# A Computing Equation (8) in Practice

With dataset $\mathcal{D}$, the density ratio in $f$-divergence becomes $\frac{p(\theta|D)}{q_\phi(\theta)} = \frac{p(D|\theta)p(\theta)}{q_\phi(\theta)p(D)}$. We estimate $p(D)$ through importance sampling and MC approximation: $p(D) = E_{\theta\sim p(\theta)}[p(D|\theta)] = E_{\theta\sim q_\phi(\theta)}[\frac{p(D|\theta)p(\theta)}{q_\phi(\theta)}] \approx \frac{1}{K}\sum_{k=1}^K \frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)}$ where $\theta_k \sim q_\phi(\theta)$. After doing this, the density ratio becomes $\frac{p(\theta_k|D)}{q_\phi(\theta_k)} = \frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)} \Big/ \frac{1}{K}\sum_{k=1}^K \frac{p(D|\theta_k)p(\theta_k)}{q_\phi(\theta_k)}$ which can be regarded as a self-normalized estimator, similar to the normalization importance weight in Li and Turner (2016). A self-normalized estimator generally helps stabilize the training especially at the beginning. We use Eq.(8) with this estimator and stochastic approximation of gradients for all experiments except for the mixture of Gaussians task where we can directly compute $p(\theta)/q_\phi(\theta)$.

# B Effect of Hyperparameter $B$

Similar to other MAML-based algorithms (Finn et al., 2017, 2018; Kim et al., 2018), the cost of our method increases as hyperparameter $B$ increases and the value of $B$ could potentially affect the results. As in prior work, we treat $B$ as a hyperparameter and tune it for each task. Empirically, we found setting $B = 1$ is enough for most tasks we considered in the experimental section. For example, we also tried $B = 2, 5$ for meta-$\alpha$ in Section 4.2 and found that they gave similar values of learned $\alpha$ as $B = 1$ ($\alpha = 0.10, 0.13, 0.16$ for $B = 1, 2, 5$ respectively). Setting $B$ larger will be costly and even cause gradients to be problematic due to requiring higher-order derivatives. We may combine our methods with recent techniques in meta-learning (Flennerhag et al., 2019, 2020; Rajeswaran et al., 2019) to allow large $B$, which is an interesting future work.

# C Additional Experimental Results and Setting Details

## C.1 Task Distribution $p(\mathcal{T})$

When the number of training tasks is finite (which is often the case in practice) such as image generation with MNIST (Section 4.3) and recommender system with MovieLens (Section 4.4), the task distribution $p(\mathcal{T})$ is defined as a uniform distribution over all training tasks for both meta-$D$ (Algorithm 1) and meta-$D\&\phi$ (Algorithm 2). When the number of training tasks is infinite such as Gaussian mixture approximation (Section 4.1) and sinusoid regression (Section 4.2), we use a uniform distribution over all training tasks as $p(\mathcal{T})$ for meta-$D\&\phi$ but a uniform distribution over a subsampled set of training tasks as $p(\mathcal{T})$ for meta-$D$ (the set size is 10 and 20 for Gaussian mixture approximation and sinusoid regression respectively). This is to avoid storing too many models since meta-$D$ allows each task $T_i$ has its own model $\phi_i$.

## C.2 Parameterization of $f$-Divergence in Practice

Based on the Proposition 1 and 2, we can parameterize $f$-divergence by parameterizing $g(t) = t^2 f''(t) = \exp(h_\eta(t))$ where $h_\eta$ is a neural network with parameter $\eta$. However, this way of parameterization makes it hard to learn the divergences whose $g(t)$ is very small when $t$ is small (because $h(t)$ has to output negative numbers with large absolute values), such as Renyi divergence with $\alpha \approx 0$. These kinds of divergences behave like approximating the expectation in Eq.(8) only with $\theta$ whose $p(\theta)/q_\phi(\theta)$ is large, which is important for modeling bimodal and heteroscedastic distributions (Depeweg et al., 2017).

