Transfer learning in Monte Carlo Methods and Beyond

Zhuo SUN

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Statistical Science

University College London

December 21, 2023

Declaration

I, Zhuo Sun, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

From computational statistics to machine learning, many methods have already achieved excellent performance for one single task given a moderate or large number of data points, e.g., kernel methods and deep learning. A task is usually a regression task, a classification task or a Monte Carlo integration task in statistical learning.

However, the performance of those methods is likely to degrade when the sample size of the training data is small; when latent information across tasks is ignored; when computational cost is prohibitively expensive. In this thesis, we focus on transfer learning for Monte Carlo methods and beyond via the scope of multi-task learning and meta-learning. It is well known that supervised learning aims to solve a specific task and often requires to train a model on some labeled data points (also known as training set). Monte Carlo methods provide us with estimators of expectations of functions of random variables with respect to some distributions. In this context, it is desirable to design novel algorithms or methods to explore and exploit transferable information across related tasks for both Monte Carlo methods and supervised learning.

This thesis includes three novel transfer learning methods. In the first work, we extend existing control variates, a powerful kind of post-processing tools for Monte Carlo methods, and propose a general framework called *vector-valued control variates* for multiple related integrals. In the second work, inspired by gradient-based meta-learning, we further generalise existing control variates to *meta-learning control variates*. In the third work, we extend gradient-based meta-learning to be a gradient-based probabilistic learning algorithm for few-shot image classification by

introducing latent class prototypes.

Impact Statement

This thesis mainly focuses on transfer learning for Monte Carlo methods and beyond. The first topic of this thesis is transfer learning for Monte Carlo methods via multi-task learning and meta-learning while the second topic is meta-learning for few-shot image classification.

For both topics, I consider to use transfer learning to improve existing methods. For Monte Carlo methods, it is often the case we can not collect a large number of samples and it can be infeasible to collect sufficient samples for all tasks. For image classification methods, it can be expensive and in-practical to acquire a large number of labeled data points, which often requires lots of human and financial resources to label huge datasets. I show that it is possible to employ transfer learning for Monte Carlo methods, which results in more efficient ways of estimation without requiring more data (but only use existing data of each task). This opens a new avenue for Monte Carlo methods. Meanwhile, my contribution on image classification methods serves as a stepping stone towards efficient transfer learning for image classification problems.

From an industry perspective, my work enhances computational efficiency while reducing costs. Thus, they can benefit many real-world applications. For instance, the two proposed transfer learning based Monte Carlo methods offer promising potential for enhancing the optimization of machine learning algorithms. It is also possible to improve medical diagnosis and vision ability of autonomous vehicles with the proposed few-shot image classification algorithm. These show the commercial potential of the proposed methods in this thesis.

Acknowledgements

I would like to thank my supervisor Dr. François-Xavier Briol for his extraordinary support and help. Over the past three years, he has always been keen to share his opinions and thoughts, and has given detailed feedback on drafts of our papers, my talk at conferences and institutions, and prize applications. He provided me with many great opportunities and guided me to be an independent researcher. I am also grateful to Department of Statistical Science and EPSRC for funding my PhD.

I would also like to thank Prof. Jing-Hao Xue for his kind support during the first year of my PhD and for being my second supervisor for the rest of my PhD. I appreciate to all the efforts of the staff members and my fellow PhD students in the department for providing support, organising departmental seminars and social events, which create a supportive research environment.

I would like to thank all my collaborators for their support. With the help from all these great people, I am able to do all these interesting and exciting projects listed in this thesis. I would also like to thank people from the data-centric engineering group at The Alan Turing Institute who were very kind and supportive when I co-organised Statistics in Data-Centric Engineering Seminar series in 2022. I am also thankful to the researchers and scholars who accepted my invitation and gave a talk at the seminar series. Meanwhile, I would also like to thank The Alan Turing Institute for providing me with the great opportunity of the Enrichment placement.

I am very grateful to my family for their support and encouragement through all my life. It is their endless love that guides me here.

UCL Research Paper Declaration Form: referencing the doctoral candidate's own published work(s)

This declaration form contains information of published works of my own that are relevant to this thesis.

For the work presented in Chapter 4:

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2)**:**
   (a) **What is the title of the manuscript?** Vector-valued Control Variates.
   (b) **Please include a link to or doi for the work:** `https://proceedings.mlr.press/v202/sun23a.html`
   (c) **Where was the work published?** Online.
   (d) **Who published the work?** Proceedings of Machine Learning Research.
   (e) **When was the work published?** July 2023.
   (f) **List the manuscript's authors in the order they appear on the publication:** Zhuo Sun, Alessandro Barp, François-Xavier Briol .
   (g) **Was the work peer reviewed?** Yes.
   (h) **Have you retained the copyright?** Yes.
   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes. `https://arxiv.org/abs/2109.08944`

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3)**:**
   (a) **What is the current title of the manuscript?**
   (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**
   If 'Yes', please please give a link or doi:
   (c) **Where is the work intended to be published?**
   (d) **List the manuscript's authors in the intended authorship order:**
   (e) **Stage of publication:**
3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4)**:** The article was written and edited by François-Xavier Briol and myself. François-Xavier Briol proposed the idea of this work and then we shaped this work together. All the experiments were done by myself, with some code from a previous publication contributed by Chris Oates. Alessandro Barp and François-Xavier Briol made contributions to most theorems. I also contributed to the theorems including Theorem 4.2.5 and the derivation of the special cases of the proposed matrix-valued Stein kernels.
4. **In which chapter(s) of your thesis can this material be found?** Chapter 4.

For the work presented in Chapter 5:

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2)**:**
   (a) **What is the title of the manuscript?** Meta-learning Control Variates: Variance Reduction with Limited Data.
   (b) **Please include a link to or doi for the work:** `https://`

proceedings.mlr.press/v216/sun23a.html

(c) **Where was the work published?** Online.

(d) **Who published the work?** Proceedings of Machine Learning Research.

(e) **When was the work published?** July 2023.

(f) **List the manuscript's authors in the order they appear on the publication:** Zhuo Sun, Chris J Oates, François-Xavier Briol.

(g) **Was the work peer reviewed?** Yes.

(h) **Have you retained the copyright?** Yes.

(i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi** Yes. https://arxiv.org/abs/2303.04756

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3)**:**

(a) **What is the current title of the manuscript?**

(b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**

If 'Yes', please please give a link or doi:

(c) **Where is the work intended to be published?**

(d) **List the manuscript's authors in the intended authorship order:**

(e) **Stage of publication:**

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4)**:** The article was written by François-Xavier Briol and myself. I proposed the idea of this work and all the experiments were done by myself, with some code from a previous

publication contributed by Kaiyu Li. Theoretical analyses were mostly done by myself with the help of Chris J Oates and François-Xavier Briol. Chris J Oates also helped to edit the paper.

4. **In which chapter(s) of your thesis can this material be found?** Chapter 5.

For the work presented in Chapter 6:

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2)**:**
   (a) **What is the title of the manuscript?** Amortized Bayesian Prototype Meta-learning: A New Probabilistic Meta-learning Approach to Few-shot Image Classification.
   (b) **Please include a link to or doi for the work:** `https://proceedings.mlr.press/v130/sun21a.html`
   (c) **Where was the work published?** Online.
   (d) **Who published the work?** Proceedings of Machine Learning Research.
   (e) **When was the work published?** April 2021.
   (f) **List the manuscript's authors in the order they appear on the publication:** Zhuo Sun, Jijie Wu, Xiaoxu Li, Wenming Yang, Jing-Hao Xue.
   (g) **Was the work peer reviewed?** Yes.
   (h) **Have you retained the copyright?** Yes.
   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**
   If 'No', please seek permission from the relevant publisher and check the box next to the below statement:
   ☒ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*
2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3)**:**
   (a) **What is the current title of the manuscript?**

(b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'?**

If 'Yes', please please give a link or doi:

(c) **Where is the work intended to be published?**

(d) **List the manuscript's authors in the intended authorship order:**

(e) **Stage of publication:**

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4)**:** The article was written mostly by myself. Jing-Hao Xue edited the paper. I proposed the idea of this work and designed all the methods and experiments. Jijie Wu and myself contributed to the code of the experiments. Xiaoxu Li and Wenming Yang provided helpful discussions and computing resources.
4. **In which chapter(s) of your thesis can this material be found?** Chapter 6.

e-Signatures confirming that the information above is accurate (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work)**:**

Candidate: Zhuo Sun

Date: July 28, 2023

Supervisor/Senior Author signature (where appropriate)**:** François-Xavier Briol

Date: July 28, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and scope of the thesis

Many methods in the fields of machine learning and computational statistics are tailored to one specific task at a time, which often require a large number of training data points to ensure satisfactory performance. In statistical learning, such a task can be a classification task, a regression task or a Monte Carlo integration task. For each task, it is often required to fit or train the model to a finite set of training dataset each time, and therefore, the latent information across different tasks is often ignored. Such latent information exists when those learning tasks are related somehow. For instance, in multi-fidelity modelling, it is often the case when there are a collection of functions being related and we are interested in the expectations of those functions. A natural and simple way is to estimate those integrals individually. However, as shown in Chapter 4, we can get more precise estimators by learning joint estimators for those multiple related integrals.

It is then natural for us to explore and exploit possible information among different tasks. To achieve this, extra assumptions are made to describe the relationship among data (e.g. either in terms of the covariates or the responses or the joint) or the relationship between different models (e.g. hard or soft parameter sharing).

In this thesis, we aim to develop novel transfer learning methods for Monte Carlo methods and classification methods through the scope of *meta-learning* and *multi-task learning*. Specifically, both the classification methods and variance re-

duction methods for Monte Carlo methods considered in this thesis fall into the category of supervised learning. We will point it out in Chapter 2.

1.2 Contributions

This thesis makes the following contributions to the fields of computational statistics and machine learning.

Contribution 1: Vector-valued Control Variates Existing control variates methods tackle integral estimation tasks one-by-one. This, however, ignores and wastes the potential relationship across many integration tasks. Therefore, in Chapter 4, we proposed a novel method for estimating related integrals jointly, *Vector-valued Control Variates*, by borrowing the strength from Stein's method and vector-valued reproducing kernel Hilbert spaces. We propose a matrix-valued Stein kernel for a sequence of distributions, provide simplified versions of it under several special cases, and give the formulation of vector-valued control variates. Our experiments show its superior performance when being compared to existing control variates methods and Monte Carlo estimators.

Contribution 2: Meta-learning Control Variates: Variance Reduction with Limited Data Inspired by meta-learning, in Chapter 5, we proposed a novel method, *Meta-learning Control Variates*, which is capable of learning task-specific control variates fast and achieve better performance even with limited data per task. Our experiments demonstrate its superior performance over existing control variate methods, e.g., control functionals, neural control variates, and also Monte Carlo estimators. Our theories provide insights and explicit conditions on the form of control variates that can be learnt well by the proposed method.

Contribution 3: Amortized Bayesian Prototype Meta-learning: A new probabilistic meta-learning approach to few-shot image classification In Chapter 6, we proposed *Amortized Bayesian Prototype Meta-learning*, a simple and novel method tailored for few-shot image classification. By introducing class prototypes as latent random variables into model-agnostic meta learning, the proposed method achieved competitive or state-of-the-art performance on multiple image datasets

when it was accepted for publication.

The works presented in Chapter 4, Chapter 5 and Chapter 6 lead to the following publications:

1. Z. Sun, C. J. Oates, and F.-X. Briol. Meta-learning Control Variates: Variance Reduction with Limited Data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2023b
    - This paper was selected for an oral presentation at UAI 2023.
2. Z. Sun, A. Barp, and F.-X. Briol. Vector-Valued Control Variates. In *International Conference on Machine Learning (ICML)*, 2023a
    - This paper received a Student Paper Award from the Section on Bayesian Statistical Science of the American Statistical Association in 2022.
3. K. Li* and Z. Sun*. Multilevel Control Functional. *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023
4. Z. Sun, J. Wu, X. Li, W. Yang, and J.-H. Xue. Amortized Bayesian Prototype Meta-learning: A New Probabilistic Meta-learning Approach to Few-shot Image Classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, 2021
5. X. Li*, Z. Sun*, J.-H. Xue, and Z. Ma. A concise review of recent few-shot meta-learning methods. *Neurocomputing*, 2020a
6. X. Li*, J. Wu*, Z. Sun*, Z. Ma, J. Cao, and J.-H. Xue. Bsnet: Bi-similarity network for few-shot fine-grained image classification. *IEEE Transactions on Image Processing*, 2020b

Outline of the Thesis The rest of the thesis is organised as follows. In Chapter 2, we present a brief summary of existing transfer learning methods in a general form and give concrete examples in both fields of computational statistics and machine learning. Chapter 3 contains necessary background on kernel methods, Stein's method, control variates and meta-learning, which are the key components of the fore-mentioned three contributions of this thesis. Chapter 4, Chapter 5 and Chap-

ter 6 present Contribution 1, 2 and 3, respectively. Each chapter is self-contained with necessary preliminary background, and additional appendices are provided at the end of the thesis. In the end, Chapter 7 presents concise summaries of this thesis and outlines future research directions.

Chapter 2

Transfer Learning in Monte Carlo Methods and Machine Learning

This chapter consists of three sections. Section 2.1 introduces general concepts and strategies of transfer learning. Section 2.2 presents how transfer learning methods can be employed for Monte Carlo methods, especially in terms of multiple related integration tasks. Section 2.3 presents how transfer learning can be used in the context of supervised learning, points out the connection between control variates methods and supervised learning, and briefly gives one example called meta-learning of such transfer learning methods.

2.1 Transfer Learning

Transfer learning (for a detailed review, see e.g., (Yang et al., 2020)) plays an important role in machine learning and statistics since it is a promising way to solve multiple tasks simultaneously, and is capable of improving the performance of algorithms on some target tasks based on the knowledge or information extracted from some source tasks. It aims to provide better performance than the traditional methods which are only designed to find or learn the decision function $f : \mathcal{X} \to \mathcal{Y}$ for a response variable $Y \in \mathcal{Y}$ given some covariates $X \in \mathcal{X}$ from one single task. For instance, such a decision function f can be a regressor or a classifier, depending on the problem of interest. In this thesis, we focus on homogeneous transfer learning. That is, the domain $\mathcal{X}$ is identical for both source and target tasks, while they may

differ in the marginal distribution of X, $p(X)$. Though it is a concept in machine learning, we actually can see it in computational statistics, e.g., importance sampling, optimal transport and gradient flow. To begin with, we give the definitions of *tasks*, *target tasks* and *source tasks*.

Definition 2.1.1 (Tasks)**.** A task $\mathcal{T}$ consists of a quantity of interest together with the data that can be used to estimate this quantity.

Definition 2.1.2 (Supervised Learning as Tasks)**.** For supervised learning, we aim to learn or estimate the relationship $f : \mathcal{X} \to \mathcal{Y}$ between a response variable $Y \in \mathcal{Y}$ and covariates X whose marginal distribution is $p(X)$ on the domain $\mathcal{X}$. Assuming that the underlying truth f belongs to a function space $\mathcal{F}$, and given a training set (e.g. m realisations of X and Y, denoted as $\{x_i, y_i\}_{i=1}^m$) of the task, the problem of interest typically involves learning an approximation, denoted as $\hat{f}$, to the function f from the function space $\mathcal{F}$. The performance of learnt approximation $\hat{f}$ is then evaluated on a testing set $\{x_i, y_i\}_{i=m+1}^{m+n}$ of the same task.

Example 2.1.1 (Estimation of Integrals as Tasks)**.** *Estimation of integrals can be categorized into the above definition of tasks since integration methods can be written as integrals of approximation methods. For instance, Bayesian quadrature (Briol et al., 2015; Xi et al., 2018; Li et al., 2023) utilizes Gaussian processes to approximate integrands of interest. Other examples include control variates (Oates et al., 2017; Si et al., 2021; Leluc et al., 2021; South et al., 2022b) which select an approximation to the integrand of interest from a zero-mean (with respect to the distribution of interest) function space; see Chapter 3 for details. This is also the case for the proposed vector-valued control variates in Chapter 4 and the proposed meta-learning control variates in Chapter 5.*

Remark 2.1.1 (Target Tasks and Source Tasks in Transfer Learning)**.** *In transfer learning, sometimes only one particular task among a group of tasks is of interest. This task is referred as the target task. The remaining tasks are referred as the source tasks. Note that there could be more than one target task. The domain of the target task $\mathcal{X}_{target}$ can be different to that of source tasks $\mathcal{X}_{source}$. The space of*

the response variable of a target task $\mathcal{Y}_{target}$ can also differ from that of source tasks $\mathcal{Y}_{source}$. Thus, the most general setting is that every task in the group has distinct domains and label spaces. In this thesis, we consider the cases when the domains of tasks are identical but may have different marginal distributions. Note that when all tasks are target tasks, this is also known as multi-task learning.

Numerous learning algorithms fall within the category of transfer learning, encompassing multi-task learning and meta-learning, all of which bear significant relevance to the three proposed methods.

Pre-training and fine-tuning Pre-training and fine-tuning are mainly designed for deep learning algorithms (LeCun et al., 2015). Pre-training (Devlin et al., 2018; Hendrycks et al., 2019; Zoph et al., 2020) means that we can pre-train a deep neural network on a large dataset (which is usually cheap to obtain), which can then be used for other tasks later. This can be particular useful when we only have access to a small number of training data points of the target tasks. Fine-tuning means that we only need to fine-tune part or all of the parameters of the pre-trained neural network without training the whole neural network from scratch. These two approaches can save lots of computing time and efforts for optimising neural networks for the target tasks and often result in significant improvements in performance, and thus they are widely used in the fields of computer vision and large language models (Devlin et al., 2018; Chen et al., 2019b; Zoph et al., 2020; Ju et al., 2022).

Multi-task learning In this case, all the tasks are treated equally. Multi-task learning aims to learn multiple tasks simultaneously and improve the performance on all tasks. This kind of method can not only be found in the machine learning community such as multitask learning with kernel methods (Carmeli et al., 2010; Álvarez et al., 2012; Ciliberto et al., 2015) or with deep neural networks (Collobert and Weston, 2008; Standley et al., 2020) but also in the computational statistics community such as multiple output Bayesian quadrature (Xi et al., 2018) and multi-task averaging (Efron and Morris, 1977; Feldman et al., 2014; Marienwald et al., 2021). In particular, multi-task kernel methods are closely related to the contribution in Chapter 4 of this thesis. More details will be presented in Chapter 3 and Chapter 4.

Bayesian hierarchical modelling Bayesian hierarchical modelling considers the cases when we have access to observations from different groups. By allowing different parametric or non-parametric models to govern the data generating processes for each of these groups, and treating some hyper-parameters as random variables shared among these groups, Bayesian hierarchical modelling constructs a hierarchical probabilistic structure on data (via likelihood), models (via priors) and hyper-parameters (via hyper-priors), e.g. hierarchical Gaussian processes when we use Gaussian processes as the middle level non-parametric models (Rasmussen, 2003) or hierarchical linear models when we use linear models as the middle level parametric models (Garson, 2013; Griffin and Brown, 2017). Instead of doing individual inference for each group, it provides us with better inference by combining all the data from all the groups as all the information are utilized and shared via the hierarchical probabilistic structure designed specifically.

Meta-learning The goal of meta-learning is to design an algorithm that is capable of learning unseen target tasks fast and efficiently. This capability is established upon: i) having access to labeled data points from a number of source tasks and ii) designing efficient learning algorithms. Meta-learning algorithms have shown promising performance in the fields of image classification (Finn et al., 2017) and reinforcement learning (Liu et al., 2019). One typical branch of meta-learning is *gradient-based meta-learning* (Finn et al., 2017, 2019, 2018; Grant et al., 2018), which is closely related to the contributions in Chapter 5 and Chapter 6 of this thesis. More details will be presented in Chapter 3, Chapter 5 and Chapter 6.

One common characteristic shared among the above methods is that they all want to transfer the knowledge extracted from the source tasks to improve performance on the target tasks, e.g. better classification accuracy if the goal is to learn a classifier $\hat{f}$, which includes parameter sharing and feature adaption. For instance, we can pre-train models using the data from source tasks and then adapt the parameters conditioning on the training samples from the target task.

In this thesis, our focus lies in the construction of novel transfer learning methods for Monte Carlo methods and supervised classification methods. We will briefly

introduce and discuss the fundamental concepts of multiple related integration tasks and supervised meta-learning in Section 2.2 and Section 2.3, respectively. More background knowledge will be presented in Chapter 3.

2.2 Transfer Learning in Monte Carlo Methods

Integrals appear almost everywhere in computational statistics, e.g., from posterior moments of model parameters in Bayesian inference to marginal likelihood in hierarchical models. These integrals are often presented in the form of the integration of a function $f : \mathcal{X} \to \mathbb{R}$ over a distribution Π. For the distributions considered here, we assume that Π is some probability distribution with Lebesgue density π. A concrete example is the expected volume of water that passes through a borehole (Xiong et al., 2013) where the parameters of the model of the borehole and water-flow are regarded as random variables with expert-specified distributions. Suppose that we have a sequence of square-integrable functions $f_1, \ldots, f_T$ and distributions $\Pi_1, \ldots, \Pi_T$, and we would like to use $\{(x_{tj}, f_t(x_{tj}))_{j=1}^{m_t}\}$ to estimate these T integrals:

$$\Pi_t[f_t] := \int_{\mathcal{X}} f_t(x)\pi_t(x)dx \quad \text{for } t = 1, \ldots, T. \tag{2.1}$$

Example 2.2.1 (A 2-dimensional Illustration Example of Multiple Related Monte Carlo Integration Tasks)**.** *For illustration purpose, we show an example where* $\mathcal{X} = \mathbb{R}^2$ *and* $\Pi[f] := (\Pi_1[f_1], \Pi_2[f_2], \Pi_3[f_3])^\top$ *in Figure 2.1. The functions are set to be* $f_1(x) = 0.6 + \sin(0.5x_1)\cos(0.5x_2)/5$*;* $f_2(x) = 0.55 + \sin(0.2x_1)\cos(0.4x_2)/5$*;* $f_3(x) = 0.65 + \sin(0.7x_1)\cos(0.4x_2)/5$*.* Π_t *is a mixture of Gaussians with three components in the form of* $\frac{1}{3}\sum_{j=1}^{3}\mathbb{N}(\mu_{tj}, \Sigma_{tj})$*. In this example,* μ_{tj} *are randomly generated while* Σ_{tj} *is a diagonal matrix with equal size elements* σ_{tj}^2*. Here,* $\sigma_{11} = 1.1, \sigma_{12} = 1.5, \sigma_{13} = 2$*;* $\sigma_{21} = 1.2, \sigma_{22} = 1.3, \sigma_{23} = 2.5$*;* $\sigma_{31} = 1.1, \sigma_{32} = 1.15, \sigma_{33} = 1.2$*. We will see how the proposed vector-valued control variates and meta-CVs solve these multiple related integrals efficiently in Chapter 4 and Chapter 5.*

Each of these integrals can be estimated individually. Such estimators include

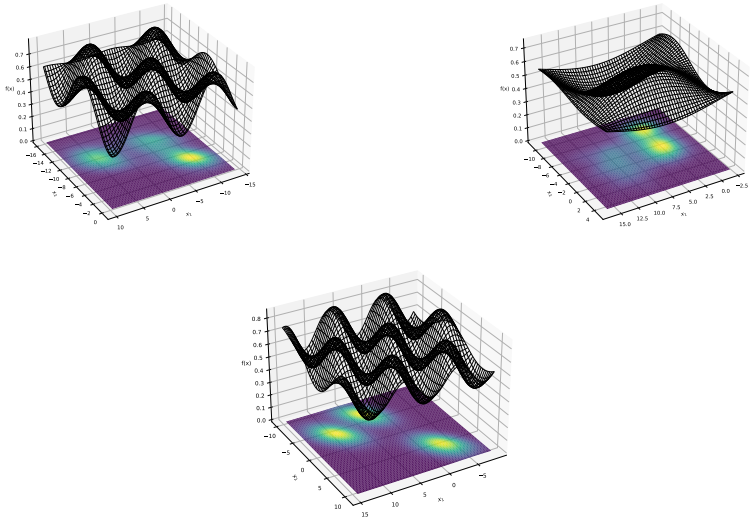

Figure 2.1: Illustration example of a vector-valued integration task. The black wireframe is the integrand $f_t(x)$ with two-dimensional input $x = (x_1, x_2)^\top$ while the bottom is the density contour plot of the distribution Π_t.

simple Monte Carlo (MC) estimator, *Markov Chain Monte Carlo* (MCMC) estimator, *importance sampling* (IS) estimator and *self-normalized importance sampling* (SNIS) estimator. For detailed reviews and analyses, we refer readers to (Owen, 2013; Robert and Casella, 2013). More sophisticated low-variance estimators include *Bayesian quadrature* (BQ) and *control variates* (CVs). These methods enjoy a much faster convergence rate if being compared to Monte Carlo estimators. This thesis will focus on Stein-based control variates for Monte Carlo methods as they can be used in the cases when we only have access to evaluations of $\tilde{\pi} = \frac{\pi}{C}$ with intractable normalising constant C. This is typically the case when π is a posterior distribution in Bayesian inference. Recent development of control variates includes stochastic-optimised control variates (Wan et al., 2019; Si et al., 2021), which is scalable to large datasets. Other interesting work includes semi-exact control variates (South et al., 2022b), which combines polynomial control variates (Mira et al., 2013) and control functionals (Oates et al., 2017) at the same time.

However, these existing control variates methods focus on one integral estimation task at a time. This restricts the flow of information among related tasks during the process of selecting/learning effective control variates for each task. The main insight is that if the functions $f_1, \ldots, f_T$ are related in some way, and if the same

can be said about the distributions $\Pi_1, \ldots, \Pi_T$, then we can estimate these integrals jointly. The intuition of jointly estimation of integrals can be traced back to *Stein's paradox* (Efron and Morris, 1977), which has shown that it is better to jointly estimate the means of three or more Gaussian random variables as such a vector-valued estimator (*James-Stein estimator*) has lower risk than the sum of the risk of each individual estimator. Recent development of multiple related integral estimation is established on Bayesian quadrature (Xi et al., 2018). However, it only be used in the cases when the kernel mean $\int k(x, x')\pi(x')dx'$ has closed form expression. This can be problematic when we only know the target distribution up-to an unknown normalising constant C, i.e., only having access to $\tilde{\pi} = \frac{\pi}{C}$ with intractable C.

Therefore, it is desirable to design new control variates methods that can learn the sequence $\Pi_1[f_1], \ldots, \Pi_T[f_t]$ simultaneously on the basis of transfer learning. Existing control variates methods (Mira et al., 2013; Oates et al., 2017; Si et al., 2021; South et al., 2022b) are post-processing tools for Monte Carlo methods. However, they are designed to estimate one integral at a time. When we have multiple related integration tasks required to be estimated and want to do better than existing control variates methods, inspired by multi-task kernel methods (Álvarez et al., 2012), we propose *vector-valued control variates* in Chapter 4. Furthermore, when the number of tasks T is extremely large (e.g. $T = 1000$) and the sample size is very small for each task, it is infeasible to use existing control variates methods or the proposed *vector-valued control variates* since the computational cost is expensive and these methods cannot provide us with satisfactory performance due to the small sample size per task. In this case, inspired by meta-learning, we propose *meta-learning control variates* in Chapter 5.

2.3 Transfer Learning in Supervised Learning

In the context of supervised learning, we seek for a predictor

$$f^* = \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \mathcal{E}(f) \text{ with } \mathcal{E}(f) := \mathbb{E}_{(X,Y)\sim p(x,y)} \left[J\left(f\left(X\right), Y\right) \right] \tag{2.2}$$

that can predict the labels for some new unseen points, where J is some loss function and $p(x, y)$ is the joint distribution of the response variable $y \in \mathcal{Y}$ and the covariates $x \in \mathcal{X}$. For example, when J is the squared loss (common loss for regression tasks), $f^*(x) = \mathbb{E}_{p(y|x)}[y|x]$; when J is the 0-1 loss (common loss for classification tasks), $f^*(x) = \arg\max_{y \in \mathcal{Y}} p(y|x)$.

However, it is hard to obtain the exact analytic expression for the population risk $\mathcal{E}(f)$ and seek for the optimal solution in the giant space of $\{f : \mathcal{X} \to \mathcal{Y}\}$ (i.e. the set of all possible functions from $\mathcal{X}$ to $\mathcal{Y}$) and have access to the joint distribution $p(x, y)$. Therefore, to tackle the above task, the common way is to minimize the empirical risk and optimize it over some function space $\mathcal{F} \subset \{f : \mathcal{X} \to \mathcal{Y}\}$, e.g. reproducing kernel Hilbert space $\mathcal{H}_k$ induced by a kernel k (Rasmussen, 2003; Wainwright, 2019). That is, given a training dataset $\{x_i, y_i\}_{i=1}^m$, we seek for $\hat{f}$ which minimizes the empirical loss,

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m J\left(f\left(x_i\right), y_i\right) \tag{2.3}$$

where $\{x_i, y_i\}_{i=1}^m$ are the training samples of the corresponding task.

Example 2.3.1 (C-class classification)**.** *A C-class classification task is a typical supervised learning task. It means that we want to categorize objects into C different classes. In this case, the response variable y takes values in the set $\{1, 2, \ldots, C\}$, which represents C different categories. The covariates x can be images or words. See Chapter 6 for the cases when we are interested in predicting the labels of images.*

Example 2.3.2 (Selecting Control Variates as Supervised Learning)**.** *Control variates can be regarded as a specific type of supervised learning as the way to selecting an effective control variate $g \in \mathcal{G}$ is done by minimizing some loss J between the integrand f and g conditioning some data points $\{x_i, y_i := f(x_i)\}_{i=1}^m$ (e.g., $\sum_{i=1}^m J(g(x_i), y_i)$), where $\mathcal{G}$ is a function space with known-mean functions. See Chapter 3, Chapter 4 and Chapter 5 for more details of the choice of $\mathcal{G}$ and the loss J.*

Suppose that we have obtained a predictor $\hat{f}$ for a C-class classification problem with a training dataset $\{(x_i, y_i)_{i=1}^m;\ \forall y_i \in \{1, \ldots, C\}\}$, we expect it to achieve satisfactory accuracy on another testing dataset from the same task. However, such a classifier can not be directly applied to a new classification task. For example, the task for training is to assign different types of dogs while the new task is to classify several kinds of birds. It becomes more difficult when the training sample size for the new task is very small, e.g., 1 labeled data point per class.

Therefore, it is desirable to design novel machine learning algorithms that are tailored for such kind of transfer learning problems. Common ways of utilizing the latent relationship among different tasks includes *parameter control* and *knowledge transfer* as discussed in Section 2.1. One example of such transfer learning methods is *meta-learning*; see Chapter 3 for a more detailed discussion of meta-learning.

Example 2.3.3 (Meta-learning)**.** Meta-learning *(Bartunov and Vetrov, 2018; Finn et al., 2017; Grant et al., 2018; Amit and Meir, 2018; Rusu et al., 2019; Iakovleva et al., 2020) aims to solve unseen new tasks based on experience established through an episodic meta-training process (Vinyals et al., 2016) of previous tasks. The goal is to seek for $\hat{f}$ that can be adapted fast to any testing task after being trained through an episodic meta-training process. To find such a $\hat{f}$, the objective is to minimise the expected loss of f on a testing set Q after being adapted on a training set S. That is,*

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{T} \sim \rho} \mathbb{E}_{S,Q \sim \mathcal{T}} \left[J\left(f\left(S\right), Q \right) \right],$$

where ρ is the population distribution (also known as environment) of tasks; $\mathcal{T}$ is a task sampled from ρ; S and Q are training and testing sets from the task $\mathcal{T}$, respectively.

Example 2.3.4 (Few-shot Sinusoidal Function Meta-learning)**.** *Consider a class of regression tasks with true underlying functions $f(x;\gamma) = \gamma_1 \sin(x + \gamma_2)$ where $\gamma := (\gamma_1, \gamma_2)^\top$. Suppose that we only have access to noisy observations $y(x;\gamma) = f(x;\gamma) + \epsilon$ at some points $\{x_i\}_{i=1}^m$, where γ_1 and γ_2 are not fixed and ϵ is some*

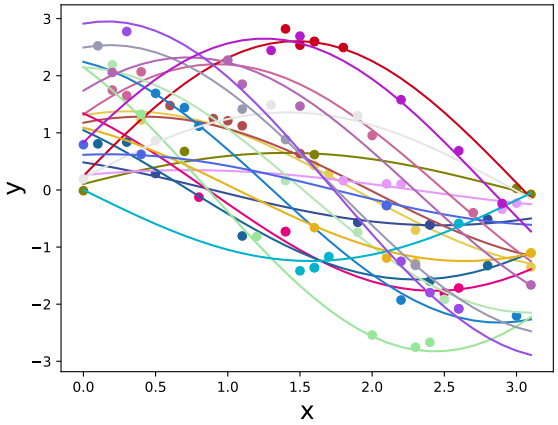

Figure 2.2: Illustration example: 5-shot sinusoidal function meta-learning. Each curve represents the true function from a sinusoidal function regression task, with observations plotted as dots.

random Gaussian noise following $\mathcal{N}(0, 0.1^2)$. *Consider that the environment* ρ *is* $\gamma_1 \sim$ *Uniform*$[0.1, 3.0]$ *and* $\gamma_2 \sim$ *Uniform*$[0, \pi]$, *each task is then a regression task with values of* γ_1 *and* γ_2 *sampled from their distributions. Through a meta-learning process, we want to find a* $\hat{f}$ *that can adapt to the underlying true function of an unseen task* $\mathcal{T}$ *from* ρ *fast. Few-shot means the number of training samples* S *of the unseen task* $\mathcal{T}$ *is small, e.g.,* 1. *When* $|S| = 1$, *this is referred to as* 1*-shot regression; when* $|S| = 5$, *this is referred to as* 5*-shot regression. Figure 2.2 gives an example of* 5*-shot meta learning for this class of sinusoidal functions.*

The above sinusoidal function regression tasks can be solved individually. However, when the true underlying function f is much more complex than sinusoidal functions and when only a small number of training data points per task are available, learning a good regressor $\hat{f}$ can be very difficult.

Example 2.3.5 (Parametric Probabilistic Models)**.** *Suppose that we have access to noisy observations from several tasks, and all of these tasks are assumed to share the family of probabilistic model* $\mathcal{P}_\Theta(\mathcal{X}, \mathcal{Y}) := \{p_\theta, \theta \sim \rho\}$ *where* θ *encodes some unknown random structure of* (X, Y) *and is sampled from some distribution* ρ. *Each task* $\mathcal{T}$ *can then be the inference problem of the joint distribution of covariates and response, i.e.,* $p_\theta(X, Y)$. *Since* θ *is random, the generative process of* (X, Y) *can be*

different across different realisations of θ but they can be related as the environment ρ is shared across different tasks and thus observations. The goal of meta-learning is to utilize a model f that is able to learn the structure $\tilde{\theta}$ given few observations from any unseen new task $\tilde{\mathcal{T}}$ fast once the model has been fit on the previous training tasks with observations $\{(x_i, y_i)\}_{i=1}^{m_t}$ for $\mathcal{T}_t$ and $t = 1, \ldots, T_{train}$. Concrete examples of this kind are included in Chapter 5 and Chapter 6.

Chapter 3

Background: Kernel Methods, Stein's Method, Control Variates and Meta-learning

This chapter contains three sections. Section 3.1 presents necessary background of kernel methods. Section 3.2 introduces Stein's method. Section 3.3 discusses the existing control variates methods. Section 3.5 presents necessary background of meta-learning. These are important and fundamental components of the three proposed methods in Chapter 4, Chapter 6 and Chapter 5, repectively.

3.1 Kernel Methods

In this section, we provide necessary background of kernel methods as they are the key components of the method proposed in Chapter 4. Kernel methods have been used to construct control variates (Oates et al., 2017; Si et al., 2021; South et al., 2022b), and they are also very popular in computational statistics and machine learning.

3.1.1 Reproducing Kernel Hilbert Spaces

A reproducing kernel Hilbert space (RKHS) is a space of real-valued functions with reproducing property (Berlinet and Thomas-Agnan, 2011; Wainwright, 2019), which makes it popular and convenient for solving statistical machine learning tasks. A concrete way to think of a RKHS $\mathcal{H}_k$ is to specify a positive semidefi-

nite kernel k. Alternatively, an equivalent way is to restrict to the Hilbert spaces in which linear functionals are bounded as a result of the *Riesz representation theorem*. For the purpose of introduction, we will focus on the first way to introduce a RKHS by defining a valid positive semidefinite kernel function.

Definition 3.1.1 (Scalar-valued Kernels)**.** A symmetric bivariate function is a function of two input variables, of which the value is the same no matter the order of its two input variable. A symmetric bivariate function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a valid scalar-valued positive semi-definite (PSD) kernel function if for any finite set of points $X := \{x_i\}_{i=1}^N$, the Gram matrix $k(X, X)$ with elements $(k(X, X))_{i,j} := k(x_i, x_j)$ is positive semidefinite, i.e. $v^\top k(X, X)v \geq 0$ for any $v \in \mathbb{R}^N$.

For example, the squared-exponential kernel $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\lambda^2})$ with $\lambda \in \mathbb{R}^+$ and preconditioned squared-exponential kernel (Oates et al., 2017) $k(x, y) = \frac{1}{(1+\alpha\|x\|_2^2)(1+\alpha\|y\|_2^2)} \exp\left(-\frac{\|x-y\|_2^2}{2\lambda^2}\right)$ with $\lambda \in \mathbb{R}^+$ and $\alpha \in \mathbb{R}^+$ are valid kernels. Polynomial kernels $k(x, y) = (x^\top y + c)^l$ with constant $c \in \mathbb{R}$ and power $l \in \mathbb{N}$ are valid scalar-valued kernels. The product of kernels are also valid kernels; see (Duvenaud, 2014) for more examples of kernels and kernel grammar.

Kernels are particularly useful as raw data points are embedded in a high dimensional feature space without specifying it directly. This is referred as reproducing property (defined in Definition 3.1.2), which is one of the most important and useful properties of kernels. It allows us to regard the kernel function as a feature map from $\mathcal{X}$ to $\mathcal{H}$. One benefit of using kernels is that we no longer need to do computation in the feature space, instead, by reproducing property we have $\langle k(\cdot, x), k(\cdot, x')\rangle_{\mathcal{H}_k} = k(x, x')$.

Definition 3.1.2 (Reproducing Property)**.** For a kernel K, given $\forall x \in \mathcal{X}$ and $\forall f \in \mathcal{H}_K$, $K(\cdot, x) \in \mathcal{H}_K$ and it satisfies,

$$\langle f, K(\cdot, x)\rangle_{\mathcal{H}_K} = f(x). \tag{3.1}$$

However, choice of kernels itself is an important and challenging task when

we do not have expert knowledge. The widely-adopted strategy is to choose kernels (and the associated hyper-parameters of kernels) by cross-validation. There are some tricks that help us to find proper candidates. For example, for regression tasks, it is often to choose the kernels which match the properties of the underlying true functions, e.g. periodicity and smoothness. Another example is more relevant to the topics of this thesis. That is, for kernel-based control variates, it is generally required in practice that the chosen kernels should match the smoothness of the integrands of interest, and ideally the integrands lie in the space of the RKHSs induced by these kernels.

Theorem 3.1.1 (Moore-Aronszajn Theorem)**.** *Every scalar-valued kernel induces a unique Hilbert space $\mathcal{H}$ in which k satisfies the reproducing property defined in Definition 3.1.2. It is also known as reproducing kernel Hilbert space (RKHS).*

See (Wainwright, 2019, Theorem 12.11) for proof.

Similarly, we can have a *vector-valued RKHS* (vv-RKHS) by specifying a valid matrix-valued PSD kernel function. The definition is provided below. Learning vector-valued functions in a vv-RKHS has been used in the literature of machine learning (Micchelli and Pontil, 2004; Evgeniou et al., 2005; Micchelli and Pontil, 2005; Minh et al., 2016) since it allows us to learn multiple tasks jointly, e.g., vector-valued regularized least squares (Micchelli and Pontil, 2005) and vector-valued multi-view support vector machines (Minh et al., 2016).

Definition 3.1.3 (Matrix-valued Kernels)**.** Suppose that $\mathcal{W}$ is a real separable Hilbert space with inner product $\langle , \rangle_{\mathcal{W}}$ and $\mathcal{L}(\mathcal{W})$ is the Banach space of bounded linear operators on $\mathcal{W}$. A bivariate function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{W})$ is said to be a valid matrix-valued positive semi-definite kernel function if $K(x, y) = K(y, x)^\top$ for $\forall x, y \in \mathcal{X}$ and $\sum_{i=1}^{N} \sum_{j=1}^{N} \langle a_i, K(x_i, x_j) a_j \rangle_{\mathcal{W}} \geq 0$ for every finite set of points $\{x_i\}_{i=1}^{N}$ in $\mathcal{X}$ and $\{a_i\}_{i=1}^{N}$ in $\mathcal{W}$.

One example of $\mathcal{W}$ is $\mathbb{R}^T$. Note that, when $\mathcal{W}$ is $\mathbb{R}$, this recovers the case of scalar-valued positive semidefinite kernel.

