Meta-learning Control Variates: Variance Reduction with Limited Data (Supplementary Material)

Zhuo Sun[1,3] **Chris J. Oates**[2,3] **François-Xavier Briol**[1,3]

[1]Department of Statistical Science, University College London, London, UK
[2]School of Mathematics, Statistics & Physics, Newcastle University, UK
[3]The Alan Turing Institute, London, UK

In Appendix A, we provide the proof of the theoretical results stated in the main text. In Appendix B, we provide more details on the implementation of Neural-CVs and Meta-CVs, together with the full experimental protocol.

A PROOF OF THEOREMS

In this section, we will firstly review the assumptions and theorems in [Ji et al., 2022] in Appendix A.1 as the proof of the theorems follows the results of [Ji et al., 2022]. We then give the proof of Theorem 1 in Appendix A.2 and proof of Corollary 1.1 in Appendix A.3.

A.1 CONVERGENCE OF MODEL-AGNOSTIC META-LEARNING

Ji et al. [2022] analysed the convergence of model-agnostic meta-learning, as we will adapt their results to the training of CVs. Letting O_t be either S_t or Q_t, and phrasing in terms of the notation and setting used in this work, the assumptions of [Ji et al., 2022] are:

(A1) $\min_t \inf_\gamma J_{O_t}(\gamma) > -\infty$;

(A2) $\chi := \max_t \sup_{\gamma \neq \zeta} \frac{\|\nabla_\gamma J_{O_t}(\gamma) - \nabla_\zeta J_{O_t}(\zeta)\|_2}{\|\gamma - \zeta\|_2} < \infty$;

(A3) $\rho := \max_t \sup_{\gamma \neq \zeta} \frac{\|\nabla^2_\gamma J_{O_t}(\gamma) - \nabla^2_\zeta J_{O_t}(\zeta)\|_2}{\|\gamma - \zeta\|_2} < \infty$;

(A4) $\sigma^2 := \max_t \sup_\gamma \|\nabla_\gamma J_{O_t}(\gamma)\|_2^2 < \infty$;

(A5) $b_t := \sup_\gamma \|J_{S_t}(\gamma) - J_{Q_t}(\gamma)\|_2 < \infty$.

Theorem A.1 (Theorem 9 and Corollary 10 [Ji et al., 2022])**.** *Let the above assumptions (A1) to (A5) hold. Then, with a meta step-size $\eta_i = \frac{1}{80\chi\eta_i}$ for $i = 1, \ldots, I_{tr}$ and $\alpha = \frac{1}{8\chi L}$ in Algorithm 1 , we attain a solution $\hat{\gamma}_{\text{meta}}$ such that*

$$\mathbb{E}\|\mathbb{E}_t[\nabla \mathcal{J}_t(\hat{\gamma}_{\text{meta}})]\|_2 = \mathcal{O}\left(\tfrac{1}{I_{tr}} + \tfrac{\sigma^2}{B} + \sqrt{\tfrac{1}{I_{tr}} + \tfrac{\sigma^2}{B}}\right),$$

where $\chi_{\eta_i} = (1+\alpha\chi)^{2L} + C_b b + C_\chi \mathbb{E}_t[\|\nabla J_{Q_t}(\hat{\gamma}_{\text{meta}})\|_2]$, with $b = \mathbb{E}_t[b_t]$ and $C_b = C_\chi = (\alpha\rho + \rho/\chi(1+\alpha\chi)^{L-1})(1+\alpha\chi)^{2L}$.

Lemma A.2 (Lemma 19 [Ji et al., 2022])**.** *Under assumptions (A1) - (A5), for any t and any $\gamma \in \mathbb{R}^{p+1}$, we have*

$$\|\mathbb{E}_t[\nabla J_{Q_t}(\gamma)]\|_2 \leq \tfrac{1}{C_1'}\|\mathbb{E}_t[\nabla \mathcal{J}_t(\gamma)]\|_2 + \tfrac{C_2'}{C_1'},$$

where $C_1' > 0$ and $C_2' > 0$ are constants given $C_1' = 2 - (1+\alpha\chi)^{2L}$ and $C_2' = ((1+\alpha\chi)^{2L} - 1)\sigma + (1+\alpha\chi)^L((1+\alpha\chi)^L - 1)b$.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

A.2 PROOF OF THEOREM 1

To prove Theorem 1, we firstly derive three useful propositions (P1-P3) based on our Assumption 1 and Assumption 2 in Section 6, and then give the proof based on the above results from [Ji et al., 2022].