To alleviate this issue, we can instead parameterize $f''(t) = \exp(h_\eta(t))$, then $g(t)$ in Eq.(8) becomes $t^2\exp(h_\eta(t))$. It is easy to see that this parameterization solves the above issue due to $t^2$ which becomes small when $t$ is small. However, it is hard to learn the divergences that put similar weights to MC samples (e.g. standard $\mathrm{KL}(q\|p)$, which gives the equal weights). These two ways of parameterization are statistically equivalent but have different inductive bias. Parameterizing $f''$ tends to learn a divergence that puts different weights to MC samples according to $p(\theta)/q_\phi(\theta)$ (due to $t^2$). On the other hand, parameterizing $g(t)$ tends to give relatively similar weights to MC samples. In the experiments, we parameterize $g(t)$ when the learned $\alpha$ from meta-$\alpha$ is close to 1 (Section 4.1 and 4.4) and parameterize $f''(t)$ when the learned $\alpha$ is close to 0 (Section 4.2 and 4.3). We found this strategy works well in practice.

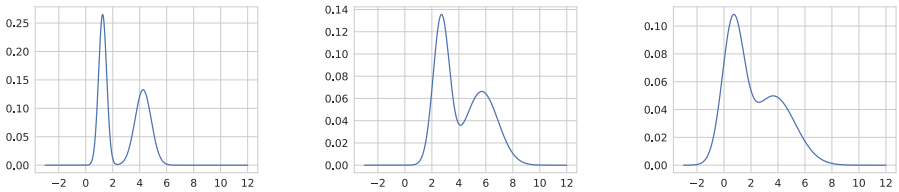


Figure 5: Three examples of mixture of Gaussians. Each task includes approximating a mixture of Gaussians by a Gaussian distribution.

### C.3 Model Architecture for $f$-divergence

On all experiments, we parameterize $h_\eta(t)$ in $f$-divergence by a neural network with 2 hidden layers with 100 hidden units and RELU nonlinearilities. In practice, we find that pretraining $h_\eta(t)$ to be the standard KL divergence can stabilize the training at the beginning. We initialize $h_\eta(t)$ in this way for all experiments.

### C.4 Approximate Mixture of Gaussians

In this experiment, each task is to approximate a mixture of Gaussians by a Gaussian distribution. We give examples of the mixture of Gaussians in Figure 5. The expectation in Eq.(3) and (8) is computed by MC approximation with 1000 particles. Note that $p(\theta)$ is computable, since we know the parameters of $p$.

#### C.4.1 TV Distance

TV is a common distance measure for probability distributions. It is defined as

$$\mathrm{TV}(p, q) = \sup_x |p(x) - q(x)| = \frac{1}{2} \int |p(x) - q(x)| \, dx.$$

For $\alpha \in (0, 1]$, TV is related to $\alpha$-divergence by $\alpha \, \mathrm{TV}^2 \leq 2 \cdot D_\alpha(p \| q)$ (Gilardoni, 2010).

Note that although TV belongs to a more general $f$-divergence family by setting $f_{\mathrm{tv}}(t) = |t - 1| / 2$, it does not belong to the $f$-divergence we defined in the paper. Since $f_{\mathrm{TV}}(t)$ is not twice-differentiable. Therefore, we can use TV as an example to test the performance of our methods when meta-loss is beyond $\alpha$- and $f$-divergence.

#### C.4.2 Bayesian Optimization

We used a standard setup of BO, following Snoek et al. (2012). To ensure fair comparisons, we implemented BO through a public and stable library [2]. We set the search region for BO to be $\alpha \in [0, 3]$. The acquisition function is the upper confidence bound with kappa 0.1. We used the same data for training meta-$D$ for BO. Specifically, the objective function that BO minimizes is the meta-loss ($D_{0.5}$ or TV). Every time BO selects an $\alpha$, we train 10 models with that $\alpha$-divergence on the training sets of 10 training tasks respectively and get the mean of log-likelihood on the test sets of the 10 training tasks. Each time the model is trained for 2000 iterations. It is possible to choose the best $\alpha$ for each test task by BO, i.e. every time we have a new test task we run BO to select an $\alpha$ for this task. However, by doing this, we are not able to extract any common knowledge from the previous tasks and running BO for each task could be very expensive. We did not include this baseline because it does not satisfy the meta-learning setting and the cost will be much higher than our methods.