Theorem 3.1.2. *Given such a $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{W})$, there exists a unique (up to an isometry) $\mathcal{W}$-valued RKHS in which K satisfies the reproducing property defined in Definition 3.1.2.*

See Micchelli and Pontil (2005, Theorem 2.1) for proof. When $\mathcal{W}$ is $\mathbb{R}^T$, one example of K is a separable kernel $K(x, y) = Bk(x, y)$ where B is a positive semi-definite matrix and k is a scalar-valued kernel. B can be regarded as the relationship among tasks since it is a kernel on the response variables. If B is unknown, Ciliberto et al. (2015) showed that one could estimate B by block-coordinate descent. This is particular useful when the relationship among several related tasks is unknown. This can also benefit the proposed *vector-valued control variates*; more details will be presented in Section 4.2.

3.1.2 Kernel Methods in Statistical Learning

In this section, we will present details of Gaussian processes regression and kernel ridge regression, which are widely used in statistical machine learning.

Gaussian Processes Gaussian process (GP) (Rasmussen, 2003) is one of most influential Bayesian non-parametric methods in statistical learning. It has been widely used in the fields of regression, Bayesian quadrature, active learning and uncertainty quantification. By assuming a GP prior on the function $f \sim \mathcal{GP}(m, k)$ with a mean function $m : \mathcal{X} \to \mathbb{R}$ and a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and conditioning on a finite set of observations $\{x_i, y_i\}_{i=1}^N$ where $y_i = f(x_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, we have $Y \sim \mathcal{N}(m(X), k(X, X))$ where X is a matrix with the i-th row $x_i^\top$, Y is a column vector with the i-th element y_i, $m(X)$ is a column vector with the i-th element $m(x_i)$ and $k(X, X)$ is a square matrix with the (i, j)-th element $k(x_i, x_j)$. As a result of Gaussian conditioning, GP provides tractable posteriors of $f|\{x_i, y_i\}_{i=1}^N \sim \mathcal{GP}(\tilde{m}_{\sigma^2}, \tilde{k}_{\sigma^2})$. When ϵ_i degrades to zero, this is the setting of interpolation. In this case, we will have $f|\{x_i, y_i\}_{i=1}^N \sim \mathcal{GP}(\tilde{m}_0, \tilde{k}_0)$.

Both of the two scenarios can be unified using the following equations:

$$\tilde{m}_{\sigma^2}(x^*) = m(x^*) + k(x^*, X)(k(X, X) + \sigma^2 I_N)^{-1}(Y - m(X))$$
$$\tilde{k}_{\sigma^2}(x^*, z^*) = k(x^*, z^*) - k(x^*, X)(k(X, X) + \sigma^2 I_N)^{-1}k(X, z^*),$$

where I_N is an indentity matrix of size N, $k(x^*, X) = (k(x^*, x_1), \ldots, k(x^*, x_m))^\top$ and $k(X, x^*) = k(x^*, X)^\top$.

Kernel Ridge Regression and Kernel Interpolation Though the above expressions are derived by Gaussian conditioning from a probabilistic point of view, it can also be derived by non-probabilistic functional approximation in the reproducing kernel Hilbert spaces $\mathcal{H}_k$ associated with the kernel k of the GP. Given observations $\{x_i, y_i\}_{i=1}^N$, the objective of kernel Ridge regression is:

$$\hat{f} := \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^N J(f(x_i), y_i) + \lambda \|f\|^2_{\mathcal{H}_k}, \tag{3.2}$$

with $J(f(x_i), y_i) := (f(x_i) - y_i)^2$ and $\lambda > 0$. The solution to the above objective is unique and is given by:

$$\hat{f}(x) = k(x, X)(k(X, X) + N\lambda I_N)^{-1}Y,$$

where $k(x, X) := (k(x, x_1), \ldots, k(x, x_N))$. In the noise-free case, i.e. without observation random error, $y_i = f(x_i)$, this corresponds to kernel interpolation. The kernel interpolation estimator is given by (3.2) with $\lambda = 0$, i.e.,

$$\hat{f}(x) = k(x, X)k(X, X)^{-1}Y.$$

See (Berlinet and Thomas-Agnan, 2011; Kanagawa et al., 2018) for more details and discussions of the connection between kernel ridge regression, kernel interpolation and Gaussian processes.

3.2 Stein's Method

Stein's method (Chen et al., 2010) can be used to construct classes of functions that have zero mean. Therefore, it has been widely used in control variates (Mira et al., 2013; Si et al., 2021; Oates et al., 2017). It is also closely related to the proposed *vector-valued control variates* in Chapter 4 and the proposed *meta-learning control variates* in Chapter 5. Therefore, in this section, we provide necessary background knowledge of Stein's method and its applications in the fields of computational statistics and machine learning; see (Anastasiou et al., 2023) for a more detailed review.

3.2.1 Stein Identity and Stein Operators

In this section, we will give definitions of Stein identity in Definition 3.2.1 and then introduce the *Langevin Stein operator* in Definition 3.2.2. They are also the key components of the proposed methods *vector-valued control variates* in Chapter 4 and *meta-learning control variates* in Chapter 5.

Definition 3.2.1 (Stein identity)**.** A Stein class of a distribution Π is a class of functions $\mathcal{U}$ associated to a Stein operator $\mathcal{S}_\Pi$, such that the Stein identity holds: $\Pi[\mathcal{S}_\Pi[u]] = 0 \ , \forall u \in \mathcal{U}$.

Definition 3.2.2 (Langevin Stein Operator)**.** For regular scalar-valued functions $u : \mathcal{X} \to \mathbb{R}$, the *second-order Langevin Stein operator* is,

$$\mathcal{L}''_\Pi[u](x) := \Delta_x u(x) + \nabla_x u(x) \cdot \nabla_x \log \pi(x), \tag{3.3}$$

where $\Delta_x u(x) := \sum_{j=1}^d \frac{\partial^2 u(x)}{\partial x_j^2}$, $\nabla_x u(x) := (\frac{\partial u(x)}{\partial x_1}, \ldots, \frac{\partial u(x)}{\partial x_d})^\top$ and $a \cdot b$ is the inner dot product for two equally sized vectors a and b. While, for regular vector-valued functions $u : \mathcal{X} \to \mathbb{R}^d$, the *first-order Langevin Stein operator* is,

$$\mathcal{L}'_\Pi[u](x) := \nabla_x \cdot u(x) + u(x) \cdot \nabla_x \log \pi(x), \tag{3.4}$$

where $\nabla_x \cdot u(x) := \sum_{j=1}^d \frac{\partial (u(x))_j}{\partial x_j}$.

With $\mathcal{L}_{\Pi}^{'}$, to check if a function $f : \mathbb{R}^d \to \mathbb{R}^d$ is in the Stein class of Π which admits a continuous differentiable density π with support $\mathcal{X} \subseteq \mathbb{R}^d$, we only need to check each element of f_j is smooth and satisfies $\int_{x\in\mathcal{X}} \nabla_x(f_j(x)\pi(x))dx = 0$ for all $j \in \{1, \ldots, d\}$ (Liu et al., 2016). With $\mathcal{L}_{\Pi}^{''}$, if $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable, $\log \pi$ is continuously differentiable and $\|\nabla_x f(x)\| \leq C\|x\|^{-\delta}\pi(x)^{-1}$ for some constant $C \in \mathbb{R}$ and $\delta > d - 1$, then f is in the Stein class of Π (South et al., 2022b). In addition, evaluation of these operators requires only point-wise evaluation of $\nabla_x \log \pi(x)$. It can also be used even when π involves an unknown normalisation constant, i.e. $\pi = \tilde{\pi}/Z$ where $\tilde{\pi}$ is only known point-wisely and $Z > 0$ is a intractable normalising constant. This is because that $\nabla \log \tilde{\pi} = \nabla \log \pi - \nabla \log Z = \nabla \log \pi$. This is a significant advantage for many applications, e.g. when the distribution of interest Π is a Bayesian posterior.

3.2.2 Stein's Operators on RKHSs

Stein's operators on RKHSs were firstly proposed and used by Liu et al. (2016); Oates et al. (2017), and then several applications and extensions have been developed in Liu and Wang (2016); Wang et al. (2019a); Gorham et al. (2019); Barp et al. (2019); Singhal et al. (2019); Matsubara et al. (2021). Here, we only present essentials of Stein's operators on RKHSs, we refer readers to (Oates et al., 2017; Liu et al., 2016; Oates et al., 2019) for more details.

Definition 3.2.3 (Scalar-valued Stein Kernels (Oates et al., 2017))**.** The image of $\mathcal{U} := \mathcal{H}_k^d := \mathcal{H}_k \times \ldots \times \mathcal{H}_k$ under $\mathcal{L}_{\Pi}^{'}$ is a RKHS $\mathcal{H}_{k_0}$ induced by the reproducing kernel k_0, which is given by

$$\begin{aligned} k_0(x, x') = &(\nabla_x \cdot \nabla_{x'})k(x, x') + (\nabla_x \log \pi(x)) \cdot (\nabla_{x'} \log \pi(x'))k(x, x') \\ &+ (\nabla_x \log \pi(x)) \cdot (\nabla_{x'} k(x, x')) + (\nabla_{x'} \log \pi(x')) \cdot (\nabla_x k(x, x')). \end{aligned} \quad (3.5)$$

The kernel mean $\mathbb{E}_{x\sim\Pi}[k_0(\cdot, x)] = 0$ almost everywhere in $\mathcal{X}$ (Oates et al., 2017, 2019).

One example is that the well-known squared-exponential kernel $k(x, x') = \exp(-\frac{\|x-x'\|_2^2}{l^2})$ with $l > 0$ is in the Stein class of distributions with smooth densities

with support $\mathcal{X} = \mathbb{R}^d$. The conditions for a base kernel $k(x, x')$ in the Stein class of a distribution Π have been discussed in (Oates et al., 2017, 2019); see also (Liu et al., 2016, Definition. 2.1 & 3.4 and Proposition 3.5). That is, a kernel k is said to be in the Stein class of Π if k has continuous second order partial derivatives, and for any fixed x, $k(\cdot, x)$ and $k(x, \cdot)$ are in the Stein class of Π.

3.2.3 Applications of Stein's method in Statistical Learning

In this thesis, Stein's method is mainly used to construct control variates, which is presented in details in Section 3.3, Chapter 4 and Chapter 5. Stein's method has also been widely used in computational statistics and machine learning. In this section, we will briefly discuss recent applications of Stein's method.

Variational Inference with Stein's Method Stein's method has also been used for statistical inference. One representative work in the field is Stein variational gradient descent (SVGD) (Liu and Wang, 2016). Other variants include SVGD without gradient (Han and Liu, 2018) (using surrogate gradient), message-passing SVGD (Zhuo et al., 2018) (using Markov blanket to solve collapse in high-dimensional scenarios) and Stein discrepancy for energy-based models (Barp et al., 2019; Grathwohl et al., 2020). Connection between SVGD and black-box variational inference (BBVI) has been discussed in (Chu et al., 2020): equivalence is established when SVGD uses the neural tangent kernel. SVGD has also been combined with amortized varitional inferece, e.g., Stein variational auto-encoder (Feng et al., 2017; Pu et al., 2017). It is also closely linked to measure transport (Fisher et al., 2021) and gradient flow (Liu, 2017).

Kernel Stein Discrepancy Stein's method has also been used for construction of kernel Stein discrepancy (KSD) which has applied in many fields. Liu et al. (2016) employ KSD for goodness-of-fit tests and Chwialkowski et al. (2016) establish theoretical consistency for these tests. Gorham and Mackey (2017) employ KSD to measure sample quality with KSD. Barp et al. (2019) establish theoretic results for minimum Stein discrepancy estimators when maximum likelihood estimators are infeasible. Matsubara et al. (2021) provide a generalized Bayesian inference framework with kernel Stein discrepancy. It is also possible to perform efficient sampling

or post-processing MCMC samples with KSD. These include Stein points (Chen et al., 2018, 2019a) (selecting samples sequentially by minimising KSD between a candidate point and existing selected samples in a greedy way) and Stein thinning (Riabiz et al., 2022) (optimally thinning the output of MCMC algorithms by minimising KSD greedily).

3.3 Scalar-valued Control Variates

In this section, we will review existing control variates methods and discuss the key steps of constructing and selecting effective control variates. The existing control variates methods aim to estimate only one single integral at a time, which is also referred to as scalar-valued control variates (sv-CVs). These sv-CVs methods are post-processing tools of Monte Carlo methods which can provide us with more accurate estimators with the same number of data points. This can be particularly useful when it is expensive to obtain samples and/or to evaluate the function of interest, e.g., when the likelihood is expensive.

3.3.1 Control Variates

In Chapter 4 and Chapter 5, we consider the integrand of interest f belonging to the space of Π-square-integrable functions on $\mathcal{X}$. This space is denoted as $\mathcal{L}^2(\Pi) = \{f : \mathcal{X} \to \mathbb{R} \text{ s.t. } \Pi[f^2] < \infty\}$. This assumption is crucial to guarantee the existence and finiteness of the variance of the target integrand f under distribution Π which is given by $\mathbb{V}_\Pi[f] := \Pi[f^2] - (\Pi[f])^2$.

The Monte Carlo estimator is the average of integrand values evaluated at N independent and identically distributed (IID) samples from Π, $\{x_i\}_{i=1}^N$, which is given by

$$\hat{\Pi}^{\text{MC}}[f] = \frac{1}{N}\sum_{i=1}^N f(x_i). \tag{3.6}$$

Since we assume that the integrand $f \in \mathcal{L}^2(\Pi)$, the Monte Carlo estimator defined

above satisfies the following central limit theorem:

$$\sqrt{N}\left(\hat{\Pi}^{\text{MC}}[f] - \Pi[f]\right) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{V}_{\Pi}[f]\right),$$

where $\xrightarrow{d}$ means converge in distribution.

The above theorem tells that the variance $\mathbb{V}_{\Pi}[f]$ plays a vital role in the performance of MC estimators. When $\mathbb{V}_{\Pi}[f]$ is large, MC estimators often require a large number of samples and integrand evaluations to ensure a satisfactory accuracy. This is also why many CVs methods take minimising $\mathbb{V}_{\Pi}[f]$ as the goal. This can also be generalised for Markov chain Monte Carlo (MCMC) (Dellaportas and Kontoyiannis, 2012; Belomestny et al., 2020, 2021; Alexopoulos et al., 2023) and quasi-Monte Carlo (Hickernell et al., 2005), e.g., to minimise the *asymptotic variance*. For instance, Oates and Girolami (2016) proposed kernel-based CVs for quasi-Monte Carlo specifically, in addition to the previous work for Monte Carlo (Oates et al., 2017). In this thesis, we will focus on the MC variance $\mathbb{V}_{\Pi}[f]$.

As discussed above, we want to have an estimator of $\Pi[f]$ but with a reduced variance. A widely adopted approach is to identity a function $g \in \mathcal{L}^2(\Pi)$, also known as a control variate, such that $\mathbb{V}_{\Pi}[f-g]$ is much smaller than $\mathbb{V}_{\Pi}[f]$. Then, we can use the sum of a MC estimator of $\Pi[f-g]$ and $\Pi[g]$ to be an estimator of the original integral of interest $\Pi[f]$. This, however, requires $\Pi[g] = 0$ or a tractable $\Pi[g]$ (e.g. $\Pi[g]$ is a constant β such that $\Pi[f-g] + \beta = \Pi[f]$) since we need to compensate for the subtraction of g from f in the integral. With a subset $\{x_i, f(x_i)\}_{i=1}^{m}$ (also called the training set of CVs) of all N samples and the corresponding integrand evaluations, we can select an effective CV $\hat{g}_m$ that minimises $\mathbb{V}_{\Pi}[f-g]$. We can then regard $f - \hat{g}_m$ as the new integrand of interest and use the remaining $N - m$ samples and the associated functional evaluations to construct a MC estimator of $\Pi[f - \hat{g}_m]$. Then, the CV estimator of the original integral of

interest $\Pi[f]$ is given by,

$$\begin{aligned}\hat{\Pi}^{\text{CV}}[f] &:= \hat{\Pi}^{\text{MC}}[f - \hat{g}_m] + \Pi[\hat{g}_m] \\ &= \tfrac{1}{N-m} \textstyle\sum_{i=m+1}^{N} \left(f(x_i) - \hat{g}_m(x_i)\right) + \Pi[\hat{g}_m].\end{aligned} \tag{3.7}$$

Note that, conditioning on the training set $\{x_i, f(x_i)\}_{i=1}^m$, we can derive a central limit theorem for $\hat{\Pi}^{\text{CV}}[f]$ with $\mathbb{V}_\Pi[f - \hat{g}_m]$ in place of $\mathbb{V}_\Pi[f]$. That is,

$$\sqrt{N-m}\left(\hat{\Pi}^{\text{CV}}[f] - \Pi[f]\right) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{V}_\Pi[f - \hat{g}_m]\right).$$

This tells us that when $\hat{g}_m$ approximates f well, the variance $\mathbb{V}_\Pi[f - \hat{g}_m]$ is very close to zero. Then, the CV estimator $\hat{\Pi}^{\text{CV}}[f]$ tends to be much more accurate than the raw MC estimator $\hat{\Pi}^{\text{MC}}[f]$.

In the following section, we will detail how to construct and select an effective CV from the data.

3.3.2 Constructing and Selecting Control Variates

As discussed in previous section, there are two key steps if we want to us control variates for Monte Carlo integration:

(i). Constructing classes of *known*-mean functions $\mathcal{G}$, e.g. $\Pi[g] = 0$ for $\forall g \in \mathcal{G}$;

(ii). Selecting an effective control variate $g^* \in \mathcal{G}$.

Zero-mean Functions by Stein's method The first challenge is that it is required a function class $\mathcal{G}$ such that for all $g \in \mathcal{G}$ have *known*-mean under the distribution of interest, Π. Though *ad-hoc* approaches, such as Taylor expansions of f (Paisley et al., 2012; Wang et al., 2013), can be used when Π is relatively simple, this is usually a challenge whenever Π is a more complex density, such as can be encountered in a Bayesian inference task. One way forward is to use Stein's method discussed in Section 3.2. That is, we can construct $g := \mathcal{S}_\Pi[u]$ by applying a Stein operator $\mathcal{S}_\Pi$ (e.g. $\mathcal{L}'_\Pi$) to $u \in \mathcal{U}$ such that $\Pi[g] = 0$. CVs constructed in this way are known as *Stein-based* CVs.

Selecting Control Variates When a family of control variates $\mathcal{G}$ has been identified, we need to select an effective CV from this family for the integration task of interest. In general, there are two ways to select the optimal g^*, either through closed form solutions (e.g., Assaraf and Caffarel, 1999; Mira et al., 2013; Oates et al., 2017) or stochastic optimization algorithms (e.g., Si et al., 2021; Wan et al., 2019). One typical choice of the objective of selecting an ideal CV $g^* \in \mathcal{G}$ is to minimize the variance,

$$g^* \in \arg\min J(g) := \Pi\left[(f - g - \Pi[f])^2\right] \tag{3.8}$$

In this thesis, we will limit ourselves to parametric families, and will aim to identify a good parameter value so that the variance of the CV estimator is minimised; see Section 3.3.3.2, Chapter 4 and Chapter 5 for more specific details. Let $g_{\beta,\theta}(x) = \beta + g_\theta(x)$ where θ consists of parameters determining the zero-mean Stein-based CV g_θ, and there is an additional parameter β that will be used to approximate $\Pi[f]$. Given a dataset $S = \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$, and following the framework of empirical loss minimisation, the parameter θ can be estimated by minimising

$$J_S(\theta, \beta) := \tfrac{1}{m} \textstyle\sum_{i=1}^m \left(f(x_i) - g_{\beta,\theta}(x_i)\right)^2.$$

The value of β minimising this objective is a consistent estimator for $\Pi[f]$ in the $m \to \infty$ limit. To avoid over-fitting when m is small, penalised objectives have also been used in practice (Oates et al., 2017; South et al., 2022a; Wan et al., 2019; Si et al., 2021). Determining the strength of the penalty can be very challenging.

CV Estimators In general, we split the dataset of size $N = m + n$ of an integral estimation task into two parts, $S := \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$ and $Q := \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=m+1}^N$. Once $g_{\hat{\beta},\hat{\theta}}$ is selected with the first dataset S, the remaining n samples are used to construct unbiased estimators of $\Pi[f]$ (as long as the

samples are IID), given by

$$\hat{\Pi}^{\text{CV}}[f] = \hat{\Pi}^{\text{MC}}\left[f - g_{\hat{\beta},\hat{\theta}}\right] = \frac{1}{N-m}\sum_{i=m+1}^{N}\left(f(x_i) - g_{\hat{\beta},\hat{\theta}}(x_i)\right) + \hat{\beta}. \tag{3.9}$$

Remark 3.3.1 (Alternative Objectives of Control Variates)**.** *Note that the objectives used to select g^* matter. The objective in* (3.8) *is known as ordinary least squares Monte Carlo. Extra penalty terms on $g \in \mathcal{G}$ would result in various variants. For instance, when including a L_1-penalty term on θ, this gives us* Lasso Monte Carlo*, one can perform control variates selection and provide a control variate estimator simultaneously; see (South et al., 2022a; Leluc et al., 2021) for a detailed discussion.*

Remark 3.3.2 (Control Variates Estimator)**.** *As shown above, $\Pi[f]$ can be included into the objective and learnt with other parameters simultaneously. For instance, when $g_\theta(x) = \sum_{j=1}^{d}\theta_j\phi_j(x)$ (Assaraf and Caffarel, 1999; Mira et al., 2013) where ϕ_j are some control variates, $\Pi[f]$ can be regarded as an intercept β in the regression with f as the response and $\{\phi_j\}_{j=1}^{d}$ as the covariates. This is also the case when we use a RKHS $\mathcal{H}_{k_0}$ with k_0 be a Stein kernel. Given a training dataset $\{x_i, \nabla\log\pi(x_i), f(x_i)\}_{i=1}^{m}$ and with the objective of kernel Ridge regression, the optimal $g \in \mathcal{H}_{k_0}$ has the form of: $g_\theta(x) = \sum_{i=1}^{m}\theta_i k_0(x, x_i)$ where $\theta := (\theta_1, \ldots, \theta_m)^\top \in \mathbb{R}^m$. We can take $g_{\beta,\theta}(x) = \sum_{i=1}^{m}\theta_i k_0(x, x_i) + \beta$ where now $(\beta, \theta_1, \ldots, \theta_m)^\top \in \mathbb{R}^{m+1}$. This will result in a biased and consistent estimator of $\Pi[f]$ given by β when the sample size m is finite. However, this biased estimator will be much more accurate if the problem of functional approximation can be solved at a faster rate than the Monte Carlo convergence rate.*

3.3.3 Choices of $\mathcal{U}$

In the previous section, the framework of existing CVs methods is discussed. In this section, we will present relevant existing Stein-based CVs methods, including polynomial CVs, kernel-based CVs and neural CVs, which consider different choices of $\mathcal{U}$. We also compare the computational complexity and properties of the CVs dis-

cussed in this section in Table 3.1; see a more detailed discussion in (South et al., 2022c).

3.3.3.1 Polynomial Control Variates

Polynomial control variates (Mira et al., 2013; Assaraf and Caffarel, 1999) utilize polynomials to be $\mathcal{U}$, e.g., first-order or second-order polynomials. For first order polynomials $u_\theta(x) = \sum_{i=1}^d \theta_i x^i$ where $\theta = (\theta_1, \ldots, \theta_d)^\top$, CVs are constructed as the following,

$$g_\theta(x) = \mathcal{L}'_\Pi[u_\theta](x) = \sum_{i=1}^d \theta_i \partial^i \log \pi(x) = \theta^\top \nabla_x \log \pi(x), \tag{3.10}$$

where $\partial^i \log \pi(x) := \frac{\partial \log \pi(x)}{\partial x^i}$. Similarly, for second order polynomials $u_\theta(x) = \sum_{i=1}^d \theta_i x^i + \sum_{i=1}^d \sum_{j \geq i}^d \theta_{ij} x^i x^j$ where $\theta = (\theta_1, \ldots, \theta_d, \theta_{11}, \theta_{12}, \ldots, \theta_{dd})^\top$, CVs are constructed as the following,

$$g_\theta(x) = \mathcal{L}'_\Pi[u_\theta](x) = 2\sum_{i=1}^d \theta_{ii} + \sum_{i=1}^d \sum_{j \geq i}^d (\theta_i + \theta_{ij} x^j) \partial^i \log \pi(x). \tag{3.11}$$

However, though polynomial control variates are simple to compute, they are not bias-correcting, which means the resulting estimators are not shown to be consistent in a biased-sampling setting as discussed in South et al. (2022c). Meanwhile, polynomials CVs sometimes are not flexible enough and can have a bad fit to complex integrands. It also requires us to determine a proper value of the degrees of polynomials.

3.3.3.2 Kernel-based Control Variates

Unlike polynomial CVs, kernel-based control variates (Oates et al., 2017, 2019; South et al., 2022b) approximate the integrand in a non-parametric way. In addition, one particular benefit of control functionals (Oates et al., 2017, 2019) and semi-exact control functionals (South et al., 2022b), is that these two methods are bias-correcting (South et al., 2022b,c) even in a biased-sampling setting.

Control Functionals One representative work is control functionals (Oates et al., 2017, 2019; Oates and Girolami, 2016). As discussed in Definition 3.2.3 and pre-

vious sections, control functional estimators $\hat{\Pi}^{\text{CF}}[f]$ are constructed based on the particular choice of reproducing kernel Hilbert space $\mathcal{H}_{k_0}$.

$$\hat{\Pi}^{\text{CF}}[f] = \begin{cases} \frac{1}{n}\mathbf{1}_n^\top [f(Z) - k_0(Z,X)k_0(X,X)^{-1}(f(X) - \hat{\beta}\mathbf{1}_m)] & n > 0 \\ \frac{\mathbf{1}_m^\top k_0(X,X)^{-1} f(X)}{\mathbf{1}_m^\top k_0(X,X)^{-1}\mathbf{1}_m} =: \hat{\beta} & n = 0 \end{cases} \tag{3.12}$$

where $X = \{x_i\}_{i=1}^m, Z = \{z_j\}_{j=m+1}^N, (k_0(X,Z))_{i,j} = k_0(x_i, z_j)$. The selected CV is: $\hat{g}_m(z) = k_0(z,X)k_0(X,X)^{-1}[f(X) - \hat{\beta}\mathbf{1}_m]$ which can be used to construct unbiased control variate estimators if we have an additional dataset of size n as demonstrated in the first row of (3.12). Meanwhile, $\hat{\beta}$ can be used as a biased estimator of $\Pi[f]$ which often has a lower mean squared error typically (Oates et al., 2017). However, the computational cost for selecting effective CVs with control functionals is $\mathcal{O}(dm^2 + m^3)$, which can be prohibitive when m is large.

Stochastic Optimization of Kernel-based Control Variates Given m observations $\{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$, it is known that an interpolant in $\mathcal{H}_{k_0}$ should have the form of $g_m(x) = \sum_{i=1}^m \theta_i k_0(x, x_i)$. Therefore, we can use stochastic optimization tools to learn $(\beta, \{\theta_i\}_{i=1}^m)^\top$; see also Oates et al. (2017); Si et al. (2021) such that the computational cost of selecting effective CVs can be reduced; see Table 3.1 for more details. This can be quite useful when the number of samples (and the corresponding function evaluations) is large.

Other Variants More generally, we can combine different types of control variates. South et al. (2022b) propose to use a hybrid of polynomials and kernels as control variates, which are known as semi-exact control functionals (SECFs). By using kernel grammar (Duvenaud, 2014), combinations of kernels can also be used to construct kernel-based control variates (Si et al., 2021).

Motivation for Chapter 4 Existing control variates methods have not yet considered to learn a joint estimator for multiple related integral estimation tasks. To this end, we propose *vector-valued control variates* in Chapter 4 through the lens of reproducing kernel Hilbert spaces induced by novel matrix-valued Stein kernels K_0, which are capable of learning a joint estimator for related tasks and provide us

with more accurate estimators.

3.3.3.3 Neural Control Variates

Another popular choice of $\mathcal{U}$ is a set of neural networks (Wan et al., 2019; Si et al., 2021), which is referred as *neural control variates* (Neural CVs). The rationale of this choice stems from that neural networks are able to approximate complex functions well, but have a fixed number of parameters, and thus a fixed memory and computational cost unlike kernel-based control variates. Meanwhile, it is also convenient to compute $\Delta \cdot u_\theta(x)$ and $\nabla_x u_\theta(x)$ by using AUTOGRAD mechanism in modern machine learning tools like PYTORCH(Paszke et al., 2019). Benefiting from stochastic optimisation and neural networks, learning the optimal control variates with (3.8) and evaluating control variates estimators with (3.9) are convenient and fast. However, due to non-convexity and complicate structures of neural networks, Neural CVs have not been shown to be bias-correcting yet.

Motivation for Chapter 5 However, as shown in Si et al. (2021), neural control variates require a large number of samples (often thousands of samples) to achieve satisfactory performance and require re-training the neural network for each individual integral estimation task. To solve these problems, we propose *meta-learning control variates* in Chapter 5 which is capable of constructing control variates at scale, sharing information across a large number of tasks and achieve better performance even with a very small number of samples (and corresponding function evaluations) per task, e.g., 5.

Table 3.1: Properties of The Existing CVs That Have Been Discussed. d is the dimension of x, k is the order of polynomials, p is the number of parameters, N is the total number of data points, m is the number of data points used to select an effective CV, $\tilde{m}$ is the size of each mini-batch and L is the total number of iterations.

Method	Computational complexity	Bias-correcting	Super-$\sqrt{m}$ convergence
Poly. CVs (Mira et al., 2013)	$\mathcal{O}(Nd + d^{3k} + md^{2k})$	No	No
CFs (Oates et al., 2017)	$\mathcal{O}(Nd + m^3 + m^2 d)$	Yes	Yes
SECFs (South et al., 2022b)	$\mathcal{O}(Nd + m^3 + d^{3k})$	Yes	Yes
Stoch. Poly. CVs (Si et al., 2021)	$\mathcal{O}(Nd + d^k \tilde{m} L)$	No	No
Neural CVs (Wan et al., 2019)	$\mathcal{O}(Nd + p\tilde{m}L)$	Not shown	Not shown

3.4 Relevant Work on Information Sharing Across Integral Estimation Tasks

The idea of sharing information across integral estimation tasks has been explored in a range of settings outside the framework of control variates. Each of these methods is built on specific assumptions and relies on the structure of the relationship across tasks. In this section, we will present and discuss these relevant methods which also consider sharing information across several integral estimation tasks.

Multi-task Averaging for Mean *Multi-task averaging* (Feldman et al., 2014; Marienwald et al., 2021) considers joint estimation of the means of several distributions with separate datasets. Though it is convenient to get an individual mean estimator for each distribution, joint estimators tend to have better performance. One early work is *James-Stein estimators* (Efron and Morris, 1977) for Gaussian distributions, which have a smaller summed mean squared error than that of individual estimators. However, these multi-task averaging methods (Feldman et al., 2014; Marienwald et al., 2021) only consider and exploit task relationship in terms of target distributions since the target integrands are fixed to be the mean of the associated distributions. The proposed *vector-valued control variates* in Chapter 4 and *meta-learning control variates* in Chapter 5 do not have such limitations as the target integrands are not restricted to be the mean.

Multi-task Learning for Monte Carlo *Multi-output Bayesian quadrature* (MoBQ) (Xi et al., 2018; Gessner et al., 2020) is a Bayesian probabilistic method for joint estimation of multiple related integrals. As an extension to Bayesian quadrature (BQ) (Briol et al., 2015), MoBQ is based on multi-output Gaussian processes. Instead of placing a GP prior for each individual f_j, these methods regard $f_1, \ldots, f_T$ as elements of a vector-valued integrand $f := (f_1, \ldots, f_T)^\top$ and place a multi-output GP prior on this vector-valued integrand f. Therefore, the relationship among the T integral estimation tasks can be modeled by specifying structures shared across outputs. Though MoBQ shows the potential to provide improved joint estimators of T related integrals, MoBQ suffers from a computational

cost between $\mathcal{O}(T^2)$ and $\mathcal{O}(T^6)$ in comparison to $\mathcal{O}(T)$ of BQ. This limit the use of these methods in cases when there is a large T number of integrals required to be estimated. Meanwhile, it is hard to use these methods for Bayesian inference. This is because BQ or MoBQ requires tractable kernel means $\int_{\mathcal{X}} k(x, x')\pi(x')dx'$, which is rare in practice. For instance, it is often the case that we only know the posterior π up-to an intractable constant Z, i.e., $\pi = \tilde{\pi}/Z$, in Bayesian inference. In these cases, these BQ methods cannot be used. The proposed *vector-valued control variates* in Chapter 4 and *meta-learning control variates* in Chapter 5 do not have such limitations.

Multilevel and Multi-fidelity Integration Multi-fidelity modelling means that we are approximating a target function f with approximations with varying levels of accuracy, which are denoted by $f_1, \ldots, f_T$; see (Peherstorfer et al., 2018) for a detailed review. In this field, methods provide estimates to $\Pi[f]$ including *multilevel Monte Carlo* (Giles, 2015), *multilevel Bayesian quadrature* (Li et al., 2023) and *multilevel control variates* (Nobile and Tesei, 2015; Fairbanks et al., 2017; Geraci et al., 2017; Li and Sun, 2023). These methods are usually based on a telescoping sum $\Pi[f] = \sum_{l=0}^{T} \Pi[f_l - f_{l-1}]$ where $f_{-1} := 0$ and $f_T := f$ and estimate $\Pi[f_l - f_{l-1}]$ for all levels $l = 1, \ldots, L$ by Monte Carlo, Bayesian quadrature or control variates. These methods are used for problems when T is small. In particular, the cost of function evaluation varies in different levels, and usually the evaluation cost of the low fidelity approximations is cheaper than that of the high fidelity approximations. Thus, these methods tend to allocate more samples for cheaper but less accurate approximations and fewer samples for expensive but more accurate approximations. This setting is different from what we consider in Chapter 4 and Chapter 5 of this thesis. Meanwhile, these methods are limited to the cases when we only have one target distribution, i.e., $\pi_1 = \cdots = \pi_T$ for all levels. The proposed *vector-valued control variates* in Chapter 4 and *meta-learning control variates* in Chapter 5 do not have such limitations.

Monte Carlo Methods for Parametric and Conditional Expectations *Parametric expectation* or *conditional expectation* methods (Krumscheid and Nobile,

2018; Alfonsi et al., 2023) consider to approximate $\mathbb{E}_{X\sim\pi(\cdot|\theta)}[f(X,\theta)]$ uniformly over some region of θ. Thus, these methods can be applied to the cases when the number of integral estimation tasks T is large. However, these methods often require assumptions on the smoothness of the values of these integrals when varying θ, which can be problematic when these assumptions are violated. This can an interesting future direction of control variates methods.

Importance Sampling Importance sampling is used when we want to estimate $\Pi[f]$ with samples from another related distribution Π' (also known as the importance distribution). This is achieved by re-weighting the samples from Π' according to the weights $\frac{\pi}{\pi'}$ where π and π' are corresponding probability densities. This is because $\Pi[f] = \mathbb{E}_{X\sim\Pi'}[\frac{\pi(X)}{\pi'(X)}f(X)]$. However, such an importance distribution Π' needs to be chosen carefully to ensure the resulting estimator has a low variance. Demange-Chryst et al. (2022) consider estimate multiple related integrals by choosing an importance distribution Π' that works well for multiple integral estimation tasks. However, choosing such a distribution Π' can be very challenging when the number of integral estimation tasks is large. The proposed *vector-valued control variates* in Chapter 4 and *meta-learning control variates* in Chapter 5 do not require to know such an importance distribution Π'.

3.5 Meta-learning

Humans are known to be able to learn new ideas or concepts from a small number of observations in a new task. This remarkable ability is on the basis of: excellent leverage of the learning experience from the past; adaption to novel ideas or concepts even with few samples from unseen tasks fast.

To mimic such ideal ability for machine learning algorithms, *meta-learning*, also known as learning to learn, has been proposed. Meta-learning methods have gained much attention and achieved rapid progress recent years, especially in the fields of image classification (Finn et al., 2017, 2019; Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018). In this section, we will introduce the necessary background knowledge of meta-learning with focus on the two branches of meta-

learning, *gradient-based meta-learning* and *metric-based meta-learning*, which are most relevant to the proposed methods in Chapter 6 and Chapter 5.

Definition 3.5.1 (Meta-Learning)**.** Suppose that we have a collection of tasks from some population distribution ρ. We seek for $\hat{f}$ that can be adapted rapidly to any testing task $\mathcal{T}$ after being trained through some meta-training process. That is,

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{T} \sim \rho} \mathbb{E}_{S,Q \sim \mathcal{T}}[J(f(S), Q)],$$

where J is some loss function, S is a training dataset of the task $\mathcal{T}$ while Q can be regarded as a testing set of $\mathcal{T}$ with labeled-samples from the task $\mathcal{T}$. S and Q are also known as the *support set* and the *query set* of the task $\mathcal{T}$. Here, f is firstly learnt on the support set S and then is tested on the query set Q. The overall goal is then to minimize the risk over the whole environment ρ. We will introduce the standard way, *episodic meta-training process*, of training meta-learning algorithms later in this chapter.

Definition 3.5.2 (Few-shot Meta-Learning)**.** Few-shot means that for each task $\mathcal{T}$, its support set S has a very small number of training points. We denote the sample size of a support set S by $|S|$. For convenience, we will use the term *few-shot learning* instead of *few-shot meta-learning* in the remaining content of this section.

One typical example of few-shot meta-learning is *C-way K-shot classification*. It is challenging since for each task only few labeled training samples are available. It is also used as a standard to evaluate the performance of different meta-learning algorithms for classification tasks.

Definition 3.5.3 (Few-shot Classification: C-way K-shot)**.** C-way K-shot few-shot classification means that each task $\mathcal{T}$ has C different classes in total. In particular, the support set S from $\mathcal{T}$ only has K labeled samples per class.

Episodic Meta-training A common way of training a meta-learning algorithm is the *episodic meta-training*, proposed by Vinyals et al. (2016). The basic concept of episodic meta-training is that the process of meta-training should be identical to

that of the meta-testing. That is, it requires to sample tasks and then sample the corresponding support sets and query sets for both the meta-training process and the meta-testing process. To make this clear, we summarize it in Algorithm 1. In Algorithm 1, the CONDITIONING step can be very general. For instance, given current task $\mathcal{T}_t$, we can use the loss on S_t (Finn et al., 2017) to adapt the model or use the feature map of the raw data from S_t (Snell et al., 2017) to be the input of the model. The UPDATE step usually utilizes the loss on the query set Q_t to update the meta-model.

Algorithm 1: Episodic Meta-training: Learn a Meta-model

Input: An Initial Meta-Model $\mathcal{M}_0$, CONDITIONING Rule, UPDATE Rule, Number of meta-iterations I_{tr}, Environment ρ.

for *i from 1 to* I_{tr} **do**
  Sample $\mathcal{T}_1, \ldots, \mathcal{T}_B$ from ρ.
  for $t \in \{1, \ldots, B\}$ **do**
    Sample S_t and Q_t for the task $\mathcal{T}_t$.
    /* Conditioning */
    CONDITIONING $\mathcal{M}$ on S_t: $\mathcal{M} \leftarrow \mathcal{M}(S_t)$.
    Evaluate the loss of $\mathcal{M}(S_t)$ on Q_t: $J(\mathcal{M}(S_t), Q_t)$
  /* Update */
  $\mathcal{M}_i \leftarrow \text{UPDATE}\left(\mathcal{M}_{i-1}, \sum_{t\in\{1,\ldots,B\}} J(\mathcal{M}_{i-1}(S_t), Q_t)\right)$.

Output: The optimized meta-model $\hat{\mathcal{M}} := \mathcal{M}_{I_{\text{tr}}}$.

Algorithm 2: Evaluation: Construct Task-specific Models from the Meta-model

Input: A task $\mathcal{T}_{t'}$ of interest, Meta-model $\hat{\mathcal{M}}$, CONDITIONING Rule.

Set the task-specific model $\mathcal{M}$ be the meta-model: $\mathcal{M} \leftarrow \hat{\mathcal{M}}$

CONDITIONING $\mathcal{M}$ on $S_{t'}$: $\mathcal{M} \leftarrow \mathcal{M}(S_{t'})$.

Output: Evaluate the performance of $\mathcal{M}(S_{t'})$ on $Q_{t'}$.