For each task t, we claim that

(P1) $\sup_{\gamma\neq\zeta} \frac{\|\nabla_\gamma J_{O_t}(\gamma)-\nabla_\zeta J_{O_t}(\zeta)\|_2}{\|\gamma-\zeta\|_2} < \infty$;

(P2) $\sup_{\gamma\neq\zeta} \frac{\|\nabla^2_\gamma J_{O_t}(\gamma)-\nabla^2_\zeta J_{O_t}(\zeta)\|_2}{\|\gamma-\zeta\|_2} < \infty$;

(P3) $\sup_\gamma \|\nabla_\gamma J_{O_t}(\gamma)\|_2 < \infty$,

for both $O_t \in \{S_t, Q_t\}$.

Proof of P1-P3. Denote the additive contribution of a single sample to the loss function as $l_t(x,\gamma) = (f_t(x) - g(x;\gamma))^2$. First we will show that under Assumption 1 and Assumption 2, we have: for each t and $x \in D_t$, the function $\gamma \mapsto \nabla_\gamma \ell_t(x;\gamma)$ is bounded and Lipschitz; and for each t and $x \in D_t$, the function $\gamma \mapsto \nabla^2_\gamma \ell_t(x;\gamma)$ is Lipschitz. Then (P1-P3) follow immediately as $J_{Q_t}(\gamma) = \frac{1}{|Q_t|}\sum_{x\in Q_t} l_t(x;\gamma)$ and $J_{S_t}(\gamma) = \frac{1}{|S_t|}\sum_{x\in S_t} l_t(x;\gamma)$.

From direct calculation, we have:

$$\begin{aligned}
\nabla_\gamma \ell_t(x;\gamma) &= -2(f_t(x) - g(x;\gamma))\nabla_\gamma g(x;\gamma) \\
\nabla^2_\gamma \ell_t(x;\gamma) &= 2(f_t(x) - g(x;\gamma))\nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - 2(f_t(x) - g(x;\gamma))\nabla^2_\gamma g(x;\gamma) \\
&= 2(f_t(x) - g(x;\gamma))\left[\nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - \nabla^2_\gamma g(x;\gamma)\right]
\end{aligned}$$

and taking differences:

$$\begin{aligned}
\|\nabla_\gamma \ell_t(x;\gamma) - \nabla_\zeta \ell_t(x;\zeta)\|_2 &= \| -2(f_t(x) - g(x;\gamma))\nabla_\gamma g(x;\gamma) + 2(f_t(x) - g(x;\zeta))\nabla_\zeta g(x;\zeta)\|_2 \\
&\le 2|f_t(x)|\|\nabla_\gamma g(x;\gamma) - \nabla_\zeta g(x;\zeta)\|_2 \\
&\quad + 2\|g(x;\gamma)\nabla_\gamma g(x;\gamma) - g(x;\zeta)\nabla_\zeta g(x;\zeta)\|_2 \\
&\le 2|f_t(x)|\|\nabla_\gamma g(x;\gamma) - \nabla_\zeta g(x;\zeta)\|_2 \\
&\quad + 2|g(x;\gamma)|\|\nabla_\gamma g(x;\gamma) - \nabla_\zeta g(x;\zeta)\|_2 + 2\|\nabla_\zeta g(x;\zeta)\|_2|g(x;\gamma) - g(x;\zeta)|.
\end{aligned}$$

So, for each t and $x \in D_t$, the function $\gamma \mapsto \nabla_\gamma \ell_t(x;\gamma)$ is bounded and Lipschitz when the functions $\gamma \mapsto g(x;\gamma)$ and $\gamma \mapsto \nabla_\gamma g(x;\gamma)$ are bounded and Lipschitz (i.e. Assumption 1).

