Harnessing meta-learning via probabilistic modelling and trajectory optimisation

by
Cuong Nguyen

A thesis submitted for the degree of
Doctor of Philosophy
in
Computer Science
of
THE UNIVERSITY OF ADELAIDE

Supervision panel:
Professor Gustavo Carneiro
Dr Toan Do

October 2021

Abstract

Meta-learning has recently flourished as one of the most promising transfer learning techniques that can adapt quickly to a new task, even if that task consists of a limited number of training examples. The main idea of meta-learning is to use a meta-parameter to model the shared structure between many observed tasks and utilise the knowledge gained from such modelling to facilitate the learning for unobserved tasks. Despite steady progress with many remarkable state-of-the-art results, existing meta-learning algorithms are often fragile due to the lack of studies in prediction uncertainty and generalisation for unseen tasks. In addition, little is known about how tasks are related to each other, potentially leading to sub-optimal solutions due to the assumption that tasks are evenly distributed – which is hardly true in practice. This thesis, therefore, aims to address such problems through the lenses of probabilistic modelling and optimisation. In particular, the thesis proposes to (i) integrate variational inference into meta-learning that considers the epistemic uncertainty into the modelling to reduce calibration errors and overfitting induced by meta-learning models, (ii) derive a PAC-Bayes upper-bound of errors evaluated on both seen and unseen tasks to enable the study of theoretical generalisation in meta-learning and use that bound to formulate a loss function applied in the training of different meta-learning methods, (iii) model tasks via a variant of Gaussian latent Dirichlet allocation and utilise the newly-obtained representation for task selection to make training more efficient, and (iv) adopt trajectory optimisation from optimal control to determine the re-weighting factor of each training task to optimise the training process of meta-learning. The results of these studies improve the robustness and provide an insightful understanding of meta-learning, and thus, enable further development of practical meta-learning approaches.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

October, 2021

Cuong Nguyen

Acknowledgement

Pursuing my PhD. degree is an exciting and at the same time challenging period in my life. Such experience has not only attributed to my intellectual growth, but also shaped my philosophical perspective. The completion of my research would not be possible without the kind support of many people whom I have had the opportunity to learn, work and collaborate with. Although I cannot thank all of them individually in writing, I would like to acknowledge some that are particular helpful.

Professor Gustavo Carneiro has fully and continuously supported the research from its inception. His immense knowledge and extraordinary patience has guided the research direction and been an instrumental in coaching me from a student without computer science background to a researcher with a specialisation in machine learning. I am extremely grateful to have him as my advisor and mentor.

Dr Toan Do has also provided a strong support to the research with insightful discussions and encouragement. His intellectual expertise and energy have aided to the direction of research. I am very grateful to have him as my co-advisor.

I would like to thank many colleagues at the Australian Institute for Machine Learning (AIML), including Toan Tran, Dung Doan, Violetta Shevchenko, Gabriel Maicas, Yutong Dai, Shin-Fang Ch'ng, Daqi Liu, Chee Kheng Chng, among others for many amusing moments. You all have made every day at AIML more enjoyable.

I would like to thank many friends outside of works, including the families of Dinh Pham, Hoa Nguyen, Dung Le, Bao Doan, Viet Vo, Khuong Nguyen, Khue Nguyen, Linh V Nguyen, Kiet Wong, among others. Hanging out with them and enjoying their delicious food have made me feel welcome and reduced my homesickness while studying in Australia.

I would also like to thank Huu Le – a long time friend – for his generous support even remotely. His encouragement and sharing about research life have made me overcome some difficult times during the period of my research. I have enjoyed talking to him about not only scientific research, but also history, life and everything.

Finally, I am grateful to my parents, my younger sister and my girlfriend, for their constant love, support and confidence in me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The vast development of machine learning has created a large number of applications to solve increasingly complex problems (Hinton et al., 2012; Krizhevsky et al., 2012; Graves et al., 2013; Simonyan and Zisserman, 2015; Havaei et al., 2017). Such successes rely on high capacity models (e.g. very deep neural networks with millions of parameters) trained on a massive amount of annotated data, which is obtained from an arduous, costly and even infeasible annotation process. This, however, contrasts with the learning ability in humans where a new task can be adapted with only a few "training" examples. Such difference might be due to the fact that humans tend to utilise prior knowledge gained from past experiences to facilitate the learning of a new task, while machine learning models are often trained from scratch without utilising any prior knowledge. This has, therefore, motivated the research of novel learning approaches, generally known as transfer learning, that exploit past experience (in the form of models learnt from other training tasks) to quickly learn a new task using relatively small training sets.

Transfer learning, in general, assumes the existence of a common latent distribution over training and testing tasks. Such assumption means that learning to solve one or many training tasks can be helpful to solve a new testing task sampled from the same or "similar" task distribution, even if the new task contains a limited number of training examples. There are many transfer learning paradigms studied in the literature depending on the configuration of the problems to be solved. For example, in *fine-tuning* (L. Y. Pratt et al., 1991; Yosinski et al., 2014) – one of the most widely-used transfer learning techniques, a part of a model pre-trained on a source task is re-used to extract relevant features. The remaining part of that model is then trained or fine-tuned on the data of the target task. In *multi-task learning* (Caruana, 1997), an agent simultaneously learns the shared representation of many related tasks and a main task that are assumed to come from the same domain. The information extracted from the related tasks tends to regularise the training of the main task, potentially improving the performance of that task. In

domain adaptation (Heckman, 1979; Shimodaira, 2000; Japkowicz and Stephen, 2002; Daume III and Marcu, 2006; Ben-David et al., 2007), a learner utilises the information of data in one or many source domains and a small amount of data in a target domain to learn domain in-variant knowledge. Such knowledge is then used to enhance the prediction made on the tasks drawn from the target domain. In *meta-learning* (Schmidhuber, 1987; Thrun and L. Pratt, 1998; Baxter, 2000), a meta-learner learns how to solve many training tasks, and use that experience to facilitate the learning of future tasks drawn from the same task distribution.

Given the ability that leverages prior knowledge to quickly adapt to new tasks or new environments, meta-learning has recently been re-visited and flourished with state-of-the-art results in several few-shot learning benchmarks (Vinyals et al., 2016; Ravi and Larochelle, 2018; Finn et al., 2017; Finn et al., 2018; Grant et al., 2018; Ravi and Beatson, 2019). This thesis focuses on investigating transfer learning techniques based on meta-learning, especially in the probabilistic and optimisation point of views.

In the remaining sections, we provide relevant background and present the formulation for meta-learning. Such explicit formulation allows to distinguish meta-learning from some other types of transfer learning techniques, such as fine-tuning, multi-task learning, domain adaptation and continual learning. The final parts of the chapter include our research questions in meta-learning and our contributions.

1.1 Background

In this section, we review relevant background in single-task learning that relates to meta-learning. The background is then used in the mathematical formulation of meta-learning presented in Section 1.2.

1.1.1 Data generation model of a task

Consider a set of tasks indexed by $i \in \mathbb{N} = \{1, 2, \ldots\}$. A data point for a task consists of an input-label pair. We denote the input for the j-th data point of the i-th task as $\mathbf{x}_{ij} \in \mathcal{X} \subseteq \mathbb{R}^d$ and its corresponding label as $\mathbf{y}_{ij} \in \mathcal{Y}$. For simplicity, only two families of tasks – regression and classification – are considered in this thesis. As a result, the label is defined as $\mathcal{Y} \subseteq \mathbb{R}$ for regression and as $\mathcal{Y} = \{0, 1, \ldots, C-1\}$ for classification, where C is the number of classes.

Each data point in a task can be generated in 2 steps:

1. generate the input $\mathbf{x}_{ij}$ by sampling from some probability distribution $\mathcal{D}_i$,
2. determine the label $\mathbf{y}_{ij} = f_i(\mathbf{x}_{ij})$, where $f_i : \mathcal{X} \to \mathcal{Y}$ is the "correct" labelling function.

Both the probability distribution $\mathcal{D}_i$ and the labelling function f_i are unknown to the learning agent.

For simplicity, we denote $(\mathbf{x}_{ij}, \mathbf{y}_{ij}) \sim (\mathcal{D}_i, f_i)$ as the data generation model of task i-th.

1.1.2 Loss function of a task

Definition 1.1: Loss function

The loss function $\ell_i : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ of the i-th task is defined as a measure of "error" between the true label $f_i(\mathbf{x}_{ij})$ and predicted label $h_i(\mathbf{x}_{ij})$ made by the model of interest h parameterised by $\mathbf{w}_i$ on a random data point $\mathbf{x}_{ij}$ generated from the underlying distribution.

Some common loss functions are:

- Negative log likelihood: $\ell_i(h(\mathbf{x}_{ij}; \mathbf{w}_i), \mathbf{y}_{ij}) = -\ln p(\mathbf{y}_{ij}|\mathbf{x}_{ij}; \mathbf{w}_i)$. Thus:
  - if the likelihood follows a Gaussian distribution as in regression, the loss function corresponds to mean-squared error,
  - if the likelihood follows a categorical distribution as in classification, the loss function is the cross-entropy loss,
  - if the likelihood follows a discrete Dirac delta distribution, then the loss function is 0-1 loss.
- Variational-free energy used in variational inference.

1.1.3 Task instance

Definition 1.2: Task (Hospedales et al., 2021)

A task or a task instance $\mathcal{T}_i$ consists of an unknown associated data generation model $(\mathcal{D}_i, f_i)$, and a loss function ℓ_i, denoted as: $\mathcal{T}_i = (\mathcal{D}_i, f_i, \ell_i)$.

To solve a task $\mathcal{T}_i$, one needs to obtain an optimal model $h(.; \mathbf{w}_i^*) : \mathcal{X} \to \mathcal{Y}$, parameterised by a task-specific parameter $\mathbf{w}_i^* \in \mathcal{W} \subseteq \mathbb{R}^n$, which minimises a loss function ℓ_i on the data of that task:

$$\mathbf{w}_i^* = \arg\min_{\mathbf{w}_i} \mathbb{E}_{(\mathbf{x}_{ij}, \mathbf{y}_{ij}) \sim (\mathcal{D}_i, f_i)} \left[\ell_i(h(\mathbf{x}_{ij}; \mathbf{w}_i), \mathbf{y}_{ij}) \right]. \tag{1.1}$$

In practice, since both $\mathcal{D}_i$ and f_i are unknown, the data generation model is replaced by a dataset consisting of a finite number of data-points generated according to the data generation model $(\mathcal{D}_i, f_i)$, denoted as $S_i = \{(\mathbf{x}_{ij}, \mathbf{y}_{ij})\}_{j=1}^{m_i}$, where $m_i \in \mathbb{N}$

is the number of data points. The objective to solve that task is often known as empirical risk minimisation:

$$\mathbf{w}_i^{\mathrm{ERM}} = \arg\min_{\mathbf{w}_i} \frac{1}{m_i} \sum_{j=1}^{m_i} \left[\ell_i(h(\mathbf{x}_{ij}; \mathbf{w}_i), \mathbf{y}_{ij})\right]. \tag{1.2}$$

Since the loss function used is the same for each task family, e.g. ℓ is mean squared error (MSE) for regression and cross-entropy for classification, the subscript on the loss function is, therefore, dropped, and the loss is denoted as ℓ throughout this chapter. Furthermore, given the commonality of the loss function across all tasks, a task can, therefore, be simply represented by either its data generation model $(\mathcal{D}_i, f_i)$ or the associated dataset S_i.

1.1.4 Hyper-parameter optimisation

In single-task learning, the common way to "tune" or optimise a hyper-parameter is to split a given dataset S_i into two disjoint subsets:

- *Training* (or *support*) subset $S_i^{(t)} = \left\{\left(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}\right)\right\}_{j=1}^{m_i^{(t)}}$,
- *Validation* (or *query*) subset $S_i^{(v)} = \left\{\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}\right)\right\}_{j=1}^{m_i^{(v)}}$,

where:

$$S_i = S_i^{(t)} \cup S_i^{(v)}, \quad S_i^{(t)} \cap S_i^{(v)} = \varnothing.$$

Note that with this definition, $m_i^{(t)} + m_i^{(v)} = m_i$, and $m_i^{(t)}$ and $m_i^{(v)}$ are not necessarily identical.

The subset $S_i^{(t)}$ is used to train the model parameter of interest $\mathbf{w}_i$, while the subset $S_i^{(v)}$ is used to validate the hyper-parameter, denoted by θ (we provide examples of the hyper-parameter in Section 1.2). Mathematically, the hyper-parameter optimisation in a single-task can be written as the following bi-level optimisation:

$$\begin{aligned} &\min_{\theta} \frac{1}{m_i^{(v)}} \sum_{k=1}^{m_i^{(v)}} \ell\left(h\left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i^*(\theta)\right), y_{ik}^{(v)}\right) \\ &\text{s.t.: } \mathbf{w}_i^*(\theta) = \arg\min_{\mathbf{w}_i} \frac{1}{m_i^{(t)}} \sum_{j=1}^{m_i^{(t)}} \ell\left(h\left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i(\theta)\right), y_{ij}^{(t)}\right). \end{aligned} \tag{1.3}$$

Note that the notation $\mathbf{w}_i(\theta)$ denotes that the task-specific parameter $\mathbf{w}_i$ is a function of the hyper-parameter θ.

We can extend the hyper-parameter optimisation in (1.3) evaluated on the two

data subsets $S_i^{(t)}$ and $S_i^{(v)}$ to the general data generation model as the following:

$$\begin{aligned} &\min_{\theta} \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i^*(\theta) \right), y_{ik}^{(v)} \right) \right] \\ &\text{s.t.: } \mathbf{w}_i^*(\theta) = \arg\min_{\mathbf{w}_i} \mathbb{E}_{(\mathbf{x}_{ik}^{(t)}, y_{ik}^{(t)}) \sim (\mathcal{D}_i^{(t)}, f_i)} \left[\ell \left(h \left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i(\theta) \right), y_{ij}^{(t)} \right) \right], \end{aligned} \tag{1.4}$$

where $\mathcal{D}_i^{(t)}$ and $\mathcal{D}_i^{(v)}$ are the probability distributions of training and validation input data, respectively, and they are not necessarily identical.

1.2 Formulation of meta-learning

The setting of the meta-learning problem considered in this paper follows the *task environment* (Baxter, 2000) that describes the unknown distribution $p(\mathcal{D}, f)$ over a family of tasks. Each task $\mathcal{T}_i$ is sampled from this task environment and can be represented as $\left(\mathcal{D}_i^{(t)}, \mathcal{D}_i^{(v)}, f_i\right)$, where $\mathcal{D}_i^{(t)}$ and $\mathcal{D}_i^{(v)}$ are the probability of training and validation input data, respectively, and are not necessarily identical. The aim of meta-learning is to use T training tasks to train a meta-learning model that can be fine-tuned to perform well on an unseen task sampled from the same task environment.

Such meta-learning methods use meta-parameters to model the common latent structure of the task distribution $p(\mathcal{D}, f)$. In this thesis, we consider meta-learning as an extension of hyper-parameter optimisation in single-task learning, where the hyper-parameter of interest – often called meta-parameter – is shared across many tasks. Similar to hyper-parameter optimisation presented in Subsection 1.1.4, the objective of meta-learning is also a bi-level optimisation:

$$\begin{aligned} &\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{D}, f_i)} \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i^*(\theta) \right), y_{ik}^{(v)} \right) \right] \\ &\text{s.t.: } \mathbf{w}_i^*(\theta) = \arg\min_{\mathbf{w}_i} \mathbb{E}_{(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}) \sim (\mathcal{D}_i^{(t)}, f_i)} \left[\ell \left(h \left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i(\theta) \right), y_{ij}^{(t)} \right) \right]. \end{aligned} \tag{1.5}$$

The difference between meta-learning presented in (1.5) and the hyper-parameter optimisation in (1.4) is that the meta-parameter (also known as hyper-parameter) θ is shared across all tasks sampled from the task environment $p(\mathcal{D}, f)$. Such difference is reflected through the expectation highlighted in red colour in (1.5).

In practice, the meta-parameter (or shared hyper-parameter) θ can be chosen as one of the followings:

- *learning rate* of gradient-based optimisation used to minimise the lower level objective function in (1.5) to learn $\mathbf{w}_i^*(\theta)$ (Z. Li et al., 2017),
- *initialisation* of model parameter when using gradient-based optimisation to minimise the lower-level in (1.5) (Finn et al., 2017),

- *data representation* or *feature extractor* (Vinyals et al., 2016; Snell et al., 2017),

- *optimiser* used to optimise the lower-level in (1.5) (Andrychowicz et al., 2016; K. Li and Malik, 2017).

The meta-parameter θ is assumed to be the initialisation of model parameters throughout this thesis. Formulation, derivation and analysis in the subsequent sections and chapters will, therefore, revolve around this assumption. Note that the analysis can be straight-forwardly extended to other types of meta-parameters with slight modifications.

In general, the objective function of meta-learning in (1.5) can be solved by gradient-based optimisation, such as gradient descent. Due to the nature of the bi-level optimisation, the optimisation are often carried out in two steps. The first step is to adapt (or fine-tuned) the meta-parameter θ to the task-specific parameter $\mathbf{w}_i(\theta)$. This corresponds to the optimisation in the lower-level, and can be written as:

$$\mathbf{w}_i^*(\theta) = \mathbf{w}_i - \alpha \mathbb{E}_{(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}) \sim (\mathcal{D}_i^{(t)}, f_i)} \left[\nabla_{\mathbf{w}_i} \ell \left(h \left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i \right), \mathbf{y}_{ij}^{(t)} \right) \right] \Big|_{\mathbf{w}_i = \theta}, \tag{1.6}$$

where α is a hyper-parameter denoting the learning rate for task $\mathcal{T}_i$. For simplicity, the adaptation step in (1.6) is carried out with only one gradient descent update.

The second step is to minimise the validation loss induced by the locally-optimal task-specific parameter $\mathbf{w}_i^*(\theta)$ evaluated on the validation subset w.r.t. the meta-parameter θ. This corresponds to the upper-level optimisation, and can be expressed as:

$$\theta \leftarrow \theta - \gamma \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{D}, f)} \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, \mathbf{y}_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\nabla_{\theta} \ell \left(h \left(\mathbf{x}_{ij}^{(v)}; \mathbf{w}_i^*(\theta) \right), \mathbf{y}_{ij}^{(v)} \right) \right], \tag{1.7}$$

where γ is another hyper-parameter representing the learning rate to learn θ. The general algorithm of meta-learning using gradient-based optimisation is shown in Algorithm 1.