3.5.1 Gradient-based Meta-learning

Gradient-based meta-learning is a class of meta-learning algorithms that utilises gradient descent to leverage similarities and relationship among tasks for meta-learning. The most well-known work in this branch is *Model-agnostic Meta-learning* (*MAML*) (Finn et al., 2017). It aims to learn a globally shared initialization of parameters for all unseen new tasks. Within few steps of gradient descent

from the shared initialization, *MAML* has shown excellent performance to an unseen task from the environment. To learn such an initialisation, *MAML* proposes a bi-level optimisation scheme which includes an inner optimization loop and an outer optimization loop. Given a task $\mathcal{T}_t$, it restricts task-specific parameters ϕ^t be initialized at the meta-parameters γ, i.e. $\phi_0^t \leftarrow \gamma$. Then, ϕ^t is further optimized according to the loss $J(\mathcal{M}_{\phi^t}, S_t)$ of the model's prediction for the support set S_t of $\mathcal{T}_t$. This inner optimisation of ϕ^t can take a few gradient descent steps which involves $\phi_l^t \leftarrow \phi_{l-1}^t - \alpha \nabla_{\phi_{l-1}^t} J(\mathcal{M}_{\phi_{l-1}^t}, S_t)$ for $l = 1, \ldots, L$ and thus the final optimised task-specific parameter is ϕ_L^t. During the process of meta-training, since we also have access to the responses/labels of data points in Q_t, γ can be gradually updated by minimising the loss $J(\mathcal{M}_{\phi_L^t}, Q_t)$ for a series of tasks, where $J(\mathcal{M}_{\phi_L^t}, Q_t)$ is the loss of the model's prediction (after adaption on S_t) for the query set Q_t. The algorithm of *MAML* is summarised in Algorithm 3.

Algorithm 3: Model-agnostic Meta-learning (Finn et al., 2017)

Input: Model $\mathcal{M}_\gamma$, learning rates α and $\eta_1, \ldots, \eta_{I_{\text{tr}}}$, number of meta-iterations I_{tr}, environment ρ.

Initialize the parameters γ of $\mathcal{M}_\gamma$ with γ_0.
for *i from 1 to* I_{tr} **do**
  Sample a mini-batch of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_B$ from ρ.
  for $t \in \{1, \ldots, B\}$ **do**
    Sample S_t and Q_t for the task $\mathcal{T}_t$.
    Initialize $\phi_0^t \leftarrow \gamma_{i-1}$.
    /* Adaption */
    for l *from* 1 *to* L **do**
      $\phi_l^t \leftarrow \phi_{l-1}^t - \alpha \nabla_{\phi_{l-1}^t} J(\mathcal{M}_{\phi_{l-1}^t}, S_t)$.
  /* Update */
  $\gamma_i \leftarrow \gamma_{i-1} - \eta_i \nabla_{\gamma_{i-1}} \frac{1}{B} \sum_{t=1}^{B} J(\mathcal{M}_{\phi_L^t}, Q_t)$.

Output: Return $\mathcal{M}_{\gamma_{I_{\text{tr}}}}$.

Remark 3.5.1 (Unrolled Learning Objectives of MAML)**.** *It is possible to unroll the learning algorithm of MAML as shown in (Fallah et al., 2020; Ji et al., 2022) later. We will present more details of the unrolled learning objective of MAML in Chapter 5 and use it for control variates. We keep the original presentation of MAML here for the most general interpretation, based on which many probabilistic*

variants are proposed (which are discussed in the next paragraph). We will also introduce a novel probabilistic meta-learning algorithm in Chapter 6.

Remark 3.5.2 (Probabilistic Variants of MAML)**.** *Many gradient-based meta-learning algorithms are based on the particular structure of* MAML*, including* MAML-HB *(Grant et al., 2018),* BMAML *(Yoon et al., 2018),* PLATIPUS *(Finn et al., 2018),* VAMPIRE *(Nguyen et al., 2020),* Meta-Mixture *(Jerfel et al., 2019),* ABML *(Ravi and Beatson, 2019) and* VERSA *(Gordon et al., 2019). These probabilistic variants aim to learn the posteriors of neural networks' parameters for each task. For instance,* MAML-HB *links the connection between* MAML *and hierarchical Bayes while* PLATIPUS *aims to learn the joint posterior of the shared initialization γ and task-specific parameters ϕ^t conditional on the support set S_t of each task $\mathcal{T}_t$.*

3.5.2 Metric-based Meta-learning

Metric-based meta-learning methods use metric or similarity measurement as criteria to evaluate how similar an object from the query set is to the labeled objects in the support set of a task. Thus, these metric-based methods are well-suited for classification tasks. However, they can not be used for regression tasks in general. Some of these methods use well-defined metrics (e.g., cosine (Vinyals et al., 2016) and Euclidean distance (Snell et al., 2017; Allen et al., 2019)) while others do not (Sung et al., 2018). *Matching Network* (Vinyals et al., 2016), *Prototype Network* (Snell et al., 2017) and *Infinite Mixture Prototype Network* (Allen et al., 2019) utilize a specific metric (e.g. cosine or Euclidean distance) for all tasks from the environment and aim to learn to learn a proper feature map (e.g. in the form of a neural network) that can capture the discriminative information for all tasks. There are also some other non-standard metric-based methods, i.e., not using a proper metric. For instance, *Relation Network* (Sung et al., 2018) introduces a so-called *deep relation* network for pair-wise similarity measurement. Metric-based meta-learning methods expect that the learned feature map can generalize well to unseen new tasks from the environment.

Remark 3.5.3 (Probabilistic Interpretation of Metric-based Meta-learning)**.** *Some metric-based meta-learning methods can be interpreted from a probabilistic perspective. For instance, Prototype Network implicitly assumes that within a task each class follows a Gaussian distribution with a class-specific mean and an identity covariance matrix. Infinite Mixture Prototype Network allows multiple clusters within one class, which is achieved by using DP-means or Chinese restaurant process. It assumes that within each class each cluster follows a Gaussian distribution with a cluster-specific mean and a shared covariance matrix among all clusters.*

Chapter 4

Vector-valued Control Variates

Monte Carlo (MC) methods are often used to construct estimators for integrals. However, MC estimators are known to have large variances. To achieve a desired accuracy, these estimators often require a large number of samples. This can be infeasible when sampling from the target distributions or evaluation of integrands is expensive. This problem can be especially severe when multiple integrals need to be estimated. Control variates are post-processing tools for Monte Carlo estimators which largely reduce the variances and thus fewer samples and function evaluations are required. However, existing control variates only consider the cases of scalar-valued integrals. When we have multiple related integrals, existing control variates methods ignore the information shared across these related integration tasks since there is no information flow during the learning processes.

In this chapter, we will extend standard *scalar-valued integration* to *vector-valued integration*. We will show how to construct and select effective *vector-valued control variates* in this case. Related background knowledge has already been presented in Chapter 3. In Section 4.2 and Section 4.3, we formally present the proposed *vector-valued control variates* and demonstrate their superior performance empirically.

4.1 Introduction

A significant computational challenge in statistics and machine learning is the approximation of intractable integrals. Examples include the computation of posterior

moments, the model evidence (or marginal likelihood), Bayes factors, or integrating out latent variables. This has lead to the development of a range of Monte Carlo (MC) methods; see Green et al. (2015) for a review. Let $f : \mathcal{X} \to \mathbb{R}$ denote some integrand of interest, and Π some distribution with Lebesgue density π known up to an intractable normalisation constant. The integration task we consider in this chapter can be expressed as estimating

$$\Pi[f] := \int_{\mathbb{R}^d} f(x)\pi(x)\mathrm{d}x$$

using evaluations of the integrand at some points in the domain: $\{x_i, f(x_i)\}_{i=1}^N$. These evaluations are usually combined to create an estimate of $\Pi[f]$ of the form $\hat{\Pi}[f] = \frac{1}{N}\sum_{i=1}^N f(x_i)$. For example, when realisations are IID, this corresponds to a MC estimator. In that case, assuming that f is square-integrable with respect to Π (i.e. $\Pi[f^2] < \infty$), we can use the central limit theorem (CLT) to show that such estimators converge to $\Pi[f]$ as $N \to \infty$, and this convergence is then controlled by the asymptotic variance of the integrand f. Analogous results can also be obtained for MCMC realisations (Jones, 2004), in which case $\{x_i\}_{i=1}^N$ are realisations from a Markov Chain with invariant distribution Π, or for randomised quasi-Monte Carlo (Hickernell et al., 2005), in which case $\{x_i\}_{i=1}^N$ form some lattice or sequence filling some hypercube domain.

As discussed in Chapter 3, the main insight behind the concept of *control variate* (CV) is that it is often possible to instead use an estimator of $\Pi[f - g]$ for some $g : \mathcal{X} \to \mathbb{R}$. This is justified if $\Pi[g]$ is known, in which case we may use $\hat{\Pi}^{\mathrm{CV}}[f] := \hat{\Pi}[f - g] + \Pi[g]$. Furthermore, if g is chosen appropriately, the variance of the CLT for this new estimator will be much smaller than that of the original, and a smaller number of samples will be required to approximate $\Pi[f]$ at a given level of accuracy.

Suppose now that $N = (N_1, \ldots, N_T) \in \mathbb{N}^T$ ($T \in \mathbb{N}_+$) is a multi-index. In this paper, we will focus on cases where we have not one integration problem, but a sequence of integrands $f_t : \mathcal{X} \to \mathbb{R}$ and distributions Π_t for which we would like

to use $\{\{x_{tj}, f_t(x_{tj})\}_{j=1}^{N_t}\}_{t=1}^{T}$ to estimate

$$\Pi_t[f_t] := \int_{\mathbb{R}^d} f_t(x)\pi_t(x)\mathrm{d}x \quad \text{for } t \in [T], \tag{4.1}$$

where $[T] := \{1, \ldots, T\}$. This is a common situation in practice; for example, our paper considers the case of multifidelity modelling (Peherstorfer et al., 2018) where $f_1, \ldots, f_T$ may be a computationally expensive physical model f, and we might be interested in expectations of that model with respect to unknown parameters. Another example we will also study is when $\pi_1, \ldots, \pi_T$ are closely related posterior distributions, such as in the case of power posteriors (Friel and Pettitt, 2008).

Of course, the estimation of integrals in (4.1) could be tackled individually, and this is in fact the most common approach. However, the main insight from this paper is that if both the integrands and distributions are related across tasks, we can improve on this by sharing computation across these tasks. We propose to construct a CV to *jointly reduce the variance* of estimators for these integrals and hence obtain a more accurate approximation. We will call such a function a *vector-valued control variate* (vv-CV). In order to encode the relationship between integration tasks, we will propose a flexible class of CVs based on interpolation in *reproducing kernel Hilbert space of vector-valued functions* (vv-RKHS). More precisely, we generalise existing constructions of Stein reproducing kernels to derive novel vv-RKHSs with the property that each output has mean zero.

We note that very few methods exist to tackle multiple integrals jointly. One exception is Xi et al. (2018), which also proposes an algorithm based on vv-RKHSs. However, that work is limited to cases where the kernel mean is known in closed-form, which is rarely possible in practice. In contrast, our vv-CVs are applicable so long as π_t is known up to an unknown constant and $\nabla_x \log \pi_t$ can be evaluated pointwise for all $t \in [T]$ (where $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_d)^\top$). This will usually be satisfied in Bayesian statistics, and is a requirement for the implementation of most gradient-based MCMC algorithms.

Notation Vectors $x \in \mathbb{R}^d$ are column vectors, $\|x\|_q = (\sum_{i=1}^d x_i^q)^{1/q}$ for $q \in \mathbb{N}$, and $\mathbf{1}_d = (1, \ldots, 1)^\top \in \mathbb{R}^d$. For a multi-index $m \in \mathbb{N}^d$, we write $|m| = \sum_{i=1}^d m_i$ for

its total degree. For a matrix $M \in \mathbb{R}^{p\times q}$, M_{ij} denotes the entry in row i and column j, $\|M\|_F^2 = \sum_{i=1}^{p}\sum_{j=1}^{q} M_{ij}^2$ is the Frobenius squared-norm, $\text{Tr}(M) = \sum_{i=1}^{m} M_{ii}$ is the trace, and $M^\dagger$ is the pseudo-inverse. I_m denotes the m-dimensional identity matrix, and S_+^m the set of symmetric strictly positive definite matrices in $\mathbb{R}^{m\times m}$. We denote by C^j the set of functions whose mixed partial derivatives of order at most j are continuous, and given a differentiable function g on $\mathbb{R}^{d_1}\times\mathbb{R}^{d_2}$, $\partial_x^r g(x,y)$ denotes its partial derivative in the r^{th}-coordinate of its first entry evaluated at (x,y).

4.2 The Proposed Method

In this chapter, we consider vector-valued RKHSs (Carmeli et al., 2006, 2010; Álvarez et al., 2012) to be $\mathcal{U}$ onto which Stein operators will be applied. The reason that we choose to use vector-valued RKHSs is because it allows us to encode relationships among integration tasks from the literature of multi-task kernel learning (Ciliberto et al., 2015). To start with, we give the definitions of integrals.

Definition 4.2.1 (Scalar-valued Integral)**.** Suppose $f : \mathcal{X} \to \mathbb{R}$ and Π is a distribution of interest.

$$\Pi[f] := \mathbb{E}_{X\sim\Pi}[f(X)] = \int_{\mathbb{R}^d} f(x)d\Pi(x) \tag{4.2}$$

Definition 4.2.2 (Vector-valued Integral)**.** Suppose $f : \mathcal{X} \to \mathbb{R}^T$ such that $f(x) = (f_1(x),\ldots,f_T(x))^\top$, and it is of interest to estimate of the following vector of integrals:

$$\Pi[f] := (\mathbb{E}_{X\sim\Pi_1}[f_1(X)],...,\mathbb{E}_{X\sim\Pi_T}[f_T(X)])^\top, \tag{4.3}$$

where Π is formally known as a *vector probability distribution*.

4.2.1 Construction Vector-valued RKHSs with Zero Means

Since now we are interested in a vector probability distribution $\Pi = (\Pi_1,\ldots,\Pi_T)^\top$, we need to define a generalised Stein identity, and also a generalised Stein operator $\mathcal{S}^{\text{vv}}$.

Definition 4.2.3 (Generalised Stein Identity and Generalised Stein Operator $\mathcal{S}^{vv}$)**.** Consider $\mathcal{H}_K$ which is a vv-RKHS with *matrix-valued reproducing kernel* (mv-kernel) $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{T\times T}$, and suppose that $K \in C^{1,1}(\mathcal{X} \times \mathcal{X})$. Furthermore, for suitably regular vv-functions $u : \mathcal{X} \to \mathbb{R}^T$, let

$$\mathcal{S}^{\text{vv}}[u] = (\mathcal{L}'_{\Pi_1}[u_1], \ldots, \mathcal{L}'_{\Pi_T}[u_T])^\top,$$

such that $\Pi_t \left[(\mathcal{S}^{\text{vv}}[u])_t\right] = 0$ for $\forall u \in \mathcal{U}$ and $\forall t \in [T]$. Thus, we have $\Pi[\mathcal{S}^{\text{vv}}[u]] = (0, \ldots, 0)^\top$.

Remark 4.2.1. *From Definition 4.2.3, the order matters since for any $t' \neq t$, we may not have $\Pi_{t'} \left[(\mathcal{S}^{vv}[u])_t\right] = 0$.*

This leads to a novel class of matrix-valued Stein reproducing kernels which is presented in the theorem below.

Theorem 4.2.1 (Matrix-valued Stein Reproducing Kernels)**.** *Consider the K and $\mathcal{S}^{vv}$ in Definition 4.2.3, the image of $\mathcal{H}_K^d = \mathcal{H}_K \times \ldots \times \mathcal{H}_K$ under $\mathcal{S}^{vv}$ is a vv-RKHS with kernel $K_0 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{T\times T}$:*

$$\begin{aligned}(K_0(x,y))_{tt'} = &\textstyle\sum_{r=1}^d \partial_x^r \partial_y^r K(x,y)_{tt'} + l_{t'r}(y)\partial_x^r K(x,y)_{tt'} \\ &+ l_{tr}(x)\partial_y^r K(x,y)_{tt'} + l_{tr}(x)l_{t'r}(y)K(x,y)_{tt'} \qquad \forall t, t' \in [T],\end{aligned}$$

where $l_{tr}(x) = \partial_x^r \log \pi_t(x)$. See Appendix A.2.3 for the proof.

Remark 4.2.2 (Evaluation of K_0)**.** *To evaluate K_0 at a pair of points (x, x'), normalization constants of $\{\Pi_t\}_{t=1}^T$ are not required but it is required to know $\nabla_x \log \pi_t(x)$ and $\nabla_{x'} \log \pi_t(x')$ for all tasks, i.e., $\forall t \in [T]$.*

The matrix-valued Stein kernel K_0 in Theorem 4.2.1 is very general. When the *base kernel* K is separable, the expression of K_0 will simplify a lot. Meanwhile, in this case, it allows us to encode the relationship among the sequence of integral estimation tasks explicitly.

Special Case I: Separable kernel K A matrix-valued kernel K is separable if it can be written as $K(x, y) = Bk(x, y)$ where k is a scalar-valued kernel and B is a positive semi-definite matrix of size $T \times T$ (i.e. $B \in S_+^T$). This allows us to encode and learn the relationship B among integral estimation tasks even when no prior knowledge of B is available. This simplifies the kernel K_0 in Theorem 4.2.1 and we have:

$$\begin{aligned}(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^{d} \partial_x^r \partial_y^r k(x,y) + l_{t'r}(y)\partial_x^r k(x,y) \\ + l_{tr}(x)\partial_y^r k(x,y) + l_{tr}(x)l_{t'r}(y)k(x,y) \qquad t, t' \in [T]. \end{aligned} \tag{4.4}$$

In this case, K_0 is not separable even though K is separable.

Example 4.2.1 (Illustration of Special Case I)**.** *To give a concrete example of K_0, here we choose $\Pi_1 = \mathcal{N}(0, 1)$, $\Pi_2 = \mathcal{N}(0, 1.25)$, $B_{11} = B_{22} = 1$ and $B_{12} = B_{21} = 0.1$. The first row corresponds to taking k to be a squared-exponential kernel, whereas the second and third row correspond to taking a polynomial kernel $k(x, y) = (x^\top y + 1)^l$ with $l = 1$ and $l = 2$ respectively. As shown in Figure 4.1, the two components of $1^\top K_0(x, y)$ are closely related. This is the key of vv-CVs.*

Special Case II: Separable kernel K with one single target distribution When all the distributions of interested are identical $\Pi_1 = \ldots = \Pi_T$, this further simplifies the evaluation of K_0 in Theorem 4.2.1 and gives,

$$(K_0(x, y))_{tt'} = B_{tt'} k_0(x, y) \qquad \forall t, t' \in [T]. \tag{4.5}$$

where k_0 is given in (3.5). And the scalar-valued Stein kernel k_0 can then be recovered by taking $T = 1$ and $B = 1$.

Form of Vector-valued Control Variates Given a dataset $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ where $\mathcal{D}_t = \{\{x_{tj}, f_1(x_{tj})\}_{j=1}^{m_t}$, and since we take $g \in \mathcal{H}_{K_0}$ to be the vector-valued

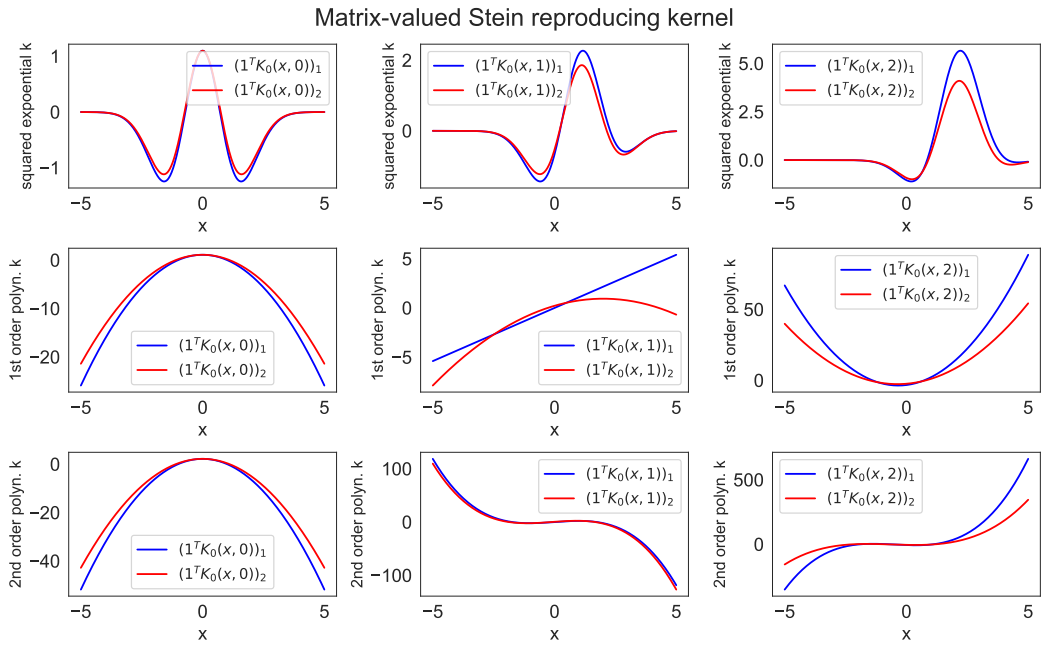

Figure 4.1: Illustration of a separable matrix-valued Stein kernel K_0 for $T = 2$ through projections with $\mathbf{1} = (1, 1)^\top$.

control variates, it has the form of,

$$g_\theta(x) = \sum_{t=1}^{T} \sum_{j=1}^{m_t} K_0(x, x_{tj})\theta_{tj}, \quad \text{where } \theta_{tj} \in \mathbb{R}^T \text{ for all } t \in [T], j \in [m_t]. \tag{4.6}$$

Remark 4.2.3 (Connection between vector-valued polynomials and vv-RKHSs with polynomial kernels)**.** *We now consider a vv-RKHS $\mathcal{H}_K$ specified by a matrix-valued kernel $K(x, x_i) = \tilde{B}k_{poly,l}(x, x_i) = \tilde{B}(\langle x, x_i\rangle + c)^l$. Conditioning on m samples, $f(x) = \sum_{i=1}^{m} \tilde{B}(\langle x, x_i\rangle + c)^l\tilde{\theta}_i$ for $f(x) \in \mathcal{H}_K$. Thus, $(f(x))_t = \sum_{i=1}^{m}\sum_{k=0}^{l} \tilde{B}_{t,\cdot}\tilde{\theta}_i\, C_l^k\, c^{l-k} \left(\sum_{j=1}^{d} x_j x_{ij}\right)^k$. A multivariate polynomial $(u_\theta(x))_t = \sum_{|\alpha|\le l} \left\{\theta_{t,\alpha} \prod_{j=1}^{d} x_j^{\alpha_j}\right\}$, where $\theta_{t,\alpha} \in \mathbb{R}$ and the subscript denotes the dependence on $\alpha = (\alpha_1, \ldots, \alpha_d)$ and the task index t. Under this formulation, the connection is then,*

$$\theta_{t,\alpha} = \sum_{i=1}^{m} \left(\sum_{t'=1}^{T} \tilde{B}_{t,t'}\tilde{\theta}_{it'}\right) \binom{l}{k} c^{l-k} \binom{k}{\alpha_1,\ldots,\alpha_d} \left[\prod_{j=1}^{d} x_{ij}^{\alpha_j}\right]$$

See Appendix A.2.2 for a detailed derivation.

Example 4.2.2 (Polynomial vv-CVs)**.** *Suppose that we have* $k_{poly,l}(x,y) = (\langle x,y\rangle + c)^l$ *and* $K(x,y) = Bk(x,y)$ *where* $B \in S_+^T$*. When* $l = 1$*, we obtain the matrix valued Stein reproducing kernel:*

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^d \left\{1 + l_{t'}^r(y)y_r + l_t^r(x)x_r + l_t^r(x)l_{t'}^r(y)\left(\langle x,y\rangle + c\right)\right\}.$$

When $l = 2$*, we get:*

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{j=1}^d \Big\{2x_r y_r + 2\left(\langle x,y\rangle + c\right) + 2y_r l_{t'}^r(y)\left(\langle x,y\rangle + c\right) + 2x_r l_t^r(x)\left(\langle x,y\rangle + c\right) + l_t^r(x)l_{t'}^r\left(\langle x,y\rangle + c\right)^2\Big\}.$$

See Appendix A.2.1 for a detailed derivation.

4.2.2 Alternative Kernel-based vv-CVs based on the Second Order Langevin Stein operator

An alternative class of matrix-valued Stein kernels can be constructed using the second-order Langevin Stein operator. The following theorem provides a characterisation of the class of *vector-valued functions* (vv-functions) obtained when applying this operator to functions in a vv-RKHS.

Theorem 4.2.2. *Consider* $\mathcal{H}_K$ *which is a vv-RKHS with mv-kernel* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{T\times T}$*, and suppose that* $K \in C^{2,2}(\mathcal{X}\times\mathcal{X})$*. Furthermore, for suitably regular vv-functions* $u = (u_1,\ldots,u_T) : \mathcal{X} \to \mathbb{R}^T$ *define the differential operator*

$$\mathcal{S}^{vv}[u] = (\mathcal{L}''_{\Pi_1}[u_1],\ldots,\mathcal{L}''_{\Pi_T}[u_T])^\top.$$

Then, the image of $\mathcal{H}_K$ *under* $\mathcal{S}^{vv}$ *is a vv-RKHS with reproducing kernel* $K_0 : \mathcal{X}\times\mathcal{X} \to \mathbb{R}^{T\times T}$*:*

$$(K_0(x,y))_{tt'} = \sum_{r,s=1}^d \partial_x^{ss}\partial_y^{rr}(K(x,y))_{tt'} + l_{t'r}(y)\partial_x^{ss}\partial_y^r(K(x,y))_{tt'} + l_{ts}(x)\partial_x^s\partial_y^{rr}(K(x,y))_{tt'} + l_{ts}(x)l_{t'r}(y)\partial_x^s\partial_y^r(K(x,y))_{tt'},$$

for $\forall t, t' \in [T]$.

See Appendix A.2.6 for proof. This theorem is very similar to Theorem 4.2.1. When $T = 1$, this more general kernel recovers the kernel proposed in Barp et al. (2022). However, one particular disadvantage from a computational perspective is that it requires higher-order derivatives of the kernel K. Another disadvantage is that it requires the evaluation of a double sum, which leads to significant increases in computational cost relative to the one in Theorem 4.2.1. Therefore, we will not explore this second-order matrix-valued Stein kernel in this thesis.

4.2.3 Learning Vector-valued Control Variates

Analogous to the scalar-valued control variates, suppose that $\mathcal{G}$ is parameterised, denoted by $\mathcal{G}_\Theta$, then the objective is,

$$J^{\text{vv}}(\theta) = \|\mathbb{V}_\Pi[f - g_\theta]\| = \|\Pi[(f - g_\theta - \Pi[f])^2]\|, \tag{4.7}$$

where $g_\theta \in \mathcal{G}_\Theta$. $\mathbb{V}_\Pi$ means to apply $\mathbb{V}_{\Pi_t}$ to $f_t - (g_\theta)_t$ elementwisely. This norm can be any norm. A natural objective then can be the sum of $\mathbb{V}_{\Pi_t}[f_t - g_t]$ for all $t \in [T]$, i.e., L_1 norm. Since we want to use functions in $\mathcal{H}_{K_0}$ as vector-valued control variates, we want to ensure the objective in (4.7) is well-defined. And this is provided by the following theorem.

Theorem 4.2.3. *Suppose that K is bounded with bounded derivatives, and $\Pi_t[\|\nabla_x \log \pi_t\|_2^2] < \infty$ for all $t \in [T]$. Then, for any $g \in \mathcal{H}_{K_0}$, g_t is square-integrable with respect to Π_t for all $t \in [T]$.*

See Appendix A.2.4 for the proof.

Empirical Risk Minimization with Penalty Given $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ where $\mathcal{D}_t = \{\{x_{tj}, f_t(x_{tj})\}_{j=1}^{m_t}$, it is then naturally to follow the empirical risk minimization scheme with penalty terms. The penalty terms often serve as the control of the

bias-variance trade-off.

$$L_m^{\text{vv}}(\theta, \beta) := J_m^{\text{vv}}(\theta, \beta) + \lambda \|g_\theta\|^2$$
$$= \sum_{t=1}^{T} \frac{1}{m_t} \sum_{j=1}^{m_t} (f_t(x_{tj}) - (g_\theta(x_{tj}))_t - \beta_t)^2 + \lambda \|g_\theta\|^2, \tag{4.8}$$

where $\beta = (\beta_1, \ldots, \beta_T)^\top \in \mathbb{R}^T$ and $\lambda \geq 0$ is the penalty coefficient. The second term in (4.8) is the penalty term which regularises the norm of vv-functions g_θ along with the emprical loss J_m^{vv}. Note that here it is not restricted to any particular choices of norms.

4.2.3.1 Closed-form Solutions of Vector-valued Control Variates

The following Theorem 4.2.4 shows that there exists a unique minimiser of the objective in (4.8) in $\mathcal{H}_{K_0}$.

Theorem 4.2.4. *Given $\mathcal{D} = \{\{x_{1j}, f_1(x_{1j})\}_{j=1}^{m_1}, \ldots, \{x_{Tj}, f_T(x_{Tj})\}_{j=1}^{m_T}\}$, the function which minimises the objective in* (4.8) *with $\|g_\theta\| := \|g_\theta\|_{\mathcal{H}_{K_0}}$ and $\beta \in \mathbb{R}^T$ is of the form:*

$$g_\theta(x) = \sum_{t=1}^{T} \sum_{j=1}^{m_t} \theta_{tj}^\top K_0(x, x_{tj}), \qquad \theta_{tj} \in \mathbb{R}^T \ \forall t \in [T], j \in [m_t],$$

with optimal parameter θ^* given by the solution of this convex linear system of equations:

$$\sum_{t'=1}^{T} \sum_{j'=1}^{m_{t'}} \Big(\sum_{t=1}^{T} \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} K_0(x_{tj}, x_{t'j'})_{t\cdot} + \lambda K_0(x_{t''j''}, x_{t'j'}) \Big) \theta^*_{t'j'}$$
$$= \sum_{t}^{T} \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} (f_t(x_{tj}) - \beta_t),$$

for $\forall t'' \in [T], j'' \in [m_T]$. When K_0 is strictly positive definite, then the system is strictly convex and θ^* is unique.

See Appendix A.2.5 for proof.

Remark 4.2.4. *In Theorem 4.2.4, $\beta = (\beta_1, \ldots, \beta_T)^\top$ is assumed to be known.*

Given a estimator $\hat{\theta}$, (4.8) *is a quadratic in* β*, and the optimal estimator is then,*

$$\beta_t^* = \tfrac{1}{m_t} \textstyle\sum_{j=1}^{m_t} f_t(x_{tj}) - (g_\theta(x_{tj}))_t, \qquad \textit{for } \forall t \in [T].$$

This naturally reminds us of block coordinate descent methods in multitask kernel methods literature, which can be implemented by using closed-form solutions or stochastic optimisation. Note that Theorem 4.2.4 does not require K *is separable. But the closed-form solution might be largely simplified when* K *is separable. In the following two sections, we are going to show how we implement this for special case I and II, i.e., when the base kernel* K *is separable.*

4.2.3.2 Stochastic Optimization with Known Relationship under Separable Kernel

One ideal scenario is when $B \in S_+^T$ is known. This happens when domain knowledge is available, e.g., from experts in the relevant fields. The pseudo-code of our proposed stochastic optimisation learning algorithm is presented in Algorithm 4.

Since our kernel-based vv-CVs in (4.6) are linear in θ and the objective L_m^{vv} is jointly convex in (θ, β) when $\|g_\theta\|^2$ is convex in θ, Algorithm 4 is able to minimise the objective L_m^{vv} globally when using popular stochastic optimization approaches under regularity conditions (Bottou et al., 2018), e.g., mini-batch stochastic gradient descent (Si et al., 2021). Note that, Algorithm 4 can also be applied to other vv-CVs whether linear or nonlinear.

Initialization Initialising the algorithm at $\theta^{(0)} = (0, \ldots, 0) \in \mathbb{R}^p$ when vv-CVs are parameterized by θ (e.g., polynomial-, kernel- and neural network-based vv-CVs). This means that no control variates are involved before optimization. Meanwhile, a natural way of initialising β is to set $\beta^{(0)}$ to be any available estimates of $\Pi[f]$ as we expect $\beta = (\beta_1, \ldots, \beta_T)^\top$ approaches $\Pi[f]$ when $m_1, \ldots, m_T \to \infty$. For instance, we can set $\beta^{(0)} = (\hat{\Pi}_1^{\text{MC}}[f_1], \ldots, \hat{\Pi}_T^{\text{MC}}[f_T])^\top$.

Construction of Mini-batches For each iteration in the stochastic optimisation algorithm for vv-CVs, we take mini-batches of size $\tilde{m} \in \mathbb{N}_+$ where $|\tilde{m}| \leq |m|$. Note that $\tilde{m} = (\tilde{m}_1, \ldots, \tilde{m}_T)^\top$ is the multi-index that determines the size of the

Algorithm 4: Stochastic optimisation for vv-CVs with known task relationship

Input: $\mathcal{D}$, $\tilde{m}$, L, λ, $\beta^{(0)}$ and $\theta^{(0)}$.

for *iterations* l *from 1 to* L **do**

  Select a mini-batch $\mathcal{D}_{\tilde{m}}$ of size $\tilde{m}$.

  $\left(\theta^{(l)}, \beta^{(l)}\right) \leftarrow \text{Update}_{\theta,\beta}(\theta^{(l-1)}, \beta^{(l-1)}, B; \mathcal{D}_{\tilde{m}})$.

Output: Return $\theta^{(L)}$, $\beta^{(L)}$.

mini-batch taken from each of the T datasets. This flexible formulation allows different sizes of mini-batches from the T datasets, which can be useful when the datasets are of different sizes. Thus, we propose to set $\tilde{m}_t \propto m_t/(\sum_{t=1}^{T} m_t)$ for all $t \in [T]$. This choice means that the number of samples for each integrand in the mini-batches is proportional to the proportion of data points for that integrand in $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$. An epoch means that the algorithm has gone through all data points in $\mathcal{D}$ and the indices are randomly shuffled after each epoch, which happens every $m_t/\tilde{m}_t$ (assuming it is an integer) iterations.

Updating In Algorithm 4, θ and β are updated abstractly as a function $\text{Update}_{\theta,\beta}(\theta, \beta, B; \mathcal{D})$. We present this general formulation here because it might also be used for other vv-CVs which might benefit from different update approaches. For instance, pre-conditioners for the gradients might be used for updating when they are available or can be estimated. In our experiments, we update the values of θ and β based on the gradient of the learning objective $\nabla_{(\theta,\beta)} L^{\text{vv}}_{\tilde{m}}(\theta, \beta)$ with the Adam optimizer (Kingma and Ba, 2015).

Regularisation Based on the literature on kernel ridge regression, a natural regularisation term can be $\|g_\theta\| := \|g_\theta\|_{\mathcal{H}_{K_0}}$. This also recovers the objective used in Theorem 4.2.4. However, it can be infeasible to use this norm due to the heavy computational cost as it requires kernel evaluations for all the training points in $\mathcal{D}$. Therefore, we follow the setting of Si et al. (2021) and use the Euclidean norm: $\|g_\theta\| = \|\theta\|_2$. Note that the learning objective is still quadratic in θ and thus still a convex optimisation problem.

4.2.3.3 Stochastic Optimization with Unknown Relationship under Separable Kernel

We now extend our method to the scenarios when the task relationship B is unknown. In this case, we need to learn B jointly with θ and β for our vv-CVs. We propose to use the same objective (4.8), but with one extra penalty term on the norm of B. That is,

$$\bar{L}_m^{\text{vv}}(\theta, \beta, B) = J_m^{\text{vv}}(\theta, \beta, B) + \lambda \|g_\theta\|^2 + \|B\|^2, \tag{4.9}$$

Note the change from $J_m^{\text{vv}}(\theta, \beta)$ to $J_m^{\text{vv}}(\theta, \beta, B)$. This is to emphasize the dependence on the task relationship B. This objective can be optimized by via Algorithm 5. In Algorithm 5, the optimization is accomplished via block coordinate descent, i.e., combining the gradient updates on (θ, β) and gradient updates on B. This block coordinate descent approach is inspired by (Ciliberto et al., 2015). Although we use gradient of the loss to updates the parameters, we once again present such updates as a general function $\text{Update}_B(\theta, \beta, B; \mathcal{D})$, which has similar reasons to those of Algorithm 4.

Remark 4.2.5 (Positive (Semi-)Definite B in Optimization)**.** *To ensure that B is positive definite at every iteration, we set $B = LL^\top$, where L is a lower triangle matrix with elements being forced to be larger than zero. This is achieved by exponential transformation. Another possible transformation is to take squared values. And in this case, B is positive semi-definite at every iteration.*

In terms of initial values of (θ, β), they can be chosen via using the discussions in the previous section. While, for the initial value of B, we suggest to set $B^{(0)} = I_T$ since this means that no relationship across tasks as we do not have access to such information before learning from the data.

Remark 4.2.6 (Convexity)**.** *Algorithm 5 is convex in (θ, β) or B but it is not jointly convex in all of the three components (θ, β, B). Though we cannot provide convergence to a global optimum due to lack of convexity in general, it works well in*

practice as shown in the experiments. However, we show the learning objective is actually jointly convex in (θ, β, B) in some scenarios; see Section 4.2.3.4 for details.

4.2.3.4 Convex Optimisation for Estimating B

As discussed in Section 4.2.3.3, estimating the matrix B for a separable kernel from data leads to a non-convex optimisation problem. Thankfully, we can approximate the optimum using a sequence of convex problems by extending the work of Dinuzzo et al. (2011); Ciliberto et al. (2015) together with Theorem 4.2.4 above. For this, we will require that the kernel K_0 is separable, and shall thus restrict ourselves to the case where we have a single target distribution (i.e. special case II).

Theorem 4.2.5. *Suppose that $\Pi_t = \Pi$ for $t \in [T]$ and $K(x, y) = Bk(x, y)$ so that $K_0(x, y) = Bk_0(x, y)$ where k_0 is defined in* (3.5)*. Then the following objective is convex in (θ, β, B) for any value of $\delta > 0$:*

$$\begin{aligned}\bar{L}^{vv}_{m,\delta}(\theta, \beta, B) = J^{vv}_m(\theta, \beta, I_T) \\ + \lambda \textstyle\sum_{t,t'=1}^{T} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \operatorname{Tr}\left[B^\dagger \left(k_0(x_{tj}, x_{t'j'})\theta_{tj}\theta_{t'j'}^\top + \delta^2 I_T\right)\right] \\ + \|B\|^2,\end{aligned}$$

and for each β and any sequence $\delta_\ell \to 0$, the associated sequence of minimisers (θ_ℓ, B_ℓ) converges to (θ_*, B_*) s.t., $(\theta_* B_*^\dagger, B_*)$ minimises the objective in (4.9)*.*

Proof. Since the kernel K_0 is separable, the objective (4.9) may be written in the form of Ciliberto et al. (2015, Problem $(\mathcal{Q})$). Has shown therein, $\sum_{t,t'=1}^{T} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \operatorname{Tr}\left[B^\dagger \left(k_0(x_{tj}, x_{t'j'})\theta_{tj}\theta_{t'j'}^\top\right)\right]$ is jointly convex in B and θ, and since the first term in $L^{\text{vv}}_{m,\delta}(\theta, \beta, B)$ is convex in β and θ jointly, $L^{\text{vv}}_{m,\delta}(\theta, \beta, B)$ is jointly convex in (θ, B, β). Moreover, by Theorem 3.1 & 3.3 in (Ciliberto et al., 2015), when $\delta \to 0$, (θ, B) converges in Frobenius norm to (θ_*, B_*), where $(\theta_* B_*^\dagger, B_*)$ a minimiser of (4.9), where $B_*^\dagger$ denotes the pseudoinverse of B_*. □

This theorem could therefore be used to construct an approach based on convex optimisation algorithms which are used iteratively for a decreasing sequence of

Algorithm 5: Block-coordinate descent for vv-CVs with unknown task relationship

Input: $\mathcal{D}, \tilde{m}, L, \lambda, \beta^{(0)}, \theta^{(0)}$ and $B^{(0)}$.

for *iterations* l *from* 1 *to* L **do**

  Select a mini-batch $\mathcal{D}_{\tilde{m}}$ of size $\tilde{m}$.

  $\left(\theta^{(l)}, \beta^{(l)}\right) \leftarrow \text{Update}_{\theta,\beta}(\theta^{(l-1)}, \beta^{(l-1)}, B^{(l-1)}; \mathcal{D}_{\tilde{m}})$.

  $B^{(l)} \leftarrow \text{Update}_{B}\left(\theta^{(l)}, \beta^{(l)}, B^{(l-1)}; \mathcal{D}_{\tilde{m}}\right)$.

Output: Return $\theta^{(L)}$, $\beta^{(L)}$ and $B^{(L)}$.

penalisation parameters in order to converge to an optimum approaching the global optimum. However, this approach is limited to the case where all distributions are identical, and is hence not as widely applicable as Algorithm 5.

4.2.4 Computational Complexity of Vector-valued Control Variates

Vector-valued control variates can be beneficial from the perspective of accuracy. However, this is achieved at the cost of computation burden, especially when the number of tasks T is large. Thus, whether to use vv-CVs should depend on the computational budget available. For instance, when the computational cost of the evaluations of the integrands or score functions are expensive, the additional computational cost of vv-CVs might be negligible.