Then taking differences and bounding terms in a similar manner, we have,

$$\begin{aligned}
\|\nabla^2_\gamma \ell_t(x;\gamma) - \nabla^2_\zeta \ell_t(x;\zeta)\|_2 \le\ & 2|f_t(x)|\|\nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - \nabla^2_\gamma g(x;\gamma) \\
& \quad - \nabla_\zeta g(x;\zeta)\nabla_\zeta g(x;\zeta)^\top + \nabla^2_\zeta g(x;\zeta)\|_2 \\
& + 2|g(x;\gamma)|\|\nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - \nabla^2_\gamma g(x;\gamma) \\
& \quad - \nabla_\zeta g(x;\zeta)\nabla_\zeta g(x;\zeta)^\top + \nabla^2_\zeta g(x;\zeta)\|_2 \\
& + 2\|\nabla_\zeta g(x;\zeta)\nabla_\zeta g(x;\zeta)^\top - \nabla^2_\zeta g(x;\zeta)\|_2|g(x;\gamma) - g(x;\zeta)|
\end{aligned}$$

So for each t and $x \in D_t$, the function $\gamma \mapsto \nabla^2_\gamma \ell_t(x;\gamma)$ is Lipschitz when the functions $\gamma \mapsto \nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - \nabla^2_\gamma g(x;\gamma)$ are bounded and Lipschitz (i.e. Assumption 2). □

Proof of Theorem 1:

Proof. Assumption (A1) is automatically satisfied. (P1) and (P2) above imply (A2) and (A3). (P3) above implies (A4).

Note that Assumption 1 implies (A5). This is because, for each t, $x \in D_t$, we have $\sup_\gamma l_t(x;\gamma) := \sup_\gamma (f_t(x) - g(x;\gamma))^2 < \infty$ as we assume that $\gamma \mapsto g(x;\gamma)$ is bounded and $f_t(x)$ is constant in γ. Thus, $\sup_\gamma J_{O_t}(\gamma) = \frac{1}{|O_t|}\sum_{x\in O_t} l_t(x;\gamma) < \infty$ where O_t can be either S_t or Q_t. So $\sup_\gamma \|J_{S_t}(\gamma) - J_{Q_t}(\gamma)\|_2 < \infty$.

Then, Theorem 1 follow from the conclusion of Theorem A.1. □

A.3 PROOF OF COROLLARY 1.1

Proof. Since Assumption 1 and Assumption 2 imply (A1) to (A5) in Appendix A.1, we will use the constants defined earlier in Appendix A.1 here as well. Firstly, note that given $\hat{\gamma}_\epsilon$, with

$$\alpha < \frac{\exp(\frac{\log 2}{2L})-1}{\chi} = \frac{2^{\frac{1}{2L}}-1}{\chi},$$

we have: $\mathbb{E}\|\mathbb{E}_t[\nabla J_{Q_t}(\hat{\gamma}_\epsilon)]\|_2 \leq \frac{1}{C_1'}\epsilon + \frac{C_2'}{C_1'}$ by taking $\gamma = \hat{\gamma}_\epsilon$ in Lemma A.2.

If then additionally $\nabla^2 J_{Q_t}(\gamma) \succeq \mu I_{p+1}$ holds, by (9.11) in Boyd et al. [2004] we have,

$$\|\gamma - \gamma_t^*\|_2 \leq \frac{2}{\mu}\|\nabla J_{Q_t}(\gamma)\|_2.$$

Taking the expectation of both sides, we then have

$$\begin{aligned}\mathbb{E}_t[\|\gamma - \gamma_t^*\|_2] &\leq \frac{2}{\mu}\mathbb{E}_t[\|\nabla J_{Q_t}(\gamma)\|_2] \\ &\overset{(i)}{\leq} \frac{2}{\mu}(\|\mathbb{E}_t[\nabla J_{Q_t}(\gamma)]\|_2 + \sigma),\end{aligned}$$

where (i) follows from [Ji et al., 2022] (Page 35, Line 8). Take $\gamma = \hat{\gamma}_\epsilon$ and take the expectation of both sides. Then by Theorem 1,

$$\begin{aligned}\mathbb{E}[\mathbb{E}_t[\|\hat{\gamma}_\epsilon - \gamma_t^*\|_2]] &\leq \frac{2}{\mu}\mathbb{E}[\|\mathbb{E}_t[\nabla J_{Q_t}(\hat{\gamma}_\epsilon)]\|_2] + \frac{2\sigma}{\mu} \\ &\leq \frac{2}{\mu}\left(\frac{1}{C_1'}\epsilon + \frac{C_2'}{C_1'}\right) + \frac{2\sigma}{\mu} \\ &= \frac{2}{\mu C_1'}\epsilon + \frac{2(\sigma C_1' + C_2')}{\mu C_1'} \\ &= \frac{C_1}{\mu}\epsilon + \frac{C_2}{\mu},\end{aligned}$$