1.2.1 Second-order meta-learning

As shown in (1.7), the optimisation for the meta-parameter θ requires the gradient of the validation loss averaged across T tasks. Given that each task-specific parameter $\mathbf{w}_i^*$ is a function of θ due to the lower-level optimisation in (1.6), the gradient of

Algorithm 1 Training procedure of meta-learning in general

procedure TRAINING(task environment $p(\mathcal{D}, f)$, learning rates γ and α)
  initialise meta-parameter θ
  while θ not converged **do**
    sample a mini-batch of T tasks from task environment $p(\mathcal{D}, f)$
    for each task $\mathcal{T}_i, i \in \{1, \dots, T\}$ **do**
      sample two data subsets $S_i^{(t)}$ and $S_i^{(v)}$ from task $\mathcal{T}_i = (\mathcal{D}_i^{(t)}, \mathcal{D}_i^{(v)}, f_i)$
      adapt meta-parameter to task $\mathcal{T}_i$:
      $\mathbf{w}_i^*(\theta) = \theta - \frac{\alpha}{m_i^{(t)}} \sum_{j=1}^{m_i^{(t)}} \nabla_{\mathbf{w}_i} \left[\ell \left(h \left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i(\theta) \right), y_{ij}^{(t)} \right) \right]$ ▷ Eq. (1.6)
    end for
    update meta-parameter:
    $\theta \leftarrow \theta - \frac{\gamma}{T} \sum_{i=1}^{T} \frac{1}{m_i^{(v)}} \sum_{k=1}^{m_i^{(v)}} \nabla_{\theta} \left[\ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i^*(\theta) \right), y_{ik}^{(v)} \right) \right]$ ▷ Eq. (1.7)
  end while
  return the trained meta-parameter θ
end procedure

interest can be expanded as:

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{D},f)} \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\nabla_\theta \ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i^*(\theta) \right), y_{ik}^{(v)} \right) \right] \\
&= \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{D},f)} \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\nabla_\theta^\top \mathbf{w}_i^*(\theta) \times \nabla_{\mathbf{w}_i} \ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i \right), y_{ik}^{(v)} \right) \Big|_{\mathbf{w}_i = \mathbf{w}_i^*(\theta)} \right] \\
&= \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{D},f)} \left\{ \left[\mathbf{I} - \alpha \mathbb{E}_{(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}) \sim (\mathcal{D}_i^{(t)}, f_i)} \left[\nabla_{\mathbf{w}_i}^2 \ell \left(h \left(\mathbf{x}_{ij}^{(t)}; \mathbf{w}_i \right), y_{ij}^{(t)} \right) \Big|_{\mathbf{w}_i = \theta} \right] \right] \right. \\
&\quad \left. \times \mathbb{E}_{(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}) \sim (\mathcal{D}_i^{(v)}, f_i)} \left[\nabla_{\mathbf{w}_i} \ell \left(h \left(\mathbf{x}_{ik}^{(v)}; \mathbf{w}_i \right), y_{ik}^{(v)} \right) \Big|_{\mathbf{w}_i = \mathbf{w}_i^*(\theta)} \right] \right\},
\end{aligned}
\tag{1.8}
$$

where the first equality is due to chain rule, and the second equality is the result that differentiates the gradient update in (1.6). Note that in the second equality, we remove the transpose notation since the corresponding matrix is symmetric.

Hence, naively implementing such gradient would require to calculate the Hessian matrix $\nabla_{\mathbf{w}_i}^2 \ell$, resulting in an intractable procedure for large models, such as deep neural networks. To obtain a more efficient implementation, one can utilise the Hessian-vector product (Pearlmutter, 1994) between the gradient vector $\nabla_{\mathbf{w}_i} \ell$ and the Hessian matrix $\nabla_{\mathbf{w}_i}^2 \ell$ to efficiently calculate the gradient of the validation loss w.r.t. θ.

Another way to calculate the gradient of the validation loss w.r.t. the meta-parameter θ is to use implicit differentiation (Domke, 2012; Rajeswaran et al., 2019; Lorraine et al., 2020). This approach is more advantaged since it does not need to stores the computational graph and takes gradient via chain rule as the methods mentioned above. Such implicit differentiation technique reduces the memory usage and therefore, allows to work with large-scale models. However, the trade-off is the

increasing computational time to estimate the gradient of interest.

Nevertheless, the implementations that compute the exact gradient of the validation loss w.r.t. θ without approximation are often referred to as "second-order" meta-learning.

1.2.2 First-order meta-learning

In practice, the Hessian matrix $\nabla^2_{\mathbf{w}_i}\ell$ is often omitted from the calculation to simplify the update for the meta-parameter θ (Finn et al., 2017). The resulting gradient consists of only the gradient of validation loss $\nabla_{\mathbf{w}_i}\ell$, which is more efficient to calculate with a single forward-pass if auto differentiation is used. This approximation is often referred as "first-order" meta-learning, and the gradient of interest can be presented as:

$$\begin{aligned}&\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{D},f)}\mathbb{E}_{(\mathbf{x}_{ik}^{(v)},y_{ik}^{(v)})\sim(\mathcal{D}_i^{(v)},f_i)}\left[\nabla_\theta\ell\left(h\left(\mathbf{x}_{ij}^{(v)};\mathbf{w}_i^*(\theta)\right),y_{ik}^{(v)}\right)\right]\\&\approx\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{D},f)}\mathbb{E}_{(\mathbf{x}_{ik}^{(v)},y_{ik}^{(v)})\sim(\mathcal{D}_i^{(v)},f_i)}\left[\nabla_{\mathbf{w}_i}\ell\left(h\left(\mathbf{x}_{ik}^{(v)};\mathbf{w}_i\right),y_{ik}^{(v)}\right)\Big|_{\mathbf{w}_i=\mathbf{w}_i^*(\theta)}\right].\end{aligned}\tag{1.9}$$

REPTILE (Nichol et al., 2018) – a variant first-order meta-learning – approximates further the gradient of validation loss $\nabla_{\mathbf{w}_i}\ell$, resulting in a much simpler approximation:

$$\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{D},f)}\mathbb{E}_{(\mathbf{x}_{ik}^{(v)},\mathbf{y}_{ik}^{(v)})\sim(\mathcal{D}_i^{(v)},f_i)}\left[\nabla_\theta\ell\left(h\left(\mathbf{x}_{ik}^{(v)};\mathbf{w}_i^*(\theta)\right),y_{ik}^{(v)}\right)\right]=\theta-\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{D},f)}\left[\mathbf{w}_i^*(\theta)\right].\tag{1.10}$$

1.3 Differentiation from other transfer learnings

Given the definition of meta-learning in Section 1.2, it is often a source of confusion to differentiate meta-learning from other transfer learning approaches. In this section, some popular transfer learning methods are described with their objective function formulated to purposely distinguish from meta-learning.

1.3.1 Fine-tuning

Fine-tuning is the most common technique in neural network based transfer learning (L. Y. Pratt et al., 1991; Yosinski et al., 2014) where the last or a couple of last layers in a neural network pre-trained on a source task are replaced and fine-tuned on a target task. Formally, if $g(.;\mathbf{w}_0)$ is denoted as the forward function of the shared layers with shared parameters $\mathbf{w}_0$, where $\mathbf{w}_s$ and $\mathbf{w}_t$ are the parameters of the remaining layers h specifically trained on source and target tasks, respectively,

then the objective of fine-tuning can be expressed as:

$$\begin{aligned} &\min_{\mathbf{w}_t} \mathbb{E}_{(\mathbf{x}_t, y_t)\sim\mathcal{T}_t} \left[\ell\left(h\left(g\left(\mathbf{x}_t; \mathbf{w}_0^*\right); \mathbf{w}_t\right), y_t\right)\right] \\ &\text{s.t.: } \mathbf{w}_0^*, \mathbf{w}_s^* = \arg\min_{\mathbf{w}_0, \mathbf{w}_s} \mathbb{E}_{(\mathbf{x}_s, y_s)\sim\mathcal{T}_s} \left[\ell\left(h\left(g\left(\mathbf{x}_s; \mathbf{w}_0\right); \mathbf{w}_s\right), y_s\right)\right], \end{aligned} \tag{1.11}$$

where $\mathbf{x}_s, y_s$ and $\mathbf{x}_t, y_t$ are the data sampled from the source task $\mathcal{T}_s$ and target task $\mathcal{T}_t$, respectively.

Although the objective of fine-tuning shown in (1.11) is still a bi-level optimisation, it is easier to solve than the one in meta-learning due to the following reasons:

- The objective in fine-tuning has only one constrain corresponding to one source task, while meta-learning has several constrains corresponding to multiple training tasks.
- In fine-tuning, $\mathbf{w}_t$ and $\mathbf{w}_0$ are inferred separately, while in meta-learning, the task-specific parameter is a function of the meta-parameter, resulting in a more complicated correlation.

The downside of fine-tuning is the requirement of a reasonable number of training examples on the target task to fine-tune $\mathbf{w}_t$. In contrast, meta-learning leverages the knowledge extracted from several training tasks to quickly adapt to a new task with only a few training examples.

1.3.2 Domain adaptation and generalisation

Domain adaptation or domain-shift refers to the case when the joint data-label distribution on source and target are different, denoted as $(\mathcal{D}_s, f_s) \neq (\mathcal{D}_t, f_t)$ (Heckman, 1979; Shimodaira, 2000; Japkowicz and Stephen, 2002; Daume III and Marcu, 2006; Ben-David et al., 2007). The aim of domain adaptation is to leverage the model $h(.; \mathbf{w})$ trained on source domain to available unlabelled data $\mathbf{x}_t$ in the target domain, so that the model obtained can perform reasonably well on the target domain. Mathematically, it can be written as:

$$\begin{aligned} &\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x}_t, y_t)\sim(\mathcal{D}_t, f_t)} \left[\ell\left(h\left(\mathbf{x}_t; \mathbf{w}\right), y_t\right)\right] \\ &\text{s.t.: } \mathbf{w} \in \mathcal{W}(\mathcal{D}_s, f_s, \mathcal{D}_t), \end{aligned} \tag{1.12}$$

where: $\mathcal{W}(\mathcal{D}_s, f_s, \mathcal{D}_t)$ is the feasible set of model parameters computed based on the labelled data of the source domain and unlabelled data of the target domain. Note that during training, the labels on the target domain is unknown to the learning agent.

Given the objective of domain adaptation in (1.12), it differs from meta-learning with the following reasons:

- Domain adaptation assumes a shift in the task environments that generate source and target tasks, while meta-learning assumes that such tasks are i.i.d. sampled from the same task environment.
- In the training phase, domain adaptation utilises information of data, and in particular, the unlabelled data $\mathbf{x}_t$, from target domain, while meta-learning does not have such access.
- In the testing phase, domain adaptation makes prediction by exploiting the trained model without any further fine-tuning nor additional training data, while meta-learning needs to fine-tune on a small set of labelled data of the targeted task.

In general, meta-learning learns a shared prior or hyper-parameters to generalise for unseen tasks, while domain adaptation produces a model to solve a particular task in a specified target domain. Recently, there is a variance of domain adaptation, named **domain generalisation**, where the aim is to learn a domain-invariant model without any information of target domain. In this view, domain generalisation is very similar to meta-learning, and there are some works that employ meta-learning algorithms for domain generalisation (D. Li et al., 2018; Y. Li et al., 2019).

1.3.3 Multi-task learning

Multi-task learning learns several related auxiliary tasks and a target task simultaneously to exploit the diversity of task representation to regularise and improve the performance on the target task (Caruana, 1997). If the input $\mathbf{x}$ is assumed to be the same across T extra tasks and the target task $\mathcal{T}_{T+1}$, then the objective of multi-task learning can be expressed as:

$$\min_{\mathbf{w}_0, \{\mathbf{w}_i\}_{i=1}^{T+1}} \frac{1}{T+1} \sum_{i=1}^{T+1} \ell_i \left(h_i \left(g \left(\mathbf{x}; \mathbf{w}_0 \right); \mathbf{w}_i \right), y_i \right), \tag{1.13}$$

where y_i, ℓ_i and h_i are the label, loss function and the classifier for task $\mathcal{T}_i$, respectively, and $g(., \mathbf{w}_0)$ is the shared feature extractor for $T+1$ tasks.

Multi-task learning is often confused with meta-learning due to their similar nature extracting information from many tasks. However, the objective function of multi-task learning in (1.13) is a single-level optimisation for the shared parameter $\mathbf{w}_0$ and multiple task-specific classifier $\{\mathbf{w}_i\}_{i=1}^{T+1}$. It is, therefore, not as complicated as a bi-level optimisation seen in meta-learning as shown in (1.5). Furthermore, multi-task learning aims to solve a number of specific tasks known during training (referred to as target tasks), while meta-learning targets the generalisation for unseen tasks in the future.

1.3.4 Continual learning

Continual or *life-long learning* refers to a situation where a learning agent has access to a continuous stream of tasks available over time, and the number of tasks to be learnt is not pre-defined (Zhiyuan Chen and B. Liu, 2018; Parisi et al., 2019). The aim is to accommodate the knowledge extracted from one-time observed tasks to accelerate the learning of new tasks without catastrophically forgetting old tasks (French, 1999). In this sense, continual learning is very similar to meta-learning. However, continual learning most likely focuses on **systematic** design to acquire new knowledge in such a way that prevents interfering to the existing one, while meta-learning is more about **algorithmic** design to learn the new knowledge more efficiently. Hence, we cannot mathematically distinguish their differences as done in Subsections 1.3.1 to 1.3.3. Nevertheless, continual learning criteria, especially catastrophic forgetting, can be encoded into meta-learning objective to advance further continual learning performance (Al-Shedivat et al., 2018; Nagabandi et al., 2019).

1.4 Open questions and contributions

1.4.1 Reliable meta-learning

To solve the meta-learning problem defined in (1.5), the loss function must be defined explicitly. One of the simplest but most popular loss functions used in meta-learning is negative log-likelihood (NLL) – corresponding to maximum likelihood estimation. In addition, since the task environment $p(\mathcal{D}, f)$ and the data generation of each task $(\mathcal{D}_i^{(t)}, f_i)$ and $(\mathcal{D}_i^{(v)}, f_i)$ are unknown, the bi-level optimisation objective for meta-learning shown in (1.5) is carried out with empirical risk minimisation. In other words, the data used for training consists of a finite number of training tasks, where each task is associated with two data subsets $S_i^{(t)}$ and $S_i^{(v)}$. Although such assumptions make the optimisation in (1.5) tractable, they poses two major concerns that need to be addressed:

1. The usage of NLL in the learning objective results in a point estimate modelling, making meta-learning fragile and easily over-fitted to the training data.

2. The reliance on empirical risk minimisation without proper regularisation in the learning objective also increases the risk of over-fitting, especially when the number of training data within each task is small.

The potential consequences are catastrophic failures, such as poorly calibrated models and overfitting, especially in applications that require high reliability, such as medical analysis or autonomous vehicles. These might hinder the deployment of meta-learning in those practical applications.

To address such problematic issues, we propose to integrate probabilistic modelling and PAC-Bayes framework into the learning objective to make meta-learning more robust. In particular, our contributions can be summarised as follows:

- In Chapter 3, we employ variational inference to take the modelling uncertainty into account. Such probabilistic modelling approach results in an efficient training with state-of-the-art calibration errors in both regression and classification tasks.

- In Chapter 4, we use the PAC-Bayes framework to derive an upper-bound of errors induced on unseen tasks and unseen samples from each training task. The meta-learning models trained by optimising the newly-derived upper-bounds show a comparable performance with respect to some state-of-the-art meta-learning methods, both in regression and classification settings.

1.4.2 Effect of training tasks on meta-learning

Recent empirical studies have shown a large variation of prediction performance when evaluating the same meta-learning model on different testing tasks (Dhillon et al., 2019, Figure 1). This means that not all testing tasks are equally related to training tasks. However, the majority of existing meta-learning algorithms is based on the bi-level optimisation in (1.5), which minimises the uniformly-weighted validation losses of training tasks. This implicitly assumes that all tasks are evenly distributed, which might not be true in practice. This has, therefore, led to the following research questions:

1. How related are tasks in meta-learning? If such relatedness exists, could it be exploited to optimise the training and prediction of meta-learning methods?

2. If tasks are not distributed evenly, training meta-learning models with uniformly weighted tasks could potentially bias the models toward some frequently-observed training tasks. Could a re-weighting method be developed to avoid such bias?

We carry out two studies to answer the above questions as follows:

- In Chapter 5, we employ a graphical modelling method to model tasks in meta-learning where each task is assumed to be generated from a Gaussian mixture model. Such modelling allows to represent tasks by their corresponding mixture vectors in the latent "task-theme" simplex. The newly-obtained representation can then be used to quantify task similarity which, in turn, benefits the training of meta-learning through task selection. We demonstrate that the task selection outperforms some common baselines in a life-long learning setting.

- In Chapter 6, we cast the training of meta-learning problem to trajectory optimisation, and employ iterative Linear Quadratic Regulator – an optimisation method in optimal control – to re-weight tasks to train a meta-learning model. The experimental results show that the model trained on re-weighting criterion out-performs the one trained on uniformly-weighted approach.

1.5 Thesis outline

The upcoming chapters of the thesis are oraganised as follows:

- Chapter 2 discusses in detail existing works in meta-learning, including probabilistic meta-learning, PAC-Bayes meta-learning, task modelling and task re-weighting.
- Chapter 3 proposes a variational inference method for meta-learning.
- Chapter 4 presents the formulation of a PAC-Bayes upper-bound for meta-learning. Unlike prior works based on PAC-Bayes framework, the proposed upper-bound is derived for the bi-level optimisation in (1.5), not the single-level optimisation often observed in multi-task learning. Thus, it is more applicable for practical meta-learning.
- Chapter 5 contributes to task modelling for meta-learning by employing a graphical model based on the Gaussian latent Dirichlet allocation. The proposal follows a generative approach that model the data generation process, while existing works on similar topics rely on discriminative approaches that use task-specific classifier to represent the task of interest.
- Chapter 6 applies an optimal control method, and in particular trajectory optimisation, to re-weight tasks to train the meta-learning model of interest.
- Chapter 7 concludes the studies carried out in this thesis, including their significance and limitations.