The computational complexity of CVs and vv-CVs is presented in Table 4.1. We emphasise the impact of T, the number of tasks. In particular, the dependence of the number of tasks T for kernel-based CVs is $\mathcal{O}(T)$, while that for vv-CVs is between $\mathcal{O}(T^4)$ and $\mathcal{O}(T^6)$. We also compare the computational complexity of the closed-form solutions by solving the linear system of equations in Theorem 4.2.4 and the stochastic optimisation methods in Section 4.2.3.2 and Section 4.2.3.3. It is found that when $\tilde{m}_t L$ is small relative to m_t, computational gains can be achieved.

In all applications considered, both m_t and T were small so the overall cost is controlled. However, this computational complexity can be further significantly reduced in special cases. When using a kernel corresponding to a finite-dimensional RKHS (e.g. a polynomial kernel), the scaling becomes linear in m_t, but is $\mathcal{O}(q^3)$

Table 4.1: Computational complexity of kernel-based CVs and vv-CVs as a function of $d, m, \tilde{m}, L$ and T. We assume that m_t is the same $\forall t \in [T]$ up to additive or multiplicative constants (and similarly for all $\tilde{m}_t$ with $t \in [T]$). The cost of stochastic optimisation algorithms is assumed to only scale with the cost of stochastic estimates of the gradient of J^{vv}.

Method	CV	vv-CV
Exact solution	$\mathcal{O}((dm_t^2 + m_t^3)T)$	$\mathcal{O}(dm_t^2T^4 + m_t^3T^6)$
Stochastic optim.	$\mathcal{O}(d\tilde{m}_t m_t LT)$	$\mathcal{O}(d\tilde{m}_t m_t LT^4)$

instead of $\mathcal{O}(m_t^3)$, where $q \ll m_t$ is the dimensionality of the RKHS. Alternatively, for certain choices of point sets and kernels, it is possible to reduce the computational complexity to $\mathcal{O}(m_t \log m_t)$ instead of $\mathcal{O}(m_t^3)$ by using scalable kernel methods such as fast Fourier features or inducing points. When the integrands are evaluated at the same set of points and the separable kernel is used, the computational cost in T also becomes $\mathcal{O}(T^2)$ instead of $\mathcal{O}(T^6)$, once again significantly reducing the computational complexity.

4.3 Experimental Results

We now illustrate the proposed approach on a range of problems with related integration tasks, including multi-fidelity models and computation of the model evidence for dynamical systems via thermodynamic integration. To make fair comparisons across methods, we will always use the same learning rate, decay coefficient, batch size, and initial values for each example. Since we are interested in gains obtained from the CVs, we will always fix $n = 0$. We describe the way of choosing kernel hyperparameters in Appendix A.3. Code is available at: `https://github.com/jz-fun/Vector-valued-Control-Variates-Code`.

4.3.1 A Synthetic Example

To start with, a synthetic example is selected from (South et al., 2022c) (denoted f_2). To make the problem fit into our framework, we introduce another integrand

(denoted f_1) which is similar to f_2:

$$\begin{aligned} f_1(x) &= 1.5 + x + 1.5x^2 + 1.75\sin(\pi x)\exp(-x^2), \\ f_2(x) &= 1 + x + x^2 + \sin(\pi x)\exp(-x^2). \end{aligned}$$

For this problem, we trained all CVs through stochastic optimisation and use $m = (50, 50)$ MC samples. This synthetic example was originally used by South et al. (2022c) to show one of the drawbacks of kernel-based CVs, namely that the fitted CV will usually tend to β in parts of the domain where we do not have any function evaluations. This phenomenon can be observed on the red lines in Figure 4.2 (left and center) which gives a CV based on a squared-exponential kernel. This behaviour is clearly one of the biggest drawbacks of existing kernel-based approaches. However, the blue curve, representing a kernel-based vv-CV with separable kernel where B was inferred through optimisation, partially overcomes this issue by using evaluations of both integrands, hence clearly demonstrating potential advantages of sharing function values across integration tasks.

The right-most plot in Figure 4.2 presents several box plots for the sum of squared errors for each integration problem calculated over 100 repetitions of the experiment. The different box plots show the impact of the difference in Π_1 and Π_2. As we observed, vv-CVs always outperform CVs, although this difference in performance is more stark when Π_2 has a larger tail than Π_1. This reinforces the previous point, since a more disperse Π_2 means that the second integrand will be evaluated more often at more extreme areas of the domain, which will help obtain a better vv-CV by improving the fit at the tails of the distribution.

4.3.2 Multi-fidelity Modelling

Many problems in the engineering and physical sciences can be tackled with multiple models of a single system of interest. These different models are often associated with varying computational costs and levels of accuracy, and their combination to solve a task is usually referred to as multi-fidelity modelling; see Peherstorfer et al. (2018) for a review. In this section, we will focus on the case with $T = 2$ models. We will consider a high-fidelity model f_H and a low-fidelity model f_L, and

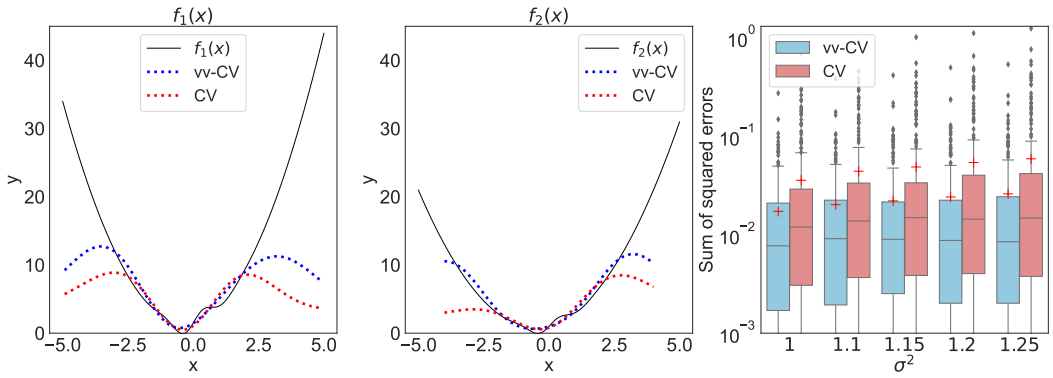

Figure 4.2: *Numerical integration of problem from South et al. (2022c). Left and center:* Illustration of f_1 and f_2, as well as the corresponding kernel-based CVs and vv-CVs obtained through stochastic optimisation when $\Pi_1 = \mathcal{N}(0, 1)$ and $\Pi_2 = \mathcal{N}(0, 1.25)$. *Right:* Sum of the squared errors in estimating $\Pi_1[f_1]$ and $\Pi_2[f_2]$. Here, $\Pi_1 = \mathcal{N}(0, 1)$ whilst $\Pi_2 = \mathcal{N}(0, \sigma^2)$ where $\sigma^2 \in \{1, 1.1, 1.15, 1.2, 1.25\}$.

will attempt to estimate the integral of f_H with our vv-CVs and using function evaluations from both the high- and low-fidelity models. For clarity, we will now denote the function $f = (f_L, f_H)$ and the vector-probability distribution $\Pi = (\Pi_L, \Pi_H)$.

We note that this is a special case of the problem considered in this section since we use evaluations of multiple functions but are only interested in $\Pi_H[f_H]$ (whereas $\Pi_L[f_L]$ is not of interest). The most common approach to tackling this type of problem is multi-level MC (Giles, 2015), and in the context of unnormalised densities, multi-level MCMC can be used (Dodwell et al., 2019). However, since vv-CVs are post-processing tools which can be used with any MC method, we will restrict ourselves to reducing the variance of simple MC estimators.

4.3.2.1 Univariate Step Function

Our first multi-fidelity experiment is a common synthetic problem in the multi-fidelity literature; see for example Xi et al. (2018). The low-fidelity function is $f_{\mathrm{L}}(x) = 2$ if $x \geq 0$ and -1 otherwise. The high-fidelity function is $f_{\mathrm{H}}(x) = 1$ if $x \geq 0$ and 0 otherwise. Although this problem is only one-dimensional, it is nonetheless not straightforward for CVs since the integrands are discontinuous but existing CVs are all continuous. The integral is over the real line and taken against $\Pi = \mathcal{N}(0, 1)$, and we fix the sample sizes to $m = (m_{\mathrm{L}}, m_{\mathrm{H}}) = (40, 40)$.

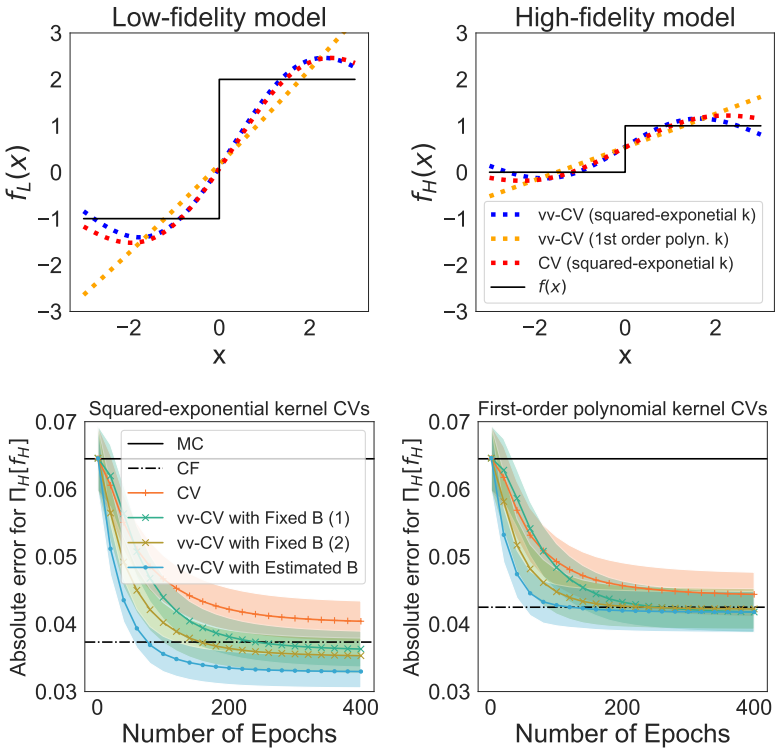

Figure 4.3: *Numerical integration of univariate discontinuous multifidelity model. Upper:* fitted CVs for both functions. *Lower Left:* performance of CVs based on a squared-exponential kernel as a number of epochs of the optimisation algorithm. The lines provide the mean over 100 repetitions of the experiment, whereas the shaded areas provide one standard deviation above and below the mean. *Lower Right:* same experiment for a polynomial kernel.

Results with a squared-exponential and 1$^{\text{st}}$ order polynomial kernel k can be found in Figure 4.3. The upper plots clearly show that the approximations are not of very high-quality, but the lower plots show that all CVs can still lead to an order of two gain in accuracy over MC methods.

We also observe that vv-CVs can lead to further gains over existing CVs by leveraging evaluations of f_L. For both kernels, we provide three different versions of the vv-CVs with separable structure to highlight the impact of the matrix B. The first two cases use Algorithm 4 with a fix value of B. In the first instance, $B_{11} = B_{22} = 0.1, B_{12} = B_{21} = 0.01$, whereas in the second instance $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$. The third case is based on estimating B through Algorithm 5. Clearly, B can have a significant impact on the performance of the vv-CV, and estimating a good value from data can provide further gains. The choice of

k is also significant; all CVs based on the squared-exponential kernel significantly outperform the CVs based on a 1st order polynomial kernel.

4.3.2.2 Multivariate Model of Water Flow

We now move on to a much more challenging task based on multi-fidelity models of the flow of water through a borehole drilled from the ground surface through aquifers (Xiong et al., 2013). The two functions have $d = 8$ inputs $x = (r_w, r, T_u, T_l, H_u, H_l, L, K_w)$ representing a range of parameters influencing the geometry and hydraulic conductivity of the borehole, as well as transmissivity of the aquifer. The two functions are given by

$$\begin{aligned} f_L(x) &= \frac{5T_u(H_u - H_l)}{\log\left(\frac{r}{r_w}\right)\left(1.5 + \frac{2LT_u}{\log\left(\frac{r}{r_w}\right)r_w^2 K_w} + \frac{T_u}{T_l}\right)} \\ f_H(x) &= \frac{2\pi T_u(H_u - H_l)}{\log\left(\frac{r}{r_w}\right)\left(1 + \frac{2LT_u}{\log\left(\frac{r}{r_w}\right)r_w^2 K_w} + \frac{T_u}{T_l}\right)}. \end{aligned}$$

They provides estimates of water flow through a borehole in m^3 per year. Prior distributions have been elicited from scientists over input parameters to account for uncertainties about their exact value (see Appendix A.4.3 for further details). One quantity of interest here is the expected water flow under these prior distributions. We hence have $\Pi_1 = \Pi_2$. Here, f_H is not more expensive than f_L and so the two-fidelity approach is not essential for this example. However, this example is popular in the multi-fidelity modelling literature (Xiong et al., 2013; Kandasamy et al., 2016; Park et al., 2017) since it is representative of the difficulty of problems commonly tackled in this field.

Results of our simulation study are presented in Table 4.2. We compare a standard MC estimator with a kernel-based CV fitted with a closed form solution (denoted *control functionals* (CF)) and two kernel-based vv-CVs corresponding to special case II in Section 4.2.1. The first with $B_{11} = B_{22} = 5 \times 10^{-4}$ and $B_{12} = B_{21} = 5 \times 10^{-5}$, and the second with B estimated using Algorithm 5. The kernel used was a tensor product of squared-exponential kernels with a separate lengthscale for each dimension. Clearly, vv-CVs significantly outperform MC in

m	vv-CV- Estimated B	vv-CV-Fixed B	CF	MC
$(10, 10)$	3.722 (0.273)	**1.943** (0.150)	2.236 (0.159)	6.418 (0.435)
$(20, 20)$	**1.290** (0.096)	1.352 (0.103)	1.960 (0.101)	4.314 (0.313)
$(50, 50)$	**1.044** (0.064)	1.766 (0.120)	1.761 (0.070)	2.629 (0.172)
$(100, 100)$	**1.074** (0.064)	1.647 (0.137)	1.712 (0.045)	1.827 (0.152)
$(150, 150)$	**0.854** (0.047)	1.302 (0.094)	1.671 (0.039)	1.423 (0.102)

Table 4.2: Expected values of the flow of water through a borehole under an expert-elicited prior distribution and using the high-fidelity model. The numbers provided give the mean absolute integration error for 100 repetition of the task of estimating $\Pi_H[f_H]$, and the number in bracket provide the sample standard deviation. To provide the absolute error, the true value of the integral (72.8904) is estimated by a MC estimator with 5×10^5 samples.

the large majority of cases, and estimating B can lead to significant gains over using a fixed B. The worst performance for vv-CVs with estimated B is when values of m are the lowest. This is because m is not large enough to learn a good value of B.

4.3.3 Computation of the Model Evidence for Dynamical Systems

We now consider Bayesian inference for non-linear differential equations such as dynamical systems, which can be particularly challenging due to the need to compute the model evidence. This is usually a computationally expensive task since sampling from the posterior repeatedly requires the use of a numerical solver for differential equations which needs to be used at a fine resolution.

In Calderhead and Girolami (2009), the authors propose to use *thermodynamic integration* (TI) (Friel and Pettitt, 2008) to tackle this problem, and Oates et al. (2016, 2017) later showed that CVs can lead to significant gains in accuracy in this context. Let Θ denote our parameter space. TI introduces a path from the prior $p(\theta)$ to the posterior $p(\theta|y)$, where y and θ represent the observations and the unknown parameters respectively. This is accomplished by the power posterior $p(\theta|y, t) \propto p(y|\theta)^t p(\theta)$, where $t \in [0, 1]$ is called the inverse temperature. When $t = 0$, $p(\theta|y, t) = p(\theta)$, whereas when $t = 1$, $p(\theta|y, t) = p(\theta|y)$. The standard TI formula for the model evidence has a simple form which can be approximated using

second-order quadrature over a discretized temperature ladder $0 = t_1 \leq \cdots \leq t_w = 1$ (Friel et al., 2014a). It takes the following form

$$\begin{aligned}\log p(y) &= \int_0^1 \left[\int_\Theta \log p(y|\theta) p(\theta|y,t) d\theta\right] dt \\ &\approx \sum_{i=1}^w \frac{t_{i+1}-t_i}{2}(\mu_{i+1}+\mu_i) - \frac{(t_{i+1}-t_i)^2}{12}(v_{i+1}-v_i),\end{aligned}$$

where μ_i is the mean and v_i the variance of $\log p(y|\theta)$ under $\pi_i = p(\theta|y,t_i)$. To estimate $\{\mu_i, v_i\}_{i=1}^w$, we need to sample from all power posteriors on the ladder, then use a MCMC estimator which can be enhanced through CVs. This gives $T = 2w$ integrals which are *related*: w integrals to compute means and w integrals to compute variances, each against different power posteriors. As we will see, this relationship between integration tasks will allow vv-CVs to provide significant gains in accuracy.

Our experiments will focus on the *van der Pol oscillator*, which is a second order oscillator $x : \mathbb{R}_+ \times \Theta \to \mathbb{R}$ (where here $\theta \in \mathbb{R}$) given by the solution of $d^2x/ds^2 - \theta(1-x^2)dx/ds + x = 0$, where s represents the time index. For this experiment, we will follow the exact setup of Oates et al. (2017)and transform the equation into a system of first order equations:

$$\frac{dx_1}{ds} = x_2, \qquad \frac{dx_2}{ds} = \theta(1-x_1^2)x_2 - x_1,$$

which can be tackled with ODE solvers. Our data will consist of noisy observation of x_1 (the first component of that system) given by $y(s) = \mathcal{N}(x_1(s;\theta), \sigma^2)$ with $\sigma = 0.1$ at each point $s \in \{0, 1, \ldots, 10\}$; see the left-most plot of Figure 4.4 for an illustration. We will take a ladder of size $w = 31$ with $t_i = ((i-1)/30)^5$ for $i \in \{1, \ldots, 31\}$. This gives a total of $T = 62$ integrals that will need to be computed simultaneously, which is likely to be too computationally expensive. As a result, we group integrands in groups of 4 means or 4 variances (except one group of 3 for mean and variance), and use vv-CVs for each group separately. To sample from the power-posterior, we use population MC with the manifold Metropolis-adjusted Langevin algorithm (Girolami and Calderhead, 2011). Due to the high

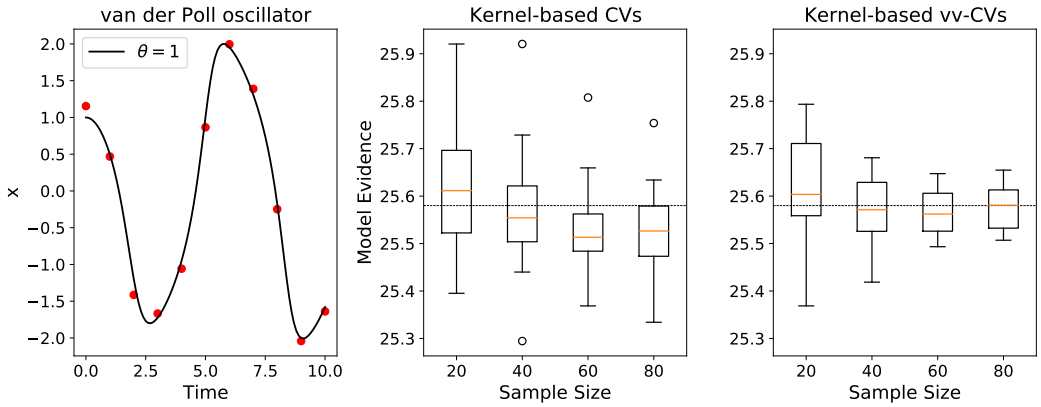

Figure 4.4: *Model evidence computation through thermodynamic integration. Left:* Illustration of the van der Pol oscillator model (black line) and corresponding observations (red dots) used to obtain a posterior distribution. *Center:* Estimates of the model evidence as a function of the number of posterior samples for kernel-based CVs. The boxplots were created by repeating the experiment 20 times and the black line gives an estimate of the truth obtained from Oates et al. (2017) and equals 25.58. *Right:* Same experiment repeated with kernel-based vv-CVs.

computational cost of using ODE solvers, our samples will be limited to less than 100 per integrand and this number will be the same for each integration task.

Our results are presented in Figure 4.4, and the boxplots present 20 repetitions of the experiment. The kernel parameters were taken to be identical to those in Oates et al. (2017). As observed in the centre plot, kernel-based CVs provide relatively accurate estimates of the model evidence. As the sample size increases, we notice less variability in these estimates, but the central 50% of the runs are contained in an interval which excludes the true value. In comparison, the right-most plot shows that kernel-based vv-CVs can provide significant further reduction in variance. The distribution of estimates is also much more concentrated and centered around the true value.

4.3.4 Bayesian Inference of Lotka-Volterra System

We now consider another model: the Lotka-Volterra system (Lotka, 1925; Volterra, 1926; Lotka, 1927) of ordinary differential equations. This system is given by:

$$\frac{dv_1(s)}{ds} = \alpha v_1(s) - \beta v_1(s) v_2(s)$$
$$\frac{dv_2(s)}{ds} = \delta v_1(s) v_2(s) - \gamma v_2(s).$$

Here, $s \in [0, S]$ for some $S \in \mathbb{R}_+$ denotes the time, and $v_1(s)$ and $v_2(s)$ are the numbers of preys and predators, respectively. The system has initial conditions $v_1(0)$ and $v_2(0)$. We have access to noisy observations of $v = (v_1, v_2)$ at points $s_1, \ldots, s_m \in [0, S]$ denoted y_{1j}, y_{2j} and which are both observed with log-normal noise with standard deviation σ_{y_1} and σ_{y_2} respectively for all $j \in \{1, \ldots, m\}$, given some unknown parameter value $x^* = (\alpha^*, \beta^*, \delta^*, \gamma^*, v_1(0)^*, v_2(0)^*, \sigma^*_{y_1}, \sigma^*_{y_2})^\top$. In practice, we reparameterise x such that the model parameters are defined in $\mathbb{R}^8$; see Appendix A.4.5 for details. Given these observations, we can construct a posterior distribution on the value of x^* and we will denote this posterior Π. We will then be interested in computing posterior expectations of v_1 at a set of time points $s'_1, \ldots, s'_T$, and hence have T integrands of the form $f_t(x) = v_1(s'_t; x)$ where x highlights the dependence on the parameter x. CVs were previously considered for individual tasks in this context by Si et al. (2021). However, these T integrands are related when $s'_1, \ldots, s'_T$ are close to each other.

We are interested in estimating $\Pi[f_t]$ for preys at time s'_t. We use the real world dataset (Hewitt, 1921) of snowshoe hares (preys) and Canadian lynxes (predators) as observations and implement Bayesian inference on model parameters x by using no-U-turn sampler (NUTS) in Stan (Carpenter et al., 2017). For sv-CVs, we estimate each individual $\Pi[f_t]$ separately; while for vv-CVs, we estimate a collection these tasks $\Pi[f] := (\Pi[f_1], \ldots, \Pi[f_T])^\top$ jointly. See Appendix A.4.5 for experimental details. In Figure A.3 (Appendix A.4.5), we demonstrate the fitting of MCMC via the posterior predictive as well as posterior trajectories of v conditional on each posterior sample of x.

Table 4.3: Posterior Expected Abundance of Preys. The numbers provided give the sum of the mean absolute integration error for 10 repetition of each task of estimating the vector-valued expectation $\Pi[f]$. To provide the absolute error, the true values of the associated expectations are estimated by MCMC estimators with 8×10^5 posterior samples.

T	m	vv-CV- Est. B	vv-CV-Fix. B	CF	MCMC
2	500	0.462	**0.404**	0.666	0.568
5	500	**0.393**	0.419	0.521	0.987
10	500	**0.938**	1.031	2.540	2.663

In Table 4.3, we compare kernel vv-CVs (special case II) with standard MCMC estimators and CF estimators. We consider two cases of vv-CVs: the first is the case when B is with $B_{tt} = 5 \times 10^{-4}$ for all t and $B_{tt'} = 5 \times 10^{-5}$ for $\forall t \neq t'$, and the second with B estimated using Algorithm 2. The kernel used is a tensor product of squared-exponential kernels with a separate lengthscale for each dimension. We increase the number of tasks from $T = 2$ to $T = 10$. Once again, vv-CVs significantly outperform MCMC, especially for large T, and estimating B provides further gains over using a fixed B in terms of statistical efficiency.

4.4 Conclusions

This chapter considers variance reduction methods that share information among multiple related integration problems. The proposed approach, *vector-valued control variates*, has shown to lead to significant variance reduction for problems including multi-fidelity modelling and Bayesian evidence computation for dynamical systems. To the best of our knowledge, the proposed approach is the first algorithm that can perform multi-task learning for numerical integration, which is achieved by using only evaluation of the score functions of the corresponding target distributions. Meanwhile, it is also the first algorithm which is able to simultaneously learn the relationship between integrands and provide estimates of the associated integrals.

There are a number of possible ways forward. Firstly, the proposed method can be tailored for a range of other interesting problems in Bayesian statistics. For instance, when doing a sensitivity analysis for the choice of prior, Markov Chain

Monte Carlo can be run for several choices of priors. We would then compare the corresponding posterior expectations of interest which differ only up due to the priors. Clearly, these integrals will be closely related and gains may be obtained by sharing information across tasks. Another example would be in settings when we have sequential data. In this case, it might be of interest to update our estimates of posterior expectations once new data arrives. This would lead to a sequence of integration tasks where information from the earlier tasks can inform something about future tasks. These are just two examples of possible uses of the proposed approach. We expect that many others could be found in practice.

On the methodology side, we propose a novel approach for constructing vv-CVs and show this can lead to significant gains in accuracy. However, further research can focus on making the approach more computationally practical. One particular line of research warranting future work is how special cases of our matrix-valued Stein kernels in Theorem 4.2.1 can be chosen to reduce the computational cost whilst still having rich classes of vv-CVs. Another line of research warranting future work is the question of when transferring information across multiple integration tasks will lead to sufficient gains in accuracy.

At last, a number of methods have been developed on the basis of Stein discrepancies, which stem from Stein reproducing kernels (Anastasiou et al., 2023). Most relevant to Bayesian computation are sampling methods such as Stein variational gradient descent (Liu and Wang, 2016; Wang et al., 2019a) and Stein points (Chen et al., 2018). One of the main contributions of this work has been the development of matrix-valued Stein reproducing kernels which produce zero mean vv-functions against vector-probability distributions. This novel tool is able to hence derive novel (and more flexible) Stein discrepancies, which further result in novel statistical methods. For instance, in terms of sampling, this can lead to methods which are able to approximate multiple distributions simultaneously.

Chapter 5

Meta-learning Control Variates: Variance Reduction with Limited Data

Control variates methods are quite useful to reduce the variance of Monte Carlo estimators. However, it can be challenging to construct and select control variates when the sample size is small.

In this chapter, we demonstrate that it is possible to construct and learn effective control variates when a large number of related integrals are required to be estimated with only few samples per task available. This is achieved by leveraging the similarity and relationship among these integration tasks. Our approach, called *meta-learning control variates* (Meta-CVs), can be used for hundreds or thousands of integration tasks. We validate the performance of the proposed Meta-CVs from two aspects: i) The empirical assessment demonstrates that Meta-CVs can result in significant variance reduction and thus more accurate estimators in such settings; ii) The theoretical analysis establishes general conditions when Meta-CVs can be learnt successfully.

5.1 Introduction

From Chapter 3 and Chapter 4, we know that to construct and select an effective CV, it often requires a large sample size N. This unfortunately restricts the usefulness

of CVs in scenarios when sampling from π or evaluating f is expensive, or when the computation budget is limited. When integrals are defined in high dimensions, it also poses challenges to CVs since such functions are difficult to approximate due to the curse of dimensionality. In the latter case, sparsity can be exploited for integrands with low *effective dimension* (South et al., 2022a; Leluc et al., 2021), but many integrands do not admit convenient structure that can be easily exploited.

In this chapter, we propose a different solution, named *meta-learning control variares* (Meta-CVs), which borrows strength from multiple related integral estimation tasks to construct and select effective CVs for each individual integral estimation task. The proposed method requires a setting where there are T integration tasks ($\Pi_1[f_1], \ldots, \Pi_T[f_T]$) to be estimated, and where the integrands of interest $f_1, \ldots, f_T$ and the target densities $\pi_1, \ldots, \pi_T$ are different, but related. This setting actually arises in a broad range of applications. For instance, multifidelity modelling (Peherstorfer et al., 2018; Li et al., 2023), sensitivity analysis (Demange-Chryst et al., 2022), policy gradient methods (Liu et al., 2018), and thermodynamic integration (Oates et al., 2016). We also consider further examples in Section 5.4. For instance, we consider the application to the marginalisation of hyper-parameters in Bayesian inference so we demonstrate it with a hierarchical Gaussian process example. Another example is the computation of predictive distributions which we illustrate with Bayesian inference of the Lotka–Volterra system. We demonstrate that when the integrands and densities are closely related, sharing information among integration tasks can be expected to result in substantial improvement in estimation accuracy.

To the best of our knowledge, *vector-valued control variates* by Sun et al. (2023a) (also presented in Chapter 4) is the only existing control variates method to explore and exploit multiple related integration tasks to obtain further improvement on the accuracy of estimators. This is achieved by learning the relationship among the integrands of interest via a multi-task kernel learning method in a vector-valued reproducing kernel Hilbert space induced by a matrix-valued Stein kernel. Though it shows the potential to have an improvement in performance, it suffers from a high

computational cost $\mathcal{O}(T^6)$ and a significant memory cost of $\mathcal{O}(T^2)$. The biggest experiment in Sun et al. (2023a) is the computation of posterior expected abundance of preys for the Lotka–Volterra System at $T = 10$ different time points. This method lacks scalability in T, which is an important limitation. It can be desirable to learn and exploit the shared information across hundreds or thousands of tasks, e.g., in the motivating examples mentioned above. Therefore, a key challenge remains to be solved: *"How can we construct control variates at scale, sharing information across a large number of integration tasks even with limited samples per task?"*

Our answer to this challenge, *Meta-learning CVs* (Meta-CVs), is established on the framework of meta-learning (Finn et al., 2017, 2018). The benefits of Meta-CVs are three-fold: (i) the computational cost grows as $\mathcal{O}(T)$, (ii) the effective number of parameters is constant in T, and (iii) the construction of the Meta-CV occurs offline and a new control variate can be constructed and selected at minimal cost whenever a new integration task arises. Before introducing Meta-CVs in Section 5.3, we first recall background on CV methods in Section 5.2.

5.2 Recap of Control Variates

The background knowledge of control variates has already been included in Chapter 3. For completeness of this chapter, we decide to still present necessary background information on general techniques used to construct and select control variates in this section.

From Chapter 3 and Chapter 4, we know that the general strategy of control variates methods is to identify a function $g \in \mathcal{L}^2(\Pi)$ (i.e. a control variate) for which $\mathbb{V}_\Pi[f - g]$ is small, and for which the expectation $\Pi[g]$ is known or can be exactly computed. For instance, as mentioned in Chapter 3, we know that Stein-based control variates $g_\theta := \mathcal{S}_\Pi[u_\theta]$ that has zero-mean under Π where $\theta \in \mathbb{R}^p$ are the parameters of the control variate. In addition, by adding an extra parameter $\beta \in \mathbb{R}$, we then have a new control variate

$$g(x; \gamma) := g_\theta(x) + \beta$$

that has β-mean under π, where $\gamma := (\theta, \beta)^\top \in \mathbb{R}^{p+1}$ and β is an additional parameter used to approximate $\Pi[f]$; see Remark 3.3.2 for a detailed discussion.

Once a suitable family of control variates is identified and employed, we then need to select an effective control variate for the integrand of interest f from this family. In this chapter, we limit ourselves to neural networks as in Wan et al. (2019). This means that we need to learn a good value of the parameters of neural networks. Given samples $S = \{x_i, \nabla \log \pi(x_i), f(x_i)\}_{i=1}^m$, and following empirical loss minimisation, the parameter γ can be estimated by minimising

$$J_S(\gamma) := \frac{1}{m} \sum_{i=1}^m \left(f(x_i) - g(x_i; \gamma)\right)^2 .$$

This is the loss we used for scalar-valued control variates in Chapter 4 which has also been discussed in Chapter 3. Penalised objectives of control variates have also been proposed to avoid over-fitting in practice (South et al., 2022a; Wan et al., 2019; Si et al., 2021). Since controlling the strength of such penalties is challenging and is not the focus of this chapter, we will proceed with the un-regularised objectives in this chapter.

5.3 The Proposed Method

We now present and set out the details of the proposed method: meta-learning control variates (Meta-CVs).

5.3.1 Problem Set-up

Consider a finite but possibly large number, T, of integration tasks

$$\Pi_1[f_1], \ldots, \Pi_T[f_T].$$

Denote by $\mathcal{T}_t := \{f_t, \pi_t\}$ the essential components of the t^{th} integration task, which consists of an integrand $f_t \in \mathcal{L}^2(\Pi_t)$ and a target density $\pi_t : \mathcal{X} \to [0, \infty)$. Meanwhile, we assume that for each task we have access to the data of the following

form

$$D_t = \{x_i, \nabla \log \pi_t(x_i), f_t(x_i)\}_{i=1}^{N_t},$$

where the sample size $N_t \in \mathbb{N}^+$ is relatively small per task.

We make the assumption that these tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ are related. In a general sense, we can consider these tasks as realisations sampled from an *environment* denoted as ρ. However, we do not aim to formally define this concept. With this set-up in place, we can then frame Meta-CVs within the framework of gradient-based meta-learning (Finn et al., 2017, 2019). This allows us to leverage gradient-based optimization techniques to learn CVs efficiently across multiple related integral estimation tasks.

5.3.2 Meta-learning CVs

From Chapter 3, we know that *gradient-based meta learning* (GBML) (Finn et al., 2017, 2019; Grant et al., 2018; Yoon et al., 2018) was proposed in the context of *model-agnostic meta-learning* (MAML) (Finn et al., 2017, 2019). Initially developed for the concept of "learning-to-learn" in a supervised learning context, these methods are primarily designed with a specific emphasis on regression tasks and large-scale image classification tasks. The fundamental idea of these GBML methods is to enhance the model's capacity to adapt to new unseen tasks fast. This is achieved by identifying a meta-model that serves as an initial model, which is capable of being adapted to a new task through a few iterations of gradient-based optimization on its parameters. By leveraging the meta-model and employing gradient-based optimization with a large number of training tasks, these methods are able to learn and perform well on a range of tasks.

In this chapter, we will adapt gradient-based meta-learning to the construction of CVs and present an algorithm that is capable of learning CVs fast and efficiently. Analogous to gradient-based meta-learning, it naturally results in a two-step approach, as outlined in Algorithm 6 and Algorithm 7. In the first step, Algorithm 6, the focus is to learn an effective *Meta-CV* that is assumably close to optimal CVs of

arbitrary tasks and exhibits reasonably good performance across most tasks. While, for the second step, as shown in Algorithm 7, we will fine-tune this Meta-CV to each specific integration task by a small number of steps of gradient-based optimisation on a task-specific objective function. This provides us with a *task-specific* CV for each specific integration task.

For each integration task $\mathcal{T}_t := \{f_t, \pi_t\}$, we split its data D_t into two disjoint sets S_t and Q_t such that $D_t = S_t \cup Q_t$. S_t and Q_t consists of the following form of data points,

$$S_t := \{x_j, \nabla \log \pi_t(x_j), f_t(x_j)\}_{j=1}^{m_t}$$
$$Q_t := \{x_j, \nabla \log \pi_t(x_j), f_t(x_j)\}_{j=m_t+1}^{N_t}.$$

Note that, $\log \pi_t(x)$ are required only due to the requirement of Stein-based CVs.

It is worthy of mentioning that S_t and Q_t correspond to the idea of the support set and the query set in the field of GBML; see Chapter 3 for details. Similar to GBML methods, the roles of S_t and Q_t will also differ depending on whether the task is used for learning the Meta-CV (Algorithm 6) or for learning a task-specific CV (Algorithm 7). More details will be presented when we return to this point below. For simplicity and since N_t is small in our experiments, all of our experiments in this chapter will consider $m_t = N_t/2$ and m_t is an integer.

5.3.2.1 Constructing the Meta-CV

The first step of the proposed method is to construct a Meta-CV. It will then be fine-tuned to a task-specific CV for each integration task. We will follow the approach in Section 5.2 and use a Neural-CV (Wan et al., 2019) as the base CV.

In order to separate the choice of the gradient-based optimization method from our overall general construction of Meta-CVs, we will use the term $\textsc{Update}_L(\gamma, \nabla_\gamma J(\gamma); \alpha)$ to represent L steps of any arbitrary gradient-based optimizer, where $\gamma \in \mathbb{R}^{p+1}$ is the initial parameter value, $\nabla_\gamma J(\gamma)$ is the gradient of an objective $J : \mathbb{R}^{p+1} \to \mathbb{R}$, and α represents (hyper-)parameters of the associated optimisation method. This decoupling allows us to focus on the general framework

and design of Meta-CVs, while leaving the specific choice of optimization technique flexible. For instance, such optimisation methods could be gradient descent and Adam (Kingma and Ba, 2015). It is possible to use more flexible alternatives (Andrychowicz et al., 2016; Grefenstette et al., 2019).

Example 5.3.1 (Gradient Descent as UPDATE_L)**.** *For example, the update corresponding to L-step gradient descent starting at γ_0 consists of $\gamma_j := \gamma_{j-1} - \alpha \nabla J(\gamma_{j-1})$ for $j = 1, \ldots, L$. This is the original setting of gradient-based meta-learning presented by Finn et al. (2017).*

By using such notation, we can now represent an an *idealised Meta-CV* as a CV whose parameters γ satisfy

$$\gamma_{\text{meta}} \in \arg\min_{\gamma \in \mathbb{R}^{p+1}} \mathbb{E}_t \left[J_t \left(\text{UPDATE}_L \left(\gamma, \nabla_\gamma J_t(\gamma); \alpha \right) \right) \right], \tag{5.1}$$

where $\mathbb{E}_t$ denotes expectation with respect to a uniformly sampled task index $t \in \{1, \ldots, T\}$.

The above objective is hard to approximate in general since it requires solving nested optimisation problems. Fortunately, we can follow the method in (Finn et al., 2017), and use a gradient-based bi-level optimisation approach presented in Algorithm 6 to minimise this objective. Note that the above objective requires estimating the gradient of the loss J_t in both the inner and outer level. To avoid over-fitting, we estimate this with two independent datasets: S_t and Q_t. The inner level then corresponds to conditioning the model on S_t while the outer level corresponds to updating the model. This indicates that the above objective is the unrolled learning objective of the gradient-based bi-level optimisation approach used in MAML.

We will then call the output of Algorithm 6 our *meta-parameter* which is denoted as $\hat{\gamma}_{\text{meta}}$. The resulting $g(\cdot; \hat{\gamma}_{\text{meta}})$ will be called the *Meta-CV*.

5.3.2.2 Task-Specific CVs

Once we have identified a meta-parameter $\hat{\gamma}_{\text{meta}}$ (and consequently the Meta-CV $g(\cdot; \hat{\gamma}_{\text{meta}})$), we only need to adapt $\hat{\gamma}_{\text{meta}}$ though few optimisation steps to obtain a task-specific parameter $\hat{\gamma}_t$ and hence the corresponding task-specific CV $g(\cdot; \hat{\gamma}_t)$ for

Algorithm 6: Learning a Meta-CV

Input: Training tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, initial parameter γ_0, UPDATE rule, # update steps L, optimiser parameters α and $\eta_1, \ldots, \eta_{I_{\text{tr}}}$, mini-batch size B, # meta-iterations I_{tr}.

for $i = 1, \ldots, I_{tr}$ **do**
  Sample $t_1, \ldots, t_B$ uniformly from $\{1, \ldots, T\}$.
  for $t \in \{t_1, \ldots, t_B\}$ **do**
    Initialize $\gamma_0^t \leftarrow \gamma_{i-1}$.
    for $j = 1, \ldots, L$ **do**
      $\gamma_j^t \leftarrow \text{UPDATE}(\gamma_{j-1}^t, \nabla_{\gamma_{j-1}^t} J_{S_t}(\gamma_{j-1}^t); \alpha)$.
  $\gamma_i \leftarrow \text{UPDATE}(\gamma_{i-1}, \frac{1}{B}\sum_{b=1}^{B} \nabla_{\gamma_{i-1}} J_{Q_{t_b}}(\gamma_L^{t_b}); \eta_i)$.

Output: The meta-parameter $\hat{\gamma}_{\text{meta}} := \gamma_{I_{\text{tr}}}$.

each task $\mathcal{T}_t$. This is summarised in Algorithm 7, and can be applied either to one of the T tasks used for learning the Meta-CV, or to any unseen new integration tasks. Once such a task-specific CV is obtained, we can use the CV estimator

$$\begin{aligned}\hat{\Pi}_t^{\text{CV}}[f_t] &:= \hat{\Pi}_t^{\text{MC}}[f_t - g(\cdot;\hat{\gamma}_t)] + \Pi_t[g(\cdot;\hat{\gamma}_t)] \\ &= \tfrac{1}{N_t - m_t}\sum_{i=m_t+1}^{N_t} (f_t(x_i) - g(x_i;\hat{\gamma}_t)) + \Pi_t[g(\cdot;\hat{\gamma}_t)].\end{aligned} \tag{5.2}$$

to estimate the corresponding integral $\Pi_t[f_t]$.