where $C_1 = \frac{2}{C_1'}$ and $C_2 = \frac{2(\sigma C_1' + C_2')}{C_1'}$. □

B EXPERIMENTAL DETAILS

In this section, we provide more experimental details and implementation details of Neural-CVs and Meta-CVs. Details of the synthetic example are presented in Appendix B.1. Details of the boundary-value ODE are provided in Appendix B.2. Details of Bayesian inference for the Lotka–Volterra system are provided in Appendix B.3. Details of the Sarcos robot arm are presented in Appendix B.4.

B.1 EXPERIMENT: OSCILLATORY FAMILY OF FUNCTIONS

Our environment ρ consists of independent distributions on each element of a. For a_1, we select a $\mathsf{Unif}(0.4, 0.6)$, whilst for all other parameters we select a $\mathsf{Unif}(4, 6)$. Each task is of the form $\mathcal{T}_t = \{f_t(x; a_t), \pi_t\}$ where $a_t := (a_{t,1}, a_{t,2:d+1})^\top$ is a sample from ρ. This creates potentially infinite number of integral estimation tasks as a is continuous. The target distributions are $\pi_1(x) = \ldots = \pi_T(x) = \mathsf{Unif}(0, 1)^d$ where d is the dimension of x.

For all experiments of this example, we set the neural network identical for both Meta CVs and Neural CVs. That is, a fully connected neural network with two hidden layers. Each layer has 80 neurons while the output layer has 1 neurons (the output then is multiplied by a identity matrix I_d to used as $\tilde{u}$ where d is the dimension of the input x). The total number of parameters of the neural network $p = 80d + 6641$ where d the dimension of the input x. The activation function is the sigmoid function. The neural network is served as $\tilde{u}$ and we apply Langevin Stein operator onto $\tilde{u}(x)\delta(x)$ where $\delta(x) = \prod_{j=1}^d x_j(1 - x_j)$ to satisfy assumptions in [Oates et al., 2019]. For experiments in this example, we use Adam as the UPDATE rule in this example and the penalty constant λ is set to be 5×10^{-6}.

2-dimensional Oscillatory Family of Functions

- For Meta-CVs: The inner step size $\alpha = 0.01$. The number of inner gradient steps is $L = 1$. The meta step size $\eta = 0.002$ for all meta iterations. The number of meta iteration I_{tr} is set to be $4,000$. The meta batch size of tasks B is set to be 5.
- For Neural-CVs: The step size (learning rate) is 0.002. The number of training epochs for each task is set to be 20 with batch size 5.
- For Control functionals: we use radius basis function $k(x, x') = \exp(-\frac{\|x-x'\|_2^2}{2v})$ with kernel hyperparameter $v > 0$ as the base kernel for control functionals. The hyper-parameter v is tuned by maximising the marginal likelihood of the Stein kernel on S_t for each task. Optimal control functionals are selected by using S_t and then unbiased control functional estimators are constructed by using Q_t of each task.

Impact of the Number of Inner Updates L

- For Meta-CVs: The inner step size $\alpha = \frac{0.01}{50\times L}$ for $L \in \{1, 3, 5, 7, 10\}$. The meta step size $\eta = 0.002$ for all meta iterations. The number of meta iteration I_{tr} is set to be $4,000$. The meta batch size of tasks B is set to be 5.

Impact of Dimensions

- For Meta-CVs: The inner step size $\alpha = 0.01$. The number of inner gradient steps is $L = 1$. The meta step size $\eta = 0.002$ for all meta iterations. The number of meta iteration I_{tr} is set to be $4,000$. The meta batch size of tasks B is set to be 5.
- For Neural-CVs: The step size (learning rate) is 0.002. The number of training epochs for each task is set to be 20 with batch size 5.
- For Control functionals: we use radius basis function $k(x, x') = \exp(-\frac{\|x-x'\|_2^2}{2v})$ with kernel hyperparameter $v > 0$ as the base kernel for control functionals. The hyper-parameter v is tuned by maximising the marginal likelihood of the Stein kernel on S_t for each task. Optimal control functionals are selected by using S_t and then unbiased control functional estimators are constructed by using Q_t of each task.