Chapter 2

Literature review

This chapter reviews existing works in the literature that cover the three main research topics of this thesis: (i) probabilistic meta-learning, (ii) task representation and relatedness, and (iii) task weighting in meta-learning

2.1 General meta-learning

The history of meta-learning or learning to learn can be rooted back in 1990s with some early works relating to inductive bias in learning (Utgoff, 1986; Rendell et al., 1987; Schmidhuber, 1987). The main idea of these studies is to explicitly model the learning process using hierarchical structures consisting of two main levels: *base-level* and *meta-level* (Thrun and L. Pratt, 1998, Section 1.3). The base-level is the problem of learning task-specific models – corresponding to the lower-level optimisation in (1.5) – which is similar to single-task learning, while the meta-level is to learn the task-invariant properties of models – corresponding to the upper-level optimisation in (1.5). Research in meta-learning is taken more formally in 1995 when many works that conceptualised and formulated meta-learning were presented at the *Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems* NIPS workshop, providing a mathematical foundation for meta-learning. In these early stages of development, research in meta-learning mainly focused on the theoretical point of view, such as algorithm complexity in some small-scale applications.

Research in meta-learning has gradually progressed and taken off, especially for deep learning models where more data with faster computational hardware has become available. By employing deep neural network to model complex data, meta-learning has achieved remarkable results in learning more efficient optimiser (Andrychowicz et al., 2016; K. Li and Malik, 2017; Ravi and Larochelle, 2018), model initialisation (Finn et al., 2017) or metric-based encoding (Vinyals et al., 2016; Snell et al., 2017). Meta-learning has also shown state-of-the-art performances in several few-shot learning benchmarks where meta-learning can quickly adapt to a

new task with only a few training examples. Despite such promising results, there are many challenges to the current meta-learning research, such as overfitting due to the reliance on point estimation of the model of interest, or how to quantify task similarity and utilise such information to train a model faster and predict more accurately. In the following sections, we review many studies that address those issues. In particular, we review probabilistic meta-learning in Section 2.2, PAC-Bayes and statistical analysis for meta-learning in Section 2.3, task-modelling in Section 2.4, and task-weighting in Section 2.5.

2.2 Probabilistic meta-learning

As majority of existing meta-learning algorithms relies on point estimate, learning from few training examples could potentially lead to overfitting, making meta-learning fragile in some reliability-required applications such as medical analysis or self-driving cars. Probabilistic meta-learning is, therefore, introduced to include uncertainty into modelling to improve the robustness of meta-learning.

One simple probabilistic meta-learning is LLAMA (Grant et al., 2018), which employs the Laplace method to approximate the true posterior of meta-parameter by a multivariate normal distribution around the mode of the true posterior. However, due to the nature of the Laplace approximation, the desired covariance matrix is proportional to the inverse Hessian matrix, making LLAMA intractable for large-scale models, such as deep neural networks. Further approximation, such as Kronecker-factored approximate curvature, is used to reduce the computational and storage complexity to make LLAMA more tractable.

Parametric variational inference (VI) is another approach to include uncertainty into meta-learning with highly-scalable capacity. Two notable parametric VI meta-learning approaches are PLATIPUS (Finn et al., 2018) and ABML (Ravi and Beatson, 2019), where multivariate normal distributions with diagonal covariance matrices are used to approximate the true posterior of meta-parameter. Despite simplicity in formulation and efficiency in computation, both approaches can be criticised since a part of the training data is used twice: one in task adaptation and the other in meta-parameter update steps (corresponding to lower- and upper-level optimisations in (1.5)), potentially leading to overfitting.

Non-parametric VI meta-learning has also been investigated, such as BMAML (Yoon et al., 2018), which employs Stein Variational Gradient Descent to update its ensemble of models, called "particles". However, due to the nature of non-parametric approaches which requires many particles, BMAML might not scale well for large models, such as very deep neural networks, since it might require a significant amount of memory storage. Another probabilistic inference-based method

is Neural Process (Garnelo et al., 2018) that trains a neural network to approximate a Gaussian-Process-like distribution over functions to achieve uncertainty quantification in few-shot learning. However, the nature of cubic complexity w.r.t. the data size induced by Gaussian Process might limit the scalability of Neural Process, making it infeasible for large-scale datasets.

The probabilistic method proposed in Chapter 3 of this thesis, in contrast, is a natural extension of the point estimate meta-learning method MAML (Finn et al., 2017). Instead of using negative log-likelihood (NLL) to substitute the loss function in (1.5) to formulate MAML algorithm, the proposed method uses the variational version of NLL – often known as variational-free energy (Blundell et al., 2015) – as the loss function, and follows the bi-level optimisation objective in (1.5) to learn an approximate posterior of the meta-parameter of interest. The proposed method can, therefore, be considered as a meta version of *Bayes by backprop* (Blundell et al., 2015) in single-task learning. The proposed method also has many advantages compared to previous works in probabilistic meta-learning. Firstly, the proposed method does not need the point estimate of any term, but uses a probabilistic approach that includes uncertainty into the modelling. Secondly, the proposed method does not require to calculate or inverse large matrices, such as the Hessian matrix in LLAMA. Lastly, the proposed method is based on a parametric approach, and hence, it is efficient in terms of computation and storage.

2.3 PAC-Bayes meta-learning

As the development of meta-learning is progressed, it is important to investigate and analyse the generalisation of meta-learning, especially when facing tasks it has never seen during training. This line of work is often relied on PAC-Bayes framework, where the true error is bounded above by an empirical error plus some regularisation.

There have been existing works that employ PAC-Bayes learning to upper-bound the true error of meta-learning with certain confidence level (Pentina and Lampert, 2014; Amit and Meir, 2018). In those works, the meta-parameter is often modelled as the regularisation shared across all tasks. In addition, these previous works rely on a train-train setting (Bai et al., 2021) without splitting the dataset of each task into a support and a query subset as in Subsection 1.1.4, resulting in a single-level optimisation objective. In contrast, the method proposed in Chapter 4 uses a meta-parameter to model the initialisation shared across all tasks. In addition, our proposed method follows the train-validation split, resulting in the bi-level optimisation objective shown in (1.5). Due to these differences, there is a slight discrepancy between our proposed method and the existing works in the literature.

Another closely-related work is *exponentially weighted aggregation for lifelong*

learning (EWA-LL) (Alquier, Pontil et al., 2017). In EWA-LL, each task-specific model is decomposed into a shared feature extractor and a task-specific classifier. Moreover, the setting of EWA-LL follows the train-train meta-learning approach, making the algorithm analogous to multi-task learning. The PAC-Bayes meta-learning method proposed in Chapter 4, in contrast, considers each task-specific model as an adapted or fine-tuned version of the meta model. The proposed method also follows a train-validation meta-learning approach with the bi-level optimisation objective as shown in (1.5).

This line of research also has a connection to the statistical analysis of meta-learning that proves generalisation upper-bound for meta-learning algorithms (Maurer and Jaakkola, 2005; Maurer et al., 2016). Some recent works include the learning of the common regularisation that is used when adapting or fine-tuning on a specific task (Denevi et al., 2018; Denevi et al., 2019a; Denevi et al., 2019b; Denevi et al., 2020) to improve the performance of meta-learning algorithms in heterogeneous task environments, or analyse and optimise the regret induced by meta-learning algorithms in online settings (Khodak et al., 2019). Our work proposed in Chapter 4 differs from previous works at how the meta-parameter is modelled. In our case, the meta-parameter of interest is the model initialisation, and our goal is to learn a variational distribution for such parameter, while existing works consider different parameter, such as the shared L2 regularisation parameters, as meta-parameter, and often learn a point estimate for such meta-parameter.

2.4 Task similarity

This section reviews existing works that models tasks in meta-learning. The purpose of task modelling is to employ task representation to quantify task similarity or task relatedness to improve further the learning and performance of meta-learning methods.

In general, task-modelling can be categorised into two main approaches: discriminative and generative. The discriminative approach relies on training a task-specific model to solve that task, and employing the information provided by such model to represent that task (Achille et al., 2019). The generative modelling approach, in contrast, does not need to train task-specific model for each task, but models the data generation process of those tasks, and use such generation models to characterise tasks. To the best of our knowledge, the majority of research in task similarity follows the discriminative approach, while there is a limited or even zero number of works related to generative modelling approach. In the following, we review some relevant studies that model tasks or estimate task similarity.

The closest work relating to task modelling in meta-learning is Task2Vec (Achille

et al., 2019). In Task2Vec, a task can be represented by an embedding vector calculated from the Fisher information matrix of the task-specific model trained on that task. Such representation can then be used to calculate the "distance" or similarity between tasks. Comparing to Task2Vec, the probabilistic task modelling (PTM) proposed in Chapter 5 follows a probabilistic modelling approach where a mixture of Gaussian distributions is used to model the task generation process. The mixture vector in the Gaussian mixture model is then used to characterise the task of interest. The two methods are therefore different at their modelling mechanisms: Task2Vec is discriminative, while PTM is generative. Such difference provides an advantage of PTM over Task2Vec, which includes modelling uncertainty into the task representation. In addition, PTM is more efficient than Task2Vec at predicting task representation, since PTM only needs a single forward pass, while Task2Vec requires to re-train or fine-tune the whole task-specific classifier and calculate the Fisher information matrix for the task that needs to be presented.

Task similarity estimation has also been studied in the field of multi-task learning. Some remarkable examples in this area include task-clustering using k-nearest neighbours (Thrun and O'Sullivan, 1996), or modelling common prior between tasks as a mixture of distributions (Bakker and Heskes, 2003; Xue et al., 2007). Another approach is to formulate multi-task learning as a convex optimisation problem either to cluster tasks and utilise the clustering results to fast track the learning (Jacob et al., 2009), or to learn task relationship through task covariance matrices (Y. Zhang and Yeung, 2012). Other approaches provided theoretical guarantees when learning the similarity or relationship between tasks (Shui et al., 2019). Recently, the *taskonomy* project (Zamir et al., 2018) was conducted to carry out extensive experiments on 26 computer-vision tasks to empirically analyse the correlation between those tasks. Other works (Tran et al., 2019; C. V. Nguyen et al., 2020) take a slightly different approach by investigating the correlation of the label distributions between the tasks of interest to measure task similarity. One commonality among all studies above is their nature of discriminative approaches that rely on task-specific classifiers to quantify task relatedness. In contrast, the proposed PTM method explicitly models tasks following the generative approach without the help of any task-specific classifier, making it more efficient in training and prediction.

The proposed PTM method in Chapter 5 is also connected to finite mixture models (Pritchard et al., 2000), such as the *latent Dirichlet allocation* (Blei et al., 2003), which analyses and summarises text data in topic modelling, or categorises natural scenes in computer vision (F.-F. Li and Perona, 2005). LDA assumes that each document within a given corpus can be represented as a mixture of finite categorical distributions, where each categorical distribution is a latent topic shared across all documents. Training an LDA model or its variants on a large text corpus

is challenging, so several approximate inference techniques have been proposed, ranging from mean-field variational inference (VI) (Blei et al., 2003), collapsed Gibbs' sampling (Griffiths and Steyvers, 2004) and collapsed VI (Teh et al., 2007). Furthermore, several online inference methods have been developed to increase the training efficiency for large corpora (Canini et al., 2009; Hoffman et al., 2010; Foulds et al., 2013). PTM is slightly different from the modelling of the conventional LDA, where we do not use the data directly, but embed it into a latent space. In short, PTM is a combination of VAE (Kingma and Welling, 2014a) and LDA to model the dataset associated with a task. The proposed approach considers "word" as continuous data, instead of the discrete data represented by a bag-of-word vector generally used by LDA-based topic modelling methods. The resultant model in the embedding latent space is, therefore, similar to the Gaussian LDA (Das et al., 2015) for word embedding in topic modelling.

2.5 Task weighting in meta-learning

This section reviews some recent works in task weighting for meta-learning. The purpose of task weighting is to avoid the trained meta-learning models from overfitting to some frequently-observed tasks.

Re-weighting tasks in meta-learning has recently attracted much research interest. One notable recent work is TR-MAML (Collins et al., 2020) which places higher weights on tasks with larger validation losses to optimise performance for worst-case scenarios. However, due to being formulated as a min-max optimisation, TR-MAML is reported to be computationally prohibitive when the number of tasks is large, requiring to cluster tasks into a small number of clusters, limiting the practicality of the approach. Another work, α-MAML (Cai et al., 2020), provides an upper-bound on the distance between the weighted risk evaluated on training tasks to the expected risk on testing tasks. The re-weight factors can then be obtained to minimise that upper-bound, reducing the variance between training and testing tasks. In reinforcement learning, MWL-MAML (Xu et al., 2021) was recently proposed to employ meta-learning to learn the local optimal re-weight factor of each trajectory using a few gradient descent steps. The downside of MWL-MAML is the need of validation trajectories (or validation tasks in meta-learning) that are representative enough to learn those weights. Furthermore, TR-MAML and MWL-MAML rely on a single mini-batch of tasks to determine the weights without considering the dynamic effect of a sequence of mini-batches when training a meta-model, potentially rendering sub-optimal solutions. In contrast, our method proposed in Chapter 6 does not need to cluster tasks nor require additional set of validation tasks. In addition, our proposed method automates the dynamic calculation of task-weighting

through an optimisation over a sequence of mini-batches, allowing to obtain better local-optimal solutions outside of a single mini-batch of tasks.

This line of work is also motivated from the observation of large variation in terms of prediction performance made by meta-learning algorithms on various testing tasks (Dhillon et al., 2019, Figure 1), implying that the trained meta-model may be biased toward certain training tasks. Such observation may be rooted in task relatedness or task similarity which is a growing research topic in the field of transfer learning, especially multi-task learning as presented in Section 2.4. This suggests the design of a mechanism to re-weight the contribution of each training task to improve the performance of the meta-model of interest.

Furthermore, the method proposed in Chapter 6 is related to finite-horizon discrete-time trajectory optimisation or open-loop optimal control which has been well studied in the field of control and robotics. The objective is to minimise a cost function that depends on the states and actions in many consecutive time steps given the state-transition dynamics. Exact solution can be obtained for the simplest problem where the cost is quadratic and the dynamics is linear using linear quadratic regulator (LQR) (Anderson and Moore, 2007). For a general non-linear problem, approximate solutions can be found via iterative approaches, such as differential dynamic programming (DDP) (Jacobson and Mayne, 1970; Murray and SJ Yakowitz, 1984; Sidney Yakowitz and Rutherford, 1984) and iterative LQR (iLQR) (Todorov and W. Li, 2005; Tassa et al., 2012).

2.6 Summary

In this chapter, we reviewed work related to the four studies presented in this thesis. These discussions include: probabilistic methods for meta-learning, PAC-Bayes upper-bound to analyse the generalisation of meta-learning, task modelling to investigate task relatedness, and task-weighting to improve further the performance of meta-learning.

Chapter 3

Variational Bayesian meta-learning

This chapter is based on the following publication:

> Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2020). 'Uncertainty in model-agnostic meta-learning using variational inference'. In: *Winter Conference on Applications of Computer Vision*, pp. 3090–3100.

The study is slightly modified from its original published content to adapt to the notations presented in Chapter 1. We also add an addendum at the beginning of this chapter to discuss how to formulate the method proposed in this chapter using the bi-level optimisation framework presented in Section 1.2.

3.1 Addendum to the publication

The current derivation of MAML and VAMPIRE presented in the following part of the chapter is a probabilistic approach that models the meta-parameter of interest, θ, as the parameter of the prior of task-specific parameter $p(\mathbf{w}_i)$. For a more general formulation without modelling θ to relate to the prior $p(\mathbf{w}_i)$, one can also use the formulation framework presented in Section 1.2. This section is, therefore, dedicated to re-formulate MAML and VAMPIRE using the bi-level optimisation approach presented in Section 1.2.

Note that the general objective function of meta-learning is a bi-level optimisation shown in (1.5). Hence, one can derive different meta-learning algorithms by simply replacing the loss function ℓ in both levels.

3.1.1 Maximum likelihood estimation and MAML

In single-task learning, the simplest objective function is maximum likelihood estimation (MLE). Hence, the simplest meta-learning algorithm can be obtained by simply

replacing the loss function ℓ by the negative log-likelihood in MLE for both levels of the meta-learning objective in (1.5). This results in the following objective function:

$$\begin{aligned}&\min_{\theta} \frac{1}{T}\sum_{i=1}^{T}\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}} -\ln p\left(\mathbf{y}_{ik}^{(v)}\left|\mathbf{x}_{ik}^{(v)};\mathbf{w}_i^*(\theta)\right.\right)\\ &\text{s.t.: } \mathbf{w}_i^*(\theta)=\arg\min_{\mathbf{w}_i}\frac{1}{m_i^{(t)}}\sum_{j=1}^{m_i^{(t)}} -\ln p\left(\mathbf{y}_{ij}^{(t)}\left|\mathbf{x}_{ij}^{(t)};\mathbf{w}_i(\theta)\right.\right), \forall i\in\{1,\ldots,T\}.\end{aligned} \tag{3.1}$$

Furthermore, if gradient descent is used to optimise (3.1), the resulting learning algorithm resembles MAML.

3.1.2 Minimum variational-free energy and VAMPIRE-2

In single-task learning, variational inference is used to infer the model parameter of interest probabilistically by minimising variational-free energy (VFE) (Blundell et al., 2015). Hence, one can replace the loss function ℓ in both levels of (1.5) by VFE to obtain a probabilistic meta-learning algorithm:

$$\begin{aligned}&\min_{\theta} \frac{1}{T}\sum_{i=1}^{T}\sum_{j=1}^{m_i^{(v)}} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[-\ln p\left(\mathbf{y}_{ij}^{(v)}|\mathbf{x}_{ij}^{(v)};\mathbf{w}_i\right)\right] + \mathrm{KL}\left[q\left(\mathbf{w}_i;\lambda_i^*\right)\|p(\mathbf{w}_i)\right]\\ &\text{s.t.: } \lambda_i^*\left(\theta\right)=\arg\min_{\lambda_i}\sum_{j=1}^{m_i^{(t)}} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[-\ln p\left(\mathbf{y}_{ij}^{(t)}|\mathbf{x}_{ij}^{(t)};\mathbf{w}_i\right)\right] \mathrm{KL}\left[q\left(\mathbf{w}_i;\lambda_i\right)\|p(\mathbf{w}_i)\right],\\ &\quad \forall i\in\{1,\ldots,T\},\end{aligned} \tag{3.2}$$

where $\mathrm{KL}(\pi||\rho)$ denotes the Kullback-Leibler (KL) divergence between two distributions π and ρ.

The bi-level optimisation in (3.2) can, therefore, be considered as a "meta" version of *Bayes-by-backprop* (Blundell et al., 2015) in sample-based learning. Note that the objective function in (3.2) is different from VAMPIRE at the additional KL divergence term highlighted in blue. For the ease of differentiation, the meta-learning algorithm that solves the bi-level optimisation in (3.2) is called as VAMPIRE-2 to distinguish from VAMPIRE.