Remark 5.3.1. *Note that we once again use two datasets, S_t and Q_t per task of which roles now differ from those in Algorithm 6: S_t will be used for learning the task-specific CV $g(\cdot;\hat{\gamma}_t)$ through Algorithm 7, whilst Q_t will be used to evaluate the CV estimator as shown in* (5.2)*.*

To understand how these task-specific CVs *borrow strength* intuitively, for unseen new tasks we highlight that each task-specific CV is constructed by using $\sum_{t=1}^{T} N_t$ samples in addition to its own data. Thus, when T is large and N_t is small, our task-specific CV may be constructed from a much larger number of samples compared to any CV constructed only using data from a single task. The closeness of the relationship among integration tasks determines the additional performance of including these additional data into the learning process of a CV. We will assess the empirical performance of Meta-CVs in Section 5.4.

Algorithm 7: Task-specific CVs from the Meta-CV

Input: Integration task $\mathcal{T}_t$, meta-parameter $\hat{\gamma}_{\text{meta}}$, UPDATE rule, # update steps L, optimiser parameters α.

Initialize $\gamma_0 \leftarrow \hat{\gamma}_{\text{meta}}$.

for $j = 1, \ldots, L$ **do**

  $\gamma_j \leftarrow \text{UPDATE}(\gamma_{j-1}, \nabla J_{S_t}(\gamma_{j-1}); \alpha)$.

Output: Task-specific parameter $\hat{\gamma}_t := \gamma_L$.

5.3.2.3 Computational Complexity

In this section, we will discuss the computational complexity of the proposed method.

Suppose that the meta-parameter $\hat{\gamma}_{\text{meta}}$ of the Meta-CV has already been identified. The additional computational complexity of training all task-specific Neural-CVs is $\mathcal{O}(TL)$, where T is total number of tasks for evaluation and L is the number of optimisation steps used to fine-tune the Meta-CV to each specific task. Usually a large number of optimisation steps (i.e. a large L) is required to learn parameters of a neural network. But due to meta-learning, we would expect the value of L to be very small. We actually take $L = 1$ in most of experiments in Section 5.4.

Remark 5.3.2 (Trade-off on the value of L)**.** *When L increases, the task-specific CV will be less and less dependent on the Meta-CV. As a result, it will take less and less into account the data from the other tasks, and revert to a case where it only depends on the very few data point from the current task. Clearly, there is therefore a trade-off between remaining close to the Meta-CV, and specialising each CV to a specific task; see Antoniou et al. (2019) for a detailed discussion.*

Moreover, taking L to be a small number indicates that fine-tuning a Meta-CV can be orders of magnitude faster compared to training Neural-CVs independently for each integration task. This will be experimentally assessed in Section 5.4.

The proposed method, of course, also needs to take the complexity of learning the Meta-CV into account. This can require a large number I_{tr} and L of optimisation steps in general, which we assess both via experiments in Section 5.4 and theoretical analysis in Section 5.5. The computational cost in p, the number of neural network parameters, is at least $\mathcal{O}(p^2)$ (Fallah et al., 2020) for the proposed method (due

to second-order derivatives in Algorithm 6) while the cost of Neural-CVs is $\mathcal{O}(p)$. This could be a challenge when p is very large. We will return to this issue and discuss it more in the conclusion section of this chapter.

5.4 Experimental Assessment

In this section, we will assess the empirical performance of the proposed Meta-CV method experimentally through a range of problems of increasing complexity where the sample size per task N_t is small and the number of integration tasks T is large. For simplicity, we restrict ourselves to the setting when the sample size per task N_t are identical and we use the Adam optimiser in all experiments.

Note that it is not feasible to apply existing methods discussed in Section 3.4 or the *vector-valued control variates* (vv-CVs) proposed in Chapter 4 to the problems in this section. Since the number of tasks T we consider in this section is a very large value, it can lead to extremely expensive computational cost for these methods. We therefore only compare to methods which do not multitask learn multiple integration tasks: Monte Carlo (MC), Neural-CVs (Wan et al., 2019) and control functionals (CFs) (Oates et al., 2017). Code is available at: `https://github.com/jz-fun/Meta_Control_Variates`.

5.4.1 A Synthetic Example

The first example we consider is trigonometric functions, which are common benchmarks for meta-learning (Finn et al., 2017; Grant et al., 2018) and CVs (Oates et al., 2017, 2019). Consider integrands of the form

$$f_t(x; a_t) = \cos\left(2\pi a_{t,1} + \textstyle\sum_{i=1}^{d} a_{t,i+1} x_i\right),$$

with parameters $a_t \in \mathbb{R}^{d+1}$, and let π_t be the uniform distribution on $\mathcal{X} = [0,1]^d$. In this example, the integrals $\Pi_t[f_t]$ can then be explicitly computed and used as a ground truth for assessment. Note that each integrand f_t is parameterised by a_t which controls the overall difficulty of the t^{th} integration task. We then sample a_t from a distribution ρ consisting of independent uniforms to generate related inte-

gration tasks. Full details are presented in Appendix B.3.1.

In Figure 5.1, we consider the case when $d = 2$. We train the Meta-CV on $T = 20,000$ integration tasks in total. To challenge Meta-CVs, we assess all methods in terms of their performance on an *additional* $T_{\text{test}} = 1,000$ tasks, which are not available during the training of the Meta-CV. On the left panel of Figure 5.1, we investigate the effect of the sample size N_t per task on the performance of CVs. It is found that Meta-CVs tend to outperform MC, CFs and Neural-CVs in terms of mean absolute error over all new unseen tasks regardless of the sample size considered. This can be explained by the fact that the proposed Meta-CVs method is the only approach that can explore and exploit transferrable information across integration tasks. Thus, it is able to leverage the large training dataset. In this case, it is also found that Neural-CVs and CFs perform even worse than MC when the sample size N_t is small, which highlights the challenge of employing CVs under these settings. In the right panel of Figure 5.1, we also investigate the effect of L, the number of gradient-based updates, which shows the robustness of Meta-CVs to L in this example. In Figure 5.2, we then investigate the effect of dimensionality d on the performance of these methods. Clearly, all CVs methods suffer from the curse of dimensionality to some extent. But Meta-CVs do improve on the other CVs when $d < 6$.

In Figure 5.3, we investigate the effect of B and I_{tr} through the lens of the performance of Meta-CVs on 1000 2-dimensional unseen new integration tasks. From Figure 5.3, we find empirically that a larger value of B helps Meta-CVs achieve the optimal performance faster; and a larger value of I_{tr} results in better performance as expected.

We conclude this example with a discussion of empirical computational cost. The empirical computational cost of computing independent Neural-CVs on all unseen test tasks is about 2 minutes. While, the offline computational time for learning the Meta-CV is about 7 minutes (with $L = 1$ when $T_{\text{train}} = 20000$, $N_t = 10$ and $d = 1$), but the online computational time for deriving task-specific CVs for the same 1000 unseen test integration tasks is approximately 6 seconds in total (i.e.,

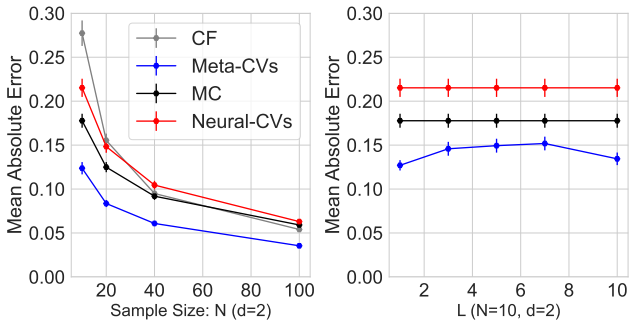

Figure 5.1: Mean absolute error (with 95% confidence intervals) for $T_{\text{test}} = 1,000$ oscillatory functions (with $N_t = N$ and $m_t = n_t = N/2$ for all t). *Left*: Increasing sample size N_t when $d = 2$ (Meta-CVs with $L = 1$); *Right*: Increasing number of inner gradient steps L of Meta-CVs.

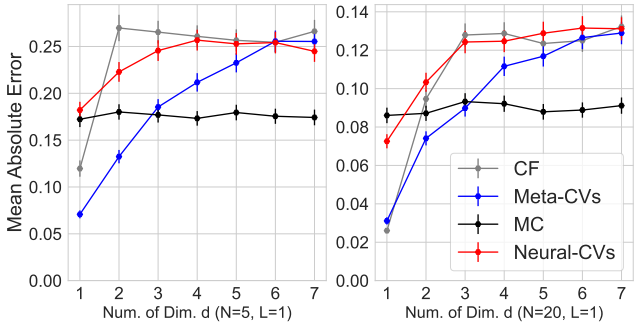

Figure 5.2: Mean absolute error (with 95% confidence intervals) for $T_{\text{test}} = 1,000$ oscillatory functions for increasing dimension d (with $N_t = N$ and $m_t = n_t = N/2$ for all t).

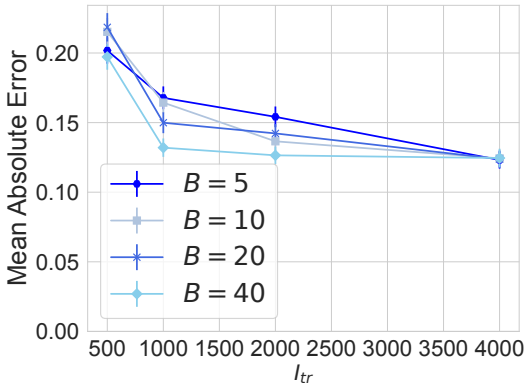

Figure 5.3: Mean absolute error (with 95% confidence intervals) of Meta-CVs for $T_{\text{test}} = 1,000$ 2-dimensional oscillatory functions for increasing B and I_{tr} (with $L = 1$, $N_t = N$ and $m_t = n_t = N/2 = 5$ for all t).

0.006 seconds per task). This shows that the Meta-CV can be adapted to new task-specific CVs of new integration tasks very fast.

5.4.2 Uncertainty Quantification for Boundary Value ODEs

The second example we consider is the computation of expectations of functionals of physical models which are often represented through differential equations. Such expectations are often taken with respect to expert-elicited distributions over parameters of these differential equations in order to perform uncertainty quantification.

In particular, we now consider a boundary-value ODE with unknown forcing closely resembling that of Giles (2015):

$$\tfrac{\mathrm{d}}{\mathrm{d}s}(c(s)\tfrac{\mathrm{d}u}{\mathrm{d}s}) = -50x^2, \qquad 0 < s < 1,$$

with boundary $u(0) = u(1) = 0$, $c(s) = 1 + as$. The integrand of interest is $f_t(x) = \int_0^1 u(s, x; a_t)\mathrm{d}s$ where a_t are draws from $\rho = \mathsf{Unif}(0, 1)$. The integral of interest is $\Pi_t[f_t] := \mathbb{E}_{X\sim\pi_t}[f_t(X)]$ where each $\pi_t = \mathcal{N}(0, 1)$. We use a finite difference approximation of f_t described in Giles (2015). See Appendix B.3.2 for full detail. Though it is a relatively simple example, it represents a broader class of challenging problems where we need improved numerical methods to approximate integrals well due to a large cost per integrand evaluation and therefore limited sample size N_t per task.

The performance of Meta-CVs on $T_{\text{test}} = 100$ unseen tasks is presented in Figure 5.4. We also compare it with MC and Neural-CVs on the same set of unseen test tasks. In this example, it is found that Meta-CVs outperform Neural-CVs and MC consistently across various settings of sample size N_t per task. This highlights once again the benefits of Meta-CVs as it shares information among a large number of tasks when N_t is small.

5.4.3 Bayesian Inference for the Lotka–Volterra System

Our next example is also relevant to uncertainty quantification for differential equation-based models. This time, we consider it in a fully Bayesian framework.

The specific model we consider here is a parametric ODE system, the Lotka–Volterra model (Lotka, 1927), which is commonly used in the fields of ecology and

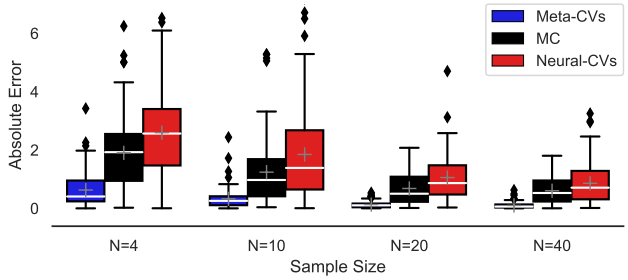

Figure 5.4: Absolute error for $T_{\text{test}} = 100$ (with $N_t = N$ and $m_t = n_t = {}^{N}/_{2}$ for all t.) unseen tasks from the boundary value ODE problem (grey crosses are mean absolute errors; white horizontal lines are medians).

epidemiology. It has the following form,

$$\frac{\mathrm{d}u_1}{\mathrm{d}s} = x_1 u_1 - x_2 u_1 u_2, \ \frac{\mathrm{d}u_2}{\mathrm{d}s} = x_3 u_1 u_2 - x_4 u_2,$$

where $u_1(s)$ and $u_2(s)$ are the numbers of preys and predators at time s, and $u_1(0) = x_5$ and $u_2(0) = x_6$.

Consider that we have access to observations of $u = (u_1, u_2)$ at time points $\{s_1, \ldots s_q\}$ with independent log-normal observational random noise with variances x_7 and x_8, respectively. A 'task' considered here is to compute the posterior expectation of model parameters x conditioning on a given observation dataset. Different observation datasets could correspond to various animal species, or to various geographical regions, and also determine the posterior distribution π_t of interest. It is challenging to perform Bayesian inference on this kind of ecological (Bolker, 2008) and epidemiological (Brauer, 2017) models due to the high computational cost of MCMC sampling, which limits the number of effective independent samples N_t per task.

We use the dataset from Hewitt (1921) on snowshoe hares (preys) and Canadian lynxes (predators) here. To mimic the process of sampling sub-populations, we sub-sample the whole dataset. The goal is to learn a Meta-CV such that it can be adapted to new sub-populations observed in the future fast. See Appendix B.3.3 for full detail.

In Figure 5.5, we compare the proposed approach to MCMC (a *No-U-Turn*

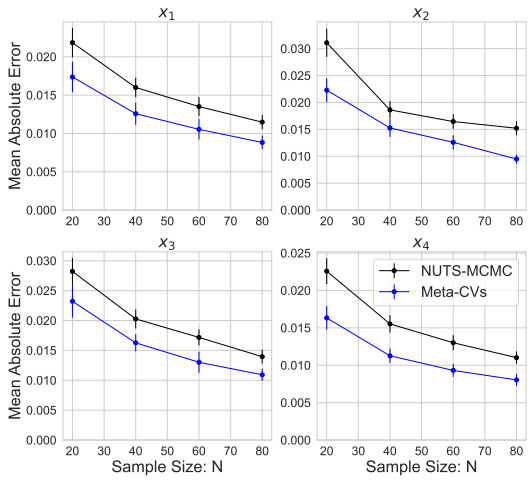

Figure 5.5: Mean absolute errors (with 95% confidence intervals) over 40 sub-populations for varying N_t. Here, $N_t = N$ and $m_t = n_t = {}^{N}/_{2}$ for all t.

Sampler (NUTS) implemented in Stan (Carpenter et al., 2017)). As previously observed and discussed, the performance of CVs is not satisfactory in high-dimensions especially when N_t is small. This is exactly what we observe experimentally: Neural-CVs perform between $5-$ and $12-$times worse than MCMC and thus we exclude it from Figure 5.5. On the contrary, the proposed Meta-CVs approach is still able to achieve a better performance than MCMC for N_t considered. This demonstrates the clear advantage of sharing information across integration tasks for higher-dimensional problems.

5.4.4 Marginalization in Hierarchical Gaussian Processes

The last example is marginalisation of hyper-parameters in Bayesian statistics. In particular, we consider a canonical example, Sarcos anthropomorphic robot arm, for hierarchical Gaussian process regression (Rasmussen, 2003). This example has been considered in both fields of hierarchical Gaussian processes (Rasmussen, 2003) and CVs (Oates et al., 2017). The problem is to recover the unknown function ν that describes a 7 degrees-of-freedom Sarcos robot arm, from a 21-dimensional input space and is based on a subset of the dataset in Rasmussen (2003).

We have access to observations $y_i = \nu(z_i) + \epsilon_i$ at inputs z_i for $i = 1, \ldots, q$, where ϵ_i are IID zero-mean Gaussian random variables with known standard deviation $\sigma > 0$. We place a zero-mean Gaussian process prior on ν, with ker-

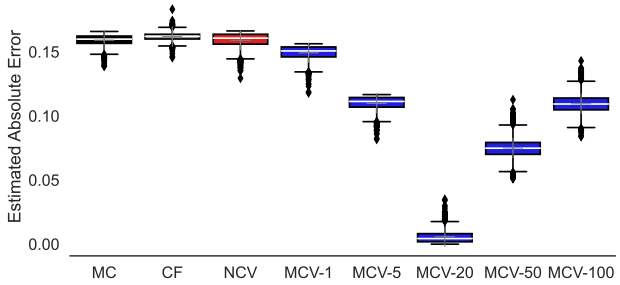

Figure 5.6: Effect of L: Estimated absolute errors over $T_{\text{test}} = 1,000$ unseen states of the Sarcos anthropomorphic robot arm (*CF*: Control functionals; *NCV*: Neural-CVs; *MCV-L*: Meta-CVs with L inner steps).

nel $k_x(z, z') = x_1 \exp(-\frac{\|z-z'\|_2^2}{2x_2^2})$. We also place priors on the hyper-parameters $x = (x_1, x_2)$, i.e., $x \sim \pi_{\text{prior}}$.

Conditioning on observations $y_{1:q} = (y_1, \ldots, y_q)^\top$, we consider a 'task' of predicting the response $\nu(x^*)$ at an unseen state z^* marginalising out the posterior $\pi(x|y_{1:q})$ on kernel hyper-parameters of the Gaussian process model. This can be achieved through the Bayesian posterior predictive mean $\mathbb{E}[Y^*|y_{1:q}] = \mathbb{E}_{X\sim\pi(\cdot|y_{1:q})}[\mathbb{E}[Y^*|y_{1:q}, X]]$. This is an integral of

$$\begin{aligned} f_{z^*}(x) &= \mathbb{E}[Y^*|y_{1:q}, x] \\ &= K_{z^*,q}(x)(K_{q,q}(x) + \sigma^2 I_q)^{-1} y_{1:q} \end{aligned}$$

against the posterior on hyperparameters $\pi(x|y_{1:q})$ (i.e. the target Π), where $(K_{q,q}(x))_{i,j} = k_x(z_i, z_j)$ and $(K_{z^*,q}(x))_j = k_x(z^*, z_j)$ for $i, j \in \{1, \ldots, q\}$. Each task has a different integrand determined by z^* which integrates against the same posterior of kernel hyper-parameters, $\Pi[f_{z^*}]$, for all z^* of interest.

Note that the integrand requires $\mathcal{O}(q^3)$ operations per evaluation (i.e. for each pair of a state and a sample from the posterior) and is therefore expensive to evaluate. It can be quite significant when q is beyond a few hundred. However, it is also common to compute the Bayesian posterior predictive mean for several new states $z_1^*, \ldots, z_T^*$, which leads to closely related integrands $f_{z_1^*}, \ldots, f_{z_T^*}$ whose relationship could potentially be leveraged and be beneficial.

We divide the dataset into two disjoint parts. The first part consists of

$q = 1,000$ data points and is used to infer the posterior of kernel hyper-parameters (which is approximated through variational inference). The second part consists of $4,449$ data points, half of which are used to learn the Meta-CV while the other half are used to as a held-out test set of tasks for assessment. See Appendix B.3.4 for full detail.

We compare the performance of Meta-CVs with MC, Neural-CVs, and CFs and present the results in Figure 5.6. All the methods are tested on $T_{\text{test}} = 1,000$ unseen tasks with $N_t = 4$ samples per task. Although we do not have access to the exact value of these integrals, the value y_t^* is an unbiased estimator, and this enables integration error to be unbiasedly estimated. It is found that Meta-CVs outperform the other methods once again. Interestingly, the performance of Meta-CVs improves when the number of inner gradient steps $L > 1$. Meanwhile, we can see the trade-off on the value of L empirically as discussed in Section 5.3.2.3. See (Antoniou et al., 2019) for a more detailed discussion.

5.5 Theoretical Analysis

In the previous section, extensive experimental results demonstrate the advantage of leveraging a large number of integration tasks for learning CVs for unseen new tasks. In this section, we will provide theoretical analysis onto the guidance of the implementation of gradient-based optimisation for *Meta-CVs*.

In particular, our analysis will mainly focus on strategies for training of Meta-CVs. This is challenging due to the dis-continuity of the global objective of learning a Meta-CV, which arises from the inner optimisation steps of these MAML-based algorithms (Finn et al., 2017; Fallah et al., 2020). Recall that the global objective (i.e. the unrolled objective) for learning a Meta-CV is

$$\begin{aligned} &\arg\min_{\gamma \in \mathbb{R}^{p+1}} \ \mathbb{E}_t \left[\mathcal{J}_t(\gamma) \right], \\ &\mathcal{J}_t(\gamma) := J_{Q_t} \left(\textsc{Update}_L \left(\gamma, \nabla_\gamma J_{S_t} \left(\gamma \right); \alpha \right) \right), \end{aligned} \tag{5.3}$$

where in what follows $\textsc{Update}_L$ is gradient descent with L steps and learning rate α.

We make the following assumptions on the properties of g with respect to its parameter γ:

Assumption 5.5.1. *For each t and $x \in D_t$, $\gamma \mapsto g(x;\gamma)$ and $\gamma \mapsto \nabla_\gamma g(x;\gamma)$ are bounded and Lipschitz.*

Assumption 5.5.2. *For each t and $x \in D_t$, $\gamma \mapsto \nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top - \nabla^2_\gamma g(x;\gamma)$ is bounded and Lipschitz.*

As a function of x, Stein-based CVs g are usually unbounded. But Assumption 5.5.1 can be satisfied by the Stein-based CVs (presented in Section 5.2 and Chapter 3) as it only requires the properties of $g(x;\gamma)$ and $\nabla_\gamma g(x;\gamma)$ as γ, rather than x. In terms of Assumption 5.5.2, since $\nabla_\gamma g(x;\gamma)\nabla_\gamma g(x;\gamma)^\top$ is a widely adopted low-rank approximation to the Hessian $\nabla^2_\gamma g(x;\gamma)$, it explicitly requires this low-rank approximation to be reasonably good.

Given the above two assumptions, we then present the following theorem. It is built on the work of Ji et al. (2022). The theorem establishes conditions under which Algorithm 6 can find an ϵ-first order stationary point of the meta-learning objective function (5.3), for any $\epsilon > 0$.

Theorem 5.5.1. *Let $\hat{\gamma}_{\text{meta}}$ be the output of Algorithm 6 with gradient descent steps, using the meta-step-sizes $\eta_1, \ldots, \eta_{I_{tr}}$, the inner-step size α and batch size B proposed in Theorem 9 and Corollary 10 of Ji et al. (2022). Then, under Assumptions 5.5.1, 5.5.2:*

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

where the outer expectation is with respect to sampling of the mini-batches of tasks in Algorithm 6.

See Appendix B.2.2 for proof.

The above theorem indicates the ideal value for the number of meta iterations I_{tr}. If we choose the meta batch size $B \geq C_B \epsilon^{-2}$ with C_B a large constant, the

output of Algorithm 6 $\hat{\gamma}_\epsilon$ will satisfy $\mathbb{E}[\|\mathbb{E}_t[\nabla \mathcal{J}_t(\hat{\gamma}_\epsilon)]\|] = \mathcal{O}(\epsilon)$ with at most $I_{\text{tr}} = \mathcal{O}(1/\epsilon^2)$ meta iterations.

When using Neural CVs as the base for Meta-CVs, there will not be a unique γ_{meta} since they are highly non-linear in the parameters γ. Therefore, it is hard to further extend the result in Theorem 5.5.1 in this case. However, for simpler CVs which are linear in their parameters, such as those based on polynomials (Assaraf and Caffarel, 1999; Mira et al., 2013; Papamarkou et al., 2014; Friel et al., 2014b; South et al., 2022a), we can reasonably assume a unique γ_{meta} and convexity of the Meta-CV objective around this point. Given these additional assumptions, we are able to show that $\hat{\gamma}_\epsilon$ tend to be close to the minimiser of the task-specific objective functional. This result is presented in the following corollary.

Corollary 5.5.1.1. *Under the setting of Theorem 5.5.1, further suppose that there exists $\mu > 0$ such that for all t and all γ, $\nabla^2 J_{Q_t}(\gamma) \succeq \mu I_{p+1}$ where I_{p+1} is an identity matrix of size $p+1$. Then there exist constants $C_1, C_2 > 0$ such that*

$$\mathbb{E}[\mathbb{E}_t[\|\hat{\gamma}_\epsilon - \gamma_t^*\|_2]] \leq \tfrac{C_1}{\mu}\epsilon + \tfrac{C_2}{\mu},$$

where γ_t^* is the (unique) minimiser of $\gamma \mapsto J_{Q_t}(\gamma)$, and here again the outer expectation is with respect to sampling of the mini-batches of tasks in Algorithm 6.

See Appendix B.2.3 for proof.

These results justify that one can learn such Meta-CVs and task-specific CVs with Algorithm 6. Meanwhile, they provide us with vital insight into the theoretically optimal choice of the hyper-parameters of the proposed method, including the value of the meta-iterations I_{tr}, mini-batch size B, inner step size α and meta-step-sizes $\eta_1, \ldots, \eta_{I_{\text{tr}}}$. The assumptions establish explicit conditions and requirements on the properties of CVs $g(x;\gamma)$ which can be trained successfully (i.e. finding an ϵ-first order stationary point of (5.3)) with Algorithm 6.

5.6 Conclusions

In this chapter, we introduced Meta-CVs, an extension of existing control variates methods. The proposed Meta-CVs approach brings meta-learning to bear on MC and MCMC. To summarise, it can achieve significant variance reduction even when the number of samples per task is small, but a large number T of similar tasks are available. In addition, most of the computational cost is an offline cost for learning a Meta-CV, and task-specific CVs for new integration tasks can be identified with minimal additional computational cost.

Our proposed algorithm is scalable in T and N_t, but it can require significant computational cost for learning the Meta-CV when dealing with flexible CVs such as Neural-CVs. For instance, the computational complexity scales as $\mathcal{O}(p^2)$ in the number of parameters p in the CV, which could prevent us from using large neural networks and thus limit the performance on more challenging tasks. One promising direction for future work is to consider first-order or Hessian-free meta-learning algorithms (Fallah et al., 2020) which can potentially alleviate the cost.

Alternatively, it is also possible to adapt online meta-learning algorithms (Finn et al., 2019) to CVs, which could be particular powerful when integration tasks arrive sequentially and Meta-CVs cannot be computed offline. Such examples include application fields where sequential importance sampling and sequential MC-type algorithms (Doucet et al., 2000, 2001) are currently being used, e.g. in the context of state-space models.

Finally, it is also possible to extend our theoretical analysis of the proposed Meta-CVs approach. The current convergence rate in I_{tr} of Meta-CVs presented in Section 5.5 aligns with (Fallah et al., 2020; Ji et al., 2022). It could be also beneficial to extend the theoretical analysis of Meta-CVs from an information theoretic aspect (Chen et al., 2021) or towards a faster convergence rate (Riou et al., 2023) with extra conditions.

Chapter 6

Amortized Bayesian Prototype Meta-learning for Few-shot Image Classification

Deep supervised learning algorithms (e.g., LeCun et al., 2015) often rely on a substantial amount of labeled data to train neural networks effectively for satisfactory performance on new tasks. However, acquiring a large labeled dataset for each new task can be very expensive or even impractical. Furthermore, re-training neural networks for every incoming task may be computationally unfeasible. As a result, learning an unseen new task with limited labeled observations poses a significant challenge. To overcome this challenge, a novel approach known as meta-learning or few-shot meta-learning has been conceptualised and proposed, which attracts lots of attention. These algorithms aim to tackle the problem of learning unseen new tasks fast and efficiently when only a small number of labeled observations are available (in the context of supervised learning). By leveraging knowledge learnt from previous tasks, meta-learning enables the efficient adaptation of neural networks to new tasks with limited data per task, and thus it provides us with a promising solution/idea to such challenges.

In this chapter, we will introduce a novel probabilistic meta-learning algorithm tailored for image classification specifically. The related background knowledge has already been provided in Chapter 3. In Section 6.3, we formally present the

proposed *amortized Bayesian prototype meta-learning*. We demonstrate its superior performance on a range of image benchmark datasets in Section 6.4.

6.1 Introduction

Meta-learning (Bartunov and Vetrov, 2018; Jamal and Qi, 2019; Finn et al., 2017; Grant et al., 2018; Amit and Meir, 2018; Li et al., 2019; Xu et al., 2020; Ren et al., 2019; Rusu et al., 2019; Sun et al., 2019; Hospedales et al., 2020; Wang et al., 2019b; Iakovleva et al., 2020; Patacchiola et al., 2020) aims to develop algorithms that can solve unseen new tasks efficiently based on the knowledge extracted from previous tasks. These methods have shown remarkable performance on few-shot classification on many image benchmark datasets, e.g. *CUB-200-2011* dataset containing 11788 birds' images from 200 different classes (Wah et al., 2011). Built on probabilistic structures over data and parameters of neural networks, probabilistic meta-learning methods (Grant et al., 2018; Gordon et al., 2019; Ravi and Beatson, 2019; Finn et al., 2018; Yoon et al., 2018; Nguyen et al., 2020; Patacchiola et al., 2020) are able to learn the posteriors of parameters and use posterior predictive samples to solve unseen new tasks that emerge during the meta-testing phase.

However, one drawback of these probabilistic methods is that they treat the parameters of large neural networks as random variables. Consequently, this introduces a substantial number of random variables, leading to severe computational and inferential challenges, such as identifiability problems. This can pose a hurdle for effectively applying probabilistic methods in practice. *Our motivation stems from here, which is: can we instead introduce an embedding space with much less random variables while still well representing the generative process of meta-learning?* To this end, we introduce latent class prototypes to probabilistic meta-learning, which largely reduce the number of random variables while achieving competitive performance.

In this chapter, we propose a novel and simple probabilistic meta-learning method, named *amortized Bayesian prototype meta-learning*, for few-shot image classification. Our approach introduces a novel latent random vector, denoted as

z, which serves as class prototypes for each task. It is capable of learning to learn the posterior distributions of these latent variables for each task effectively whenever at meta-training or meta-testing stage. This capability is obtained by careful design of the prior and variational distributions for the latent class prototypes and is based on gradient-based meta-learning like model-agnostic meta-learning (Finn et al., 2017). In particular, we propose task-dependent priors for latent class prototypes conditional on the support set of each task, and replace the Kullback–Leibler (KL) divergence term in the evidence lower bound of the marginal log-likelihood of the support set with an unbiased estimator to its expectation. Notably, the proposed method does not require any pre-training and can be trained end-to-end. Experimental results demonstrate its competitive performance on many real-world image benchmark datasets, such as *mini-ImageNet* (Vinyals et al., 2016), *Stanford-dogs* (Khosla et al., 2011) and *CUB-200-2011* (Wah et al., 2011).

6.2 Related Work

Gradient-based meta-learning and metric-based meta-learning methods have already been introduced in Chapter 3. In this section, we will discuss the probabilistic variants of gradient-based meta-learning methods which are closely related to the proposed method and summarize the key differences from these previous probabilistic algorithms.

Probabilistic Meta-learning Methods The proposed method is based on *MAML* (Finn et al., 2017), a gradient-based meta-learning method aiming to learn a shared initialization of neural network's parameters that can be adapted to an unseen novel task with only few steps of gradient descent. Probabilistic variants of *MAML* include *MAML-HB* (Grant et al., 2018), *BMAML* (Yoon et al., 2018), *PLATIPUS* (Finn et al., 2018), *VAMPIRE* (Nguyen et al., 2020), *Meta-Mixture* (Jerfel et al., 2019) and *ABML* (Ravi and Beatson, 2019). *MAML-HB* interpreted *MAML* from a hierarchical Bayes learning perspective. *PLATIPUS* proposed to learn the joint posterior of meta initialization γ and task-specific parameters conditional on the support set of each task $\mathcal{T}_t$, while *BMAML*, *VAMPIRE* and *ABML* learned the posterior distributions of

the task-specific parameters conditional on the γ and the support set. More specific, *BMAML* learned the posteriors of task-specific parameters through Stein variational gradient descent, which was distinct from the others. *ABML* proposed to minimize the loss of the support and query sets of a task jointly (i.e. equivalent to maximizing $\mathbb{E}[\log p(S, Q)]$), which did not explicitly encourage neural networks to maximize $\mathbb{E}[\log p(Q|S)]$. *VAMPIRE* was similar to *ABML*. The main difference was that *VAMPIRE* only used the loss of the query set of a task to update the global shared parameter γ, while *ABML* used both the support and query sets. *VERSA* (Gordon et al., 2019) proposed to directly maximize $\log p(Q|S)$, which was achieved by only learning the posteriors of parameters of the linear classifier via an extra amortization network.

Key Differences from Previous Work The proposed approach has some key differences from previous work in the following aspects. 1) Previous probabilistic meta-learning methods (e.g., Gordon et al., 2019; Ravi and Beatson, 2019; Finn et al., 2018; Yoon et al., 2018; Nguyen et al., 2020; Jerfel et al., 2019) treat the parameters of neural networks as random variables, while our method introduces latent prototypes as random variables. 2) The proposed approach learns to learn the posterior distributions of latent class prototypes in an amortized inference way with no need for an extra amortization network. 3) Assuming that the support images and the query images come from same data generating process, we replace the $\mathbb{KL}$ term in the evidence lower bound of $\log p(S)$ with an unbiased estimator to its expectation (6.7), which is purely dependent on the support set of a task. In addition, the proposed method provides us with a more interpretable and simpler way of modelling, and achieves state-of-the-art or competitive performance on various benchmark datasets.

6.3 The Proposed Method

In this section, we will discuss how meta-learning can be achieved by maximizing the expectation of the posterior predictive likelihood in Section 6.3.1 and present the proposed *amortized Bayesian prototype meta-learning* for few-shot image clas-

sification in Section 6.3.2.

6.3.1 Meta-learning via Maximizing Expectation of Posterior Predictive Likelihood

Suppose that we can sample a series of tasks $\{\mathcal{T}_t\}_{t=1}^{T}$ from an environment (or meta population) ρ, meta-learning aims to learn an algorithm $\mathcal{M}$ that can minimize the transfer risk $\mathcal{R}$ on a new task $\mathcal{T}$. $\mathcal{R}$ is defined as follows,

$$\mathcal{R} = \mathbb{E}_{\mathcal{T}\sim\rho}\mathbb{E}_{\{x_i,y_i\}_{i=1}^{m}\sim\mathcal{T}}\mathbb{E}_{\{\tilde{x},\tilde{y}\}\sim\mathcal{T}}[J(\mathcal{M}(\{x_i,y_i\}_{i=1}^{m}),\{\tilde{x},\tilde{y}\})],$$

where J is some loss function measuring the difference of the model's prediction given a testing data point $\tilde{x}$ and the corresponding ground-truth $\tilde{y}$, and $\{x_i,y_i\}_{i=1}^{m}$ are the m training samples from the same task $\mathcal{T}$. The environment ρ refers to a distribution of tasks $\mathcal{T}$ (Denevi et al., 2018, 2019; Baxter, 2000; Ji et al., 2022; Fallah et al., 2020). Without loss of generality and to simplify notation, we increase the number of test samples to n, i.e. $\{\tilde{x}_j,\tilde{y}_j\}_{j=1}^{n}$, and denote $\{x_i,y_i\}_{i=1}^{m}$ by S and $\{\tilde{x}_j,\tilde{y}_j\}_{j=1}^{n}$ by Q. The above risk can then be re-written as follows,

$$\mathcal{R} = \mathbb{E}_{\mathcal{T}\sim\rho}\mathbb{E}_{S\sim\mathcal{T}}\mathbb{E}_{Q\sim\mathcal{T}}\left[J\Big(\mathcal{M}(S),Q\Big)\right]. \tag{6.1}$$

This risk is often approximated by empirical risk, commonly through Monte Carlo estimators. When confronted with a new unseen task $\mathcal{T}$ from which we have access to its support set S and query set Q, the algorithm $\mathcal{M}$ can be adapted with the support set S. The performance of $\mathcal{M}$ on this task is then evaluated by measuring the empirical loss on the query set Q. To assess the overall performance of $\mathcal{M}$, it is crucial to measure its average performance on tasks sampled from ρ. This provides a comprehensive evaluation of $\mathcal{M}$'s generalisation ability and performance across a range of diverse tasks from the environment.

Suppose we have two random variables, S and Q, representing a support set and a query set, respectively. Now consider a probabilistic generative model that is parameterized by γ. This parameterization defines a prior distribution, denoted

as $p_\gamma(z)$, over the latent variables z, as well as a conditional likelihood, denoted as $p_\gamma(S|z)$, for the support set S. These components collectively characterize the underlying probabilistic structure. In order to approximate the posterior distribution $p_\gamma(z|S)$, we can leverage the evidence lower bound (ELBO) of $\log p_\gamma(S)$ (Kingma and Welling, 2013):

$$\log p_\gamma(S) \geq \mathbb{E}_{z\sim q_\phi(z)}[\log p_\gamma(S|z)] - \mathbb{KL}[q_\phi(z)||p_\gamma(z)], \tag{6.2}$$

where $\mathbb{KL}$ denotes the Kullback–Leibler divergence. This lower bound is tight when $q_\phi(z) = p_\gamma(z|S)$. However, the true posteriors are intractable. Fortunately, we can optimize the evidence lower bound with respect to the variational parameters ϕ, which allows us to obtain the approximate posterior $q_\phi(z)$ (Kingma and Welling, 2013). Given $q_\phi(z) \approx p_\gamma(z|S)$, the log posterior predictive likelihood of Q, $\log p_\gamma(Q|S)$, can be approximated. It is given by,

$$\begin{aligned}\log p_\gamma(Q|S) &= \log \textstyle\int p_\gamma(Q|z)p_\gamma(z|S)\, dz \approx \log \mathbb{E}_{z\sim q_\phi(z)}[p_\gamma(Q|z)] \\ &\geq \mathbb{E}_{z\sim q_\phi(z)}[\log p_\gamma(Q|z)].\end{aligned} \tag{6.3}$$

The above discussion indicates that, given γ, we need to first optimize the ELBO of $\log p_\gamma(S)$ to approximate the posterior $p_\gamma(z|S)$. We can then optimize the lower bound of $\log p_\gamma(Q|S)$ (right hand side of (6.3)) with respect to γ. We can find that the equivalence between maximizing the expectation of $\log p_\gamma(Q|S)$ and the risk $\mathcal{R}$ in (6.1) can be established when the loss $J(\mathcal{M}(S), Q) := -\log p_\gamma(Q|S)$.

In practice, the expectation $\mathbb{E}[\log p_\gamma(Q|S)]$ can be approximated well by the average of $\{\log p_\gamma(Q_t|S_t)\}_{t=1}^T$ of the tasks $\{\mathcal{T}_t\}_{t=1}^T$ when the number of training tasks T is large. This is often the case in few-shot meta-learning (Vinyals et al., 2016; Finn et al., 2017). However, tasks are often encountered in a sequential manner, which poses a challenge when using Monte Carlo estimators to approximate the corresponding expectation. Due to the requirement of knowing all tasks and computing all relevant terms simultaneously, we cannot employ Monte Carlo estimators directly in this scenario. Alternatively, we can optimize γ iteratively through each

task once arriving using $\log p_\gamma(Q_t|S_t)$, for $t = 1, \ldots, T$. As discussed by Ravi and Beatson (2019), since T is large, the uncertainty of the posteriors of γ is low. It is reasonable to use a point estimate of γ as in (Ravi and Beatson, 2019). By introducing auxiliary latent random variables z for each task, we can deal with intractable likelihoods and posteriors. In the proposed method, we assume that there exists a generative process wherein each image is generated based on the corresponding latent class prototype. This assumption implies that the latent prototypes capture essential characteristics and features of images. As a result, the approximate posterior $q_\phi(z)$ can then be used as a learnable discriminative classifier tailored for each few-shot image classification task.

6.3.2 Amortized Bayesian Prototype Meta-learning

In this section, we will firstly give a brief overview of the proposed *amortized Bayesian prototype meta-learning* approach, highlight its novelty and strength, and then discuss the details of variational distributions, prior distributions and the loss designed.