Impact of B and I_{tr} of Meta-CVs

- The inner step size $\alpha = 0.01$. The number of inner gradient steps is $L = 1$. The meta step size is $\eta = 0.002$ for all meta iterations.

B.2 EXPERIMENT: BOUNDARY VALUE ODES

For all experiments of this example, we set the neural network identical for both Meta-CVs and Neural-CVs. That is, a fully connected neural network with three hidden layers. Each layer has 80 neurons while the output layer has 1 neurons. The total number of parameters of the neural network $p = 13,201$. The activation function is the sigmoid function. We use Adam as the UPDATE rule in this example and the penalty constant λ is set to be 5×10^{-6}.

- For Meta-CVs: The inner step size $\alpha = 0.01$ and the meta step size $\eta = 0.002$ for all meta iterations. The number of inner updates is $L = 1$. The number of meta iteration I_{tr} is set to be $2,000$. The meta batch size of tasks is set to be 5.
- For Neural-CVs: The step size (learning rate) is 0.002. The number of training epochs for each task is set to be 20 with batch size 5.

B.3 EXPERIMENT: BAYESIAN INFERENCE OF LOTKA-VOLTERRA SYSTEM

The log-exp transform is used on the model parameters x to avoid constrained parameters on the ODE directly. We reparameterised the Lotka—Volterra system as,

$$\begin{aligned}\frac{\mathrm{d}u_1(s)}{\mathrm{d}s} &= \tilde{x}_1 u_1(s) - \tilde{x}_2 u_1(s) u_2(s)\\ \frac{\mathrm{d}u_2(s)}{\mathrm{d}s} &= \tilde{x}_3 u_1(s) u_2(s) - \tilde{x}_4 u_2(s),\end{aligned}$$

where

$$\tilde{x}_1 = \exp(x_1), \tilde{x}_2 = \exp(x_2),$$
$$\tilde{x}_3 = \exp(x_3), \tilde{x}_4 = \exp(x_4),$$

where u_1 and u_2 represents the number of preys and predators, respectively.

The model is,

$$y_1(0) \sim \text{Log-Normal}(\log \tilde{x}_5, \tilde{x}_7)$$
$$y_2(0) \sim \text{Log-Normal}(\log \tilde{x}_6, \tilde{x}_8)$$
$$y_1(s) \sim \text{Log-Normal}(\log u_1(s), \tilde{x}_7)$$
$$y_2(s) \sim \text{Log-Normal}(\log u_2(s), \tilde{x}_8)$$

where

$$\tilde{x}_5 := \exp(x_5), \tilde{x}_6 := \exp(x_6)$$
$$\tilde{x}_7 := \exp(x_7), \tilde{x}_8 = \exp(x_8).$$

By doing so, x is then on the whole $\mathbb{R}^8$. As a result, the prior distribution $\pi(x)$ is defined on $\mathbb{R}^8$ and Stan will return the scores of these parameters directly as these 8 parameters x themselves are unconstrained through manually reparameterisation directly.

Priors are,

$$x_1, x_4 \sim \text{Normal}(0, 0.5^2)$$
$$x_2, x_3 \sim \text{Normal}(-3, 0.5^2)$$
$$x_5, x_6 \sim \text{Normal}(\log 10, 1^2)$$
$$x_7, x_8 \sim \text{Normal}(-1, 1^2)$$

Inference of x_1 and x_2

- For both Meta-CVs and Neural-CVs: We use a fully connected neural network with 3 hidden layers. Each layer has 5 neurons while the output layer has 8 neurons. The total number of parameters of the neural network $p = 153$. The activation function is the tanh function. All parameters of neural networks are initialised with a Gaussian distribution with zero mean and standard deviation 0.01 except of $\gamma_{t,0}$ is initialised at the Monte Carlo estimator of each task. We use Adam as the UPDATE rule in this example and the penalty constant λ is set to be 5×10^{-5}.
- For Meta-CVs: The inner step size $\alpha = 0.0001$. The number of inner gradient steps is $L = 1$. The meta step size was initialised at 0.001 with a step size decay ($\eta_{i+10} = 0.9\eta_i$) every 10 meta iterations. The number of meta iteration I_{tr} is set to be $2,000$. The meta batch size of tasks B is set to be 5. We only use 100 tasks (sub-populations) for learning the Meta-CVs. For each of these 100 tasks, we have more than N_t data points (also because MCMC sampler will return more than N_t samples, so we reuse all of them) such that we can learn Meta-CV with $I_{\text{tr}} = 2000$ and $B = 5$.
- For Neural-CVs: The step size (learning rate) is 0.001. The number of training epochs for each task is set to be 20 with batch size 5.