#### C.4.3 Analytical Expression of $h_f(t)$

When $f$-divergence is $D_{0.5}$, the $f$ function is $f(t) = \frac{t^{0.5}}{-0.5^2}$. Then we can write out the corresponding $h_f(t)$ as

$$h_f(t) = \log g_f(t) = \log f''(t) + 2 \log t = 0.5 \log t.$$

[2] https://github.com/fmfn/BayesianOptimization.

Because the definition of $f$-divergence is invariant to constant scaling of the function $f$, i.e. $f$ and $af$ define the same divergence for $\forall a > 0$, we consider the corresponding $h_f(t)$ for $af$ which is

$$h_f(t) = 0.5 \log t + \log a.$$

In Figure 2, we compare the learned $h_\eta(t)$ and the ground truth $h_f(t)$. We found that the learned $h_\eta(t)$ is very close to $0.5 \log t + 1.25$, which means that our method has learned the optimal divergence $D_{0.5}$. We conjecture that the constant $a$ for the learned $f$ is related to the learning rate. In fact, this gives $f$-divergence the ability to automatically adjust the learning rate through the scaling constant. For example, the constant $a$ is $\exp(1.25) > 1$ which may suggest that the learning rate $\beta$ is a bit small. Note that $\alpha$-divergence does not have this ability. We plot $f$ for $t \in [0, 3]$ in Figure 2 since we find most $t$ lie in this range.

### C.4.4 Additional Experimental Results

For meta-$D$, we report in Table 8 the meta-losses on 10 test tasks, which are obtained by executing the learned divergence minimization algorithm for 2000 iterations. The error bar is large due to the large variance among different tasks, so we report the ranking in Table 2, similar to Ma et al. (2019), to clearly show the adavantages of meta-$D$ over BO. Similarly, we also report the meta-losses for meta-$D\&\phi$ over 10 tasks in Table 9.

Table 8: Meta-$D$ on MoG: value of meta-loss over 10 test tasks.

| Methods | $\alpha = 0.5$ | TV |
|---|---|---|
| ground truth | 0.0811±0.0277 | - |
| meta-$\alpha$ | 0.0811±0.0277 | 0.2143±0.0936 |
| meta-$f$ | **0.0795**±0.0301 | **0.2020**±0.1024 |
| BO (8 iters) | 0.0833±0.0289 | 0.2203±0.0898 |
| BO (16 iters) | 0.0811±0.0277 | 0.2143±0.0936 |

Table 9: Meta-$D\&\phi$ on MoG: value of meta-loss over 10 test tasks.

| Methods\Meta-loss | $\alpha = 0.5$ (20 iters) | TV (20 iters) | $\alpha = 0.5$ (100 iters) | TV (100 iters) |
|---|---|---|---|---|
| VI&$\phi$ | 0.1237±0.0539 | 0.2572±0.1137 | 0.0905±0.0332 | 0.2321±0.0961 |
| meta-$\alpha\&\phi$ | 0.1207±0.0500 | 0.2462±0.1043 | 0.0879±0.0305 | **0.2263**±0.0936 |
| meta-$f\&\phi$ | **0.0793**±0.0237 | **0.2344**±0.0955 | **0.0784**±0.0332 | 0.2301±0.0949 |

## C.5 Regression Tasks with Bayesian Neural Networks

Each task includes a regression problem on a sinusoid wave; see Figure 6 for examples of the sinusoid waves. The BNN model is a two-layer neural network with hidden layer size 20 and RELU nonlinearities. For meta-learning divergence only, the training set size is 1000 and is obtained by sampling $x \in [-4, 4]$ uniformly. We use $K = 100$ and batch size 40 of which 20 data points are for updating $\phi_i$ and 20 points are for updating $\eta$. We train meta-$D$ for 1000 epochs. To evaluate the performance, we train the model with the learned divergence and VI respectively on new tasks for 1000 epochs. For learning both the divergence objective and initial variational parameters, we sample 20 tasks each time where each task has 40 data points. We use 20 points for updating $\phi_i$ and the other 20 points for updating divergence $\eta$ and the shared initialization $\phi$. $B = 4$ for meta-$\alpha\&\phi$. To evaluate, we start with the learned initialization and train the variational parameters with the learned divergence for 300 epochs.