6.3.2.1 Overview

Rather than inferring task-specific variational parameters ϕ^t for each task $\mathcal{T}_t$, we can employ a global model V which learns to estimate ϕ^t conditioned on both the support set S_t and the shared set of parameters γ (Marino et al., 2018). It allows us to obtain ϕ^t as a function of S_t and γ, i.e., $\phi^t = V(S_t, \gamma)$ for all tasks. This is also known as amortized variational inference (Kingma and Welling, 2013; Marino et al., 2018; Ravi and Beatson, 2019). In the proposed method, we follow the settings of (Grant et al., 2018; Ravi and Beatson, 2019) and set the task-specific variational distribution $q_{\phi^t}(z) = \mathcal{N}(z; \mu_{\phi^t}, \Sigma_{\phi^t})$. The variational parameters $\phi^t := (\mu_{\phi^t}, \Sigma_{\phi^t})$ can be optimized by $\phi^t = V(S_t, \gamma) := \gamma + \alpha \nabla_\gamma \log p_\gamma(S_t)$ for all t. Hence, we can see that inference for all tasks is amortized by the globally shared set of parameters γ and the model V. To simplify notation, we drop the subscript t in the following content.

In the proposed method, we assume that there is a random vector $z =$

$(z_1, \cdots, z_C)^\top$ acting as class prototypes for each C-way K-shot image classification task. Each element of z is allowed to have a specific Gaussian distribution parameterised by ϕ representing the corresponding generative processes. From the discussion above, we know that we can learn ϕ by $\phi = V(S, \gamma)$. Note that the first term in the evidence lower bound of $\log p_\gamma(S)$ (R.H.S. of (6.2)) and the lower bound of $\log p_\gamma(Q|S)$ (6.3) represents negative reconstruction loss. Hence, we can reformulate them into a negative classification loss $-\mathcal{L}_{\text{PR}}$ in (6.9). In this way, given a task $\mathcal{T}$, we can rewrite (6.2) and (6.3) into a loss on S, $J(S)$, for learning the task-specific variational parameters ϕ, and a meta-loss $J_{\text{meta}}(Q)$ for learning the globally shared parameters γ, respectively.

$$J(S) := \mathcal{L}_{\text{PR}}(S|z) + \mathbb{KL}[q_\phi(z|S) \,||\, p_\gamma(z)], \tag{6.4}$$

$$J_{\text{meta}}(Q) := \mathcal{L}_{\text{PR}}(Q|z). \tag{6.5}$$

Given the current values of γ, the variational parameters ϕ can be optimized using $J(S)$ in (6.4) which are initialized at γ, e.g., several steps of gradient descent or even more sophisticated approach like Adam (Kingma and Ba, 2015). The globally shared set of parameters γ can be updated using $J_{\text{meta}}(Q)$ conditioning on the current query set Q, e.g., $\gamma \leftarrow \gamma - \eta \nabla_\gamma J_{\text{meta}}(Q)$. The pseudo-algorithms of meta-training, meta-validation and meta-testing are summarized in Algorithm 8 and Algorithm 9.

Novelty and Strength Our approach targets at learning to learn $p_\gamma(z|S)$ of the latent class prototypes z for each C-way image classification task efficiently by amortized variational inference. A significant difference from previous probabilistic meta-learning methods (Gordon et al., 2019; Ravi and Beatson, 2019; Finn et al., 2018; Yoon et al., 2018) are summarised as follows. Posteriors of the class prototypes are learnt efficiently. They are used as classifiers directly, rather than generating stochastic classifiers (i.e neural networks parameterized by random weights sampled from their posteriors). With the approximate posterior $q_\phi(z|S)$, the proposed method is able to encourage better classification on the query set through $\mathcal{L}_{\text{PR}}$ explicitly.

6.3.2.2 Design of Variational Distributions, Prior Distributions and Classification Loss

In this section, we will present the details of variational distributions, prior distributions and classification loss designed for the proposed method.

Variational Distributions The proposed approach learns the uncertainty of each image's deep representation through an embedding network f_γ. The embedding network f_γ consists of a convolutional neural network and a fully connected layer; see Section 6.4.1 for details. To be more specific, after an image x being fed forward into the embedding network f_γ, we then obtain a feature vector $f_\gamma(x) \in \mathbb{R}^{2p}$ which is then further split into two parts, i.e., $\mu(x) \in \mathbb{R}^p$ and $\sigma(x) \in \mathbb{R}^p$. We further set $\sigma^2(x) = |w| \odot \text{Sig}(\sigma^2(x)) \oplus |b|$ to ensure a finite $\sigma^2(x)$, where $\text{Sig}(\cdot)$ is a element-wise sigmoid function, and $\{w, b\}$ are two learnable scalars broadcasted to match the dimension of $\text{Sig}(\sigma^2(x)) \in \mathbb{R}^p$. Here, $\odot$ and $\oplus$ represent element-wise multiplication and element-wise addition operators, respectively. Consequently, an image x can be represented by a Gaussian distribution $\mathcal{N}(\mu(x), \Sigma(x))$ where the covariance matrix $\Sigma(x) \in \mathbb{R}^{p\times p}$ is a diagonal matrix with the diagonal elements $\sigma^2(x)$. To aggregate distributions of images from one class into one Gaussian distribution, we consider a matrix-version of harmonic mean. Note that other forms of average can also be taken into account, e.g., simple average. For a C-way K-shot classification task, suppose that S_c denotes the subset of S containing K samples from the class $c \in \{1, \ldots, C\}$, the matrix-version of harmonic mean is,

$$\begin{aligned}\mu_c &= \left\{\textstyle\sum_{x_n\in S_c} \Sigma(x_n)^{-1}\right\}^{-1} \left\{\textstyle\sum_{x_n\in S_c} \Sigma(x_n)^{-1}\mu(x_n)\right\} \ , \\ \Sigma_c &= \left\{\tfrac{1}{|S_c|}\textstyle\sum_{x_n\in S_c}[\Sigma(x_n)^{-1}]\right\}^{-1} \ . \end{aligned} \tag{6.6}$$

We can then set $q(z_c|S_c) = \mathcal{N}(z_c; \mu_c, \Sigma_c)$ for all $c \in \{1, \ldots, C\}$.

Prior Distributions In existing methods (Grant et al., 2018; Gordon et al., 2019; Ravi and Beatson, 2019; Finn et al., 2018; Yoon et al., 2018; Nguyen et al., 2020; Jerfel et al., 2019), z are the parameters of neural networks and are regarded as random variables. In this chapter, since z are no longer the weights of neural net-

works, trivial choices of prior distributions such as standard Gaussian distributions are not desirable. Task-dependent priors are proposed for z in our approach, i.e., $\mathcal{N}(\mu(x), \Sigma(x))$ conditional on a sample x sampled from the support set S of a task $\mathcal{T}$. The reason why we choose this to be the prior is because it has already been extracted by the neural network and thus requires no additional efforts. Due to the randomness induced by x, we propose to replace the $\mathbb{KL}$ divergence term in (6.4) with its expectation,

$$\mathbb{E}_{\{x,y\}\sim\mathcal{T}}\left[\mathbb{KL}\left[q_\phi(z|S)||p_\gamma(z;\mu(x),\Sigma(x))\right]\right]. \tag{6.7}$$

One unbiased estimator of the above expectation is given by

$$\tfrac{1}{CK}\sum_{c=1}^{C}\sum_{n=1}^{K}\mathbb{KL}[q_\phi(z_c|S_c)||\mathcal{N}(z_c;\mu(x_n^{(S_c)}),\Sigma(x_n^{(S_c)}))],$$

where $x_n^{(S_c)}$ is the n_{th} image from the subset S_c. This is valid when $K > 1$. When $K = 1$, it is possible to define priors with both the query set and the support set of a task. However, using samples from the query set to construct priors is arguable since label information may leak during such processes. Hence, for C-way 1-shot classification, we use the same rule in (6.6) to aggregate the Gaussian distributions of the C support samples into a single Gaussian distribution $\mathcal{N}(\mu_{\text{union}}, \Sigma_{\text{union}})$ to represent the prior distribution $p_\gamma(z)$. Note that this is a quite strong prior on z which could result in shrinkage. However, it is still reasonable if being compared to standard Gaussian distributions. In this case, an estimator of the $\mathbb{KL}$ term in (6.4) now becomes $\frac{1}{C}\sum_{c=1}^{C}\ \mathbb{KL}[q_\phi(z_c|S_c)\ ||\ \mathcal{N}(\mu_{\text{union}}, \Sigma_{\text{union}})]$.

Classification Loss For a sample x, we measure its class membership to each of the C classes by

$$\Pr[\mu(x)|z_c] = \mathcal{N}(\mu(x);\mu_c,\Sigma_c)\ , \tag{6.8}$$

for $c = 1,\ldots,C$. Since the sum of the above probabilities over C classes is not restricted to be 1, it is essential to normalize these C values of x to ensure a valid

loss function for classification. Thus, $\mathcal{L}_{\text{PR}}$ in (6.4) and (6.5) is then set to be

$$\mathcal{L}_{\text{PR}}(\tau|z) = -\frac{1}{|\tau|}\sum_{n=1}^{|\tau|}\left[\log\left(\frac{\Pr[\mu(x_n)|z_{y_n}]}{\sum_{c=1}^{C}\Pr[\mu(x_n)|z_c]}\right)\right], \tag{6.9}$$

where τ represents either a support set S or a query set Q.

6.3.2.3 Connection between Loss and Reconstruction Error

In this section, we make the connection between the classification loss and the negative reconstruction error $\mathbb{E}_z[\log p_\gamma(\tau|z)]$ clear where τ can be either S or Q.

Note that

$$\log p_\gamma(\tau|z) = \log(\textstyle\prod_{n=1}^{|\tau|} p_\gamma(y_n|x_n, z)) = \sum_{n=1}^{|\tau|}\log p_\gamma(y_n|x_n, z).$$

The response variable y_n is discrete and takes values in $\{1, \ldots, C\}$ in a C-way classification task. Thus, it should have a probability mass function, which can be accomplished by normalizing over the sum of class membership:

$$p_\gamma(y_n|x_n, z) = \frac{\Pr[\mu(x_n)|z_{y_n}]}{\sum_{c=1}^{C}\Pr[\mu(x_n)|z_c]}.$$

Taking the logarithm of the above equation, we have,

$$\log p_\gamma(\tau|z) = \sum_{n=1}^{|\tau|}\log\left(\frac{\Pr[\mu(x_n)|z_{y_n}]}{\sum_{c=1}^{C}\Pr[\mu(x_n)|z_c]}\right).$$

One unbiased estimator of the negative reconstruction loss $\mathbb{E}_z[\log p_\gamma(\tau|z)]$ is $\log p_\gamma(\tau|z)$ when the sample size of z is one. This is well-behaved in our experiments as z are latent prototypes of each task. Therefore, under this setting, one unbiased estimator of $-\mathbb{E}_z[\log p_\gamma(\tau|z)]$ is then

$$-\sum_{n=1}^{|\tau|}\log\left(\frac{\Pr[\mu(x_n)|z_{y_n}]}{\sum_{c=1}^{C}\Pr[\mu(x_n)|z_c]}\right)$$

which is our $\mathcal{L}_{\text{PR}}(\tau|z)$ multiplied by a factor of $1/|\tau|$.

Remark 6.3.1. *Though the* $\mathbb{KL}$ *term in the evidence lower bound of* $\log p_\gamma(S)$ *is re-*

placed with (6.7) *since the proposed prior of* z *depends on* S*, our estimator to* (6.4) *is still an unbiased estimator to the evidence lower bound of* $\log p_\gamma(S)$ *(scaled by a constant). Therefore, our approach is still able to learn to approximate the posteriors of* z *conditional on* S*. To appreciate this, first note that during the inference stage* τ *is the support set* S *(and we have* $|\tau| = CK$*). After putting the unbiased estimator of* (6.7) *and* (6.8) *into* (6.4)*, the loss in* (6.4) *can be then rewritten as*

$$J(S) = \frac{1}{CK}\sum_{c=1}^{C}\sum_{n=1}^{K}\left[-\log\left(\frac{\Pr\left[\mu(x_n^{(S_c)})|z_c\right]}{\sum_{k=1}^{C}\Pr\left[\mu(x_n^{(S_c)})|z_k\right]}\right) + \mathbb{KL}\left[q_\phi(z_c|S_c)||p_\gamma(z_c;\mu(x_n^{(S_c)}),\Sigma(x_n^{(S_c)}))\right]\right].$$

By taking the negative sign on both sides of the above equation, we have

$$-J(S) = \frac{1}{CK}\sum_{c=1}^{C}\sum_{n=1}^{K}\left[\log p_\gamma(y_n^{(S_c)}|x_n^{(S_c)},z_c) - \mathbb{KL}\left[q_\phi(z_c|S_c)||p_\gamma(z_c;\mu(x_n^{(S_c)}),\Sigma(x_n^{(S_c)}))\right]\right].$$

Note that the terms in the double summation is an unbiased estimator of the evidence lower bound of $\log p_\gamma(y_n^{(S_c)}|x_n^{(S_c)})$*. Since*

$$\frac{1}{CK}\sum_{c,n}\log p_\gamma(y_n^{(S_c)}|x_n^{(S_c)}) = \frac{1}{CK}\log p_\gamma(S),$$

it tells that $-J(S)$ *is an unbiased estimator of the evidence lower bound of* $\log p_\gamma(S)$ *scaled by a factor of* $1/CK$*.*

6.4 Applications to Few-shot Image Classification

In this section, we will evaluate the performance of the proposed method on a range of real-world image datasets. Meanwhile, implementation details and ablation studies are provided.

Algorithm 8: Meta-training for C-way K-shot classification

Input: Model $\mathcal{M}_\gamma$ with initial parameter γ_0, number meta-iterations I_{tr}, inner learning rate α, outer learning rate η, C-way, K-shot, environment ρ .

for *i from 1 to I_{tr}* **do**
- Sample a mini-batch of tasks $\mathcal{T}_t$ from ρ, for $t = 1, ..., B$.
- **for** *each task $\mathcal{T}_t$ from 1 to B* **do**
  - Sample S_t and Q_t from $\mathcal{T}_t$.
  - Initialize $\phi_0^t \leftarrow \gamma_{i-1}$.
  - `/* Approximate inference for posteriors of z conditional on` S_t `*/`
  - **for** *l from 1 to L* **do**
    - $\phi_l^t \leftarrow \phi_{l-1}^t - \alpha \nabla_{\phi_{l-1}^t} \{\mathcal{L}_{PR}(S_t|z) + \mathbb{KL}[q_{\phi_{l-1}^t}(z|S_t) \,||\, p_{\gamma_{i-1}}(z)]\}$.
  - Compute prediction loss: $\mathcal{L}_{\text{PR}}(Q_t|z)$ given ϕ_L^t.
- `/* Update globally shared meta-parameters` γ `*/`
- $\gamma_i \leftarrow \gamma_{i-1} - \eta \nabla_{\gamma_{i-1}} \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_{\text{PR}}(Q_t|z)$.

Output: Return $\mathcal{M}_{\gamma_{I_{\text{tr}}}}$.

Algorithm 9: Meta-validation/Meta-testing of the proposed method

Input: Meta-model $\mathcal{M}_{\gamma_{I_{\text{tr}}}}$, set of tasks for meta-validation/meta-testing $\mathcal{T}_1, \ldots, \mathcal{T}_T$.

for *t from 1 to T* **do**
- Initialize $\phi_0^t \leftarrow \gamma_{I_{\text{tr}}}$.
- `/* Approximate inference for the current task */`
- **for** *l from 1 to L* **do**
  - $\phi_l^t \leftarrow \phi_{l-1}^t - \alpha \nabla_{\phi_{l-1}^t} \{\mathcal{L}_{PR}(S_t|z) + \mathbb{KL}[q_{\phi_{l-1}^t}(z|S_t) \,||\, p_{\gamma_{I_{\text{tr}}}}(z)]\}$.
- Denote $S_{t,c}$ the subset of S_t containing samples from the class $c \in [C]$.
- Predict for an image x: $\hat{y} = \arg\max_c \Pr(\mu(x)|q_{\phi_L^t}(z_c|S_{t,c})),\ c \in [C]$.
- Compute prediction accuracy a_t on the query set Q_t.

Output: Average accuracy $\frac{1}{T} \sum_{t=1}^{T} a_t$.

6.4.1 Implementation Details

We ensure a fair comparison with other methods by adhering to standardized practices in terms of the network architecture, image datasets used for benchmarking, and experimental setup.

Network Architecture Our approach only requires a neural network f_γ for deep feature embedding of raw image data. Such embedding neural networks can be *VGG* (Simonyan and Zisserman, 2015) or *ResNet* (He et al., 2016). To make fair comparisons with existing methods (details in Appendix C), we use a shallow convolution neural network consisting of four convolution blocks and a fully-connected linear layer. This is used as the feature extractor f_γ. Each convolution block has 64 3-by-3 filters, followed by batch-normalization, ReLU activation, and 2-by-2 max-pooling. The fully-connected layer then transforms the flattened features from the previous four convolution blocks into vectors $\in \mathbb{R}^{128}$. By following the steps in Section 6.3.2, we can formulate a Gaussian distribution for $f_\gamma(x)$ of each sample x.

Image Datasets *Omniglot* (Lake et al., 2011) is widely used as a image dataset for benchmarking, which contains 1623 classes from 50 languages. Each class contains 20 samples. For *Omniglot*, we follow the settings of (Vinyals et al., 2016; Snell et al., 2017; Chen et al., 2019b) to augment the classes by rotations in 90, 180 and 270 degrees, which leads to 6492 classes in total. We also split these classes into 4112 classes for meta-training, 688 classes for meta-validation, and 1692 classes for meta-testing by following the settings of Snell et al. (2017); Chen et al. (2019b). All images are down-sampled to $28 \times 28 \times 1$ as a pre-processing step. Another popular real-world image dataset used for object recognition is *mini-ImageNet* (Vinyals et al., 2016). It is a subset of *ImageNet* (Deng et al., 2009), firstly used by Vinyals et al. (2016) to investigate few-shot meta-learning. *mini-ImageNet* has images from 100 different classes. Each class has 600 images with labels. In our experiments, we follow the settings of Ravi and Larochelle (2016); Chen et al. (2019b). The dataset is randomly split into 64 classes for meta-training, 16 classes for meta-validation, 20 classes for meta-testing. The proposed approach is also tested on two fine-grained image datasets, i.e., *CUB-200-2011* (Wah et al., 2011)

and *Stanford-dogs* (Khosla et al., 2011). *CUB-200-2011* consists of 11788 birds' images from 200 different classes. We randomly split the dataset into 100 classes for meta-training, 50 classes for meta-validation and 50 classes for meta-testing, which is the same setting as that of (Chen et al., 2019b). *Stanford-dogs* has 20580 dogs' images from 120 classes. It is also randomly split into three mutually disjoint subsets, 60 classes for meta-training, 30 classes for meta-validation and 30 classes for meta-testing. We down-sample all images from *mini-ImageNet*, *CUB-200-2011* and *Stanford-dogs* to $84 \times 84 \times 3$, which are then fed into the embedding neural network f_γ. Meanwhile, standard data augmentation techniques are employed, i.e., random sized crop, random horizontal flip, and image jitter.

Setup All experiments in this chapter are implemented and evaluated by the use of the episodic meta-training/meta-evaluation processes (Vinyals et al., 2016), which is the widely accepted standard in this field. For few-shot image classification, an *episode* actually refers to a C-way K-shot image classification task (see Definition 3.5.3 for definition) or a mini-batch of such tasks. As discussed in the previous paragraph, each image dataset is randomly split into three disjoint parts. By doing so, we set up three procedures, i.e., meta-training, meta-validation and meta-testing. The optimal number of meta-training epochs is selected according to the accuracy on the meta-validation set. During meta-testing, 600 novel tasks are randomly sampled from the meta-testing set. We report the mean accuracy with its 95% confidence interval. *PyTorch* (Paszke et al., 2019) is used for all experiments. See Appendix C for detailed setting of hyper-parameters, e.g., B, L, α and η in Algorithm 8.

6.4.2 Experimental Results

We now report the performance of the proposed method evaluated on a range of real-world image datasets.

Comparisons to Probabilistic Meta-learning Methods For 5-shot classification on *mini-ImageNet*, the proposed approach achieves state-of-the-art performance for 5-way 5-shot tasks with a lower variance, as shown in Table 6.1. In terms of 1-shot classification on *mini-ImageNet*, the mean accuracy of the proposed approach is

Table 6.1: *Meta-testing Accuracy for* 5*-way Classification on Mini-ImageNet and Omniglot.* These methods all use a comparable feature embedding, i.e. shallow convolution networks (see the supplementary material for details). Bold text indicates the highest mean accuracy and results that overlap with the confidence intervals of those highest mean accuracy.

	Omniglot (%)		*mini-ImageNet* (%)	
	5-way 1-shot	*5-way 5-shot*	*5-way 1-shot*	*5-way 5-shot*
BMAML (Yoon et al., 2018)	-	-	$\mathbf{53.80 \pm 1.46}$	$64.23 \pm 0.69^*$
PLATIPUS (Finn et al., 2018)	-	-	50.13 ± 1.86	-
ABML (Ravi and Beatson, 2019)	-	-	45.00 ± 0.60	-
Amortized VI (Gordon et al., 2019)	97.77 ± 0.55	98.71 ± 0.22	44.13 ± 1.78	55.68 ± 0.91
VERSA (Gordon et al., 2019)	$\mathbf{99.70 \pm 0.20}$	$\mathbf{99.75 \pm 0.13}$	$\mathbf{53.40 \pm 1.82}$	67.37 ± 0.86
Meta-Mixture (Jerfel et al., 2019)	-	-	51.20 ± 1.52	65.00 ± 0.96
VAMPIRE (Nguyen et al., 2020)	98.41 ± 0.19	$\mathbf{99.56 \pm 0.08}$	51.54 ± 0.74	64.31 ± 0.74
DKT (Patacchiola et al., 2020)	-	-	49.73 ± 0.07	64.00 ± 0.09
Ours	98.83 ± 0.17	$\mathbf{99.54 \pm 0.08}$	$\mathbf{53.28 \pm 0.91}$	$\mathbf{70.44 \pm 0.72}$

Table 6.2: *Meta-testing Accuracy for* 5*-way Classification on CUB-200-2011, Stanford-dogs and Mini-ImageNet.* These methods all use the same feature embedding architecture in (Chen et al., 2019b), i.e. four convolution blocks. Results with superscript $*$ means training and testing locally.

	CUB-200-2011 (%)		*Stanford-dogs* (%)		*mini-ImageNet* (%)	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
MatchingNet (Vinyals et al., 2016)	60.52 ± 0.88	75.29 ± 0.75	$45.65 \pm 0.90^*$	$60.87 \pm 0.71^*$	48.14 ± 0.78	63.48 ± 0.66
ProtoNet (Snell et al., 2017)	50.46 ± 0.88	76.39 ± 0.64	$41.07 \pm 0.84^*$	$62.47 \pm 0.69^*$	44.42 ± 0.84	64.24 ± 0.72
RelationNet (Sung et al., 2018)	62.34 ± 0.94	77.84 ± 0.68	$47.20 \pm 0.89^*$	$66.12 \pm 0.71^*$	49.31 ± 0.85	66.60 ± 0.69
Baseline++ (Chen et al., 2019b)	60.53 ± 0.83	79.34 ± 0.61	$44.15 \pm 0.71^*$	$64.42 \pm 0.66^*$	48.24 ± 0.75	66.43 ± 0.63
IMP (Allen et al., 2019)	$59.50 \pm 0.93^*$	$79.50 \pm 0.65^*$	$48.29 \pm 0.84^*$	$68.00 \pm 0.67^*$	49.60 ± 0.80	68.10 ± 0.80
MAML (Finn et al., 2017)	56.10 ± 1.01	75.41 ± 0.74	43.35 ± 0.85	60.55 ± 0.77	48.70 ± 1.84	63.11 ± 0.92
Ours	$\mathbf{63.46 \pm 0.98}$	$\mathbf{80.94 \pm 0.62}$	$\mathbf{54.45 \pm 0.94}$	$\mathbf{72.61 \pm 0.64}$	$\mathbf{53.28 \pm 0.91}$	$\mathbf{70.44 \pm 0.72}$

slightly lower than that of *BMAML* (Yoon et al., 2018). However, it still falls in the 95% confidence interval of the performance of *BMAML*. Experimental results also show that the performance of our approach slightly degrades on *Omniglot* but it still achieves comparable results if being compared to recent probabilistic meta-learning methods.

Comparisons to Metric-based Methods Since Chen et al. (2019b) provide fair comparisons of recent metric-based methods, the proposed method uses a neural network which consists of the same four convolution blocks as those in (Chen et al., 2019b). Our experiments are also follow the same settings as those in Chen et al. (2019b). To make fair comparisons, the results of *MatchingNet*, *ProtoNet* and *RelationNet* reported in Chen et al. (2019b) are presented here. As shown in Table 6.2, the proposed method achieves state-of-the-art performance on all three datasets.

6.4.3 Ablation Studies

In this section, we investigate and report the robustness and effectiveness of the proposed method.

On Robustness The proposed approach enables us to have a luxury of changing the number of ways C (i.e. the number of classes) or the number of shots K (i.e. the number of labeled images per class) without re-training the whole model. We study the robustness of the proposed method in these scenarios on *Omniglot* and *mini-ImageNet*.

- *Omniglot*. The proposed model is trained for 5-way 5-shot or 10-way 5-shot on the meta-training set of *Omniglot*. Then, during meta-testing stage for evaluation the models' performance, the number of ways C (Figure 6.1-a) or the number of shots K (Figure 6.1-b) are varied. From Figure 6.1-a, it is found that our approach still has a high mean accuracy above 96% even when $C = 50$.
- *mini-ImageNet*. The proposed model is trained for 5-way 5-shot on the meta-training set of *mini-ImageNet*, and is then tuned on the meta-validation set. Then, we test its performance for a higher C-way classification problem during meta-testing, which is a more challenging problem. In Table 6.3, it shows that the proposed approach outperforms other metric-based methods. For example, it preserves a high accuracy above 56% for 10-way 5-shot tasks without the need of re-training.

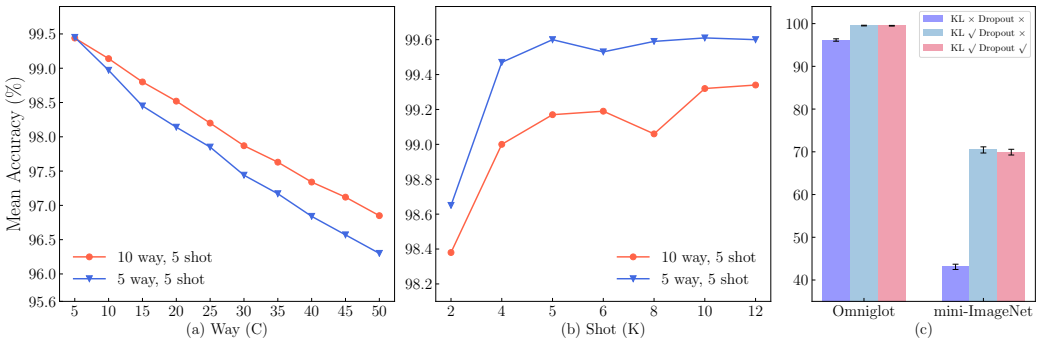

Figure 6.1: *Ablation Studies.* (a)&(b): Robustness of the proposed method on *Omniglot*. (c): Effectiveness of our probabilistic inference and extra dropout regularization. Details are presented in the supplementary material.

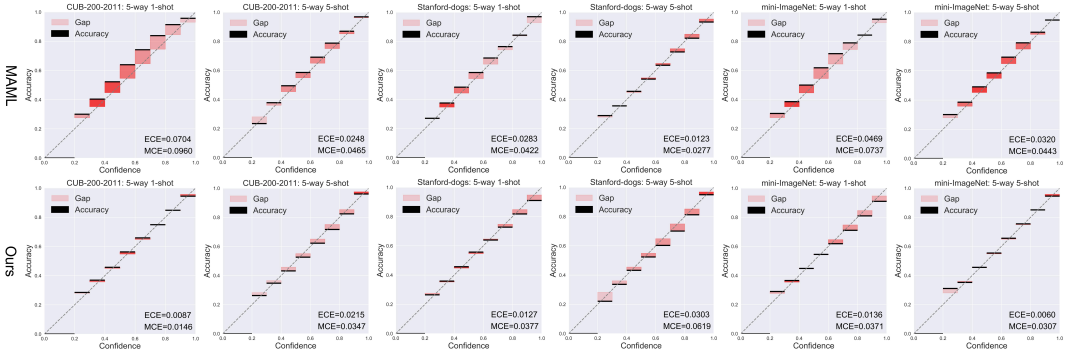

Figure 6.2: *Reliability Diagrams on Various Image Datasets.*

On Effectiveness of Inference We investigate the effectiveness of the proposed probabilistic inference, which is to learn the posteriors of class prototypes for each task. This is investigated by comparing with the removal of the $\mathbb{KL}$ term in (6.4). Additionally, since we do not use *dropout* (Kingma et al., 2015; Gal and Ghahramani, 2016) as an extra regularization (Gordon et al., 2019), the effectiveness of *dropout* is also investigated here. We present the results of 5-way 5-shot classification on *Omniglot* and *mini-ImageNet* in Figure 6.1-c. It shows that the proposed probabilistic inference plays a vital role in performance as it boosts performance, e.g., $43.08 \pm 0.62\%$ versus $70.44 \pm 0.72\%$ on *mini-ImageNet*. Experimental results also indicate that *dropout* is not effective to the proposed approach.

Table 6.3: *Ablation Study: Robustness on Mini-ImageNet.* All methods are trained for 5-way 5-shot at meta-training stage and are tested for C-way 5-shot at meta-testing stage.

C-way	$C = 5$ (%)	$C = 10$ (%)	$C = 20$ (%)
MatchingNet	63.48 ± 0.66	47.61 ± 0.44	33.97 ± 0.24
ProtoNet	64.24 ± 0.68	48.77 ± 0.45	34.58 ± 0.23
RelationNet	66.60 ± 0.69	47.77 ± 0.43	33.72 ± 0.22
Baseline++	66.43 ± 0.63	52.26 ± 0.40	38.03 ± 0.24
Ours	$\mathbf{70.44 \pm 0.72}$	$\mathbf{56.21 \pm 0.47}$	$\mathbf{43.43 \pm 0.25}$

On Quality of Predictive Uncertainty We measure the quality of predictive uncertainty of our approach by the expected calibration error (ECE) and maximum calibration error (MCE) (Guo et al., 2017; Naeini et al., 2015). They are presented jointly with reliability diagrams in Figure 6.2. As discussed in (Guo et al., 2017; Naeini et al., 2015), a perfect calibration would have its predicting probabilities

equal to the true correctness likelihood, i.e., $\Pr[\hat{y} = y|\hat{p} = p] = p$, where $p \in [0, 1]$, where $\hat{y}$ and $\hat{p}$ are the prediction of the model and its corresponding prediction confidence/probability, respectively. This indicates that a well calibrated model should have its bars close to the diagonal of reliability diagrams. Meanwhile it should also have small values of MCE and ECE. As shown in Figure 6.2, the proposed approach is well calibrated among various datasets and tasks.

6.5 Conclusions

In this chapter, a simple yet effective probabilistic meta-learning approach, *amortized Bayesian prototype meta-learning*, is proposed. The proposed approach can be trained end-to-end without pre-training. It is able to learn posteriors of latent prototypes efficiently whenever doing meta-training or meta-testing. Inference is amortized via learning a set of globally shared initializations of parameters. By doing so, within a few steps of gradient descent, the algorithm can produce well-behaved approximate posteriors of class prototypes. Randomness is taken into account by the learned posteriors of latent class prototypes. The proposed approach does not need an extra amortization network, and meanwhile it achieves competitive performance on various real-world image datasets, e.g., *mini-ImageNet*, *CUB-200-2011* and *Stanford-dogs*. We also demonstrate its robustness and effectiveness through ablation studies.

It is possible to extend the approach. Firstly, more flexible ways of aggregating distributions can be taken into account. For instance, one possible extension is to utilize normalizing flows for more flexible and general distributions of latent class prototypes. Secondly, it is possible to further improve the efficiency of the proposed method. For instance, with careful design of the structure of neural networks, it allows us to obtain posterior distributions of class prototypes by a single forward pass of the model such as Garnelo et al. (2018) instead of the current bi-level optimisation scheme (Finn et al., 2017).

Chapter 7

Conclusions and Future Work

This last chapter consists of two sections. Section 7.1 presents concise conclusions and reviews the key contributions of this thesis. Section 7.2 highlights additional future research directions including federated-learning control variates and more scalable Meta-CVs.

7.1 Conclusions

In this thesis, three novel transfer learning approaches in Monte Carlo methods and supervised classification are proposed, which encompass a diverse range of topics and tools, including control variates, kernel methods, Stein's method and meta-learning. The following paragraphs present the conclusions and contributions of the three proposed novel methods.

Chapter 4 presents *vector-valued control variates* which multitask-learn control variates' estimators for multiple related integrals. From the theoretical aspect, this is achieved by deriving novel matrix-valued Stein kernels and learning vector-valued control variates in the RKHSs induced by these novel Stein kernels. Meanwhile, extensive experiments demonstrate the effectiveness of the proposed approach on a range of problems from multi-fidelity modelling to Bayesian inference for dynamic systems.

Chapter 5 addresses an important limitation of control variates. That is, existing control variates methods tend to perform badly when the number of samples per task is small. Inspired by meta-learning, we propose the *Meta-CVs* approach which

is capable of learning effective control variates even when the number of samples per task is small. From the theoretical aspect, we establish the general conditions when Meta-CVs can be successfully trained. Extensive experiments show that the proposed method works well on a range of problems from Bayesian inference of differential equation-based models to marginalisation in hierarchical Gaussian processes.

Chapter 6 introduces *amortized Bayesian prototype meta-learning* for few-shot image classification. Instead of performing Bayesian inference on neural networks' parameters, it introduces and performs Bayesian inference on latent class prototypes per task, which largely reduces the number of latent variables involved. Inference is amortised by learning a set of globally shared parameters, and therefore the proposed approach is capable of learning unseen new few-shot image classification tasks fast. Extensive experiments demonstrate that the proposed method is effective and performs well on a range of real-world image datasets.

7.2 Future Work

Potential future work of the three proposed methods has already been discussed in detail in the respective associated chapters. In this section, we will further introduce novel control variate methods that offer fresh and innovative approaches to address various challenges and provide an insightful discussion on the application of transfer learning control variates in machine learning.

Further Theoretical Guarantees on vv-CVs and Meta-CVs Though we have provided numerous theoretical analyses of vv-CVs and Meta-CVs, it is still desirable to understand the maximum variance reduction that vv-CVs and meta-CVs can provide from a theoretical perspective. This, however, requires sophisticated assumptions on the properties of data points and integrands, which is very challenging.

More Scalable Meta-CVs The proposed meta-CVs method provides us with significant advantages and improvements over existing control variates methods. However, the algorithm itself is not highly scalable due to the bi-level optimisation for

learning meta-CVs. It could be particularly valuable to design more scalable algorithms with better optimisation efficiency to further reduce the computational cost. One way to move forward is to use conditional neural processes (Garnelo et al., 2018) such that a singe forward propagation of data points can produce task-specific CVs immediately. This, however, needs careful design of the structure of neural networks and neural processes to ensure that they have zero-mean under target distributions.

Federated-learning Control Variates When a dataset is not allowed to be shared directly due to privacy, how can we construct control variates efficiently in a transfer learning way? One way to move forward is to use federated-learning (McMahan et al., 2017). We will call this novel kind of control variates *federated-learning control variates*. Unlike vv-CVs or meta-CVs in which the original datasets from all tasks are gathered together, the federated control variate can be optimised by combining the gradient of the loss of each individual control variate and each individual control variate can then be updated using the resulting gradient of the federated control variate. This is similar to meta-CVs but they are fundamentally different as no datasets are shared directly during training. This is an very important future direction of control variates and can help to improve data privacy.

Meta Probabilistic Numerical Integration Methods Probabilistic numerical methods, such as Bayesian quadrature (BQ), provide us with uncertainty quantification over estimates of integrals. Ott et al. (2023) propose *Bayesian Stein neworks* which are scalable for large datasets, unlike the high computational cost of other probabilistic numerical methods like BQ. However, this approach requires a large number of samples to ensure satisfactory performance. Therefore, the remaining challenge is: can we have a novel probabilistic numerical integration method which is scalable and works efficiently even with a small number of samples per task? One way to move forward is to combine this work of Ott et al. (2023) with meta-learning. Analogous to meta-CVs, we will call this novel probabilistic numerical integration approach as *meta Bayesian Stein networks*. It is expected to provide us with the benefits of uncertainty quantification over estimates and satisfactory accuracy even

when we only have access to a small number of samples per task.

Transfer-learning Control Variates for Related Learning Objectives Control variates have also been employed as variance reduction tools within machine learning algorithms. However, most of these algorithms only employ simple approaches to perform variance reduction on their learning objectives. For instance, the widely used "baseline" in reinforcement learning is actually the first-order polynomial Stein-based CVs. One possible way forward is to apply more sophisticated CVs like vv-CVs and meta-CVs onto related learning objectives of machine learning algorithms, e.g., evidence lower bound in meta approximate variational inference (Liu et al., 2019; Nguyen et al., 2020; Iakovleva et al., 2020). For instance, in terms of the proposed meta-CV approach, the additional cost of incorporating it for related learning objectives is simply one additional neural network with a predetermined Stein operator.

Bibliography

A. Alexopoulos, P. Dellaportas, and M. K. Titsias. Variance reduction for Metropolis–Hastings samplers. *Statistics and Computing*, 33(6), 2023.

A. Alfonsi, B. Lapeyre, and J. Lelong. How many inner simulations to compute conditional expectations with least-square monte carlo? *Methodology and Computing in Applied Probability*, 25(3):71, 2023.

K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pages 232–241, 2019.

M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

R. Amit and R. Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*, pages 205–214, 2018.

A. Anastasiou, A. Barp, F.-X. Briol, R. E. Ebner, B.and Gaunt, F. Ghaderinezhad, J. Gorham, A. Gretton, C. Ley, Q. Liu, et al. Stein's method meets computational statistics: a review of some recent developments. *Statistical Science*, 38(1):120–139, 2023.

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29, 2016.

A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. In *International Conference on Learning Representations*, 2019.

R. Assaraf and M. Caffarel. Zero-variance principle for Monte Carlo algorithms. *Physical Review Letters*, 83(23):4682, 1999.

A. Barp, F.-X. Briol, A. Duncan, M. Girolami, and L. Mackey. Minimum Stein discrepancy estimators. In *Advances in Neural Information Processing Systems 32*, pages 12964–12976. 2019.

A. Barp, C. J. Oates, E. Porcu, and M. Girolami. A Riemann–Stein kernel method. *Bernoulli*, 28(4):2181–2208, 2022.

S. Bartunov and D. Vetrov. Few-shot generative modelling with generative matching networks. In *International Conference on Artificial Intelligence and Statistics*, pages 670–678. PMLR, 2018.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

D. Belomestny, L. Iosipoi, E. Moulines, A. Naumov, and S. Samsonov. Variance reduction for Markov chains with application to MCMC. *Statistics and Computing*, 30:973–997, 2020.

D. Belomestny, L. Iosipoi, E. Moulines, A. Naumov, and S. Samsonov. Variance reduction for dependent sequences with applications to stochastic gradient MCMC. *SIAM-ASA Journal on Uncertainty Quantification*, 9(1):507–535, 2021.

A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

B. M. Bolker. Ecological models and data in R. In *Ecological Models and Data in R*. Princeton University Press, 2008.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. ISSN 00361445. doi: 10.1137/16M1080173.

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

F. Brauer. Mathematical epidemiology: Past, present, and future. *Infect. Dis. Model.*, 2(2):113–127, 2017.

F.-X. Briol, C. Oates, M. Girolami, and M. A. Osborne. Frank-Wolfe Bayesian quadrature: Probabilistic integration with theoretical guarantees. In *Advances in Neural Information Processing Systems*, pages 1162–1170, 2015.

B. Calderhead and M. Girolami. Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics and Data Analysis*, 53 (12):4028–4045, 2009.

C. Carmeli, E. De Vito, and A. Toigo. Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 4 (4):377–408, 2006.

C. Carmeli, E. De Vito, A. Toigo, and V. Umanita. Vector valued reproducing kernel Hilbert spaces and universality. *Analysis and Applications*, 8(1):19–61, 2010.

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *J. Stat. Softw.*, 76(1), 2017.

L. H. Chen, L. Goldstein, and Q.-M. Shao. *Normal approximation by Stein's method*. Springer Science & Business Media, 2010.

Q. Chen, C. Shui, and M. Marchand. Generalization bounds for meta-learning: An information-theoretic analysis. *Advances in Neural Information Processing Systems*, 34:25878–25890, 2021.

W. Y. Chen, L. Mackey, J. Gorham, F.-X. Briol, and C. Oates. Stein points. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2018.

W. Y. Chen, A. Barp, F.-X. Briol, J. Gorham, M. Girolami, L. Mackey, and C. J. Oates. Stein point Markov chain Monte Carlo. In *International Conference on Machine Learning, PMLR 97*, pages 1011–1021, 2019a.

W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019b. URL `https://openreview.net/forum?id=HkxLXnAcFQ`.