Statement of Authorship

Title of Paper	Uncertainty in model-agnostic meta-learning using variational inference
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2020). "Uncertainty in model-agnostic meta-learning using variational inference". In Winter Conference on Applications of Computer Vision, pp. 3090-3100

Principal Author

Name of Principal Author (Candidate)	Cuong Nguyen		
Contribution to the Paper	- Developed the conception of the paper - Formulated the objective function of the research problem - Implemented the proposed algorithm - Drafted and revised the paper		
Overall percentage (%)	70		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	20/10/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Thanh-Toan Do		
Contribution to the Paper	- Discussed and refined the conception of the paper - Verified the mathematical formulation of the research problem - Discussed the experiment setup and results - Wrote and revised the paper		
Signature		Date	20/10/2021

Name of Co-Author	Gustavo Carneiro		
Contribution to the Paper	- Discussed and refined the conception of the paper - Verified the mathematical formulation of the research problem - Suggested ideas to implement the proposed algorithm - Wrote and revised the paper		
Signature		Date	20/10/2021

Please cut and paste additional co-author panels here as required.

Abstract

We introduce a new, rigorously-formulated Bayesian meta-learning algorithm that learns a probability distribution of model parameter prior for few-shot learning. The proposed algorithm employs a gradient-based variational inference to infer the posterior of model parameters for a new task. Our algorithm can be applied to any model architecture and can be implemented in various machine learning paradigms, including regression and classification. We show that the models trained with our proposed meta-learning algorithm are well calibrated and accurate, with state-of-the-art calibration and classification results on three few-shot classification benchmarks (Omniglot, mini-ImageNet and tiered-ImageNet), and competitive results in a multi-modal task-distribution regression.

3.2 Introduction

Machine learning, in particular deep learning, has thrived during the last decade, producing results that were previously considered to be infeasible in several areas. For instance, outstanding results have been achieved in speech and image understanding (Hinton et al., 2012; Graves et al., 2013; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015), and medical image analysis (Havaei et al., 2017). However, the development of these machine learning methods typically requires a large number of training samples to achieve notable performance. Such requirement contrasts with the human ability of quickly adapting to new learning tasks using few "training" samples. This difference may be due to the fact that humans tend to exploit prior knowledge to facilitate the learning of new tasks, while machine learning algorithms often do not use any prior knowledge (e.g., training from scratch with random initialisation (Glorot and Bengio, 2010)) or rely on weak prior knowledge to learn new tasks (e.g., training from pre-trained models (Rosenstein et al., 2005)). This challenge has motivated the design of machine learning methods that can make more effective use of prior knowledge to adapt to new learning tasks using few training samples (Lake et al., 2015).

Such methods assume the existence of a latent distribution over classification or regression tasks that share a common structure. This common structure means that solving many tasks can be helpful to solve a new task, sampled from the same task distribution, even if it contains a limited number of training samples. For instance, in *multi-task learning* (Caruana, 1997), an agent simultaneously learns the shared representation of many related tasks and a main task that are assumed to come from the same domain. The extra information provided by this multi-task training tends to regularise the main task training, particularly when it contains few

training samples. In *domain adaptation* (Bridle and Cox, 1991; Ben-David et al., 2010), a learner transfers the shared knowledge of many training tasks drawn from one or several source domains to perform well on tasks (with small training sets) drawn from a target domain. *Bayesian learning* (F.-F. Li et al., 2006) has also been explored, where prior knowledge is represented by a probability density function on the parameters of the visual classes' probability models. In *learning to learn* or *meta-learning* (Schmidhuber, 1987; Thrun and L. Pratt, 1998), a meta-learner extracts relevant knowledge from many tasks learned in the past to facilitate the learning of new future tasks.

From the methods above, meta-learning currently produces state-of-the-art results in many benchmark few-shot learning datasets (Santoro et al., 2016; Ravi and Larochelle, 2018; Munkhdalai and H. Yu, 2017; Snell et al., 2017; Finn et al., 2017; Yoon et al., 2018; R. Zhang et al., 2018; Rusu et al., 2019). Such success can be attributed to the way meta-learning leverages prior knowledge from several training tasks drawn from a latent distribution of tasks, where the objective is to perform well on unseen tasks drawn from the same distribution. However, a critical issue arises with the limited amount of training samples per task combined with the fact that most of these approaches (Santoro et al., 2016; Vinyals et al., 2016; Ravi and Larochelle, 2018; Finn et al., 2017; Snell et al., 2017) do not try to estimate model uncertainty – this may result in overfitting. This issue has been recently addressed with Laplace approximation to estimate model uncertainty, involving the computationally hard estimation of a high-dimensional covariance matrix (Grant et al., 2018), and with variational Bayesian learning (Finn et al., 2018; Yoon et al., 2018) containing sub-optimal point estimate of model parameters and inefficient optimisation.

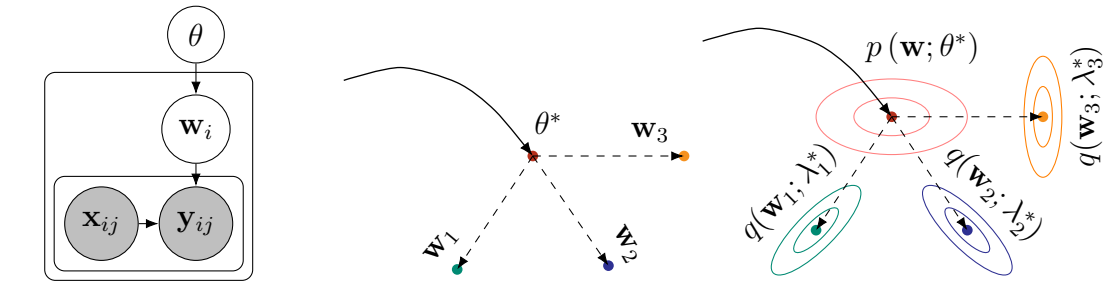
In this work, we propose a new variational Bayesian learning by extending model-agnostic meta-learning (MAML) (Finn et al., 2017) based on a rigorous formulation that is efficient and does not require any point estimate of model parameters. In particular, compared to MAML (Finn et al., 2017), our approach explores probability distributions over possible values of meta-parameters, rather than having a fixed value. Learning and prediction using our proposed method are, therefore, more robust due to the perturbation of learnt meta-parameters that coherently explains data variability. Our evaluation shows that the models trained with our proposed meta-learning algorithm is at the same time well calibrated and accurate, with competitive results in terms of Expected Calibration Error (ECE) and Maximimum Calibration Error (MCE), while outperforming state-of-the-art methods in some few-shot classification benchmarks (Omniglot, mini-ImageNet and tiered-ImageNet).

3.3 Related work

Meta-learning has been studied for a few decades (Schmidhuber, 1987; Naik and Mammone, 1992; Thrun and L. Pratt, 1998), and recently gained renewed attention with the use of deep learning methods. As meta-learning aims at the unique ability of learning how to learn, it has enabled the development of training methods with limited number of training samples, such as few-shot learning. Some notable meta-learning approaches include memory-augmented neural networks (Santoro et al., 2016), deep metric learning (Vinyals et al., 2016; Snell et al., 2017), learning how to update model parameters (Andrychowicz et al., 2016; Ravi and Larochelle, 2018) and learning good prior (Finn et al., 2017) using gradient update. These approaches have generated some of the most successful meta-learning results, but they lack the ability to estimate model uncertainty. Consequently, their performances may suffer in uncertain environments and real world applications.

Bayesian meta-learning techniques have, therefore, been developed to incorporate uncertainty into model estimation. Among those, MAML-based meta-learning has attracted much of research interest due to the straightforward use of gradient-based optimisation. LLAMA (Grant et al., 2018) uses Laplace method to extend the point estimates made in MAML to Gaussian distributions to improve the robustness of the trained model, but the need to estimate and invert the Hessian matrix makes this approach computationally challenging, particularly for large-scale models used in deep learning. Variational inference (VI) addresses such scalability issue – remarkable examples of VI-based methods are PLATIPUS (Finn et al., 2018), BMAML (Yoon et al., 2018) and the methods similar to our proposal, Amortised Bayesian Meta-Learning (ABML) (Ravi and Beatson, 2019) and VERSA (Gordon et al., 2019) [1]. However, PLATIPUS optimises the lower bound of data prediction, leading to the need to approximate a joint distribution between the task-specific and meta parameters. This approximation complicates the implementation and requires a point estimate of the task-specific parameters to reduce the complexity of the estimation of this joint distribution. Employing point estimate may, however, reduce its ability to estimate model uncertainty. BMAML uses a closed-form solution based on Stein Variational Gradient Descent (SVGD) that simplifies the task adaptation step, but it relies on the use of a kernel matrix, which increases its computational complexity. ABML uses the both train and validation subsets to update meta-parameters, potentially resulting in overfitting. VERSA takes a slightly different approach by employing an external neural network to learn the variational distribution for certain parameters, while keeping other parameters shared across all tasks. Another inference-based method

[1] ABML (Ravi and Beatson, 2019) and VERSA (Gordon et al., 2019) have been developed in parallel to our proposed VAMPIRE.



(a) Hierarchical graphical model

(b) MAML (reproduced from (Finn et al., 2017))

(c) VAMPIRE

Figure 3.1: (a) Hierarchical graphical model of the few-shot meta-learning, where a prior parameterised by θ is shared across many tasks; (b) and (c) Visualisation between MAML and VAMPIRE, respectively, where VAMPIRE extends both the deterministic prior $p(\mathbf{w}_i;\theta)$ and posterior $p(\mathbf{w}_i|S_i^{(t)},\theta)$ in MAML by using probabilistic distributions.

is Neural Process (Garnelo et al., 2018) that employs the train-ability of neural networks to model a Gaussian-Process-like distribution over functions to achieve uncertainty quantification in few-shot learning. However, due to the prominent weakness of Gaussian Process that suffers from cubic complexity to data size, this might limit the scalability of Neural Process and makes it infeasible for large-scale datasets.

Our approach, in contrast, is a straightforward extension of MAML, which uses VI to model the distributions of task-specific parameters and meta-parameters, where we do not require the use of point estimate of any term, nor do we need to compute Hessian or kernel matrices or depend on an external network. Our proposed algorithm can be considered a rigorous and computationally efficient Bayesian meta-learning algorithm. A noteworthy non-meta-learning method that employs Bayesian methods is the neural statistician (Edwards and Storkey, 2017) that uses an extra variable to model data distribution within each task, and combines that information to solve few-shot learning problems. Our proposed algorithm, instead, does not introduce additional parameters, while still being able to extract relevant information from a small number of examples.

3.4 Methodology

In this section, we first define and formulate the few-shot meta-learning problem. We then describe MAML, derive our proposed algorithm, and mention the similarities and differences between our method and recently proposed meta-learning methods that are relevant to our proposal.

3.4.1 Few-shot Learning Problem Setup

While conventional machine learning paradigm is designed to optimise the performance on a single task, few-shot learning is trained on a set of conditional independent and identically distributed (i.i.d.) tasks given meta-parameters. We employ the notation of "task environment" (Baxter, 2000), where tasks are sampled from an unknown task distribution $\mathcal{D}$ over a family of tasks. Each task $\mathcal{T}_i$ in this family is indexed by $i \in \{1, ..., T\}$ and consists of a support (or training) set $S_i^{(t)} = \{(\mathbf{x}_{ij}^{(t)}, \mathbf{y}_{ij}^{(t)})\}_{j=1}^{m_i^{(t)}}$ and a query (or validation) set $S_i^{(v)} = \{(\mathbf{x}_{ij}^{(v)}, \mathbf{y}_{ij}^{(v)})\}_{j=1}^{m_i^{(v)}}$. The aim of few-shot learning is to predict the output $\mathbf{y}_{ij}^{(v)}$ of the query input $\mathbf{x}_{ij}^{(v)}$ given the small support set for task $\mathcal{T}_i$ (e.g. $m_i^{(t)} \leq 20$). We rely on a Bayesian hierarchical model (Grant et al., 2018) to model the few-shot meta-learning problem. In the graphical model shown in Figure 3.1a, θ denotes the meta-parameters of interest, and $\mathbf{w}_i$ represents the task-specific parameters for task $\mathcal{T}_i$. One typical example of this modelling approach is MAML (Finn et al., 2017), where $\mathbf{w}_i$ are the neural network weights adapted to task $\mathcal{T}_i$ by performing truncated gradient descent using the data from the support set $S_i^{(t)}$ and the initialisation θ.

The objective function of few-shot learning is, therefore, to find a meta-learner, parameterised by θ, across tasks sampled from $\mathcal{D}$ that maximises the following log-likelihood:

$$\max_{\theta} \frac{1}{T} \sum_{i=1}^{T} \frac{1}{m_i^{(v)}} \sum_{j=1}^{m_i^{(v)}} \ln p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{(v)}, S_i^{(t)}, \theta\right), \tag{3.3}$$

where T denotes the number of tasks.

Each term of the predictive probability on the right hand side of (3.3) can be expanded by applying the sum rule of probability and lower-bounded by Jensen's inequality:

$$\begin{aligned} \ln p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{(v)}, S_i^{(t)}, \theta\right) &= \ln \sum_{\mathbf{w}_i} p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{(v)}, \mathbf{w}_i\right) p\left(\mathbf{w}_i \middle| S_i^{(t)}, \theta\right) \\ &= \ln \mathbb{E}_{p(\mathbf{w}_i|S_i^{(t)},\theta)}\left[p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{v}, \mathbf{w}_i\right)\right] \\ &\geq \mathbb{E}_{p(\mathbf{w}_i|S_i^{(t)},\theta)}\left[\ln p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{v}, \mathbf{w}_i\right)\right]. \end{aligned} \tag{3.4}$$

Hence, instead of maximising the log-likelihood in (3.3), we maximise the lower-bound of the corresponding log-likelihood shown in (3.4). Our alternative objective function can, therefore, be written as:

$$\max_{\theta} \frac{1}{T} \sum_{i=1}^{T} \frac{1}{m_i^{(v)}} \sum_{j=1}^{m_i^{(v)}} \mathbb{E}_{p(\mathbf{w}_i|S_i^{(t)},\theta)}\left[\ln p\left(\mathbf{y}_{ij}^{(v)} \middle| \mathbf{x}_{ij}^{v}, \mathbf{w}_i\right)\right]. \tag{3.5}$$

If each task-specific posterior, $p(\mathbf{w}_i|S_i^{(t)}, \theta)$, is well-behaved, we can apply Monte

Carlo to approximate the expectation in (3.5) by sampling model parameters $\mathbf{w}_i$ from $p(\mathbf{w}_i|S_i^{(t)}, \theta)$. Thus, depending on how the task-specific posterior $p(\mathbf{w}_i|S_i^{(t)}, \theta)$ is modelled and approximated, we can formulate different algorithms to solve the problem of few-shot learning. We review a deterministic method that is widely used in the literature in Subsection 3.4.2, and then present our proposed approach in Subsection 3.4.3.

3.4.2 Point Estimate - MAML

A simple way is to approximate $p(\mathbf{w}_i|S_i^{(t)}, \theta)$ by a Dirac delta function at its local mode:

$$p\left(\mathbf{w}_i \left| S_i^{(t)}, \theta \right.\right) = \delta\left(\mathbf{w}_i - \mathbf{w}_i^{\mathrm{MLE}}(\theta)\right), \tag{3.6}$$

where the local mode $\mathbf{w}_i^{\mathrm{MLE}}$ can be obtained by using maximum likelihood estimate:

$$\begin{aligned}\mathbf{w}_i^{\mathrm{MLE}}(\theta) &= \arg\max_{\mathbf{w}_i} \ln p\left(\mathbf{w}_i \left| S_i^{(t)}, \theta \right.\right) \\ &= \arg\max_{\mathbf{w}_i} \frac{1}{m_i^{(t)}} \sum_{j=1}^{m_i^{(t)}} \ln p\left(\mathbf{y}_{ij}^{(t)} | \mathbf{x}_{ij}^{(t)}, \mathbf{w}_i\right) + \text{const. w.r.t. } \mathbf{w}_i.\end{aligned} \tag{3.7}$$

If gradient descent is used with θ as the initialisation of $\mathbf{w}_i$, the local mode can be determined as:

$$\mathbf{w}_i^{\mathrm{MLE}}(\theta) = \mathbf{w}_i + \frac{\alpha}{m_i^{(t)}} \sum_{j=1}^{m_i^{(t)}} \nabla_{\mathbf{w}_i} \left[\ln p\left(\mathbf{y}_{ij}^{(t)} \left| \mathbf{x}_{ij}^{(t)}, \mathbf{w}_i \right.\right)\right] \Bigg|_{\mathbf{w}_i=\theta}, \tag{3.8}$$

where α is the learning rate, and the truncated gradient descent consists of a single step of (3.8) (the extension to a larger number of steps is trivial).

Given the point estimate assumption in (3.6), the maximisation on the lower-bound of the log-likelihood in (3.5) can be simplified to:

$$\max_{\theta} \frac{1}{T} \sum_{i=1}^{T} \frac{1}{m_i^{(v)}} \sum_{j=1}^{m_i^{(v)}} \ln p\left(\mathbf{y}_{ij}^{(v)} \left| \mathbf{x}_{ij}^{(v)}, \mathbf{w}_i^{\mathrm{MLE}}(\theta) \right.\right). \tag{3.9}$$

Maximising the resultant lower-bound in (3.9) w.r.t. θ by gradient-based optimisation represents the MAML algorithm (Finn et al., 2017). This derivation also explains the intuition behind MAML, which finds a good initialisation of model parameters as illustrated in Figure 3.1b.