### C.5.1 Additional Experimental Results

We provide the learned value of $\alpha$ from meta-$\alpha$ and meta-$\alpha\&\phi$ in Table 10. The predictive distributions on an example test task are given in Figure 7. Similar to the results of meta-$D$, meta-$D\&\phi$ is also able to model heteroskedastic noise while VI&$\phi$ cannot.

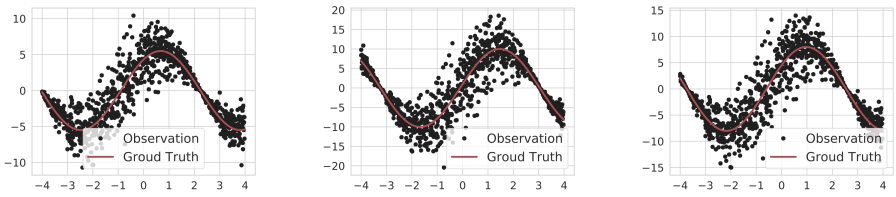


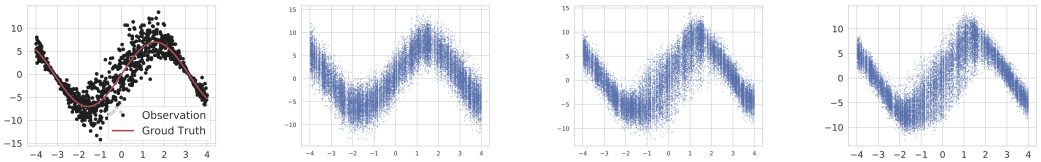

Figure 6: Three examples of sinusoid waves. Each task includes a regression on a sinusoid wave.

Table 10: Learned value of $\alpha$ of meta-$\alpha$ and meta-$\alpha\&\phi$ on sinusoid regression.

| | meta-$\alpha$ | meta-$\alpha\&\phi$ |
|---|---|---|
| $\alpha$ | 0.10 | 0.12 |

Table 11: Learned value of $\alpha$ of meta-$\alpha$ and meta-$\alpha\&\phi$ on MNIST.

| | meta-$\alpha$ | meta-$\alpha\&\phi$ |
|---|---|---|
| $\alpha$ | 0.14 | 0.80 |




(a) Ground Truth (b) VI$\&\phi$ (c) meta-$\alpha\&\phi$ (d) meta-$f\&\phi$

Figure 7: Meta-$D\&\phi$ on sin: the predictive distribution on a sinusoid wave.

### C.6 Image Generation with Variational Auto-Encoders

During meta-training, we sample 128 images of each task/digit and use half of the images to compute Eq.(6) and the other half for computing the meta-loss. The number of training epochs is 600. We set $K = 10$. For all methods, we use the same architecture (100 hidden units and 3 latent variables) and the marginal log-likelihood estimator as in Kingma and Welling (2014). We report the learned value of $\alpha$ from meta-$\alpha$ and meta-$\alpha\&\phi$ in Table 11. The meta-loss is the negative marginal log-likelihood.

#### C.6.1 Examples of Reconstructed Images

Besides comparing marginal likelihood in Section 4.3, we visualize the reconstructed images on two examples of digit 5. As shown in Fig 8, our methods produce higher quality of images because the reconstructed images are sharper and are able to capture different writing styles whereas the images from standard VI are blurry.

(a) VI$\&\phi$ (b) meta-$\alpha\&\phi$ (c) meta-$f\&\phi$

Figure 8: Reconstructed images of digit 5 by Meta-$D\&\phi$.

### C.7 Recommender System

We split the users into seven age groups: under 18, 18-24, 25-34, 35-44, 45-49, 50-55 and above 56, and regard predicting the ratings of users within the same age group as a task since the users with similar age may have similar preferences. We select 4 age groups (under 18, 25-34, 45-49, above 56) as training tasks, and use the remaining as test tasks. The meta-loss is the negative log-likelihood as used in Ma et al. (2018).

For meta-$D$ (Algorithm 1), during meta-training, we sample 100 users per task (400 users in total) and use half of the observed ratings to compute Eq.(6) and the other half for computing the meta-loss. The number of training epochs is 400. During meta-testing, we use 90%/10% training-test split for the three test tasks and train p-VAE with the learned divergence.