Inference of x_3 and x_4

- For both Meta-CVs and Neural-CVs: We use a fully connected neural network with 3 hidden layers. Each layer has 3 neurons while the output layer has 8 neurons. The total number of parameters of the neural network $p = 83$. The activation function is the tanh function. All parameters of neural networks are initialised with a Gaussian distribution with zero mean and standard deviation 0.01 except of $\gamma_{t,0}$ is initialised at the Monte Carlo estimator of each task. We use Adam as the UPDATE rule in this example and the penalty constant λ is set to be 5×10^{-5}.
- For Meta-CVs: The inner step size $\alpha = 0.001$. The number of inner gradient steps is $L = 1$. The meta step size was initialised at 0.001 with a step size decay ($\eta_{i+10} = 0.9\eta_i$) every 10 meta iterations. The number of meta iteration I_{tr} is set to be $2,000$. The meta batch size of tasks B is set to be 5. We only use 100 tasks (sub-populations) for learning the Meta-CVs. For each of these 100 tasks, we have more than N_t data points (also because MCMC sampler will return more than N_t samples, so we reuse all of them) such that we can learn Meta-CV with $I_{\text{tr}} = 2000$ and $B = 5$.
- For Neural-CVs: The step size (learning rate) is 0.001. The number of training epochs for each task is set to be 20 with batch size 5.

B.4 EXPERIMENT: SARCOS ROBOT ARM

Approximate Inference of Full Bayesian Gaussian Process Regression We learn full Bayesian hierarchical Gaussian processes by variational inference [Kucukelbir et al., 2017, Lalchand and Rasmussen, 2020].

We set $\sigma = 0.1$, $\pi(x_1) = \mathsf{Gamma}(25, 25)$ and $\pi(x_2) = \mathsf{Gamma}(25, 25)$, which is the prior used in [Oates et al., 2017]. We transform the kernel hyper-parameters $x \in \mathbb{R}^{2+}$ to $\eta = g(x) = \log x$ such that we can learn a variational distribution $q_\phi(\eta)$ of η in $\mathbb{R}^2$ and then transform back to $q(x)$. We use full rank approximation which means the variational family takes the following form:

$$q_\phi(\eta) = \mathsf{N}(\mu, VV^\top),$$

with variational parameter $\phi := \{\mu, V\} \in \mathbb{R}^{p+p(p+1)/2}$ where μ is a column vector and V is a lower triangular matrix. The objective of variational inference is to maximize the evidence lower with respect to ϕ, which is given by,

$$\begin{aligned}\mathsf{ELBO}(\phi) &= \mathbb{E}_{q_\phi}[\log p(y_{1:q}, e^\eta) + \log|\mathsf{Jacobian}_{g^{-1}}(\eta)|] - \mathbb{E}_{q_\phi}[\log q_\phi(\eta)] \\ &= \mathbb{E}_{q_\phi}[\log p(y_{1:q}|e^\eta) + \log \pi(e^\eta) + \log|\mathsf{Jacobian}_{g^{-1}}(\eta)|] - \mathbb{E}_{q_\phi}[\log q_\phi(\eta)]\end{aligned}$$

The expectations involved in $\mathsf{ELBO}(\phi)$ are approximated by Monte Carlo estimators and we use re-parametrization trick [Kingma and Welling, 2014] to learn ϕ. Figure 1 demonstrates the prior and the corresponding posterior of the kernel hyper-parameters $x = (x_1, x_2)$ (in the form of 2d histograms).