3.4.3 Gradient-based Variational Inference

In contrast to the deterministic method presented in Subsection 3.4.2, we use a variational distribution $q(\mathbf{w}_i;\lambda_i)$, parameterised by λ_i – a function of $S_i^{(t)}$ and θ, to approximate the true task-specific posterior $p(\mathbf{w}_i|S_i^{(t)},\theta)$. In variational inference, $q(\mathbf{w}_i;\lambda_i)$ can be obtained by minimising the following Kullback-Leibler (KL) divergence:

$$\begin{aligned}
\lambda_i^* &= \arg\min_{\lambda_i} \mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)\,\middle\|\, p\left(\mathbf{w}_i\,\middle|S_i^{(t)},\theta\right)\right] \\
&= \arg\min_{\lambda_i} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[\ln q(\mathbf{w}_i;\lambda_i) - \ln p\left(\mathbf{w}_i\,\middle|S_i^{(t)},\theta\right)\right] \\
&= \arg\min_{\lambda_i} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\Bigg[\ln q(\mathbf{w}_i;\lambda_i) - \ln p(\mathbf{w}_i) \\
&\quad -\frac{1}{m_i^{(t)}}\sum_{j=1}^{m_i^{(t)}} \ln p\left(\mathbf{y}_{ij}^{(t)}\,\middle|\mathbf{x}_{ij}^{(t)},\mathbf{w}_i\right) + \ln p\left(\mathbf{y}_{ij}^{(t)}\,\middle|\mathbf{x}_{ij}^{(t)}\right)\Bigg] \\
&= \arg\min_{\lambda_i} \underbrace{\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)\|p\left(\mathbf{w}_i\right)\right] - \frac{1}{m_i^{(t)}}\sum_{j=1}^{m_i^{(t)}} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[\ln p\left(\mathbf{y}_{ij}^{(t)}\,\middle|\mathbf{x}_{ij}^{(t)},\mathbf{w}_i\right)\right]}_{\mathsf{L}\left(S_i^{(t)},\lambda_i(\theta)\right)}. \qquad (3.10)
\end{aligned}$$

The resulting cost function $\mathsf{L}(S_i^{(t)},\lambda_i(\theta))$ in (3.10) is often known as the variational free energy (VFE). The first term of VFE can be considered as a regularisation that penalises the difference between the prior $p\left(\mathbf{w}_i\right)$ and the approximated posterior $q(\mathbf{w}_i;\lambda_i)$, while the second term is referred as likelihood cost. Exactly minimising the cost function in (3.10) is computationally challenging, so gradient descent is used with θ as the initialisation of λ_i:

$$\lambda_i^* = \lambda_i - \alpha\nabla_{\lambda_i}\left[\mathsf{L}\left(S_i^{(t)},\lambda_i(\theta)\right)\right]\Big|_{\lambda_i=\theta}, \qquad (3.11)$$

where α is the learning rate, and only one gradient-descent step is carried out to simplify the analysis. Note that the extension for multiple gradient-descent steps can straight-forwardly be applied.

Given the approximated posterior $q(\mathbf{w}_i;\lambda_i^*)$ with parameter λ_i^* obtained according to (3.11), we can calculate and optimise the lower-bound of the log-likelihood of interest shown in (3.5) to find a local-optimal meta-parameter θ.

In Bayesian statistics, the prior $p\left(\mathbf{w}_i\right)$ represents a modelling assumption, and the approximated posterior $q(\mathbf{w}_i;\lambda_i)$ is a flexible function that can be adjusted to achieve a good trade-off between performance and complexity. For simplicity, we assume that both $p\left(\mathbf{w}_i\right)$ and $q(\mathbf{w}_i;\lambda_i)$ are Gaussian distributions with diagonal covariance

Algorithm 2 VAMPIRE training

task distribution $(\mathcal{D}, f)$
hyper-parameters: T, L_t, L_v, α and γ
initialise θ
while the cost in (3.5) is not converged **do**
  sample a mini-batch of tasks $(S_i^{(t)}, S_i^{(v)}) \sim (\mathcal{D}, f)$, $i = 1:T$
  for each task $\mathcal{T}_i$ **do**
    $\lambda_i \leftarrow \theta$
    draw L_t samples $\hat{\mathbf{w}}_i^{(l_t)} \sim q(\mathbf{w}_i; \lambda_i)$, $l_t = 1:L_t$
    calculate $\mathsf{L}(S_i^{(t)}, \lambda_i)$ using Monte Carlo to approximate $\mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}$
    obtain $\lambda_i^* = \lambda_i - \alpha \boldsymbol{\nabla}_{\lambda_i} [\mathsf{L}(S_i^{(t)}, \lambda_i(\theta))]\Big|_{\lambda_i=\theta}$ ▷ Eq (3.11)
    draw L_v samples $\hat{\mathbf{w}}_i^{(l_v)} \sim q(\mathbf{w}_i; \lambda_i^*)$, $l_v = 1:L_v$
    calculate the validation loss $\mathsf{L}(S_i^{(v)}, \lambda_i^*)$
  end for
  update meta-parameter: $\theta \leftarrow \theta + \frac{\gamma}{T} \sum_{i=1}^{T} \boldsymbol{\nabla}_\theta [\mathsf{L}(S_i^{(v)}, \lambda_i^*)]$
end while

matrices:

$$\begin{cases} p(\mathbf{w}_i) & = \mathcal{N}[\mathbf{w}_i | \boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta = \text{diag}(\boldsymbol{\sigma}_\theta^2)] \\ q(\mathbf{w}_i; \lambda_i) & = \mathcal{N}\left[\mathbf{w}_i | \boldsymbol{\mu}_{\lambda_i}, \boldsymbol{\Sigma}_{\lambda_i} = \text{diag}\left(\boldsymbol{\sigma}_{\lambda_i}^2\right)\right], \end{cases} \tag{3.12}$$

where $\boldsymbol{\mu}_\theta, \boldsymbol{\mu}_{\lambda_i}, \boldsymbol{\sigma}_\theta, \boldsymbol{\sigma}_{\lambda_i} \in \mathbb{R}^d$, with d denoting the number of model parameters, and the operator diag(.) returns a diagonal matrix of the vector in its input parameter.

Given the prior $p(\mathbf{w}_i)$ and the posterior $q(\mathbf{w}_i; \lambda_i)$ in (3.12), we can compute the KL divergence of VFE shown in (3.10) by using either Monte Carlo sampling or a closed-form solution. According to (Blundell et al., 2015), sampling model parameters from the approximated posterior $q(\mathbf{w}_i; \lambda_i)$ to compute the KL divergence term and optimise the cost function in (3.10) does not perform better or worse than using the closed-form of the KL divergence between two Gaussian distributions. Therefore, we employ the closed-form formula of the KL divergence to speed up the training process.

For numerical stability, we parameterise the standard deviation point-wisely as $\sigma = \exp(\rho)$ when performing gradient update for the standard deviations of model parameters. The meta-parameters $\theta = (\boldsymbol{\mu}_\theta, \exp(\boldsymbol{\rho}_\theta))$ are the initial mean and standard deviation of neural network weights, and the variational parameters $\lambda_i = (\boldsymbol{\mu}_{\lambda_i}, \exp(\boldsymbol{\rho}_{\lambda_i}))$ are the optimised mean and standard deviation of those network weights adapted to task $\mathcal{T}_i$. We also implement the re-parameterisation trick (Kingma and Welling, 2014a) when sampling the network weights from the approximated posterior to compute the expectation of the data log-likelihood in (3.10):

$$\mathbf{w}_i = \boldsymbol{\mu}_{\lambda_i} + \epsilon \odot \exp(\boldsymbol{\rho}_{\lambda_i}), \tag{3.13}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I}_d)$, and $\odot$ is the element-wise multiplication.

After obtaining the variational parameters λ_i in (3.11), we can apply Monte Carlo approximation by sampling L_v sets of model parameters from the approximated posterior $q(\mathbf{w}_i; \lambda_i^*)$ to calculate and optimise the objective function in (3.5) w.r.t. θ. This approach leads to the general form of our proposed algorithm, named Variational Agnostic Modelling that Performs Inference for Robust Estimation (VAMPIRE), shown in Algorithm 2.

3.4.4 Differentiating VAMPIRE and Other Bayesian Meta-learning Methods

VAMPIRE is different from the "probabilistic MAML" - PLATIPUS (Finn et al., 2018) in several ways. First, PLATIPUS uses VI to approximate the joint distribution $p(\mathbf{w}_i, \theta | S_i^{(t)}, S_i^{(v)})$, while VAMPIRE uses VI to approximate the task-specific posterior $p(\mathbf{w}_i | S_i^{(t)}, \theta)$. To handle the complexity of sampling from a joint distribution, PLATIPUS relies on the same point estimate of the task-specific posterior as MAML, as shown in (3.6). Second, to adapt to task $\mathcal{T}_i$, PLATIPUS learns only the mean, without varying the variance. In contrast, VAMPIRE learns both $\boldsymbol{\mu}_\theta$ and Σ_θ for each task $\mathcal{T}_i$. Lastly, when adapting to a task, PLATIPUS requires 2 additional gradient update steps, corresponding to steps 7 and 10 of Algorithm 1 in (Finn et al., 2018), while VAMPIRE needs only 1 gradient update step as shown in step 10 of Algorithm 2. Hence, VAMPIRE is based on a simpler formulation that does not rely on any point estimate, and it is also more flexible and efficient because it allows all meta-parameters to be learnt while performing less gradient update steps.

VAMPIRE is also different from the PAC-Bayes meta-learning method designed for multi-task learning (Amit and Meir, 2018) at the relation between the shared prior $p(\mathbf{w}_i; \theta)$ and the variational task-specific posterior $q(\mathbf{w}_i; \lambda_i)$. While the PAC-Bayes meta-learning method does not relate the "posterior" to the "prior" as in the standard Bayesian analysis, VAMPIRE relates these two probabilities through a likelihood function by performing a fixed number of gradient updates as shown in (3.11). Due to this discrepancy, the PAC-Bayes meta-learning needs to maintain all the task-specific posteriors, requiring more memory storage, consequently resulting in an un-scalable approach, especially when the number of tasks is very large. In contrast, VAMPIRE learns only the shared prior, and hence, is a more favourable method for large-scaled applications, such as few-shot learning.

VAMPIRE is also different from the PAC-Bayes meta-learning method designed for multi-task learning (Amit and Meir, 2018) at the relation between the shared prior $p(\mathbf{w}_i; \theta)$ and the variational task-specific posterior $q(\mathbf{w}_i; \lambda_i)$. While the PAC-Bayes meta-learning method does not relate the "posterior" to the "prior" as in the

standard Bayesian analysis, VAMPIRE relates these two probabilities through a likelihood function by performing a fixed number of gradient updates as shown in (3.11). Due to this discrepancy, the PAC-Bayes meta-learning needs to maintain all the task-specific posteriors, requiring more memory storage, consequently resulting in an un-scalable approach, especially when the number of tasks is very large. In contrast, VAMPIRE learns only the shared prior, and hence, is a more favourable method for large-scaled applications, such as few-shot learning.

Our proposed algorithm is different from BMAML (Yoon et al., 2018) at the methods used to approximate task-specific posterior $p(\mathbf{w}_i|S_i^{(t)}, \theta)$: BMAML is based on SVGD, while VAMPIRE is based on a variant of amortised inference. Although SVGD is a non-parametric approach that allows a flexible variational approximation, its downsides are the computational complexity of kernel matrix, and high memory usage when increasing the number of particles. In contrast, our approach uses a straightforward VI using parametric functions, resulting in a simpler computational and memory-efficient approach. One advantage of BMAML compared to our method in Algorithm 2 is the use of Chaser Loss, which may be an effective way of preventing overfitting. Nevertheless, in principle, we can also implement the same loss for our proposed algorithm.

VAMPIRE is different from ABML (Ravi and Beatson, 2019) at the data subset used to update the meta-parameters θ: whole data set $S_i = S_i^{(t)} \cup S_i^{(v)}$ of task $\mathcal{T}_i$ in ABML versus only the query subset $S_i^{(v)}$ in VAMPIRE. This discrepancy is due to the differences in the objective function. In particular, ABML maximises the lower bound of marginal likelihood, while VAMPIRE maximises the predictive probability in (3.3). Moreover, when deriving a lower bound of marginal log-likelihood using VI (Ravi and Beatson, 2019, Derivation right before Eq. (1)), the variational distribution q must be strictly greater than zero for all θ and variational parameters. The assumption that approximates the variational distribution $q(\theta; \psi)$ by a Dirac delta function made in Amortised ML (Ravi and Beatson, 2019, Eq. (4)) is, therefore, arguable.

Another Bayesian meta-learning approach similar to VAMPIRE is VERSA (Gordon et al., 2019). The two methods are different at the methods modelling the parameters of interest θ. VAMPIRE relies on gradient update to relate the prior and posterior through likelihood function, while VERSA is based on an amortisation network to output the parameters of the variational distributions. To scale up to deep neural network models, VERSA models only the parameters of the last fully connected layer, while leaving other parameters as point estimates that are shared across all tasks. As a result, VAMPIRE is more flexible since it does not need to define which parameters are shared or not shared, nor does it require any additional network.

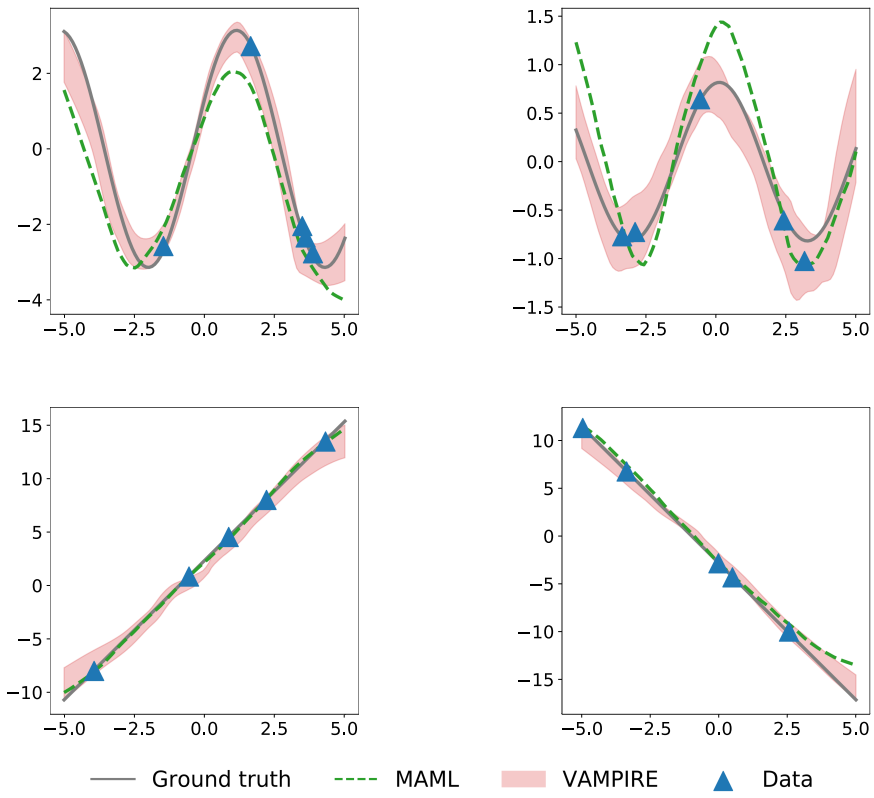


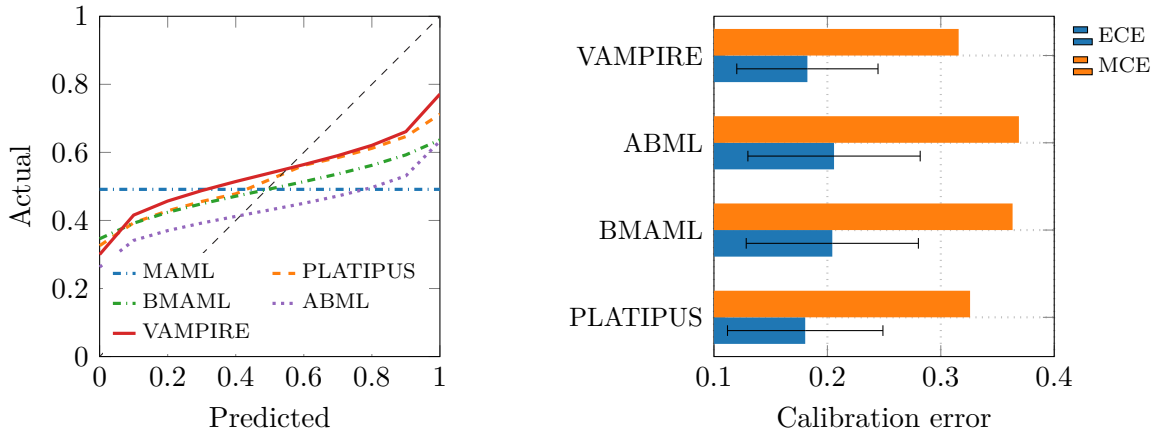
Figure 3.2: Qualitative results on multi-modal data – half of the tasks are generated from sinusoidal functions, and the other half are from linear functions with visualisation of MAML and VAMPIRE, where the shaded area is the prediction made by VAMPIRE $\pm$ $2\times$ standard deviation.

3.5 Experiments

The goal of our experiments is to present empirical evaluation of VAMPIRE compared to state-of-art meta-learning approaches. The experiments include both regression and few-shot classification problems often used to benchmark meta-learning algorithms.

3.5.1 Few-shot regression

VAMPIRE is evaluated using the popular multi-modal task distribution described in Subsection 3.5.1, where half of the tasks generated are sinusoidal functions, and the other half are linear functions (Finn et al., 2018). The model used in this experiment is a 3-hidden fully connected neural network with 100 hidden units per each hidden layer. Output from each layer is activated by ReLU without batch normalisation. Adam is used as the optimiser to optimise the meta-parameter of interest. Please refer to Table A.1 in the Appendix for the details of hyperparameters used. The performance of VAMPIRE is then compared to its closely-related meta-learning algorithm MAML.



(a) Reliability diagram (b) Calibration errors

Figure 3.3: Quantitative results on sinusoidal-linear 5-shot regression problem: (a) reliability diagram of various meta-learning methods averaged over 1000 tasks, and (b) ECE and MCE of the Bayesian meta-learning methods.

The qualitative results in Figure 3.2 show that VAMPIRE can effectively reason which underlying function generates the training data points as the predictions are all sinusoidal or linear. In addition, VAMPIRE is able to vary the prediction variance, especially when there is more uncertainty in the training data. In contrast, due to the deterministic nature, MAML can only output a single value at each input.

To quantitatively compare the performance between VAMPIRE and other few-shot meta-learning methods, a reliability diagram based on the quantile calibration for regression (Song et al., 2019) is constructed. A model is perfectly calibrated when its predicted probability equals to the actual probability, resulting in a curve that is well-aligned with the diagonal $y = x$. Some popular few-shot meta-learning methods are re-implemented, trained until convergence, and their reliability diagrams for 1000 tasks are plotted in Figure 3.3a. To have a fair comparison, BMAML is trained without Chaser Loss, and ABML is trained with a uniform hyper-posterior. Due to the deterministic nature, the performance curve of MAML is presented as a horizontal line. In addition, the expected calibration error (ECE), which averages the absolute errors measuring from the diagonal, and the maximum calibration error (MCE), which returns the maximum of absolute errors, are plotted in Figure 3.3b to further quantitatively compare.