C. Chu, K. Minami, and K. Fukumizu. The equivalence between stein variational gradient descent and black-box variational inference. *arXiv preprint arXiv:2004.01822*, 2020.

K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *International Conference on Machine Learning*, pages 2606–2615. PMLR, 2016.

C. Ciliberto, Y. Mroueh, T. Poggio, and L. Rosasco. Convex learning of multiple tasks and their structure. In *International Conference on Machine Learning*, pages 1548–1557, 2015.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, pages 160–167, 2008.

P. Dellaportas and I. Kontoyiannis. Control variates for estimation based on reversible Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 74(1):133–161, 2012.

J. Demange-Chryst, F. Bachoc, and J. Morio. Efficient estimation of multiple expectations with the same sample by adaptive importance sampling and control variates. *arXiv:2212.00568*, 2022.

G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. Learning to learn around a

common mean. In *Advances in Neural Information Processing Systems*, pages 10169–10179, 2018.

G. Denevi, C. Ciliberto, R. Grazzi, and M. Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, pages 1566–1575, 2019.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

F. Dinuzzo, C. S. Ong, P. V. Gehler, and G. Pillonetto. Learning output kernels with block coordinate descent. In *International Conference on Machine Learning*, 2011.

T. J. Dodwell, C. Ketelsen, R. Scheichl, and A. L. Teckentrup. Multilevel Markov chain Monte Carlo. *SIAM Review*, 61(3):509–545, 2019.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

A. Doucet, N. De Freitas, and N. J. Gordon. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

D. Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

B. Efron and C. Morris. Stein's paradox in statistics. *Scientific American*, 236(5): 119–127, 1977.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

H. R. Fairbanks, A. Doostan, C. Ketelsen, and G. Iaccarino. A low-rank control variate for multilevel Monte Carlo simulation of high-dimensional uncertain systems. *Journal of Computational Physics*, 341:121–139, 2017.

A. Fallah, A. Mokhtari, and A. Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

S. Feldman, M. R. Gupta, and B. A. Frigyik. Revisiting stein's paradox: multi-task averaging. *Journal of Machine Learning Research*, 15(1):3441–3482, 2014.

Y. Feng, D. Wang, and Q. Liu. Learning to draw samples with amortized stein variational gradient descent. *arXiv preprint arXiv:1707.06626*, 2017.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. In *International Conference on Machine Learning*. PMLR, 2019.

M. Fisher, T. Nolan, M. Graham, D. Prangle, and C. Oates. Measure transport with kernel stein discrepancy. In A. Banerjee and K. Fukumizu, editors, *International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1054–1062. PMLR, 13–15 Apr 2021. URL `http://proceedings.mlr.press/v130/fisher21a.html`.

N. Friel and A. N. Pettitt. Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70 (3):589–607, 2008.

N. Friel, M. Hurn, and J. Wyse. Improving power posterior estimation of statistical evidence. *Statistics and Computing*, 24(5):709–723, 2014a.

N. Friel, A. Mira, and C. J. Oates. Exploiting multi-core architectures for reduced-variance estimation with intractable likelihoods. *Bayesian Analysis*, 11(1):215–245, 2014b.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

G. D. Garson. *Hierarchical linear modeling: Guide and applications*. Sage, 2013.

G. Geraci, M. S. Eldred, and G. Iaccarino. A multifidelity multilevel Monte Carlo method for uncertainty propagation in aerospace applications. In *19th AIAA non-deterministic approaches conference*, page 1951, 2017.

A. Gessner, J. Gonzalez, and M. Mahsereci. Active multi-information source Bayesian quadrature. In *Uncertainty in Artificial Intelligence*, pages 712–721, 2020.

M. B. Giles. Multilevel Monte Carlo methods. *Acta numerica*, 24:259–328, 2015.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(2):123–214, 2011.

J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkxStoC5F7`.

J. Gorham and L. Mackey. Measuring sample quality with kernels. In *International Conference on Machine Learning*, pages 1292–1301. PMLR, 2017.

J. Gorham, A. B. Duncan, S. J. Vollmer, and L. Mackey. Measuring sample quality with diffusions. *Annals of Applied Probability*, 29(5):2884–2928, 2019.

E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJ_UL-k0b`.

W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pages 3732–3747. PMLR, 2020.

P. Green, K. Latuszyski, M. Pereyra, and C. Robert. Bayesian computation: a summary of the current state, and samples backwards and forwards. *Statistics and Computing*, 25:835–862, 2015.

E. Grefenstette, B. Amos, D. Yarats, P. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala. Generalized inner loop meta-learning. *arXiv:1910.01727*, 2019.

J. Griffin and P. Brown. Hierarchical shrinkage priors for regression models. *Bayesian Analysis*, 12(1):135, 2017.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

J. Han and Q. Liu. Stein variational gradient descent without gradient. In *International Conference on Machine Learning*, pages 1900–1908. PMLR, 2018.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

C. G. Hewitt. *The conservation of the wild life of Canada*. New York: C. Scribner, 1921.

F. J. Hickernell, C. Lemieux, and A. B. Owen. Control variates for quasi-Monte Carlo. *Statistical Science*, 20(1):1–31, 2005.

T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

E. Iakovleva, J. Verbeek, and K. Alahari. Meta-learning with shared amortized variational inference. In *International Conference on Machine Learning*, pages 4572–4582. PMLR, 2020.

M. A. Jamal and G.-J. Qi. Task agnostic meta-learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

G. Jerfel, E. Grant, T. Griffiths, and K. A. Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Advances in Neural Information Processing Systems*, pages 9119–9130, 2019.

K. Ji, J. Yang, and Y. Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *Journal of Machine Learning Research*, 23:29–1, 2022.

G. L. Jones. On the Markov chain central limit theorem. *Probability Surveys*, 1: 299–320, 2004.

H. Ju, D. Li, and H. R. Zhang. Robust fine-tuning of deep neural networks with hessian-based generalization guarantees. In *International Conference on Machine Learning*, pages 10431–10461. PMLR, 2022.

M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

K. Kandasamy, G. Dasarathy, J. Oliva, J. Schneider, and B. Poczos. Gaussian process optimisation with multi-fidelity evaluations. In *Neural Information Processing Systems*, pages 992–1000, 2016.

A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

S. Krumscheid and F. Nobile. Multilevel monte carlo approximation of functions. *SIAM-ASA Journal on Uncertainty Quantification*, 6(3):1256–1293, 2018.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2017.

B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

V. Lalchand and C. E. Rasmussen. Approximate inference for fully bayesian gaussian process regression. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12. PMLR, 2020.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

R. Leluc, F. Portier, and J. Segers. Control variate selection for monte carlo integration. *Statistics and Computing*, 31(4):1–27, 2021.

H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B.-G. Hu. LGM-Net: Learning to generate matching networks for few-shot learning. In *International Conference on Machine Learning*, pages 3825–3834, 2019.

K. Li and Z. Sun. Multilevel Control Functional. *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

K. Li, D. Giles, T. Karvonen, S. Guillas, and F.-X. Briol. Multilevel Bayesian Quadrature. In *International Conference on Artificial Intelligence and Statistics*, pages 1845–1868. PMLR, 2023.

X. Li*, Z. Sun*, J.-H. Xue, and Z. Ma. A concise review of recent few-shot meta-learning methods. *Neurocomputing*, 2020a.

X. Li*, J. Wu*, Z. Sun*, Z. Ma, J. Cao, and J.-H. Xue. Bsnet: Bi-similarity network for few-shot fine-grained image classification. *IEEE Transactions on Image Processing*, 2020b.

H. Liu, Y. Feng, Y. Mao, D. Zhou, J. Peng, and Q. Liu. Action-dependent control variates for policy optimization via stein's identity. In *International Conference on Learning Representations*, 2018.

H. Liu, R. Socher, and C. Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2019.

Q. Liu. Stein variational gradient descent as gradient flow. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, pages 2378–2386, 2016.

Q. Liu, J. Lee, and M. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pages 276–284. PMLR, 2016.

A. J. Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

A. J. Lotka. Fluctuations in the abundance of a species considered mathematically. *Nature*, 119(2983):12–12, 1927.

H. Marienwald, J.-B. Fermanian, and G. Blanchard. High-dimensional multi-task averaging and application to kernel mean embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 1963–1971. PMLR, 2021.

J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412, 2018.

T. Matsubara, J. Knoblauch, F.-X. Briol, C. Oates, et al. Robust generalised Bayesian inference for intractable likelihoods. *arXiv preprint arXiv:2104.07359*, 2021.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

C. Micchelli and M. Pontil. Kernels for multi–task learning. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

M. Micheli and J. A. Glaunes. Matrix-valued kernels for shape deformation analysis. *Geometry, Imaging and Computing*, 1(1):57–139, 2014.

H. Q. Minh, L. Bazzani, and V. Murino. A unifying framework in vector-valued reproducing kernel hilbert spaces for manifold regularization and co-regularized multi-view learning. *Journal of Machine Learning Research*, 2016.

A. Mira, R. Solgi, and D. Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662, 2013.

M. P. Naeini, G. F. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *AAAI Conference on Artificial Intelligence*, volume 2015, page 2901, 2015.

C. Nguyen, T.-T. Do, and G. Carneiro. Uncertainty in model-agnostic meta-learning using variational inference. In *IEEE Winter Conference on Applications of Computer Vision*, pages 3090–3100, 2020.

F. Nobile and F. Tesei. A multilevel Monte Carlo method with control variate for elliptic PDEs with log-normal coefficients. *Stochastic Partial Differential Equations: Analysis and Computations*, 3:398–444, 2015.

C. J. Oates and M. Girolami. Control functionals for quasi-Monte Carlo integration. In *International Conference on Artificial Intelligence and Statistics*, volume 51, pages 56–65, 2016.

C. J. Oates, T. Papamarkou, and M. Girolami. The controlled thermodynamic integral for Bayesian model comparison. *Journal of the American Statistical Association*, 111(514):634–645, 2016.

C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718, 2017.

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

K. Ott, M. Tiemann, P. Hennig, and F.-X. Briol. Bayesian numerical integration with neural networks. *arXiv preprint arXiv:2305.13248*, 2023.

A. B. Owen. Monte carlo theory, methods and examples. 2013.

J. Paisley, D. M. Blei, and M. I. Jordan. Variational bayesian inference with stochastic search. In *International Conference on Machine Learning*, 2012.

T. Papamarkou, A. Mira, and M. Girolami. Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Analysis*, 9(1):97–128, 2014.

C. Park, R. T. Haftka, and N. H. Kim. Remarks on multi-fidelity surrogates. *Structural and Multidisciplinary Optimization*, 55(3):1029–1050, 2017.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. Storkey. Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3): 550–591, 2018.

Y. Pu, Z. Gan, R. Henao, C. Li, S. Han, and L. Carin. VAE learning via Stein variational gradient descent. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

S. Ravi and A. Beatson. Amortized Bayesian meta-learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rkgpy3C5tX`.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2016.

M. Ren, R. Liao, E. Fetaya, and R. Zemel. Incremental few-shot learning with attention attractor networks. In *Advances in Neural Information Processing Systems*, pages 5276–5286, 2019.

M. Riabiz, W. Chen, J. Cockayne, and P. Swietach. Optimal thinning of MCMC output. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2022.

C. Riou, P. Alquier, and B.-E. Chérief-Abdellatif. Bayes meets bernstein at the meta level: an analysis of fast rates in meta-learning with pac-bayes. *arXiv preprint arXiv:2302.11709*, 2023.

C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJgklhAcK7`.

S. Si, C. J. Oates, A. B. Duncan, L. Carin, and F.-X. Briol. Scalable control variates for Monte Carlo methods via stochastic optimization. *Proceedings of the 14th Conference on Monte Carlo and Quasi-Monte Carlo Methods. arXiv:2006.07487*, 2021.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

R. Singhal, X. Han, S. Lahlou, and R. Ranganath. Kernelized complete conditional Stein discrepancy. *arXiv preprint arXiv:1904.04478*, 2019.

J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

L. South, C. Oates, A. Mira, and C. Drovandi. Regularized zero-variance control variates. *Bayesian Analysis*, 1(1):1–24, 2022a.

L. F. South, T. Karvonen, C. Nemeth, M. Girolami, and C. J. Oates. Semi-exact control functionals from sard's method. *Biometrika*, 109(2):351–367, 2022b.

L. F. South, M. Riabiz, O. Teymur, and C. J. Oates. Post-Processing of MCMC. *Annual Review of Statistics and Its Application*, 2022c. doi: 10.1146/annurev-statistics-040220-091727.

T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

Z. Sun, J. Wu, X. Li, W. Yang, and J.-H. Xue. Amortized Bayesian Prototype Meta-learning: A New Probabilistic Meta-learning Approach to Few-shot Image Classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, 2021.

Z. Sun, A. Barp, and F.-X. Briol. Vector-Valued Control Variates. In *International Conference on Machine Learning (ICML)*, 2023a.

Z. Sun, C. J. Oates, and F.-X. Briol. Meta-learning Control Variates: Variance Reduction with Limited Data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2023b.

F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

V. Volterra. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Società anonima tipografica" Leonardo da Vinci", 1926.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

R. Wan, M. Zhong, H. Xiong, and Z. Zhu. Neural control variates for variance reduction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, page 533–547, 2019.

C. Wang, X. Chen, A. J. Smola, and E. P. Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

D. Wang, Z. Tang, C. Bajaj, and Q. Liu. Stein variational gradient descent with matrix-valued kernels. In *Advances in Neural Information Processing Systems*, pages 7836–7846, 2019a.

Y. Wang, Q. Yao, J. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *arXiv: 1904.05046*, 2019b.

X. Xi, F.-X. Briol, and M. Girolami. Bayesian quadrature for multiple related integrals. In *International Conference on Machine Learning*, pages 5373–5382, 2018.

S. Xiong, P. Z. Qian, and C. J. Wu. Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55(1):37–46, 2013.

J. Xu, J.-F. Ton, H. Kim, A. R. Kosiorek, and Y. W. Teh. Metafun: Meta-learning with iterative functional updates. In *International Conference on Machine Learning*, pages 10617–10627, 2020.

Q. Yang, Y. Zhang, W. Dai, and S. J. Pan. *Transfer learning*. Cambridge University Press, 2020.

J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.

J. Zhuo, C. Liu, J. Shi, J. Zhu, N. Chen, and B. Zhang. Message passing stein variational gradient descent. In *International Conference on Machine Learning*, pages 6018–6027. PMLR, 2018.

B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le. Rethinking pre-training and self-training. In *Advances in neural information processing systems*, volume 33, pages 3833–3845, 2020.

Appendix A

Supplementary Material of Vector-valued Control Variates

A.1 Overview

In this Appendix, we present proofs for Chapter 4 in Appendix A.2, implementation details in Appendix A.3 and additional details and results for experimental study in Appendix A.4 All experiments are implemented with PyTorch.

A.2 Proofs

A.2.1 First-order K_0 for Polynomial-based vv-CVs

Consider the case when we are interested in using a polynomial kernel $k_l(x, y) = (\langle x, y\rangle + c)^l$, together with a separable kernel $K(x, y) = k(x, y)B$ and we decide to use the framework of first-order K_0. Note that $x, y \in \mathbb{R}^d$.

Degree 1 polynomials When the degree $l = 1$, to derive the closed form expression of first-order K_0, firstly note that,

$$
\begin{aligned}
&k(x, y) = \langle x, y\rangle + c = \textstyle\sum_{j=1}^{d} x_j y_j + c \\
&\Rightarrow \partial_y^r k(x, y) = x_r \\
&\Rightarrow \partial_x^r k(x, y) = y_r \\
&\Rightarrow \partial_x^r \partial_y^r k(x, y) = 1
\end{aligned}
$$

Thus, under such cases,

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r=1}^{d} \{1 + l_{t'}^r(y)y_r + l_t^r(x)x_r + l_t^r(x)l_{t'}^r(y)\left(\langle x,y\rangle + c\right)\}$$

Degree 2 polynomials When the degree $l = 2$, to derive the closed form expression of first-order K_0, firstly note that,

$$\begin{aligned}
&k(x,y) = \left(\sum_{j=1}^{d} x_j y_j + c\right)^2 \\
&\Rightarrow \partial_x^r k(x,y) = 2y_r \left(\sum_{j=1}^{d} x_j y_j + c\right) \\
&\Rightarrow \partial_y^r k(x,y) = 2x_r \left(\sum_{j=1}^{d} x_j y_j + c\right) \\
&\Rightarrow \partial_x^r \partial_y^r k(x,y) = 2x_r y_r + 2\left(\langle x,y\rangle + c\right)
\end{aligned}$$

Thus, under such cases,

$$\begin{aligned}
(K_0(x,y))_{tt'} = B_{tt'} \sum_{j=1}^{d} \Big\{ & 2x_r y_r + 2\left(\langle x,y\rangle + c\right) + 2y_r l_{t'}^r(y)\left(\langle x,y\rangle + c\right) + 2x_r l_t^r(x)\left(\langle x,y\rangle + c\right) \\
& + l_t^r(x) l_{t'}^r \left(\langle x,y\rangle + c\right)^2 \Big\}
\end{aligned}$$

A.2.2 Connection between vvpolynomials and vvRKHS with polynomial kernel

In this section, d is the dimension of x; T is the number of tasks; l is the degree of polynomial.

A vvRKHS $\mathcal{H}_K$ specified by a matrix-valued kernel $K(x, x_i) = \tilde{B} k_{\text{poly},l}(x, x_i) = \tilde{B}(\langle x, x_i\rangle + c)^l$, for any function $f(x) \in \mathcal{H}_K$,

$$\begin{aligned}
f(x) &= \sum_{i=1}^{m} \tilde{B}(\langle x, x_i\rangle + c)^l \tilde{\theta}_i \\
&= \sum_{i=1}^{m} \left(\sum_{j=1}^{d} x_j x_{ij} + c\right)^l \tilde{B}\tilde{\theta}_i
\end{aligned}$$

$$
\begin{aligned}
(f(x))_t &= \sum_{i=1}^{m} \left\{ \sum_{k=0}^{l} \left[C_l^k \left(\sum_{j=1}^{d} x_j x_{ij} \right)^k \left(c^{l-k} \right) \right] \underbrace{\tilde{B}_{t,\cdot}\, \tilde{\theta}_i}_{r_{ti}} \right\} \\
&= \sum_{i=1}^{m} \sum_{k=0}^{l} r_{ti}\, C_l^k\, c^{l-k} \left(\sum_{j=1}^{d} x_j x_{ij} \right)^k \qquad \text{(A.1)}
\end{aligned}
$$

A multivariate polynomial (where $\theta_{t,\alpha} \in \mathbb{R}^1$ and the subscript denotes the dependence on the choice of $\alpha = (\alpha_1, \ldots, \alpha_d)$ and task index t),

$$
(u_\theta(x))_t = \sum_{|\alpha| \le l} \left\{ \theta_{t,\alpha} \prod_{j=1}^{d} x_j^{\alpha_j} \right\} \quad , \qquad \text{(A.2)}
$$

When $d = 1$,

$$
\begin{aligned}
\text{Eq.(A.1)} &= \underbrace{\sum_{k=0}^{l}}_{\text{serves as } \alpha \le l} \left(\sum_{i=1}^{m} r_{ti}\, C_l^k\, c^{l-k}\, x_i^k \right) x^k \\
\text{Eq.(A.2)} &= \sum_{\alpha \le l} \theta_{t,\alpha}\, x^\alpha = \sum_{k=0}^{l} \theta_{t,\alpha} x^k \\
\Rightarrow \theta_{t,\alpha} &= \sum_{i=1}^{m} r_{ti}\, C_l^k\, c^{l-k}\, x_i^k = \sum_{i=1}^{m} \left(\sum_{t'=1}^{T} \tilde{B}_{t,t'} \tilde{\theta}_{it'} \right) C_l^k c^{l-k} x_i^k
\end{aligned}
$$

When $d \ge 2$,

$$
\begin{aligned}
\text{Eq.(A.1)} &= \sum_{i=1}^{m} \sum_{k=0}^{l} r_{ti} \binom{l}{k} c^{l-k} \left(x_1 x_{i1} + \cdots + x_d x_{id} \right)^k \\
&= \sum_{i=1}^{m} \sum_{k=0}^{l} r_{ti} \binom{l}{k} c^{l-k} \sum_{\alpha_1 + \cdots \alpha_d = k} \binom{k}{\alpha_1, \ldots, \alpha_d} \left[\prod_{j=1}^{d} (x_j x_{ij})^{\alpha_j} \right] \\
&= \sum_{k=0}^{l} \sum_{\alpha_1 + \cdots \alpha_d = k} \sum_{i=1}^{m} r_{ti} \binom{l}{k} c^{l-k} \binom{k}{\alpha_1, \ldots, \alpha_d} \left[\prod_{j=1}^{d} (x_j x_{ij})^{\alpha_j} \right] \\
&= \underbrace{\sum_{k=0}^{l} \sum_{\alpha_1 + \cdots \alpha_d = k}}_{\text{serves as } |\alpha| \le l} \sum_{i=1}^{m} r_{ti} \binom{l}{k} c^{l-k} \binom{k}{\alpha_1, \ldots, \alpha_d} \left[\prod_{j=1}^{d} x_{ij}^{\alpha_j} \right] \prod_{j=1}^{d} x_j^{\alpha_j} \\
\text{Eq.(A.2)} &= \sum_{|\alpha| \le l} \left\{ \theta_{t,\alpha} \prod_{j=1}^{d} x_j^{\alpha_j} \right\} \\
\Rightarrow \theta_{t,\alpha} &= \sum_{i=1}^{m} r_{ti} \binom{l}{k} c^{l-k} \binom{k}{\alpha_1, \ldots, \alpha_d} \left[\prod_{j=1}^{d} x_{ij}^{\alpha_j} \right] \\
&= \sum_{i=1}^{m} \left(\sum_{t'=1}^{T} \tilde{B}_{t,t'} \tilde{\theta}_{it'} \right) \binom{l}{k} c^{l-k} \binom{k}{\alpha_1, \ldots, \alpha_d} \left[\prod_{j=1}^{d} x_{ij}^{\alpha_j} \right]
\end{aligned}
$$

, where $\binom{l}{k} = \frac{l!}{k!(l-k)!}$ and $\binom{k}{\alpha_1, \ldots, \alpha_d} = \frac{k!}{\alpha_1! \alpha_2! \cdots \alpha_d!}$ with $\sum_{j=1}^{d} \alpha_j = k$ (which is also required in one of those summation operators). The expression when $d \ge 2$ is more

general, and covers the expression of $d = 1$.

A.2.3 Proof of Theorem 4.2.1

We will show K_0 is a kernel and derive its matrix components by constructing an appropriate feature map. The first order Stein operator maps matrix-valued functions $u = (u_1, u_2 \ldots, u_T) : \mathbb{R}^d \to \mathbb{R}^{d\times T}$ to the vv-function $\mathcal{S}^{\text{vv}}[u] : \mathbb{R}^d \to \mathbb{R}^T$ given by

$$\mathcal{S}^{\text{vv}}[u] = \left(\mathcal{L}'_{\Pi_1}[u_1], \ldots, \mathcal{L}'_{\Pi_T}[u_T]\right)^\top$$
$$\text{where} \quad \mathcal{L}'_{\Pi_t}[u_t](x) = \nabla_x \cdot u_t(x) + \nabla_x \log \pi_t(x) \cdot u_t(x) \quad \forall t \in [T].$$

Since $K \in C^{1,1}(\mathcal{X} \times \mathcal{X})$, we can use (Steinwart and Christmann, 2008, Corollary 4.36) to conclude that $\mathcal{H}_K$ is a vector-valued RKHS of continuously differentiable functions from $\mathbb{R}^d$ to $\mathbb{R}^T$, hence the tensor product $\mathcal{H}_K^d$ consists of suitable functions $u \in \mathcal{H}_K^d$, with components $u^i = (u_1^i, \ldots, u_T^i) \in \mathcal{H}_K$ for $i \in [d]$. Now recall that (see for example Theorem 2.11 (Micheli and Glaunes, 2014)):

$$\langle \partial_x^j K(\cdot, x)e_t, u^i\rangle_{\mathcal{H}_K} = e_t \cdot \partial^j u^i(x) = \partial^j u_t^i(x) \equiv \tfrac{\partial u_t^i}{\partial x^j}(x) \quad \forall t \in [T]$$

where $e_t \in \mathbb{R}^T$ is a vector of zeros with value 1 in the t^{th} component. Then, writing $K_x^{e_t} \equiv K(\cdot, x)e_t$,

$$\begin{aligned}
\mathcal{S}^{\text{vv}}[u](x) &= \textstyle\sum_{t=1}^T \sum_{r=1}^d (\partial_x^r u_t^r(x) + \partial_x^r \log \pi_t \, u_t^r(x))e_t \\
&= \textstyle\sum_{t=1}^T \sum_{r=1}^d \langle \partial_x^r K_x^{e_t} + l_{tr}(x) K_x^{e_t}, u^r\rangle_{\mathcal{H}_K} e_t \\
&= \textstyle\sum_{t=1}^T \langle \partial_x^\bullet K_x^{e_t} + l_{t\bullet}(x) K_x^{e_t}, u\rangle_{\mathcal{H}_K^d} e_t,
\end{aligned}$$

where $l_{tr}(x) = \partial_x^r \log \pi_t(x)$, and $\partial_x^\bullet K_x^{e_t}$ and $l_{t\bullet}(x)$ denote respectively the tuples $(\partial_x^1 K_x^{e_t}, \ldots, \partial_x^d K_x^{e_t}) \in \mathcal{H}_K^d$ and $(l_{t1}(x), \ldots, l_{td}(x)) \in \mathbb{R}^d$.

We have thus obtained a feature map, i.e., a map $\gamma : \mathcal{X} \to \mathcal{B}(\mathcal{H}_K^d, \mathbb{R}^T)$, where $\mathcal{B}(\mathcal{H}_K^d, \mathbb{R}^T)$ denotes the space of bounded linear maps from $\mathcal{H}_K^d$ to $\mathbb{R}^T$, via the

relation

$$\gamma(x)[u] = \mathcal{S}^{\text{vv}}[u](x),$$

with adjoint $\gamma(x)^* = \sum_{t=1}^{T}(\partial_x^{\bullet} K_x^{e_t} + l_{t\bullet}(x)K_x^{e_t})e_t$. Recall the adjoint map $\gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathcal{H}_K^d)$ to $\gamma(y)$, is defined for any $a \in \mathbb{R}^T, u \in \mathcal{H}_K^d$ by the relation

$$\langle \gamma(y)^*[a], u \rangle_{\mathcal{H}_K^d} = \gamma(y)[u] \cdot a.$$

In particular, by Proposition 1 of Carmeli et al. (2010) we have that:

$$K_0(x,y) \equiv \gamma(x) \circ \gamma(y)^* \in \mathbb{R}^{T \times T}$$

will then be the kernel associated to the "feature operator" (that is, a surjective partial isometry whose image is $\mathcal{H}_{K_0}$) $\mathcal{S}^{\text{vv}} : \mathcal{H}_K^d \to \mathcal{H}_{K_0}$. Subbing in the expressions for the feature map and its adjoint derived above, and using the equalities

$$\left\langle \partial_x^s K(\cdot,x)e_t, \partial_y^r K(\cdot,y)e_{t'} \right\rangle_{\mathcal{H}_K} = e_t \cdot \partial_x^s \partial_y^r K(x,y)e_{t'} = (\partial_x^s \partial_y^r K(x,y))_{tt'} \qquad \forall t,t' \in [T]$$
$$\text{and} \qquad \left\langle \partial_x^{ss} K(\cdot,x)e_t, \partial_y^r K(\cdot,y)e_{t'} \right\rangle_{\mathcal{H}_K} = (\partial_x^{ss} \partial_y^r K(x,y))_{tt'} \qquad \forall t,t' \in [T],$$

which hold for any $C^{1,1}(\mathcal{X} \times \mathcal{X})$ mv-kernel (Micheli and Glaunes, 2014), we obtain the following expression for the components of K_0

$$\begin{aligned}(K_0(x,y))_{tt'} = \textstyle\sum_{r=1}^{d}(\partial_x^r \partial_y^r K(x,y))_{tt'} + l_{t'r}(y)(\partial_x^r K(x,y))_{tt'} \\ + l_{tr}(x)\partial_y^r (K(x,y))_{tt'} + l_{tr}(x)l_{t'r}(y)(K(x,y))_{tt'}.\end{aligned}$$

In particular for separable kernels (i.e. $K(x,y) = Bk(x,y)$) we have

$$(K_0(x,y))_{tt'} = B_{tt'} \textstyle\sum_{r=1}^{d} \partial_x^r \partial_y^r k(x,y) + l_{t'r}(y)\partial_x^r k(x,y) + l_{tr}(x)\partial_y^r k(x,y) + l_{tr}(x)l_{t'r}(y)k(x,y).$$

A.2.4 Proof of Theorem 4.2.3

Recall that if a scalar kernel k satisfies $\int_{\mathcal{X}} k(x,x)\mathrm{d}\mu(x) < \infty$, then its RKHS consists of square μ-integrable functions (for any finite measure μ) (Steinwart and Christmann, 2008, Theorem 4.26).

If $g \in \mathcal{H}_{K_0}$ then g_t belongs to the RKHS with scalar-valued kernel (this follows from (Carmeli et al., 2010, Prop. 1), using as feature operator the dot product with respect to e_t, where e_t is defined in appendix A.2.3)

$$\begin{aligned}(K_0(x,y))_{tt} = \textstyle\sum_{r=1}^{d}(\partial_x^r \partial_y^r K(x,y))_{tt} + l_{tr}(y)\partial_x^r(K(x,y))_{tt} \\ + l_{tr}(x)\partial_y^r(K(x,y))_{tt} + l_{tr}(x)l_{tr}(y)(K(x,y))_{tt} \qquad \forall t \in [T].\end{aligned}$$

In particular since K is bounded with bounded derivatives, and

$$\Pi_t\left[|l_{tr}|\right] + \Pi_t\left[|l_{tr}|^2\right] \leq \sqrt{\Pi_t\left[|l_{tr}|^2\right]} + \Pi_t\left[|l_{tr}|^2\right] \qquad \forall t \in [T], r \in [d]$$

then $\int_{\mathcal{X}}(K_0(x,x))_{tt}\mathrm{d}\Pi_t(x) < \infty$ if $\|\nabla_x \log \pi_t(x)\|_2$ is square integrable with respect to Π_t, and the result follows.

A.2.5 Proof of Theorem 4.2.4

Proof. We want to find

$$\begin{aligned}&\arg\min_{g\in\mathcal{H}_{K_0}} L_m^{\mathrm{vv}}(g,\beta) \\ \text{where} \qquad &L_m^{\mathrm{vv}}(g,\beta) := \textstyle\sum_{t=1}^{T}\frac{1}{m_t}\sum_{j=1}^{m_t}(f_t(x_{tj}) - g_t(x_{tj}) - \beta_t)^2 + \lambda\|g\|^2_{\mathcal{H}_{K_0}}.\end{aligned}$$

Note that the objective is the same as that in (4.8), with the only difference being that the first input is now a function as opposed to the parameter value parameterising this function. We will abuse notation by using the same mathematical expression for both objectives.

By Ciliberto et al. (2015, Section 2.1), any solution of the minimization problem has the form $\hat{g}(\cdot) \equiv \sum_{t'=1}^{T}\sum_{j'=1}^{m_{t'}} K_0(\cdot\,, x_{t'j'})\theta_{t'j'}$. Subbing this solution into

$L_m^{\text{vv}}(g, \beta)$ yields

$$\begin{aligned} L_m^{\text{vv}}(\hat{g}, \beta) &= \textstyle\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} (f_t(x_{tj}) - (\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'} - \beta_t)^2 \\ &\quad + \lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ &= \lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ &\quad + \textstyle\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} \Big(y_{tj}^2 + (\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'})^2 \\ &\qquad\qquad - 2 \textstyle\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} y_{tj} K_0(x_{tj}, x_{t'j'})_t \theta_{t'j'} \Big), \end{aligned}$$

where $y_{tj} \equiv f_t(x_{tj}) - \beta_t$. The problem thus becomes a minimization problem over the coefficients θ,

$$\begin{aligned} \arg\min_{\theta \in \mathbb{R}^{|\mathcal{D}|}} & \lambda \textstyle\sum_{t',t''=1}^T \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{t''j''}) \theta_{t''j''} \\ & - 2 \textstyle\sum_{t,t'=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{tj})_{\cdot t} y_{tj} + \sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} y_{tj}^2 \\ & + \textstyle\sum_{t,t',t''=1}^T \sum_{j=1}^{m_t} \sum_{j'=1}^{m_{t'}} \sum_{j''=1}^{m_{t''}} \theta_{t'j'}^T K_0(x_{t'j'}, x_{tj})_{\cdot t} \frac{1}{m_t} K_0(x_{tj}, x_{t''j''})_{t\cdot} \theta_{t''j''}. \end{aligned}$$

Since the quadratic terms are semi-positive definite, the resulting objective is a convex function of θ, thus, by differentiating it, we obtain that the solution θ is the solution to

$$\begin{aligned} \textstyle\sum_{t'=1}^T \sum_{j'=1}^{m_{t'}} \Big(\sum_{t=1}^T \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} K_0(x_{tj}, x_{t'j'})_{t\cdot} + \lambda K_0(x_{t''j''}, x_{t'j'}) \Big) \theta_{t'j'} \\ = \textstyle\sum_t^T \frac{1}{m_t} \sum_{j=1}^{m_t} K_0(x_{t''j''}, x_{tj})_{\cdot t} (f_t(x_{tj}) - \beta_t), \end{aligned}$$

$\forall t'' \in [T], j'' \in [m_T]$.

□

A.2.6 Proof of Theorem 4.2.2

Proof. We proceed as for the proof of Theorem 4.2.1 and shall derive a feature map for K_0. Recall that $g = \mathcal{S}^{\text{vv}}[u] = (\mathcal{L}''_{\Pi_1}[u_1], \ldots, \mathcal{L}''_{\Pi_T}[u_T])^\top$, where $\mathcal{L}''_{\Pi_i}$ is the second-order Stein operator, which maps scalar functions to scalar functions. Here u belongs to a RKHS of $\mathbb{R}^T$-valued functions with matrix kernel K. From the

differentiability assumption on K, we have $\mathcal{H}_K \subset C^2$, i.e., it is a space of twice continuously differentiable functions. Note that (here $\partial^{jj} = \partial^j \partial^j = \frac{\partial^2}{\partial x_j \partial x_j}$)

$$\langle \partial_x^{jj} K(\cdot, x) e_t, u \rangle_{\mathcal{H}_K} = \partial^{jj} u_t(x) \equiv \tfrac{\partial^2 u_t}{\partial x_j \partial x_j}(x) \qquad \forall t \in [T],$$

where e_t is the t^{th} standard basis vector of $\mathbb{R}^T$ as before. Thus

$$\begin{aligned}
\mathcal{L}''_{\Pi_t}[u_t](x) &= \Delta_x u_t(x) + \nabla_x \log \pi_t(x) \cdot \nabla_x u_t(x) \\
&= \textstyle\sum_{s=1}^d \partial^{ss} u_t(x) + \sum_{s=1}^d l_{ts}(x) \partial^s u_t(x) \\
&= \textstyle\sum_{s=1}^d \langle \partial_x^{ss} K(\cdot, x) e_t, u \rangle_{\mathcal{H}_K} + \sum_{s=1}^d \langle l_{ts}(x) \partial_x^s K(\cdot, x) e_t, u \rangle_{\mathcal{H}_K} \\
&= \textstyle\sum_{s=1}^d \langle \partial_x^{ss} K(\cdot, x) e_t + l_{ts}(x) \partial_x^s K(\cdot, x) e_t, u \rangle_{\mathcal{H}_K} \qquad \forall t \in [T].
\end{aligned}$$

Hence

$$\mathcal{S}^{\text{vv}}[u](x) = \begin{pmatrix} \left\langle \sum_{s=1}^d \partial_x^{ss} K(\cdot, x) e_1 + l_{1s}(x) \partial_x^s K(\cdot, x) e_1, u \right\rangle_{\mathcal{H}_K} \\ \vdots \\ \left\langle \sum_{s=1}^d \partial_x^{ss} K(\cdot, x) e_T + l_{Ts}(x) \partial_x^s K(\cdot, x) e_T, u \right\rangle_{\mathcal{H}_K} \end{pmatrix} \in \mathbb{R}^T.$$

Note that for each $x \in \mathcal{X}$, each component of the above is a bounded linear operator $\mathcal{H}_K \to \mathbb{R}$ (i.e., the map $u \mapsto (\mathcal{S}^{\text{vv}}(u)(x))_s \in \mathbb{R}$ to the s-component is a bounded linear operator), then we have obtained a a feature map, i.e., a map $\gamma : \mathcal{X} \to \mathcal{B}(\mathcal{H}_K, \mathbb{R}^T)$, where $\mathcal{B}(\mathcal{H}_K, \mathbb{R}^T)$ denotes the space of bounded linear maps from $\mathcal{H}_K$ to $\mathbb{R}^T$. Specifically

$$\gamma(x) \equiv \mathcal{S}^{\text{vv}}[\cdot](x) \in \mathcal{B}(\mathcal{H}_K, \mathbb{R}^T).$$

In particular, as before

$$K_0(x, y) \equiv \gamma(x) \circ \gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathbb{R}^T)$$

will thus be the kernel associated to the "feature operator" $\mathcal{S}^{\text{vv}} : \mathcal{H}_K \to \mathcal{H}_{K_0}$. Recall that $\gamma(y)^* \in \mathcal{B}(\mathbb{R}^T, \mathcal{H}_K)$ is the adjoint map to $\gamma(y)$, i.e., it satisfies for any

$a \in \mathbb{R}^T, u \in \mathcal{H}_K$:

$$\langle \gamma(y)^*[a], u \rangle_{\mathcal{H}_K} = \gamma(y)[u] \cdot a.$$

From this we obtain

$$\gamma(y)^* : a \mapsto \sum_{r=1}^{d} \sum_{t=1}^{T} a_t \left(\partial_y^{rr} K(\cdot, y) e_t + l_{tr}(y) \partial_y^r K(\cdot, y) e_t \right) \in \mathcal{H}_K.$$

From $K_0(x, y)a = \gamma(x) \circ \gamma(y)^*[a]$ for all $a \in \mathbb{R}^T$ and the above expressions we can finally calculate K_0. We have

$$K_0(x, y)a = \mathcal{S}^{\text{vv}}[\gamma(y)^* a](x) = \begin{pmatrix} \left\langle \sum_{s=1}^{d} \partial_x^{ss} K(\cdot, x) e_1 + l_{1s}(x) \partial_x^s K(\cdot, x) e_1, \gamma(y)^* a \right\rangle_{\mathcal{H}_K} \\ \vdots \\ \left\langle \sum_{s=1}^{d} \partial_x^{ss} K(\cdot, x) e_T + l_{Ts}(x) \partial_x^s K(\cdot, x) e_T, \gamma(y)^* a \right\rangle_{\mathcal{H}_K} \end{pmatrix}.$$

We obtain that $K_0(x, y)a$ is a vector with components:

$$(K_0(x, y)a)_t = \sum_{r,s=1}^{d} \sum_{t'=1}^{T} a_{t'} \Big((\partial_x^{ss} \partial_y^{rr} K(x, y))_{tt'} + l_{t'r}(y) (\partial_x^{ss} \partial_y^r K(x, y))_{tt'} + l_{ts}(x) (\partial_x^s \partial_y^{rr} K(x, y))_{tt'} + l_{ts}(x) l_{t'r}(y) (\partial_x^s \partial_y^r K(x, y))_{tt'} \Big),$$

$\forall t \in [T]$.

Thus the components of $K_0(x, y) \in \mathbb{R}^{T \times T}$ are

$$(K_0(x, y))_{tt'} = \sum_{r,s=1}^{d} (\partial_x^{ss} \partial_y^{rr} K(x, y))_{tt'} + l_{t'r}(y) (\partial_x^{ss} \partial_y^r K(x, y))_{tt'} + l_{ts}(x) (\partial_x^s \partial_y^{rr} K(x, y))_{tt'} + l_{ts}(x) l_{t'r}(y) (\partial_x^s \partial_y^r K(x, y))_{tt'},$$

$\forall t, t' \in [T]$.

□

Analogously to the mv-kernel in Theorem 4.2.1, there are several cases of practical interest. The first is when $K(x, y) = Bk(x, y)$ is a separable kernel, in

which case:

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r,s=1}^{d} \partial_x^{ss} \partial_y^{rr} k(x,y) + l_{t'r}(y) \partial_x^{ss} \partial_y^{r} k(x,y) \\ + l_{ts}(x) \partial_x^{s} \partial_y^{rr} k(x,y) + l_{ts}(x) l_{t'r}(y) \partial_x^{s} \partial_y^{r} k(x,y) \qquad \forall t, t' \in [T].$$

The second is when K is separable and $\Pi_1 = \ldots = \Pi_T$, in which case $l_r(x) := l_{1r}(x) = \ldots = l_{Tr}(x)\ \forall r \in [d]$ and:

$$(K_0(x,y))_{tt'} = B_{tt'} \sum_{r,s=1}^{d} \partial_y^{ss} \partial_x^{rr} k(x,y) + l_r(x)\ \partial_y^{ss} \partial_x^{r} k(x,y) \\ + l_s(y)\ \partial_y^{s} \partial_x^{rr} k(x,y) + l_s(y) l_r(x) \partial_y^{s} \partial_x^{r} k(x,y) \qquad \forall t, t' \in [T].$$

A.3 Implementation Details

In this appendix, we focus on implementation details which may be helpful for implementing the algorithms in the main text. Firstly, in Appendix A.3.1 we derive the derivatives of several common kernels; this is essential for the implementation of Stein reproducing kernels. Then, in Appendix A.3.2, we provide details on how to select hyperparameters.