For meta-$D\&\phi$ (Algorithm 2), we compare our method with getting a p-VAE model initialization only, which can be regarded as a combination of MAML and p-VAE (denoted VI$\&\phi$). During evaluation, we apply 60%/40% training-test split for the test tasks and train the learned p-VAE model with the learned divergence.

#### C.7.1 Additional Experimental Results

We provide the learned value of $\alpha$ from meta-$\alpha$ and meta-$\alpha\&\phi$ in Table 12. And we visualize the learned $h_\eta(t)$ from meta-$f$ and meta-$f\&\phi$ in Figure 9. Besides the test log-likelihood, there are other popular evaluation metric being used in recommender system and sometimes they are not consistent with each other. Therefore, we also evaluate the performance of our method in terms of other common metrics: test root mean square error (RMSE) and test mean absolute error (MAE). For both metrics, our methods performs better than the baseline in the setting of learning inference algorithm and the setting of learning inference algorithm and model parameters (see Figure 10 and 11).

Table 12: Learned value of $\alpha$ of meta-$\alpha$ and meta-$\alpha\&\phi$ on MovieLens.

| | meta-$\alpha$ | meta-$\alpha\&\phi$ |
|---|---|---|
| $\alpha$ | 0.90 | 1.06 |

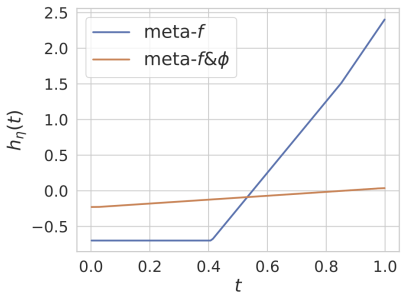



Figure 9: Visualization of learned $h_\eta(t)$ from meta-$f$ and meta-$f\&\phi$ on MovieLens.

### C.8 Comparison with MLAP (Amit and Meir, 2018)

Meta-Learning by Adjusting Priors (MLAP) is a method that meta-learns the Bayesian prior. We compared our method with it to show the importance of learning divergence. We tested on the permuted pixels experiment on MNIST, following the same experimental setup in Amit and Meir (2018). As this is a few-shot learning setup, we run meta-$\alpha\&\phi$ (meta-learning divergences and variational parameter). Our method attained test error 2.97% which outperformed the best result 3.40% (attained by MLAP-M) in Amit and Meir (2018) significantly. This further demonstrates the importance of learning divergence and the effectiveness of our method on finding the suitable divergence.

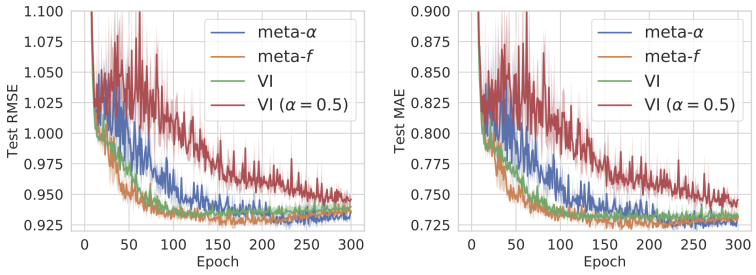



Figure 10: Meta-$D$ on ML: Comparison of meta-$D$ and VI in terms of test RMSE and test MAE.

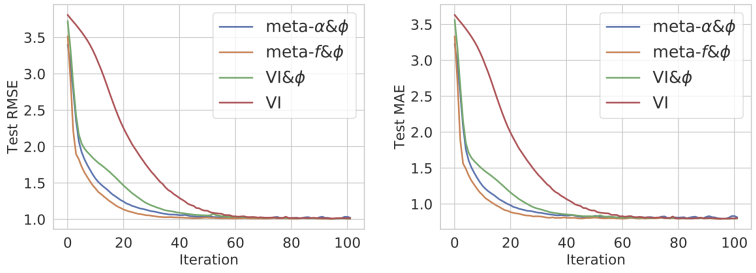



Figure 11: Meta-$D$&$\phi$ on ML: Comparison of meta-$D$&$\phi$ and VI&$\phi$ in terms of test RMSE and test MAE.