Overall, in terms of ECE and MCE, the model trained with VAMPIRE is better than BMAML and ABML, while competitive with PLATIPUS. The performance of BMAML could be higher if more particles and Chaser Loss are used. Another observation is that ABML has slightly lower performance than MAML, although the training procedures of the two methods are very similar. This might be due to overfitting induced by using the whole training data subset $S_i^{(t)} \cup S_i^{(v)}$ to update meta-parameter, while MAML and VAMPIRE use only the validation data subset

$S_i^{(v)}$ to learn the meta-parameters.

3.5.2 Few-shot classification

The experiments in this sub-section are based on the N-way k-shot learning task, where a meta learner is trained on many related tasks containing N classes and small training sets of k samples for each class. Omniglot (Lake et al., 2015), mini-ImageNet (Vinyals et al., 2016; Ravi and Larochelle, 2018) and tiered-ImageNet (Ren et al., 2018) are the three datasets used to benchmark the results against the state-of-the-art in the few-shot classification setting. Note that the experiments on tiered-ImageNet is carried with input as features extracted by a residual network that was pre-trained on data and classes from training meta-set (Rusu et al., 2019, Section 4.2.2) instead of using raw image data to have a consistent comparison to existing results reported in the literature.

For Omniglot and mini-ImageNet, the same network architecture of state-of-the-art methods (Vinyals et al., 2016; Finn et al., 2017; Ravi and Larochelle, 2018) is employed to fairly compare different meta-learning algorithms. The network consists of 4 hidden convolution modules, each containing 64 filters of size 3-by-3, followed by batch normalisation (Ioffe and Szegedy, 2015), ReLU activation, and a 2-by-2 strided convolution. For the mini-ImageNet, the strided convolution is replaced by a 2-by-2 max-pooling layer, and only 32 filters are used on each convolution layer to avoid over-fitting (Finn et al., 2017; Ravi and Larochelle, 2018). For tiered-ImageNet with extracted features, a 2-hidden-layer fully-connected network with 128 and 32 hidden units is used as a backbone. The evaluation subset $S_i^{(v)}$ has 15 samples per class, resulting in $m_i^{(v)} = 15 \times N$ samples to be consistent with the previous works in the literature (Finn et al., 2017; Ravi and Larochelle, 2018). The training is carried out by using Adam. The learning rate of the meta-parameters θ is set to be $\gamma = 10^{-3}$, and decayed by a factor of 0.99 after every 10,000 tasks. Other hyperparameters used are specified in Table A.3 in the Appendix. The numbers of ensemble models L_t and L_v sampled from the variational posterior $q(\mathbf{w}_i; \lambda_i)$ are selected to fit into the memory of one Nvidia 1080 Ti GPU. Higher values of L_t and L_v are desirable to achieve a better Monte Carlo approximation, but the trade-off is slower computation per task.

A 2-hidden fully connected layer with 128 and 32 hidden units is used in the experiments with features extracted from mini-ImageNet and tiered-ImageNet presented in Table A.4, and the bottom part of Table 3.2, respectively (Rusu et al., 2019). The learning rate α is set as 0.01 and 5 gradient updates were carried out. The learning rate for meta-parameters was $\gamma = 0.001$.

The N-way k-shot classification accuracy measured on Omniglot, and mini-

Table 3.1: Few-shot classification accuracy (in percentage) on Omniglot, tested on 1000 tasks and reported with 95% confidence intervals. The results of VAMPIRE are competitive to the state-of-the-art baselines which are carried out on a standard 4-convolution-layer neural networks. The top of the table contains methods trained on the original split defined in (Lake et al., 2015), while the middle part contains methods using a standard 4-layer CNN trained on random train-test split. The bottom part presents results of different methods using different network architectures, or requiring external modules and additional parameters trained on random split. Note that the Omniglot results on random split cannot be fairly compared.

	5-way		**20-way**	
	1-shot	5-shot	1-shot	5-shot
Omniglot (Lake et al., 2015) - **Original Split, standard 4-layer CNN**				
MAML	$\mathbf{96.68 \pm 0.57}$	98.33 ± 0.22	84.38 ± 0.64	$\mathbf{96.32 \pm 0.17}$
VAMPIRE	96.27 ± 0.38	$\mathbf{98.77 \pm 0.27}$	$\mathbf{86.60 \pm 0.24}$	96.14 ± 0.10
Omniglot (Lake et al., 2015) - **Random Split, standard 4-layer CNN**				
Matching nets (Vinyals et al., 2016)	98.1	98.9	93.8	98.5
Prototypical nets (Snell et al., 2017) †	98.8	99.7	96.0	**98.9**
MAML (Finn et al., 2017)	$\mathbf{98.7 \pm 0.4}$	$\mathbf{99.9 \pm 0.1}$	$\mathbf{95.8 \pm 0.3}$	$\mathbf{98.9 \pm 0.2}$
VAMPIRE	98.43 ± 0.19	99.56 ± 0.08	93.20 ± 0.28	98.52 ± 0.13
Omniglot (Lake et al., 2015) - **Random Split, non-standard CNNs**				
Siamese nets (Koch et al., 2015)	97.3	98.4	88.2	97.0
Neural statistician (Edwards and Storkey, 2017)	98.1	99.5	93.2	98.1
Memory module (Kaiser et al., 2017)	98.4	99.6	95.0	98.6
Relation nets (Sung et al., 2018)	99.6 ± 0.2	$\mathbf{99.8 \pm 0.1}$	97.6 ± 0.2	$\mathbf{99.1 \pm 0.1}$
VERSA (Gordon et al., 2019)	$\mathbf{99.70 \pm 0.20}$	99.75 ± 0.13	$\mathbf{97.66 \pm 0.29}$	98.77 ± 0.18

† Trained with 60-way episodes

ImageNet, tiered-ImageNet data sets are shown in Tables 3.1 and 3.2, respectively. Overall, the results of VAMPIRE are competitive to the state-of-the-art methods that use the same network architecture (Vinyals et al., 2016; Finn et al., 2017; Ravi and Larochelle, 2018).

On Omniglot, the results of VAMPIRE on a random train-test split are competitive in most scenarios. VAMPIRE outperforms some previous works in few-shot learning, such as siamese networks (Koch et al., 2015), matching networks (Vinyals et al., 2016) and memory models (Kaiser et al., 2017), although they are designed with a focus on few-shot classification. The result of VAMPIRE on the 20-way 1-shot is slightly lower than prototypical networks (Snell et al., 2017) and VERSA (Gordon et al., 2019), but prototypical networks need more classes (higher "way") per training episode to obtain advantageous results and VERSA requires an additional amortised networks to learn the parameters of variational distributions. The results of VAMPIRE are also slightly lower than MAML, potentially due to the difference of train-test split. To obtain a fair comparison, we run the public code provided by MAML's authors, and measure its accuracy on the original split suggested in (Lake et al., 2015). Using this split, VAMPIRE achieves competitive performance, and outperforms MAML in some cases.

On mini-ImageNet, VAMPIRE outperforms all reported methods that use the standard 4-layer CNN architecture on the 1-shot tests, while being competitive on the 5-shot episodes. Prototypical Networks achieve a higher accuracy on the 5-shot tests due to, again, the use of extra classes during training. Although the current work does not aim to achieve the state-of-the-art results in few-shot learning, an additional experiment using input as features extracted by a residual network (Rusu et al., 2019, section 4.2.2) is carried out for reference purpose. The results, including the state-of-the-art methods that employ much deeper networks with various architectures, are presented in Table A.4. Note that deeper networks tend to reduce intra-class variation, resulting in a smaller gap of performance among many meta-learning methods (W.-Y. Chen et al., 2019).

On tiered-ImageNet, VAMPIRE outperforms many methods published previously by a large margin on both 1- and 5-shot settings.

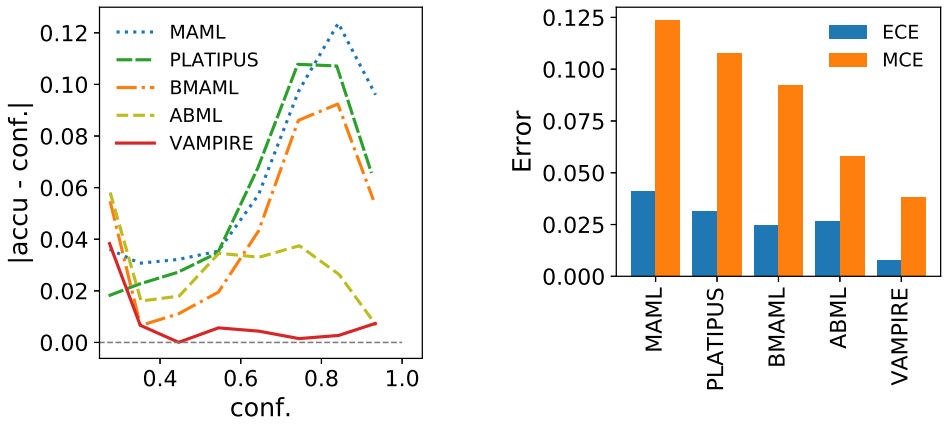
To evaluate the predictive uncertainty of the models trained with different meta-learning methods, we show in Figure 3.4a the "normalised" reliability diagrams (C. Guo et al., 2017) which presents the absolute errors averaged over many unseen tasks. A perfectly calibrated model will have its "normalised" values overlapped with the y-axis, indicating that the probability associated with the label prediction is the same as the true probability. To have a fair comparison, we train all the

[2]Trained with 30-way episodes for 1-shot classification and 20-way episodes for 5-shot classification
[3]Produced locally without Chaser loss

Table 3.2: The few-shot 5-way classification accuracy results (in percentage) of VAMPIRE averaged over 600 mini-ImageNet tasks and 5000 tiered-ImageNet tasks are competitive to the state-of-the-art methods.

	Mini-ImageNet	
	1-shot	5-shot
Standard 4-block CNN		
Matching nets (Vinyals et al., 2016)	43.56 ± 0.84	55.31 ± 0.73
Meta-learner LSTM (Ravi and Larochelle, 2018)	43.44 ± 0.77	60.60 ± 0.71
MAML (Finn et al., 2017)	48.70 ± 1.84	63.15 ± 0.91
Proto. nets (Snell et al., 2017) [2]	49.42 ± 0.78	$\mathbf{68.20 \pm 0.66}$
LLAMA (Grant et al., 2018)	49.40 ± 1.83	_
PLATIPUS (Finn et al., 2018)	50.13 ± 1.86	_
BMAML (Yoon et al., 2018)[3]	49.17 ± 0.87	64.23 ± 0.69
Amortised ML (Ravi and Beatson, 2019)	45.00 ± 0.60	_
VAMPIRE	$\mathbf{51.54 \pm 0.74}$	64.31 ± 0.74
	Tiered-ImageNet	
	1-shot	5-shot
Different settings and network architectures		
MAML (Y. Liu et al., 2018)	51.67 ± 1.81	70.30 ± 0.08
Prototypical Networks (Ren et al., 2018)	53.31 ± 0.89	72.69 ± 0.74
Relation Networks (Y. Liu et al., 2018)	54.48 ± 0.93	71.32 ± 0.78
Transductive Propagation Nets (Y. Liu et al., 2018)	57.41 ± 0.94	71.55 ± 0.74
LEO (Rusu et al., 2019)	66.33 ± 0.05	81.44 ± 0.09
MetaOptNet (Lee et al., 2019)	65.81 ± 0.74	81.75 ± 0.53
VAMPIRE	$\mathbf{69.87 \pm 0.29}$	$\mathbf{82.70 \pm 0.21}$

methods of interest under the same configuration, e.g. network architecture, number of gradient updates, while keeping all method-specific hyper-parameters the same as the reported values. Due to the constrain of GPU memory, BMAML is trained with only 8 particles, while PLATIPUS, Amortimised Meta-learner and VAMPIRE are trained with 10 Monte Carlo samples. According to the reliability graphs, the model trained with VAMPIRE shows a much better calibration than the ones trained with the other methods used in the comparison. To further evaluate, we compute the expected calibration error (ECE) and maximum calibration error (MCE) (C. Guo et al., 2017) of each models trained with these methods. Intuitively, ECE is the weighted average error, while MCE is the largest error. The results plotted in Figure 3.4b show that the model trained with VAMPIRE has smaller ECE and MCE compared to all the state-of-the-art meta-learning methods. The slightly low performance of ABML might be due to the usage of the whole task-specific dataset, potentially overfitting to the training data. Another factor contributed might be



(a) "Normalised" reliability diagram (b) ECE and MCE

Figure 3.4: (a) Uncertainty evaluation between different meta-learning methods using reliability diagrams, and (b) expected calibration error (ECE) and maximum calibration error (MCE), in which the evaluation is carried out on 5-way 1-shot setting for $\binom{20}{5} = 15504$ unseen tasks sampled from mini-ImageNet dataset.

the arguable Dirac-delta hyper-prior used, which can be also the cause for the low prediction accuracy shown in Table 3.2.

3.6 Summary

We introduce and formulate a new Bayesian algorithm used for few-shot meta-learning. The proposed algorithm, VAMPIRE, employs variational inference to optimise a well-defined cost function to learn a distribution of model parameters. The uncertainty, in the form of the learnt distribution, can introduce more variability into the decision made by the model, resulting in well-calibrated and highly-accurate prediction. The algorithm can be combined with different models that are trainable with gradient-based optimisation, and is applicable in regression and classification. We demonstrate that the algorithm can make reasonable predictions about unseen data in a multi-modal 5-shot learning regression problem, and achieve state-of-the-art calibration and classification results with only 1 or 5 training examples per class on public image data sets.

Chapter 4

PAC-Bayesian upper-bound for meta-learning

The content of this chapter was submitted at the time the thesis was examined, and then accepted to the IEEE Transaction on Pattern Analysis and Machine Intelligence with the following publication:

> Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2022). 'PAC-Bayes meta-learning with implicit task-specific posteriors'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. DOI: 10.1109/TPAMI.2022.3147798.

We note that Section 4.3, and in particular, Subsections 4.3.1 to 4.3.3, are largely overlapped with Section 1.1 presented in Chapter 1.

Statement of Authorship

Title of Paper	PAC-Bayesian meta-learning with implicit variational posterior
Publication Status	☐ Published ☐ Accepted for Publication ☑ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2021). "PAC-Bayesian meta-learning with implicit variational posterior". Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

Principal Author

Name of Principal Author (Candidate)	Cuong Nguyen		
Contribution to the Paper	- Developed the conception of the paper - Formulated the objective function - Implemented the proposed algorithm - Drafted the paper		
Overall percentage (%)	70		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	20/10/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Thanh-Toan Do		
Contribution to the Paper	- Discussed and refined the idea of the paper - Verified the mathematical formulation - Sugguested ideas to implement the proposed method - Wrote and revised the paper		
Signature		Date	20/10/2021

Name of Co-Author	Gustavo Carneiro		
Contribution to the Paper	- Discussed and refined the idea of the paper - Verified the mathematical formulation of the proposed method - Advised and discussed some difficulties of the implementation - Wrote and revised the paper		
Signature		Date	20/10/2021

Please cut and paste additional co-author panels here as required.

Abstract

We introduce a new and rigorously-formulated PAC-Bayes meta-learning algorithm that solves few-shot learning. Our proposed method extends the PAC-Bayes framework from a single-task setting to the meta-learning multiple-task setting to upper-bound the error evaluated on any, even unseen, tasks and samples. We also propose a generative-based approach to estimate the posterior of task-specific model parameters more expressively compared to the usual assumption based on a multivariate normal distribution with a diagonal covariance matrix. We show that the models trained with our proposed meta-learning algorithm are well-calibrated and accurate, with state-of-the-art calibration errors while still being competitive on classification results on few-shot classification (mini-ImageNet and tiered-ImageNet) and regression (multi-modal task-distribution regression) benchmarks.

4.1 Introduction

Ounique ability of humans is to quickly learn new tasks with only a few *training* examples. This is due to the fact that humans tend to exploit prior experience to facilitate the learning of new tasks. Such exploitation is markedly different from conventional machine learning approaches, where no prior knowledge (e.g. training from scratch with random initialisation) (Glorot and Bengio, 2010), or weak prior knowledge (e.g., fine-tuning from pre-trained models) (Rosenstein et al., 2005) are employed to learn a new task. This motivates the development of novel learning algorithms that can effectively encode the knowledge learnt from training tasks, and exploit that knowledge to quickly adapt to future tasks (Lake et al., 2015).

Prior knowledge can be helpful for future learning only if all tasks are assumed to be distributed according to a latent task distribution. Learning this latent distribution is, therefore, useful for solving an unseen task, even if the task contains a limited number of training examples. Many approaches have been proposed and developed to achieve this goal, namely: *multi-task learning* (Caruana, 1997), *domain adaptation* (Bridle and Cox, 1991; Ben-David et al., 2010) and *meta-learning* (Schmidhuber, 1987; Thrun and L. Pratt, 1998). Among these, meta-learning has flourished as one of the most effective methods due to its ability to leverage the knowledge learnt from many training tasks to quickly adapt to unseen tasks.

Recent advances in meta-learning have produced state-of-the-art results in many benchmarks of few-shot learning data sets (Santoro et al., 2016; Ravi and Larochelle, 2018; Munkhdalai and H. Yu, 2017; Snell et al., 2017; Finn et al., 2017; R. Zhang et al., 2018; Rusu et al., 2019). Learning from a few training examples is often difficult and easily leads to over-fitting, especially when no model uncertainty is taken into

account. This issue has been addressed by several recent probabilistic meta-learning approaches that incorporate model uncertainty into prediction, e.g., LLAMA (based on Laplace method) (Grant et al., 2018), or PLATIPUS (Finn et al., 2017), Amortised Bayesian Meta-learner (ABML) (Ravi and Beatson, 2019) and VERSA (Gordon et al., 2019) that use variational inference. However, these studies have not thoroughly investigated the errors evaluated on arbitrary tasks (including seen and unseen) sampled from the same task distribution and arbitrary samples generated from the same task. This results in limited theoretical generalisation guarantees. Moreover, most of these studies are based on variational functions that may not represent well the richness of the underlying distributions. For instance, a common choice for the variational posterior relies on a multivariate normal distribution with a diagonal covariance matrix, which can potentially worsen the prediction accuracy given its limited representability.

