf{x}, g_t^{(m)}(\boldsymbol{\theta}_t^\circ))$ as dotted red lines. Training data are plotted as blue dots.

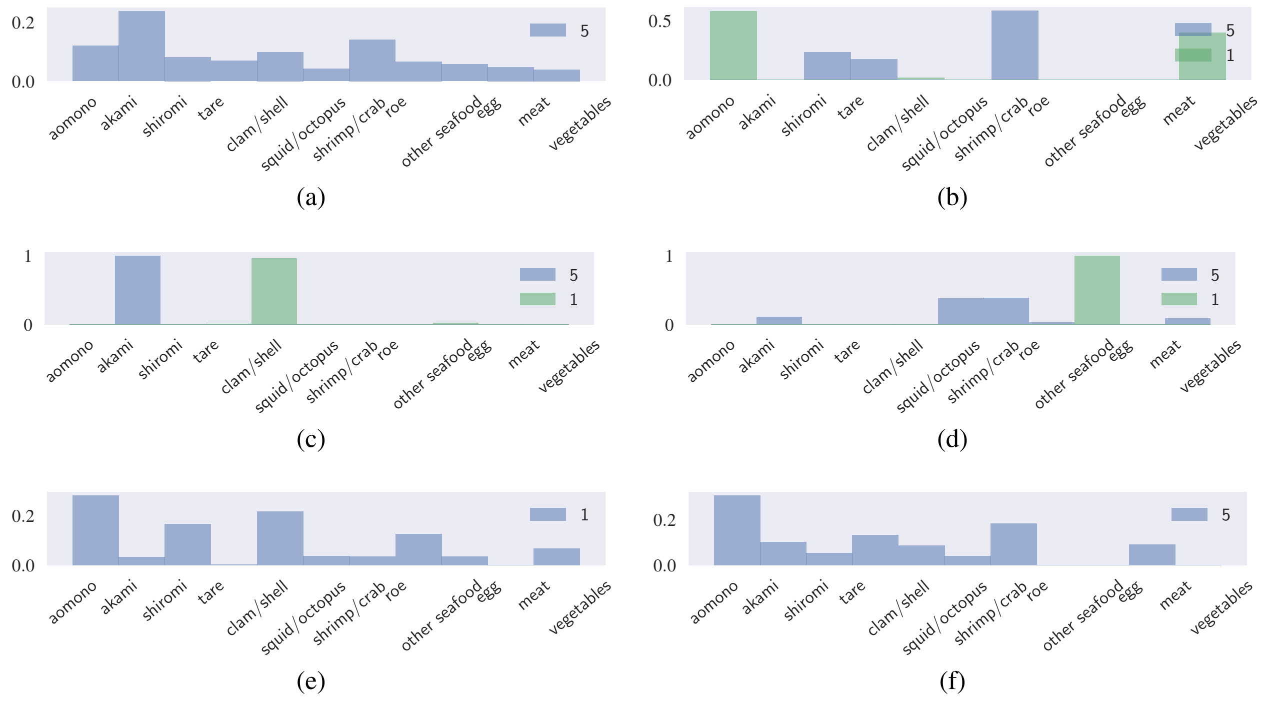


Figure 10: Plots of the predicted ratings over different sushi groups given (a) the initializer learned using MAML; and (b-f) 5 different inducing tasks ($f(\mathbf{x}, \boldsymbol{\theta}^*_{t_u})$) learned using GPML.