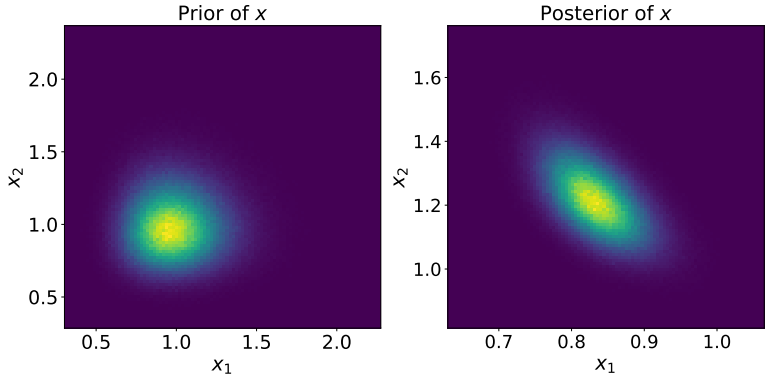

Figure 1: Priors and Posteriors of Kernel Hyper-parameters x.

Settings

- For both Meta-CVs and Neural-CVs, a fully connected neural network with 5 hidden layers. Each layer has 20 neurons while the output layer has 2 neurons (the output then is timed by a identity matrix I_2 to used as u since 2 is the dimension of the input x). The total number of parameters of the neural network $p = 10,401$. The activation function is the sigmoid function. All parameters of neural networks are initialised with a Gaussian distribution with zero mean and standard deviation 0.001. We use Adam as the UPDATE rule in this example and the penalty constant λ is set to be 1×10^{-10}.
- For Meta-CVs: The inner step size $\alpha = 0.01$. The meta step size was initialised at 0.001 with a step size decay ($\eta_{i+10} = 0.9\eta_i$) every 10 meta iterations. The number of meta iteration I_{tr} is set to be $1,000$. The meta batch size of tasks B is set to be 1.
- For Neural CV: The step size (learning rate) is 0.001. The number of training epochs for each task is set to be 20 with batch size 5.
- For Control functionals: we use radius basis function $k(x, x') = \exp(-\frac{\|x-x'\|_2^2}{2v})$ with kernel hyperparameter $v > 0$ as the base kernel for control functionals. The hyper-parameter v is tuned by maximising the marginal likelihood with the Stein kernel on S_t for each task. Optimal control functionals are selected by using S_t and then unbiased control functional estimators are constructed by using Q_t of each task.

Extra Experiments In addition, we test the performance of Meta-CVs on the same tasks used for learning the Meta-CV. Under the same setting described above, the comparisons between Meta-CVs and other methods are presented in Figure 2.

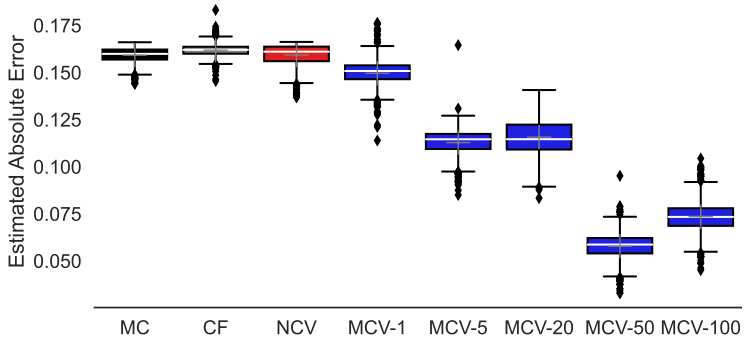

Figure 2: Estimated absolute errors over the same training states (which are used for learning the Meta-CV) of the Sarcos anthropomorphic robot arm (CF: Control functionals; NCV: Neural-CVs; MCV-L: Meta-CVs with L inner steps).

References

S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

K. Ji, J. Yang, and Y. Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *J. Mach. Learn. Res.*, 23: 29–1, 2022.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *J. Mach. Learn. Res.*, 2017.

V. Lalchand and C. E. Rasmussen. Approximate inference for fully Bayesian Gaussian process regression. In *AABI*, pages 1–12. PMLR, 2020.

C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *J. R. Stat. Soc. Series B*, 79(3): 695–718, 2017.

C. J. Oates, J. Cockayne, F-X. Briol, and M. Girolami. Convergence rates for a class of estimators based on Stein's method. *Bernoulli*, 25(2):1141–1159, 2019.