In this paper, we address the two problems listed above with the following technical novelties: (i) derivation of a rigorous meta-learning objective that upper-bounds the errors evaluated on any tasks and any samples of few-shot learning setting based on the PAC-Bayes framework, and (ii) proposal of a novel implicit modelling approach to expressively approximate the posterior of task-specific model parameters. Our evaluation shows that the models trained with our proposed meta-learning algorithm are at the same time well-calibrated and accurate, with state-of-the-art Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) in few-shot classification (mini-ImageNet and tiered-ImageNet) and regression (multi-modal task-distribution regression) benchmarks, while still being competitive in terms of classification accuracy.

4.2 Related Work

Our paper is related to probabilistic few-shot meta-learning techniques that have been developed to incorporate uncertainty into model estimation. LLAMA (Grant et al., 2018) employs the Laplace method to extend the deterministic estimation assumed in MAML (Finn et al., 2017) to a multivariate normal distribution. However, the need to estimate and invert the Hessian matrix of a loss function makes this approach computationally challenging for large-scale models, such as deep neural networks. Variational inference (VI) addresses such scalability issue – remarkable examples of VI-based methods are PLATIPUS (Finn et al., 2018), BMAML (Yoon et al., 2018), ABML (Ravi and Beatson, 2019) and VERSA (Gordon et al., 2019). Although these VI-based approaches have demonstrated impressive results in regression, classification as well as reinforcement learning, they do not provide any theoretical guarantee on the error induced by arbitrary or even unseen tasks sampled from the same

task distribution as well as any samples belonging to the same task. Moreover, the variational distributions used in most of these works are overly-simplified as multivariate normal distributions with diagonal covariance matrices. This assumption, however, limits the expressiveness of the variational approximation, resulting in a less accurate prediction.

Our work is also related to the PAC-Bayes framework used in meta-learning that upper-bounds errors with certain confidence levels (Pentina and Lampert, 2014; Amit and Meir, 2018). The main difference between those previous studies and ours is in the modelling of meta-parameters and the objective functions which are used to train such parameters. In the previous studies, the meta-parameters are the prior of task-specific parameters which are analogous to regularisation in task adaptation step (also known as "inner-loop"), while in our proposed method, the meta-parameters of interest are the model initialisation. In addition, the existing works rely on a train-train setting (Bai et al., 2021) where all data of a task is used for task adaptation such as REPTILE (Nichol et al., 2018), while ours follows the train-validation split with the bi-level optimisation objective shown below in (4.3). Such differences lead to a discrepancy in the formulation of the corresponding PAC-Bayes bounds. Another work closely related to our proposed method is *exponentially weighted aggregation for lifelong learning* (EWA-LL) (Alquier, Pontil et al., 2017). In EWA-LL, each task-specific model is decomposed into a shared feature extractor and a task-specific classifier, while in our approach, each task-specific model is an adapted or a fine-tuned version of the meta-parameters. Moreover, the setting of EWA-LL follows the train-train meta-learning approach, making the algorithm analogous to multi-task learning, while our proposed method is a train-validation meta-learning approach with the bi-level optimisation objective.

Our work has a connection to the statistical analysis of meta-learning that proves generalisation upper-bound for meta-learning algorithms (Maurer and Jaakkola, 2005; Maurer et al., 2016). Some typical recent works include the learning of the common regularisation that is used when adapting or fine-tuning on a specific task (Denevi et al., 2018; Denevi et al., 2019a; Denevi et al., 2019b; Denevi et al., 2020) to improve the performance of meta-learning algorithms in heterogeneous task environments, or analyse and optimise the regret induced by meta-learning algorithms in an online setting (Khodak et al., 2019). Our work differs from this line of research at how the meta-parameters are modelled. In our case, the meta-parameters of interest are the model initialisation, and our goal is to learn an approximated posterior for such parameters, while existing works consider different hyper-parameters, such as the shared L2 regularisation parameters, as meta-parameters, and often follow maximum likelihood estimation (MLE) or maximum a posterior (MAP) to learn a point estimate for such meta-parameters.

The most closely related work to ours is the *PAC optimal hyper-posterior* (PACOH) (Rothfuss et al., 2021), submitted later but published earlier. Both works study the generalisation of meta-learning via PAC-Bayes framework. There are only some minor differences between the two works. In PACOH, the meta-parameter of interest is used to model the common prior of task-specific parameter, while in our work, such parameter is analogous to the hyper-parameter shared over tasks sampled from the same task environment. In addition, PACOH relies on a "modern" PAC-Bayes bound (Alquier et al., 2016) which does not require bounded loss function, but moment generating function, while ours is extended from the classical PAC-Bayes bound (McAllester, 1999). Nevertheless, such differences are minimal and the two derived upper-bounds will coincide if the same PAC-Bayes bound for single-task learning setting is used. Another related work is PACMAML (Ding et al., 2021) that presents an upper-bound of the generalisation error when the testing task is sampled from a different task environment. In a loose sense, PACMAML can be considered as a variant of PACOH applied when the testing task environment is not the same as the training.

4.3 Background

4.3.1 Data generation model of a task

Consider a task indexed by $i \in \mathbb{N} = \{1, 2, \ldots\}$. A data point of the i-th task consists of an input $\mathbf{x}_{ij} \in \mathcal{X} \subseteq \mathbb{R}^d$ and a corresponding label $\mathbf{y}_{ij} \in \mathcal{Y}$ with $j \in \mathbb{N}$. Such data points are generated in two steps. The first step is to generate the input $\mathbf{x}_{ij}$ by sampling from some probability distribution $\mathcal{D}_i$. The second step is to determine the label $\mathbf{y}_{ij} = f_i(\mathbf{x}_{ij})$, where $f_i : \mathcal{X} \to \mathcal{Y}$ is the "correct" labelling function. Note that both the probability distribution $\mathcal{D}_i$ and the labelling function f_i are unknown. To simplify the notations, $(\mathbf{x}_{ij}, \mathbf{y}_{ij}) \sim (\mathcal{D}_i, f_i)$ is then used to denote such data generation.

4.3.2 Task instance

We restate the definition of a task as presented in Section 1.1.

Definition 1.2: Task (Hospedales et al., 2021)

A task or a task instance $\mathcal{T}_i$ consists of an unknown associated data generation model $(\mathcal{D}_i, f_i)$, and a loss function ℓ_i, denoted as: $\mathcal{T}_i = (\mathcal{D}_i, f_i, \ell_i)$.

Remark 4.1

The loss function ℓ_i is defined abstractly, and not necessarily some common loss functions, such as mean squared error (MSE) or cross-entropy. For example, ℓ_i could be referred to as negative log-likelihood if the objective is maximum likelihood estimation, or variational-free energy if the objective is based on variational inference.

To solve a task $\mathcal{T}_i$, one needs to obtain an optimal task-specific model $h(.;\mathbf{w}_i^*) : \mathcal{X} \to \mathcal{Y}$, parameterised by $\mathbf{w}_i^* \in \mathcal{W} \subseteq \mathbb{R}^n$, which minimises a loss function ℓ_i on the data of that task:

$$\mathbf{w}_i^* = \arg\min_{\mathbf{w}_i} \mathbb{E}_{(\mathbf{x}_{ij},\mathbf{y}_{ij})\sim(\mathcal{D}_i,f_i)} \left[\ell_i(\mathbf{x}_{ij},\mathbf{y}_{ij};\mathbf{w}_i)\right]. \tag{4.1}$$

In practice, since both $\mathcal{D}_i$ and f_i are unknown, the data generation model is replaced by a dataset consisting of a finite number of data-points generated according to the data generation model $(\mathcal{D}_i, f_i)$, denoted as $S_i = \{\mathbf{x}_{ij}, \mathbf{y}_{ij}\}_{j=1}^{m_i}$. The objective to solve that task is often known as empirical risk minimisation (ERM):

$$\mathbf{w}_i^{\text{ERM}} = \arg\min_{\mathbf{w}_i} \frac{1}{m_i} \sum_{j=1}^{m_i} \left[\ell_i(\mathbf{x}_{ij},\mathbf{y}_{ij};\mathbf{w}_i)\right]. \tag{4.2}$$

For simplicity, this paper considers two families of tasks: regression and classification. As a result, the label is a scalar $\mathcal{Y} \subseteq \mathbb{R}$ for regression and $\mathcal{Y} = \{0, 1, \ldots, C-1\}$ for classification, where C is the total number of classes. In addition, the loss function used will be the same for each task family, hence, the subscript on the loss function will be dropped, and the loss is denoted as ℓ throughout the paper. Due to the commonality of the loss function across all tasks, we will drop the notation of ℓ when referring to a task. In other words, a task can be simply represented by either its data generation model $(\mathcal{D}_i, f_i)$ or the associated dataset S_i.

4.3.3 Meta-learning

The setting of the meta-learning problem considered in this paper follows the *task environment* (Baxter, 2000) that describes the unknown distribution $p(\mathcal{D}, f)$ over a family of tasks. Each task $\mathcal{T}_i$ sampled from this task environment can be represented as $(\mathcal{D}_i^{(t)}, \mathcal{D}_i^{(v)}, f_i)$, where $\mathcal{D}_i^{(t)}$ and $\mathcal{D}_i^{(v)}$ are the probability distributions generate the training and validation input data, respectively, and they are not necessarily identical. The aim of meta-learning is to obtain a model trained on available training tasks such that the model can be fine-tuned on some labelled data of a testing task drawn from the same task environment to predict the label of unlabelled data on the same task accurately.

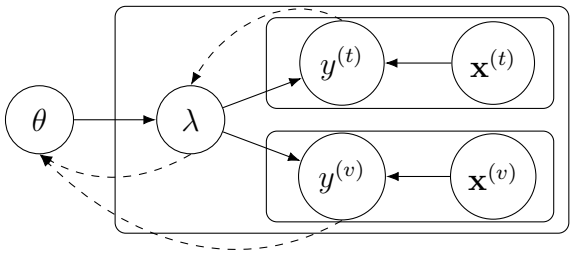


Figure 4.1: Meta-learning is an extension of hyper-parameter optimisation, where the meta-parameter θ is shared across all tasks. The solid arrows denote forward pass, while the dashed arrows indicate parameter inference, and rectangles illustrate the plate notations. The training subset $\{(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)})\}_{j=1}^{m_i^{(t)}}$ of task $\mathcal{T}_i$ and the meta-parameter θ are used to learn the task-specific parameter λ_i, corresponding to the lower-level optimisation in (4.3). The obtained λ_i is then used to evaluate the error on the validation subset $\{(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)})\}_{j=1}^{m_i^{(v)}}$ to learn the meta-parameter θ, corresponding to the upper-level optimisation in (4.3).

Such meta-learning methods use meta-parameters to model the common latent structure of the task distribution $p(\mathcal{D}, f)$. Examples of meta-parameters are: model initialisation (Finn et al., 2017; Finn et al., 2018; Ravi and Beatson, 2019; C. Nguyen et al., 2020), learning rate when fine-tuning the model for a task (Z. Li et al., 2017), feature extractor (Vinyals et al., 2016; Snell et al., 2017), and optimiser (Andrychowicz et al., 2016; Ravi and Larochelle, 2018). From this point of view, meta-learning can be considered as an extension of hyper-parameter optimisation in single-task learning, where the hyper-parameters of interest or meta-parameters are shared across many tasks. Mathematically, the objective of meta-learning can be written as a bi-level optimisation:

$$\begin{aligned} &\min_{\psi} \mathbb{E}_{q(\theta;\psi)} \mathbb{E}_{p(\mathcal{D},f)} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*(\theta))} \mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)} \left[\ell \left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i \right) \right], i \in \mathbb{N} \\ &\text{s.t.: } \lambda_i^*(\theta) = \arg\min_{\lambda_i(\theta)} \mathbb{E}_{(\mathcal{D}_i^{(t)},f_i)} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i(\theta))} \left[\ell \left(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}; \mathbf{w}_i \right) \right], \end{aligned} \tag{4.3}$$

where $q(\theta;\psi)$, parameterised by ψ, is a distribution over the meta-parameter θ, $q(\mathbf{w}_i;\lambda_i(\theta))$, parameterised by $\lambda_i(\theta)$, is a distribution of task-specific parameter $\mathbf{w}_i$, $\mathbb{E}_{(\mathcal{D}_i^{(t)},f_i)}$ means the expectation evaluated on input $\mathbf{x}_{ij}^{(t)}$ sampled from $\mathcal{D}_i^{(t)}$ and its corresponding label $y_{ij}^{(t)} = f_i(\mathbf{x}_{ij}^{(t)})$, and $\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}$ is defined similarly. In addition, to simplify the notations, we drop the dependence of θ from $\lambda_i(\theta)$.

Note that the difference of the objective in (4.3) from hyper-parameter optimisation in single-task learning is at the upper-level where (4.3) consists of the additional expectation over all training tasks, denoted as $\mathbb{E}_{p(\mathcal{D},f)}$.

Depending on how the loss function ℓ and distributions $q(\theta;\psi)$ and $q(\mathbf{w}_i;\lambda_i)$ are defined, one can obtain different meta-learning algorithms. For example, if ℓ corresponds to the loss in MLE and the lower-level is optimised by gradient descent

with θ as the initialisation of λ_i:

$$\begin{cases} \ell(\mathbf{x}, y; \mathbf{w}) & = -\ln p(y|\mathbf{x}; \mathbf{w}) \\ \theta & = \text{initialisation of } \lambda_i \\ q(\mathbf{w}_i; \lambda_i) & = \delta(\mathbf{w}_i - \lambda_i) \\ q(\theta; \psi) & = \delta(\theta - \psi), \end{cases} \tag{4.4}$$

where $\delta(.)$ is the Dirac delta function, then the objective in (4.3) can be simplified to:

$$\begin{aligned} &\min_{\theta} \mathbb{E}_{p(\mathcal{D},f)} \mathbb{E}_{\left(\mathcal{D}_i^{(v)}, f_i\right)} \left[-\ln p\left(y_{ij}^{(v)} | \mathbf{x}_{ij}^{(v)}; \mathbf{w}_i\right) \right] \\ &\text{s.t.: } \mathbf{w}_i^* = \arg\min_{\mathbf{w}_i} \mathbb{E}_{\left(\mathcal{D}_i^{(t)}, f_i\right)} \left[-\ln p\left(y_{ij}^{(t)} | \mathbf{x}_{ij}^{(t)}; \mathbf{w}_i\right) \right], i \in \mathbb{N}, \end{aligned} \tag{4.5}$$

which resembles the MAML algorithm (Finn et al., 2017).

Another example is the probabilistic meta-learning algorithm that replaces the loss function ℓ by a form of the variational-free energy and uses some similar assumptions in MAML:

$$\begin{cases} \ell(\mathbf{x}, y; \mathbf{w}) & = -\ln p(y|\mathbf{x}; \mathbf{w}) + \ln \left[\dfrac{q(\mathbf{w}_i; \lambda_i)}{p(\mathbf{w}_i)} \right] \\ \theta & = \text{initialisation of } \lambda_i \\ q(\mathbf{w}_i; \lambda_i) & = \mathcal{N}\left(\mathbf{w}_i; \boldsymbol{\mu}_{\lambda_i}, \text{diag}\left(\boldsymbol{\sigma}_{\lambda_i}^2\right)\right) \\ q(\theta; \psi) & = \delta(\theta - \psi), \end{cases} \tag{4.6}$$

where $\mathcal{N}(.)$ denotes multivariate normal distribution, diag(.) denotes a diagonal matrix, and $p(\mathbf{w}_i)$ is the prior of $\mathbf{w}_i$. This formulation resembles ABML (Ravi and Beatson, 2019) and VAMPIRE (C. Nguyen et al., 2020) algorithms.

4.3.4 PAC-Bayes upper-bound in single-task learning

In practice, the data probability distribution $\mathcal{D}_i$ and the labelling function f_i of a task $\mathcal{T}_i$ are unknown, but only a dataset S_i consisting of finite input data and labels is available. Since the aim is to minimise the loss averaged over all data generated from $(\mathcal{D}_i, f_i)$, it is, therefore, important to analyse the difference between such "true" loss and the empirical loss evaluated on a given dataset. Such difference can be upper-bounded by the KL divergence shown in Theorem 4.1 with a certain level of confidence.

Theorem 4.1: (McAllester, 1999)

If a dataset S_i consists of m_i inputs $\mathbf{x}_{ik}$ i.i.d. sampled from a data probability distribution $\mathcal{D}_i$ and being labelled by f_i, $\mathcal{H}$ is a hypothesis class, $\varepsilon \in (0, 1]$

and a loss function $\ell : \mathcal{H} \times \mathcal{Y} \to [0, 1]$, then for any "posterior" $q(\mathbf{w}_i; \lambda_i)$ over a hypothesis $h(.; \mathbf{w}_i) \in \mathcal{H}$, parameterised by $\mathbf{w}_i$, the following holds with a probability at least $1 - \varepsilon$:

$$\mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\mathbb{E}_{(\mathcal{D}_i,f_i)}\left[\ell(\mathbf{x}_{ij}, y_{ij}; \mathbf{w}_i)\right] \leq \frac{1}{m_i}\sum_{k=1}^{m_i}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[\ell(\mathbf{x}_{ik}, y_{ik}; \mathbf{w}_i)\right] + \sqrt{\frac{\mathrm{KL}\left[q(\mathbf{w}_i; \lambda_i)||p(\mathbf{w}_i)\right] + \frac{\ln m_i}{\varepsilon}}{2(m_i - 1)}},$$

where $p(\mathbf{w}_i)$ is the prior of $\mathbf{w}_i$.

Instead of minimising the "true" loss of a task, denoted as the left-hand side term in Theorem 4.1, which is intractable, one should minimise both the empirical loss and the KL divergence on the right-hand side. Indeed, the upper-bound in Theorem 4.1 is often used as a tractable learning objective function for the model of interest.