A.3.1 Kernels and Their Derivatives

We now provide details of all the kernels used in the paper, as well as expressions for their derivatives.

Polynomial Kernel The polynomial kernel $k_l(x,y) = (x^\top y + c)^l$ with constant $c \in \mathbb{R}$ and power $l \in \mathbb{N}$ has derivatives given by

$$\begin{aligned} \nabla_x k_l(x,y) &= l(x^\top y + c)^{l-1} y, \quad \nabla_y k_l(x,y) = l(x^\top y + c)^{l-1} x, \\ \nabla_x \cdot \nabla_y k_l(x,y) &= \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j \partial y_j} k_l(x,y) = \sum_{j=1}^{d} \frac{\partial}{\partial x_j} \left[l(x^\top y + c)^{l-1} x_j \right] \\ &= \sum_{j=1}^{d} l(l-1)(x^\top y + c)^{l-2} y_j x_j + l(x^\top y + c)^{l-1} \\ &= l(l-1)(x^\top y + c)^{l-2} x^\top y + dl(x^\top y + c)^{l-1}. \end{aligned}$$

Squared-Exponential Kernel The squared-exponential kernel (sometimes called Gaussian kernel) $k(x,y) = \exp(-\frac{\|x-y\|_2^2}{2\lambda})$ with lengthscale $\lambda > 0$ has derivatives given by

$$\begin{aligned}\nabla_x k(x,y) &= -\tfrac{(x-y)}{\lambda}k(x,y), \qquad \nabla_y k(x,y) = \tfrac{(x-y)}{\lambda}k(x,y),\\ \nabla_x \cdot \nabla_y k(x,y) &= \textstyle\sum_{j=1}^d \frac{\partial^2}{\partial y_j \partial x_j}k(x,y) = \sum_{j=1}^d \frac{\partial}{\partial y_j}\left[-\frac{(x_j-y_j)}{\lambda}k(x,y)\right]\\ &= \textstyle\sum_{j=1}^d \left[\frac{1}{\lambda} - \frac{(x_j-y_j)^2}{\lambda^2}\right]k(x,y) = \left[\frac{d}{\lambda} - \frac{(x-y)^\top(x-y)}{\lambda^2}\right]k(x,y).\end{aligned}$$

Preconditioned Squared-Exponential Kernel Following Oates et al. (2017), we also considered a preconditioned squared-exponential kernel:

$$k(x,y) = \tfrac{1}{(1+\alpha\|x\|_2^2)(1+\alpha\|y\|_2^2)}\exp\left(-\tfrac{\|x-y\|_2^2}{2\lambda^2}\right).$$

with lengthscale $\lambda > 0$ and preconditioner parameter $\alpha > 0$. This kernel has derivatives given by:

$$\begin{aligned}\nabla_x k(x,y) &= \left[\tfrac{-2\alpha x}{1+\alpha\|x\|_2^2} - \tfrac{(x-y)}{\lambda^2}\right]k(x,y),\ \nabla_y k(x,y) = \left[\tfrac{-2\alpha y}{1+\alpha\|y\|_2^2} + \tfrac{(x-y)}{\lambda^2}\right]k(x,y),\\ \nabla_x \cdot \nabla_y k(x,y) &= \textstyle\sum_{j=1}^d \frac{\partial^2}{\partial x_j \partial y_j}k(x,y) = \sum_{j=1}^d \frac{\partial}{\partial y_j}\left[\left(\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2}\right)k(x,y)\right]\\ &= \textstyle\sum_{j=1}^d \left(\frac{1}{\lambda^2}k(x,y) + \left[\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2}\right]\frac{\partial}{\partial y_j}k(x,y)\right)\\ &= \textstyle\sum_{j=1}^d \left(\frac{1}{\lambda^2}k(x,y) + \left[\frac{-2\alpha x_j}{1+\alpha\|x\|_2^2} - \frac{(x_j-y_j)}{\lambda^2}\right]\left[\frac{-2\alpha y_j}{1+\alpha\|y\|_2^2} + \frac{(x_j-y_j)}{\lambda^2}\right]k(x,y)\right)\\ &= k(x,y)\left[\tfrac{4\alpha^2 x^\top y}{(1+\alpha\|x\|_2^2)(1+\alpha\|y\|_2^2)} + \tfrac{2\alpha(x-y)^\top y}{\lambda^2(1+\alpha\|y\|_2^2)} - \tfrac{2\alpha(x-y)^\top x}{\lambda^2(1+\alpha\|x\|_2^2)} + \tfrac{d}{\lambda^2} - \tfrac{(x-y)^\top(x-y)}{\lambda^4}\right].\end{aligned}$$

Product of Kernels Finally, some of our examples will also use products of well-known kernels. Consider the kernel $k(x,y) = \prod_{j=1}^d k_j(x_j,y_j)$. The derivatives of this kernel can be expressed in terms of the components of the product and their

derivates as follows:

$$\begin{aligned}
\nabla_x k(x,y) &= \left(\tfrac{\partial k_1(x_1,y_1)}{\partial x_1}\prod_{j\neq 1} k_j(x_j,y_j),\ldots,\tfrac{\partial k_d(x_d,y_d)}{\partial x_d}\prod_{j\neq d} k_j(x_j,y_j)\right)^\top \\
\nabla_y k(x,y) &= \left(\tfrac{\partial k_1(x_1,y_1)}{\partial y_1}\prod_{j\neq 1} k_j(x_j,y_j),\ldots,\tfrac{\partial k_d(x_d,y_d)}{\partial y_d}\prod_{j\neq d} k_j(x_j,y_j)\right)^\top \\
\nabla_y \cdot \nabla_x k(x,y) &= \sum_{j=1}^d \tfrac{\partial^2}{\partial x_j \partial y_j} k(x,y) = \sum_{j=1}^d \tfrac{\partial}{\partial y_j}\left(\tfrac{\partial k_j(x_j,y_j)}{\partial x_j}\prod_{i\neq j} k_i(x_i,y_i)\right) \\
&= \sum_{j=1}^d \left[\tfrac{\partial^2 k_j(x_j,y_j)}{\partial y_j \partial x_j}\prod_{i\neq j} k_i(x_i,y_i)\right].
\end{aligned}$$

A.3.2 Hyper-parameters Selection

Most kernels (whether scalar- or matrix-valued) will have hyperparameters which we will have to select. For example, the squared-exponential kernel will often have a lengthscale or amplitude parameter, and these will have a significant impact on the performance.

We propose to select kernel hyperparameters through a marginal likelihood objective by noticing the equivalence between the optimal vv-CV based on the objective in (4.8) and the posterior mean of a zero-mean Gaussian process model with covariance matrix $K_0(x,y)$; see Oates et al. (2017) for a discussion in the sv-CV case. Unfortunately, computing the marginal likelihood in the general case can be prohibitively expensive due to the need to take inverses of large kernel matrices; the exact issue we were attempting to avoid through the use of the stochastic optimisation approaches. For simplicity, we instead maximise the marginal likelihood corresponding to $B = I_T$:

$$\nu^* := \arg\max_\nu -\tfrac{1}{2}\sum_{t=1}^T \left(\sum_{j,j'=1}^{m_t} f_t(x_{tj})(K_{\Pi_t}(\nu)+\lambda I_{m_t})^{-1}_{jj'} f_t(x_{tj'}) + \log\det[K_{\Pi_t}(\nu)+\lambda I_{m_t}]\right).$$

where $K_{\Pi_t}(\nu)$ is a matrix with entries $K_{\Pi_t}(\nu)_{ij} = k_{\Pi_t}(x_{ti}, x_{tj}; \nu)$ where k_{Π_t} is a Stein reproducing kernel of the form in (3.5) specialised to Π_t which has hyperparameters given by some vector ν. This form is not optimal when $B \neq I_T$, but we found that it tend to perform well in our numerical experiments. The regularisation parameter λ can also be selected through the marginal likelihood. However, in practice we are in an interpolation setting and therefore choose λ as small as possible

whilst still being large enough to guarantee numerically stable computation of the matrix inverses above.

A.4 Additional Details and Results for the Experimental Study

This last Appendix provides additional details on our numerical experiments from Section 4.3.

A.4.1 Experimental Details of the Illustration Example

The experiment was replicated 100 times for all methods. The exact details of the implementation are as follows.

- CV
  - Sample size: 50.
  - Hyper-parameter tuning: batch size 5; learning rate 0.05; total number of epochs 30.
  - Base kernel: squared exponential kernel
  - Optimisation: $\lambda = 0.001$; batch size is 5; learning rate is 0.001; total number of epochs 400.
- vv-CV (estimated B)
  - Sample size: $(50, 50)$ from (Π_1, Π_2) for (f_1, f_2).
  - Hyper-parameter tuning: batch size 5 (10 in total for (f_1, f_2)); learning rate 0.05; total number of epochs 30.
  - Base kernel: squared exponential kernel
  - Optimisation: $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 0.001$; batch size is 5 (10 in total for (f_1, f_2)); learning rate is 0.001; total number of epochs 400.

A.4.2 Experimental Details of the Multifidelity Univariate Step Functions

The experiment was replicated 100 times for all methods. Details of their implementation is given below:

Squared-exponential kernel

- CV
  - Sample size: 40.
  - Base kernel: squared exponential kernel.
  - Hyper-parameter tuning: batch size is 10; learning rate 0.02; total number of epochs 15.
  - Optimisation: $\lambda = 1 \times 10^{-5}$; batch size is 10; learning rate is 3×10^{-4}; total number of epochs 400.
- vvCV (estimated B/fixed B)
  - Sample size: $(40, 40)$ from $(\mathcal{N}(0,1), \mathcal{N}(0,1))$ for (f_L, f_H).
  - Hyper-parameter tuning: batch size is 5 (10 in total for (f_L, f_H)); learning rate 0.02; total number of epochs 15.
  - Base kernel: squared exponential kernel.
  - Optimisation: When B is fixed, we set $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$; otherwise, $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 1 \times 10^{-5}$; batch size is 5 (10 in total for (f_L, f_H)); learning rate is 3×10^{-4}; total number of epochs 400.

First-order polynomial kernel

- CV
  - Sample size: 40.
  - Base kernel: first order polynomial kernel.
  - Optimisation: $\lambda = 1 \times 10^{-5}$; batch size is 10; learning rate is 3×10^{-4}; total number of epochs 400.
- vv-CV (estimating B/fixed B)
  - Sample size: $(40, 40)$ from $(\mathcal{N}(0,1), \mathcal{N}(0,1))$ for (f_L, f_H).
  - Base kernel: first order polynomial kernel.
  - Optimisation: When B is fixed, we set $B_{11} = B_{22} = 0.5, B_{12} = B_{21} = 0.01$; otherwise, $B^{(0)}$ is initialized at the identity matrix I_2. $\lambda = 1 \times 10^{-5}$; batch size is 5 (10 in total for (f_L, f_H)); learning rate is 3×10^{-4}; total number of epochs 400.

Random variable	Distributions	Random variable	Distributions
r_w	$\mathcal{N}(0.1, 0.0161812^2)$	r	$\mathcal{N}(100, 0.01)$
T_u	$\mathcal{N}(89335, 20)$	T_l	$\mathcal{N}(89.55, 1)$
H_u	$\mathcal{N}(1050, 1)$	H_l	$\mathcal{N}(760, 1)$
L	$\mathcal{N}(1400, 10)$	K_w	$\mathcal{N}(10950, 30)$

Table A.1: *Prior Distributions for the inputs of the Borehole function.*

A.4.3 Experimental Details of the Multifidelity Modelling of Waterflow

In this section, we provide details on the Borehole example from the main text, and provide complementary experiments. The distributions with respect to which the integral is taken is an eight-dimensional Gaussian with independent marginals provided in Table A.1.

A.4.3.1 Experiment in the main text: Balanced vv-CVs

The number of replications is 100 for all methods. Details of their implementation is given below:

- Base kernel: Instead of using $k(x, x') = \exp(-\|x - x'\|_2^2/2\nu)$ with $l > 0$ which implicitly assumes that the length-scales are identical in all directions, we now allow that each dimension can have its own length-scale. That is,

$$k(x, x') := \prod_{j=1}^{d} k_j(x_j, x_j') \qquad \text{where} \qquad k_j(x_j, x_j') = \exp\left(-\frac{(x_j - x_j')_2^2}{2\nu_j}\right).$$

  Each of the components has its own length-scale $\nu_j > 0$ to be determined.
- Since $\pi(x) = \prod_{j=1}^{d} \pi_j(x_j)$, the score function is $\nabla_x \log \pi(x) = \left(\frac{\partial \log \pi_1(x)}{\partial x_1}, \ldots, \frac{\partial \log \pi_d(x)}{\partial x_d}\right)^\top$.
- Hyper-parameter tuning: batch size 5 (10 in total for (f_L, f_H)); learning rate of tuning 0.05; epochs of tuning 20.
- Optimisation (estimated B/pre-fixing B): When B is fixed, we set $B_{11} = B_{22} = 5 \times 10^{-4}, B_{12} = B_{21} = 5 \times 10^{-5}$; otherwise, $B^{(0)}$ is initialized at $1 \times 10^{-5} \times I_2$. $\lambda = 1 \times 10^{-5}$; batch size 5 (10 in total for (f_L, f_H)); learning rate for the cases when sample sizes are $(10, 20, 50, 100, 150)$ are

$(0.09, 0.06, 0.012, 0.0035, 0.002)$, respectively.

A.4.3.2 Additional Experiments: Unbalanced vv-CVs

In Figure A.1, we present the results of vv-CVs when the sample sizes are unbalanced; that is, we have a different number of samples for the low-fidelity and high-fidelity models. The exact setup is given below, and we replicated the experiment 100 times.

- Sample size: m_H is fixed to be 20, while $m_L \in \{20, 40, 60\}$.
- Base kernel: product of squared exponential kernels. $k(x, x') := \prod_{j=1}^{d} k_j(x_j, x'_j)$, where each $k_j(x_j, x'_j) = \exp(-(x_j - x'_j)_2^2/2\nu_j)$ has its own length-scale $\nu_j > 0$ to be determined.
- Hyperparameter tuning: batch size of tuning 5 (10 in total for (f_L, f_H)); learning rate of tuning is 0.05; epochs of tuning is 20.
- Optimisation (estimated B/pre-fixing B): When B is fixed, we set $B_{11} = B_{22} = 5 \times 10^{-4}, B_{12} = B_{21} = 5 \times 10^{-5}$; otherwise, $B^{(0)}$ is initialized at $1 \times 10^{-5} \times I_2$. $\lambda = 1 \times 10^{-5}$; learning rate is $(0.06, 0.04, 0.02)$ when $m_L \in \{20, 40, 60\}$, respectively.

Interestingly, we notice that not much is gained when increasing the number of samples for the low-fidelity model. In fact, in the case of a fixed B, the performance tends to decrease with a larger m_L. This is likely due to the "negative transfer" phenomenon which is well-known in machine learning. This phenomenon can occur when two tasks are not similar enough to provide any gains in accuracy. In this case, there is clearly no advantage in using a larger m_L since this increases computational cost and does not provide any gains in accuracy.

A.4.4 Experimental Details of the Computation of the Model Evidence through Thermodynamic Integration

To implement our vv-CVs, we need to derive the corresponding score functions. For a power posterior, the score function is of the form:

$$\nabla_\theta \log p(\theta|y, t) = t\nabla_\theta \log p(y|\theta) + \nabla_\theta \log p(\theta)$$

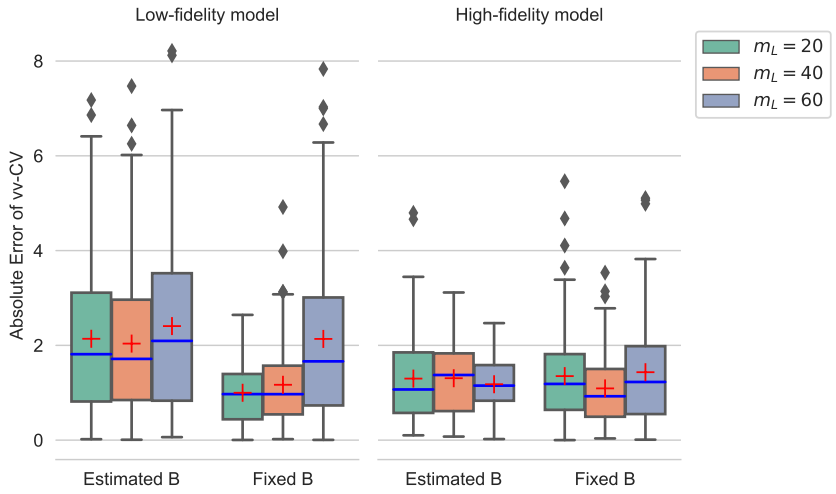

Figure A.1: Performance of vv-CVs with unbalanced sample sizes. Here we fix $m_H = 20$, and changing m_L to be $20, 40, 60$. Each experiment is repeated 100 times.

where $\nabla_\theta \log p(\theta)$ is the score function corresponding to the prior. In our case, the prior is a log-normal distribution $\log \theta \sim \mathcal{N}(\mu, \sigma^2)$ (where $\sigma = 0.25$), and its score function is given by:

$$\nabla_\theta \log p(\theta) = -\tfrac{1}{\theta} - \tfrac{\log \theta - \mu}{x\sigma^2}.$$

The score functions for all temperatures are plotted in Figure A.2; as observed, temperatures' consecutive score functions are very similar to one another.

In order to keep the computational cost manageable, we split the $T = 62$ integration problems into groups of closely related problems. In particular, we jointly estimate the means in terms of 4 consecutive temperatures on the ladder (group 1 is $\mu_1, \mu_2, \mu_3, \mu_4$, group 2 is $\mu_5, \mu_6, \mu_7, \mu_8$, etc...). Since 31 is not divisible by 4, our last group consists of three means $\mu_{29}, \mu_{30}, \mu_{31}$. Then, the same approach is taken to create groups of 4 (or 3 for the last group) variances.

The number of replications was 20 for each method. Details are given below:

- CV
  - Base kernel: Preconditioned squared-exponential kernel (Oates et al., 2017).
  - Hyperparameter tuning: we use the values $(0.1, 3)$ in (Oates et al.,

2017).

- Optimisation: $\lambda = 1 \times 10^{-3}$; batch size is 5; total number of epochs is 400.

- vv-CV(estimated B)
  - Base kernel: Preconditioned squared-exponential kernel (Oates et al., 2017).
  - Hyperparameter tuning: we use the values $(0.1, 3)$ in (Oates et al., 2017).
  - Optimisation: $\lambda = 1 \times 10^{-3}$; batch size is 5; learning rate is 0.01; number of epochs is 400.

A.4.5 Experimental Details of the Lotka-Volterra System

We implement $\log - \exp$ transform on model parameters and avoid constrained parameters on the ODE directly. Lotka—Volterra system can be re-parameterized as,

$$\frac{dv_1(s)}{ds} = \tilde{\alpha} v_1(s) - \tilde{\beta} v_1(s) v_2(s)$$
$$\frac{dv_2(s)}{ds} = \tilde{\delta} v_1(s) v_2(s) - \tilde{\gamma} v_2(s),$$

where

$$\tilde{\alpha} = \exp(\alpha), \tilde{\beta} = \exp(\beta),$$
$$\tilde{\delta} = \exp(\delta), \tilde{\gamma} = \exp(\gamma),$$

where v_1 and v_2 represents the number of preys and predators, respectively.

The model is,

$$y_{10} \sim \text{Log-Normal}(\log \tilde{v}_1(0), \tilde{\sigma}_{y_1})$$
$$y_{20} \sim \text{Log-Normal}(\log \tilde{v}_2(0), \tilde{\sigma}_{y_2})$$
$$y_{1s} \sim \text{Log-Normal}(\log v_1(s), \tilde{\sigma}_{y_1})$$
$$y_{2s} \sim \text{Log-Normal}(\log v_2(s), \tilde{\sigma}_{y_2})$$

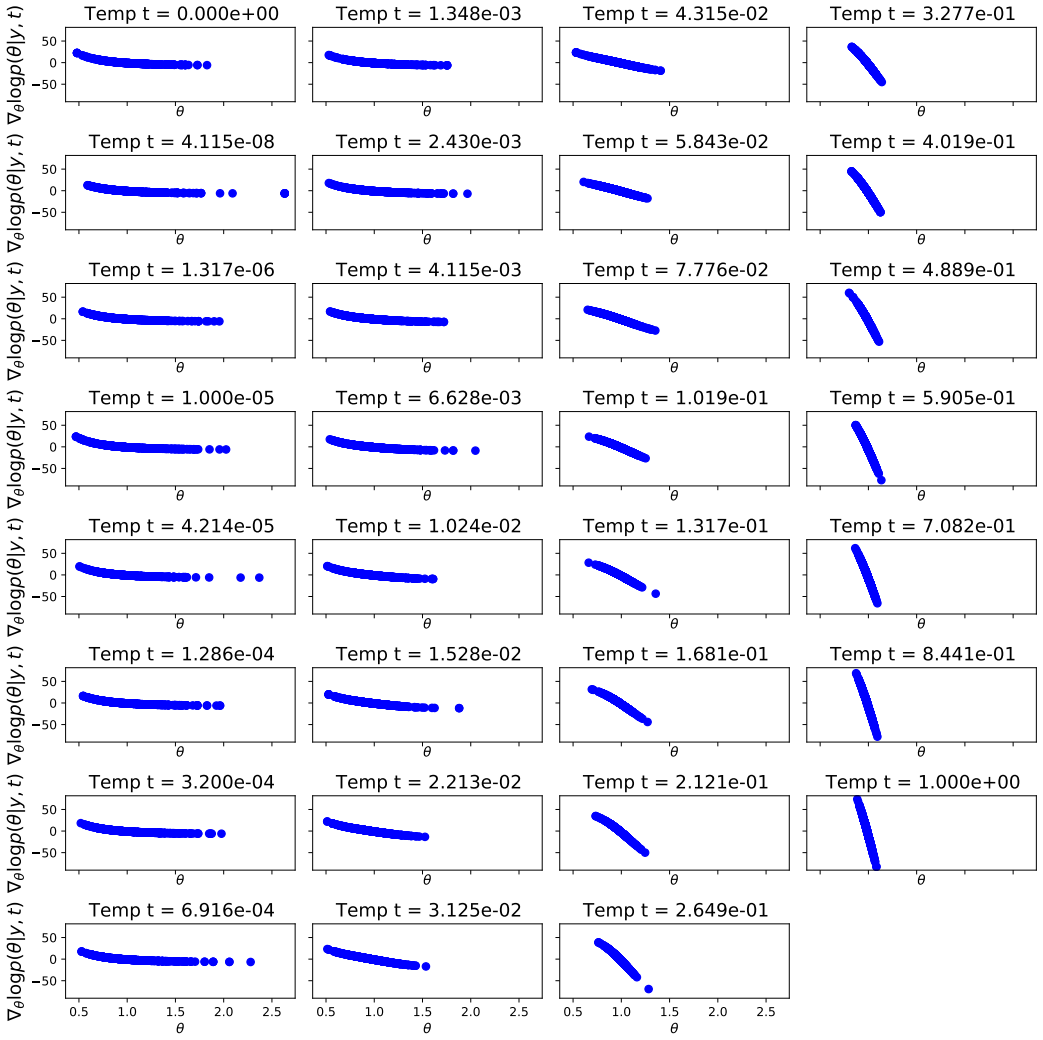

Figure A.2: Score functions corresponding to the power posteriors at different temperatures on the temperature ladder.

where

$$\tilde{v}_1(0) := \exp(v_1(0)), \tilde{v}_2(0) := v_2(0)$$
$$\tilde{\sigma}_{y_1} := \exp(\sigma_{y_1}), \tilde{\sigma}_{y_2} = \exp(\sigma_{y_2}).$$

By doing so, $x := (\alpha, \beta, \delta, \gamma, v_1(0), v_2(0), \sigma_x, \sigma_y)^\top$ can be defined on the whole $\mathbb{R}^8$ since the exponential transformation will make them be larger than zero. Therefore, x can be assigned priors on $\mathbb{R}^8$, e.g., Gaussian. As a result, the expectations associated with $\pi(x)$ are defined on $\mathbb{R}^8$ and Stan will return the scores of these parameters directly as these 8 parameters x themselves are unconstrained through

manual reparameterisation.

Priors are,

$$\alpha, \gamma \sim \text{Normal}(0, 0.5^2)$$
$$\beta, \delta \sim \text{Normal}(-3, 0.5^2)$$
$$\sigma_x, \sigma_y \sim \text{Normal}(-1, 1^2)$$
$$v_1(0), v_2(0) \sim \text{Normal}(\log 10, 1^2)$$

The fitting for predators y_{1s} and $v_1(s)$ at points $s_1, \ldots, s_m$ are shown in Figure A.3. The fitting for predators y_{2s} and $v_2(s)$ at points $s_1, \ldots, s_m$ are shown in Figure A.4.

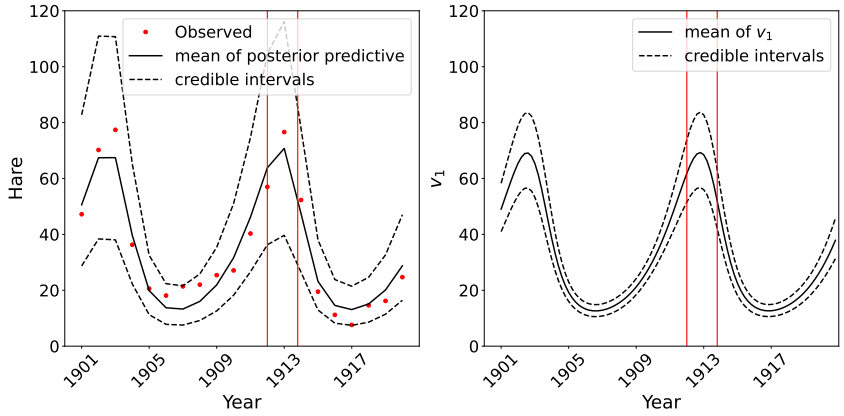

Figure A.3: *Bayesian inference of abundance of preys of Lotka-Volterra system.* Dots are observations; lines are the posterior means while dotted lines are the corresponding 95% credible intervals. Tasks are chosen in the area between the two vertical red lines.

The number of replications was 10 for each method and for each task. Details are given below:

- Tasks $\Pi[f_t]$ at time $s'_1, \ldots, s'_T$ (the base unit is 1 year):
  - $T = 2$: 1913., 1913.2;
  - $T = 5$: 1912., 1912.2, 1912.4, 1912.6, 1912.8;
  - $T = 10$: 1912., 1912.2, 1912.4, 1912.6, 1912.8, 1913., 1913.2, 1913.4, 1913.6, 1913.8.
- Base kernel: same one as in A.4.3: product of squared-exponential kernels.
- Hyperparameter tuning: batch size is 10; learning rate 0.01; total number of epochs 10.

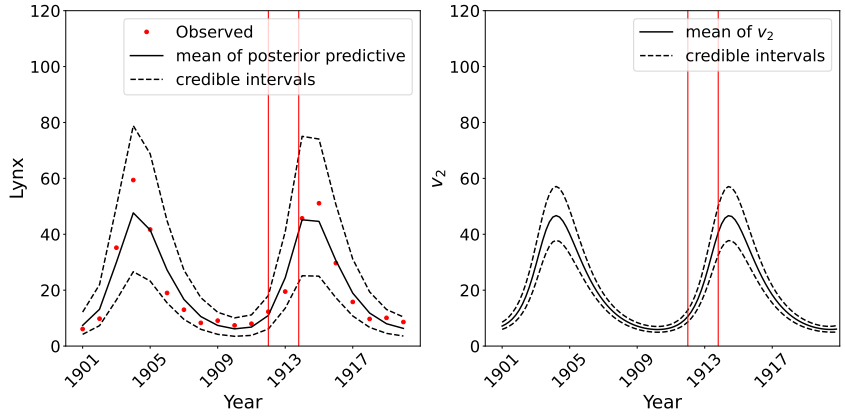

Figure A.4: *Bayesian inference of abundance of predators of Lotka-Volterra system.* Dots are observations; lines are the posterior means while dotted lines are the corresponding 95% credible intervals. Tasks are chosen in the area between the two vertical red lines.

Table A.2: *Lotka Example: Sum of mean absolute error of each task.* Number of replication is 10.

T	m	vv-CV- Estimated B	vv-CV-Fixed B	CF	MC
2	500	0.169	0.144	0.242	0.275
5	500	0.322	0.245	1.246	0.661
10	500	0.916	0.792	5.835	1.797

- Optimisation: $\lambda = 10^{-5}$; batch size 10; learning rate 10^{-3}; total number of epochs 400.

A.4.5.1 Addition Experiments for Bayesian inference of abundance of preys of Lotka-Volterra system

We present additional experiments in Table A.2 under the same settings as those in the above section. We consider to estimate $\Pi[f_t]$ at time $s'_1, \ldots, s'_T$ (the base unit is 1 year),

- $T = 2$: $1915., 1915.2$;
- $T = 5$: $1915., 1915.2, 1915.4, 1915.6, 1915.8$;
- $T = 10$: $1914., 1914.2, 1914.4, 1914.6, 1914.8, 1915., 1915.2, 1915.4, 1915.6, 1915.8$.

Appendix B

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

B.1 Overview

In Appendix B.2, we provide the proof of the theoretical results stated in the main text. In Appendix B.3, we provide more details on the implementation of Neural-CVs and Meta-CVs, together with the full experimental protocol.

B.2 Proof of Theorems

In this section, we will firstly review the assumptions and theorems in Ji et al. (2022) in Appendix B.2.1 as the proof of the theorems follows the results of (Ji et al., 2022). We then give the proof of Theorem 5.5.1 in Appendix B.2.2 and proof of Corollary 5.5.1.1 in Appendix B.2.3.

B.2.1 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 B.2.1 (Theorem 9 and Corollary 10 (Ji et al., 2022))**.** *Let the above assumptions (A1) to (A5) hold. Then, with a meta step-size $\eta = \frac{1}{80\chi_{\eta_e}}$ in Algorithm 6 , we attain a solution $\hat{\gamma}_{\text{meta}}$ such that*

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

where $\chi_{\eta_e} = (1+\alpha\chi)^{2L} + C_b b + C_\chi \mathbb{E}_t[\|\nabla J_{Q_t}(\hat{\gamma}_{\text{meta}})\|_2]$, 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 B.2.2 (Lemma 19 (Ji et al., 2022))**.** *Under assumptions (A1) - (A5), for any t and any $\gamma \in \mathbb{R}^{p+1}$, we have*

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

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

B.2.2 Proof of Theorem 5.5.1

To prove Theorem 5.5.1, we firstly derive three useful propositions (P1-P3) based on our Assumption 5.5.1 and Assumption 5.5.2 in Section 5.5, 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 5.5.1 and Assumption 5.5.2, 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 5.5.1).

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

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

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 5.5.2). □

Proof of Theorem 5.5.1:

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

Note that Assumption 5.5.1 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 5.5.1 follow from the conclusion of Theorem B.2.1. □

B.2.3 Proof of Corollary 5.5.1.1

Proof. Since Assumption 5.5.1 and Assumption 5.5.2 imply (A1) to (A5) in Appendix B.2.1, we will use the constants defined earlier in Appendix B.2.1 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 B.2.2.

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 5.5.1,

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

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

B.3 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.3.1. Details of the boundary-value ODE are provided in Appendix B.3.2. Details of Bayesian inference for the Lotka–Volterra system are provided in Appendix B.3.3. Details of the Sarcos robot arm are presented in Appendix B.3.4.

B.3.1 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 neuron (the output then is multiplied by an identity matrix I_d to be 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 learning rate $\alpha = 0.01$. The number of inner gradient steps is $L = 1$. The meta learning rate $\eta = 0.002$ 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 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 learning rate $\alpha = \frac{0.01}{50 \times L}$ for $L \in \{1, 3, 5, 7, 10\}$. The meta learning rate $\eta = 0.002$ 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 learning rate $\alpha = 0.01$. The number of inner gradient steps is $L = 1$. The meta learning rate $\eta = 0.002$ 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 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.

B.3.2 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 learning rate $\alpha = 0.01$ and the meta learning rate $\eta = 0.002$ 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 learning rate is 0.002. The number of training epochs for each task is set to be 20 with batch size 5.

B.3.3 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,

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

where

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

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

The model is,

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

where

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

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,

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

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 learning rate $\alpha = 0.0001$. The number of inner gradient steps is $L = 1$. The meta learning rate was $\eta = 0.001$ for all 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.
- For Neural-CVs: The 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 learning rate $\alpha = 0.001$. The number of inner gradient steps is $L = 1$. The meta learning rate was $\eta = 0.001$ for all 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.
- For Neural-CVs: The learning rate is 0.001. The number of training epochs for each task is set to be 20 with batch size 5.

B.3.4 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 hyperparameters $x \in \mathbb{R}^{2+}$ 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, 2013) to learn ϕ. Figure B.1 demonstrates the prior and the corresponding posterior of the kernel hyper-parameters $x = (x_1, x_2)$ (in the form of 2d histograms).

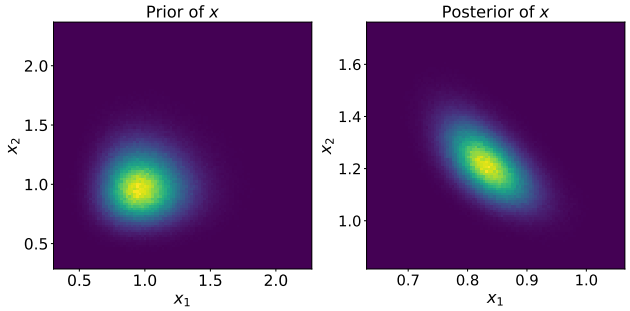

Figure B.1: Priors and posteriors of kernel hyper-parameters x.

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 learning rate $\alpha = 0.01$. The meta learning rate was $\eta = 0.001$ for all 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 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 B.2.

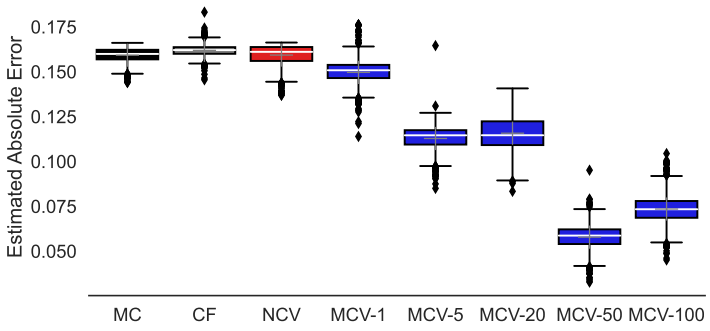

Figure B.2: Estimated absolute errors over the same training states (which are used for learning the Meta-CV) of the Sarcos anthropomorphic robot arm (CF: Control functionals; NCV: Neural-CVs; MCV-L: Meta-CVs with L inner steps).

Appendix C

Supplementary Material of Amortized Bayesian Prototype Meta-learning

C.1 Overview

In this appendix, we present details of experimental settings, including hyper-parameters including batch size and learning rates. We also provide detailed statistics in plots and figures. All experiments are implemented with *PyTorch*.

C.2 Experimental Details

At the meta-training stage, except that the maximum training epoch is 12000 for 1-shot classification on *mini-ImageNet*, the maximum training epoch is set to be 3500 epochs for all the other experiments. We use a mini-batch of tasks consisting T tasks to update the shared θ during meta-training.

We select the optimal meta-training epoch on the meta-validation set according to classification accuracy. At the meta-testing stage, we randomly sample 600 novel tasks from the meta-testing set, and report the mean accuracy with its 95% confidence interval, i.e., $\text{mean acc.} \pm 1.96\frac{\text{std}}{\sqrt{600}}$. For C-way K-shot, a task is constructed by sampling C classes and then subsequently sampling $K + M$ instances for each class, with K being the number of support images in each class. In our

experiments,

- *Omniglot*: $M = 15$ for meta-training/meta-validation/meta-testing;
- *mini-ImageNet*: $M = 16$ for meta-training and meta-validation, $M = 15$ for meta-testing;
- *CUB-200-2011*: $M = 16$ for meta-training and meta-validation, $M = 15$ for meta-testing;
- *Stanford-dogs*: $M = 16$ for meta-training and meta-validation, $M = 15$ for meta-testing.

The values of B, L, α and η in Algorithm 8 are set to be

- *Omniglot*: $B = 32$, $L = 1$, $\alpha = 0.1$, $\eta = 0.001$;
- *mini-ImageNet*: $B = 4$, $L = 5$, $\alpha = 0.01$, $\eta = 0.001$;
- *CUB-200-2011*: $B = 4$, $L = 5$, $\alpha = 0.01$, $\eta = 0.001$;
- *Stanford-dogs*: $B = 4$, $L = 5$, $\alpha = 0.01$, $\eta = 0.001$.

In addition, we use standard stochastic gradient descent to generate variational parameters ϕ_i, during meta-training/meta-validation/meta-testing, for a task $\mathcal{T}_i$ and for all i. We use the *Adam* optimizer to update the shared parameter θ at meta-training stage.

C.3 Details of Figures

In this section, we present details of Figure 6.1 in the following tables, i.e. Table C.1, Table C.2 and Table C.3. In particular, the setting of dropout used in Table C.3 is set to be:

- *Omniglot*: *Dropout* with a keep probability of 0.9.
- *mini-ImageNet*: *Dropout* with a keep probability of 0.5.

C.4 Comparisons of Convolution Networks

Here, we present details of shallow convolution networks used in the probabilistic meta-learning methods in Table C.4. CONV-X means a convolution network with X convolution blocks.

Table C.1: *Ablation study in Figure 6.1-(a).*

	Meta-training conditions	
C-way at meta-testing	5-way 5-shot (%)	10-way 5-shot (%)
$C = 5$	99.45 ± 0.09	99.44 ± 0.08
$C = 10$	98.97 ± 0.08	99.14 ± 0.08
$C = 15$	98.45 ± 0.09	98.80 ± 0.09
$C = 20$	98.14 ± 0.09	98.52 ± 0.08
$C = 25$	97.85 ± 0.09	98.20 ± 0.08
$C = 30$	97.44 ± 0.09	97.87 ± 0.08
$C = 35$	97.17 ± 0.09	97.63 ± 0.08
$C = 40$	96.84 ± 0.08	97.34 ± 0.08
$C = 45$	96.57 ± 0.08	97.12 ± 0.08
$C = 50$	96.30 ± 0.08	96.85 ± 0.08

Table C.2: *Ablation study in Figure 6.1-(b).*

	Meta-training conditions	
K-shot at meta-testing	5-way 5-shot (%)	10-way 5-shot (%)
$K = 2$	98.65 ± 0.27	98.38 ± 0.15
$K = 4$	99.47 ± 0.11	99.00 ± 0.11
$K = 5$	99.60 ± 0.10	99.17 ± 0.09
$K = 6$	99.53 ± 0.11	99.19 ± 0.10
$K = 8$	99.59 ± 0.10	99.06 ± 0.13
$K = 10$	99.61 ± 0.09	99.32 ± 0.10
$K = 12$	99.60 ± 0.09	99.34 ± 0.09

Table C.3: *Ablation study in Figure 6.1-(c).*

KL	*Dropout*	*Omniglot* (%)	*mini-ImageNet* (%)
-	-	96.16 ± 0.28	43.08 ± 0.62
✓	-	99.54 ± 0.08	70.44 ± 0.72
✓	✓	99.50 ± 0.08	69.92 ± 0.67

C.5 Effect of L

In Table C.5, we also take the effect of L into account. Recall that L is the number of updates of the inner loop for the approximate inference. We consider the cases

Table C.4: *Convolution networks of methods in Table 6.1.*

	Omniglot	*mini-ImageNet*
BMAML	CONV-5	CONV-5
PLATIPUS	CONV-4	CONV-4
VAMPIRE	CONV-4	CONV-4
ABML	CONV-4	CONV-4
Amortized VI	CONV-4	CONV-5
VERSA	CONV-4	CONV-5
Meta-Mixture	CONV-4	CONV-4
DKT	CONV-4	CONV-4
Ours	CONV-4	CONV-4

when $L = 1$, $L = 3$ and $L = 5$. Performance for each choice of L is measured on the meta-testing set.

Table C.5: *Effect of L.*

mini-ImageNet	$L = 1$	$L = 3$	$L = 5$
5-way 1-shot	$52.79 \pm 0.94(\%)$	$53.29 \pm 0.89(\%)$	$53.28 \pm 0.91(\%)$
5-way 5-shot	$69.63 \pm 0.70(\%)$	$70.56 \pm 0.70(\%)$	$70.44 \pm 0.72(\%)$