4.4 Methodology

4.4.1 PAC-Bayes meta-learning

In practice, the training and validation data probability distributions, $\mathcal{D}_i^{(t)}$ and $\mathcal{D}_i^{(v)}$, and their corresponding labelling function f_i, are unknown, and only two disjoint datasets with finite examples:

$$S_i^{(t)} = \left\{\left(\mathbf{x}_{ij}^{(t)}, y_{ij}^{(t)}\right)\right\}_{j=1}^{m_i^{(t)}}, \quad S_i^{(v)} = \left\{\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}\right)\right\}_{k=1}^{m_i^{(v)}}$$
$$S_i = S_i^{(t)} \cup S_i^{(v)}, \quad S_i^{(t)} \cap S_i^{(v)},$$

associated with task $\mathcal{T}_i$, are provided as illustrated in Figure 4.1. Note that $m_i^{(t)}$ and $m_i^{(v)}$ are not necessarily identical, and the label $y_{ij}^{(v)}$ in the validation dataset $S_i^{(v)}$ are known in training, while being unknown in testing. In addition, the task distribution $p(\mathcal{D}, f)$ is unknown, but only T tasks sampled from such task distribution are available for training. Hence, it is important to derive an upper-bound, and in particular PAC-Bayes upper-bound, for both the "generalisation" losses in upper- and lower-levels of (4.3). Since the optimisation in the lower-level of (4.3) corresponds to solving a single task, the PAC-Bayes upper-bound presented in Theorem 4.1 can be straight-forwardly applied as the learning objective to upper-bound the error on unseen samples generated from that task. The remaining problem lies on the formulation of the PAC-Bayes upper-bound for the loss in the upper-level of (4.3). This novel bound is shown in Theorem 4.2 with its detailed proof presented in Appendix B.1.

Theorem 4.2

Given T tasks sampled from the same task environment $p(\mathcal{D}, f)$, where each task has an associated pair of datasets $(S_i^{(t)}, S_i^{(v)})$ with samples generated from the task-specific data generation model $(\mathcal{D}_i^{(t)}, \mathcal{D}_i^{(v)}, f_i)$, then for a bounded loss function $\ell : \mathcal{W} \times \mathcal{Y} \to [0, 1]$ and any distributions $q(\theta; \psi)$ of meta-parameter θ and $q(\mathbf{w}_i; \lambda_i)$ of task-specific parameter $\mathbf{w}_i$, the following holds with the probability at least $1 - \varepsilon, \forall \varepsilon \in (0, 1]$:

$$\begin{aligned}
&\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{p(\mathcal{D},f)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right] \\
&\leq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda)}\left[\ell\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}; \mathbf{w}_i\right)\right] \\
&\quad + \sqrt{\frac{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)||p(\mathbf{w}_i)\right]\right] + \frac{T^2}{(T-1)\varepsilon}\ln m_i^{(v)}}{2\left(m_i^{(v)} - 1\right)}} \\
&\quad + \sqrt{\frac{\mathrm{KL}\left[q(\theta;\psi)||p(\theta)\right] + \frac{T\ln T}{\varepsilon}}{2(T-1)}},
\end{aligned}$$

where $p(\mathbf{w}_i), \forall i \in \{1, \ldots, T\}$ is the prior of task-specific parameter $\mathbf{w}_i$ and $p(\theta)$ is the prior of meta-parameter θ.

Proof sketch. The proof is carried out in three steps: (i) derive a PAC-Bayes upper-bound for unseen samples generated from task-specific data generation model $(\mathcal{D}_i^{(v)}, f_i), \forall i \in \{1, \ldots, T\}$ by adapting the proof of Theorem 4.1 as shown in Appendix B.1.1, (ii) derive a PAC-Bayes upper-bound for unseen tasks by applying Theorem 4.1 as shown in Appendix B.1.2, and (iii) combine the two obtained results as shown in Appendix B.1.3 to complete the proof. □

Remark 4.2

One limitation of Theorems 4.1 and 4.2 is the assumption of bounded loss which restricts the loss function within $[0, 1]$. Although there are several studies that extend PAC-Bayes bound to unbounded losses (Catoni, 2004; Germain et al., 2016; Alquier et al., 2016; Alquier and Guedj, 2018), their formulation still needs to assume the boundedness of the moment generating function of some particular loss functions. This assumption is, however, impractical since in practice, some common loss functions, such as mean squared error or cross-entropy, do not possess such property. Nevertheless, our main focus in this paper is to provide a theoretical generalisation guarantee for meta-learning using PAC-Bayes theory.

Such extension is not necessary since in the implementation our loss is clipped to be within $[0, 1]$.

4.4.2 Practical meta-learning objective

Instead of following the objective in (4.3), which is intractable due to the unknown data generation model of each task and the unknown task environment, we utilise the results in Theorems 4.1 and 4.2 to propose a new objective function for a practical meta-learning that theoretically guarantees the generalisation errors due to unseen samples of a task and unseen tasks sampled from the task environment. The new objective is formulated by replacing the optimisation in the lower-level of (4.3) by the minimisation of the corresponding upper-bound in Theorem 4.1, and the optimisation in the upper-level of (4.3) by the minimisation of the upper-bound in Theorem 4.2.

4.4.3 Meta-learning with implicit task-specific posterior

Given the new objective function where both the optimisations in the lower- and upper-levels of (4.3) are replaced by the minimisation of the PAC-Bayes upper-bounds in Theorems 4.1 and 4.2, there are a total of four distributions of interest: the "posteriors" $q(\theta; \psi)$ and $q(\mathbf{w}_i; \lambda_i), i \in \{1, \ldots, T\}$, and the priors $p(\theta)$ and $p(\mathbf{w}_i)$.

As priors represent the modelling assumption and are chosen before observing data, we assume that both $p(\theta)$ and $p(\mathbf{w}_i)$ follows multivariate normal distributions with diagonal covariance matrices:

$$p(\theta) = \mathcal{N}\left(\theta; \mathbf{0}, \sigma_\theta^2 \mathbf{I}\right) \tag{4.7}$$

$$p(\mathbf{w}_i) = \mathcal{N}\left(\mathbf{w}_i; \mathbf{0}, \sigma_w^2 \mathbf{I}\right), \tag{4.8}$$

where $\mathbf{0}$ is a vector containing all zeros, $\mathbf{I}$ is the identity matrix, $\sigma_\theta > 0$ and $\sigma_w > 0$ are hyper-parameters.

Remark 4.3

The assumption of multivariate normal priors is to simplify the analysis presented in this section, especially easily sampled from to apply Monte Carlo approximation for the lower-bound of the KL divergence shown in Eq. (4.12). One can also select different distributions as priors, but that might result in a more complicated formulation and implementation.

For the posterior $q(\theta; \psi)$ of the meta-parameter θ, it is often assumed to be a Dirac delta function: $q(\theta; \psi) = \delta(\theta - \psi)$ (Finn et al., 2017; Ravi and Beatson, 2019; C. Nguyen et al., 2020). In this paper, $q(\theta; \psi)$ is assumed to be a multivariate normal

distribution with a fixed diagonal covariance matrix, denoted as:

$$q(\theta;\psi) = \mathcal{N}\left(\theta;\boldsymbol{\mu}_\theta,\sigma_0\mathbf{I}\right), \tag{4.9}$$

where σ_0 is a hyper-parameter. Such notation also means that $\psi = \boldsymbol{\mu}_\theta$.

The only distribution left is the task-specific "posterior" $q(\mathbf{w}_i;\lambda_i)$. In general, $q(\mathbf{w}_i;\lambda_i)$ can be modelled following one of the two general types: prescribed and implicit Diggle and Gratton, 1984. For example, ABML Ravi and Beatson, 2019 and VAMPIRE C. Nguyen et al., 2020 are prescribed approaches where $q(\mathbf{w}_i;\lambda_i)$ is assumed to be a multivariate normal distribution with a diagonal covariance matrix. Such approximation is, however, inexpressive, resulting in a poor estimation. In contrast, implicit modelling only has access to the samples generated from the distribution of interest without assuming any analytical form of such distribution, e.g. the generator in generative adversarial networks (Goodfellow et al., 2014). In this paper, we use the implicit modelling approach to expressively represent $q(\mathbf{w}_i;\lambda_i)$.

The distribution $q(\mathbf{w}_i;\lambda_i)$ is now defined at a more fundamental level whereby data is generated through a stochastic mechanism without specifying a parametric distribution. A parameterised model (i.e., a generator G represented by a deep neural network) is used to model the sample generation:

$$\mathbf{w}_i \sim q(\mathbf{w}_i;\lambda_i) \Leftrightarrow \mathbf{w}_i = G(\mathbf{z};\lambda_i), \mathbf{z} \sim p(\mathbf{z}), \tag{4.10}$$

where $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^z$ is the latent noise sampled from a latent noise distribution $p(\mathbf{z})$, which is often selected as the uniform in $[0,1]^z$ or the standard normal distribution. In our implementation, we observe that due to the unconstrained support space of the standard normal distribution, latent noise sampled from such distribution may vary drastically, resulting in a large variation of the generated task-specific model parameter $\mathbf{w}_i$ and potentially, making the training difficulty, especially at the beginning of the training. We, therefore, use the uniform distribution on $[0,1]^z$ as the latent noise distribution $p(\mathbf{z})$ to bound the support space $\mathcal{Z}$ of the latent noise to make the training more stable.

Due to the nature of implicit models, the KL divergence term $\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)||p(\mathbf{w}_i)\right]$ in the lower-level and $\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]$ in the upper-level of the bi-level optimisation objective cannot be evaluated either analytically or symbolically. We, therefore, propose to employ the lower-bound of the KL divergence shown in Lemma 4.3 to estimate the value of the KL divergence to train the meta-learning model.

Lemma 4.3: Compression lemma (Banerjee, 2006)

For any measurable function $\phi(h)$ on a set of predictors under consideration $\mathcal{H}$,

and any distributions P and Q on $\mathcal{H}$, the following holds:

$$\mathbb{E}_Q\left[\phi(h)\right] - \ln \mathbb{E}_P\left[e^{\phi(h)}\right] \leq \mathrm{KL}\left[Q\|P\right].$$

Further,

$$\sup_{\phi} \mathbb{E}_Q\left[\phi(h)\right] - \ln \mathbb{E}_P\left[e^{\phi(h)}\right] = \mathrm{KL}\left[Q\|P\right].$$

Proof. Please refer to the proof in Appendix B.2. □

Remark 4.4

Naively substituting the lower-bound of the KL divergence terms into the objective function makes the upper-bound in Theorem 4.2 no longer the upper-bound of the generalisation error. However, since the supremum (or maximum) of such KL lower-bound equals to the KL divergence, the upper-bound derived in Theorem 4.2 still holds if the maximum for the KL lower-bound is obtained. In this paper, we present another optimisation to estimate such optimal lower-bound to approximate the KL divergence terms.

To model the function $\phi : \mathcal{W} \to \mathbb{R}$ in Lemma 4.3 to estimate the lower-bound of KL divergences, we use a neural network with parameter ω_i. The objective to obtain ϕ is to maximise the left-hand side in Lemma 4.3, which can be expressed as:

$$\omega_i^* = \arg\max_{\omega_i} \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[\phi(\mathbf{w}_i;\omega_i)\right] - \ln \mathbb{E}_{p(\mathbf{w}_i)}\left[e^{\phi(\mathbf{w}_i;\omega_i)}\right]. \tag{4.11}$$

The KL divergence between the task-specific posterior $q(\mathbf{w}_i;\lambda_i)$ and prior $p(\mathbf{w}_i)$ can then be estimated as:

$$\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)||p(\mathbf{w}_i)\right] = \mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\left[\phi(\mathbf{w}_i;\omega_i^*)\right] - \ln \mathbb{E}_{p(\mathbf{w}_i)}\left[e^{\phi(\mathbf{w}_i;\omega_i^*)}\right]. \tag{4.12}$$

One problem that arises when estimating the losses in both the lower- and upper-level of the meta-learning objective is to learn a different ω_i for each different task $\mathcal{T}_i$ by training the neural network ϕ from scratch. The downside of such naive implementation is the significant increase in training time. We, therefore, propose to learn a good initialisation of ω_i using MAML (Finn et al., 2017) to reduce the time of the KL divergence estimation. With this assumption, we define ω_0 as the meta-parameters (or initialisation) of ω_i. Within each task, we initialise ω_i at ω_0 and optimise the loss in (4.11) w.r.t. ω_i using gradient descent. ω_0 is then obtained by optimising the validation loss evaluated on ω_i^* w.r.t. ω_0.

The proposed meta-learning method, therefore, consists of two parameters of interest: the hyper-meta-parameter ψ and the meta-parameter ω_0 of the ϕ-network.

Algorithm 3 SImPa

procedure TRAIN
  initialise hyper-meta-parameter $\psi = \boldsymbol{\mu}_\theta$
  initialise ϕ-network meta-parameter ω_0
  while ψ and ω_0 not converged **do**
    sample: $\theta \sim \mathcal{N}(\theta; \boldsymbol{\mu}_\theta, \sigma_0 \mathbf{I})$
    sample a mini-batch of T tasks
    for each task $\mathcal{T}_i$ **do**
      $\lambda_i, \omega_i \leftarrow$ OPTIMISE LOWER-LEVEL$(\theta, \omega_0, \mathcal{T}_i)$
      use λ_i to calculate PAC-Bayes upper-bound in Theorem 4.2
      use ω_i to calculate the KL lower-bound in (4.12)
    end for
    $\psi \leftarrow$ SGD (average of T losses obtained in step 9)
    $\omega_0 \leftarrow$ SGA(average of T KL lower-bounds obtained in step 10)
      ▷ SGD/SGA = stochastic gradient descent/ascent
  end while
  return ψ and ω_0
end procedure

procedure OPTIMISE LOWER-LEVEL$(\theta, \omega_0, \mathcal{T}_i)$
  initialise task-specific parameter: $\lambda_i \leftarrow \theta$
  initialise task-specific ϕ-net: $\omega_i \leftarrow \omega_0$
  $\omega_i^* \leftarrow \arg\max_{\omega_i}$ KL lower-bound in (4.11)
  use ω_i^* to calculate KL divergence in Eq. (4.12)
  $\lambda_i^* \leftarrow \arg\min_{\lambda_i}$ PAC-Bayes upper-bound in Theorem 4.1
  return λ_i^* and ω_i^*
end procedure

Each of the parameters is optimised following their corresponding bi-level optimisation objective functions similar to (4.3). Algorithm 3 shows the training procedure of the proposed approach. In addition, we name the proposed approach as *statistical implicit PAC-Bayes meta-learning* or SImPa for short, to simplify the text when comparing to other meta-learning methods.

One potential drawback of the implicit modelling approach is the curse of dimensionality, resulting in a computationally expensive training process. This is an active research question when dealing with generative models. This issue can be addressed by encoding the high-dimensional data, such as images, to a feature embedding space by supervised-learning on the same training data set. For example, in one of our experiments, we also show how to use image features extracted from a wide-residual network trained on tiered-ImageNet training set (Rusu et al., 2019) to mitigate this issue. This strategy reduces the dimension of the input space, leading to smaller task-specific model parameter $\mathbf{w}_i$, and eventually decreasing the size of the generator. The trade-off lies in the possibility of losing relevant information that can affect the performance on hold-out tasks.

It is also worth noting that our proposed method is easier to train than prior probabilistic meta-learning methods (Finn et al., 2018; Ravi and Beatson, 2019) because we no longer need to tune the weighting factor of KL $[q(\mathbf{w}_i; \lambda_i)||p(\mathbf{w}_i)]$ in both levels of the bi-level optimisation objective. Although weighting such KL divergence terms can be justified by casting the optimisation in each level of the meta-learning objective to a constrained optimisation as shown in β-VAE (Higgins et al., 2016), the weighting factor in such case is the corresponding Lagrange multiplier of the constrained optimisation. Thus, simply setting that weighting factor as a "tunable" hyper-parameter may result in a sub-optimal solution. In contrast, our proposed approach does not need to re-weight the KL divergences. The trade-off of our approach lies in the need to set the confidence parameter ε, but tuning ε is arguably more intuitive than tuning the correct weighting factor for the KL divergence terms done in previous works.

4.5 Experimental evaluation

In this section, SImPa is evaluated on few-shot regression and classification problems and compared to common meta-learning baselines, such as MAML (Finn et al., 2017), ABML (Ravi and Beatson, 2019), PLATIPUS (Finn et al., 2018) and BMAML (Yoon et al., 2018).

The loss functions used are mean-squared error (MSE) for regression and cross-entropy for classification. Following the assumption of bounded losses made in Section 4.4, the data-related losses in both the lower- and upper-level, $\ell(\mathbf{x}_{ij}, y_{ij}; \mathbf{w}_i)$ are clipped to $[0, 1]$. In addition, Monte Carlo (MC) sampling is used to evaluate the expectation over $q(\theta; \psi)$ and $q(\mathbf{w}_i; \lambda_i)$. In particular, one sample of meta-parameter θ and eight samples of task-specific parameter $\mathbf{w}_i$ are used in training, while these numbers are one and thirty-two in testing, respectively. The asymmetric choice of those hyper-parameters is to optimise training time, while maximising the prediction performance in evaluation. For the hyper-parameters defined in Eqs. (4.7) to (4.9), $\sigma_\theta = \sigma_{\mathbf{w}} = 1$ and $\sigma = 10^{-6}$. In addition, the confidence parameters is selected as $\varepsilon = \varepsilon_i = 0.1, \forall i \in \{1, \ldots, T\}$. The number of tasks per an update of the parameters of interest is $T = 20$.

In terms of generating $\mathbf{w}_i$ for each task $\mathcal{T}_i$, we use latent noise vectors $\mathbf{z}$ sampled from the uniform distribution in $[0, 1]^{128}$. The generator in (4.10) is modelled by a fully-connected neural network with 2 hidden layers. Each of these layers consists of 256 and 512 hidden units, respectively, and is activated by rectified linear unit without any batch normalisation. The output of the final layer is then activated by hyperbolic tangent function to constrain the parameter of the base network, avoiding the loss value from exploding. The ϕ-network has an "inverted" architecture of the

generator, which consists of 3-hidden layers. These layers consist of 512, 256 and 128 hidden units, respectively, and are also activated by rectified linear unit without any batch normalisation. Adam optimiser (Kingma and Ba, 2015) is used to optimise both the hyper-meta-parameter ψ and ω_0 with the same learning rate of 10^{-4}. To train the ϕ-network for each task, 512 MC samples of the base network parameters are sampled from both $q(\mathbf{w}_i; \lambda_i)$ and $p(\mathbf{w}_i)$ to evaluate the lower-bound of the KL divergence in (4.11). To optimise (4.11) w.r.t. ω_i, gradient descent is used with a learning rate of 10^{-4}.

4.5.1 Regression

The experiment in this subsection is a multi-modal task distribution where half of the data is generated from sinusoidal functions, while the other half is from linear functions (Finn et al., 2018). The sinusoidal function used in this experiment has the form $y = A\sin(x + \Phi) + \epsilon$, where A and Φ are uniformly sampled from $[0.1, 5]$ and $[0, \pi]$, respectively, while the linear function considered is in the form $y = ax + b + \epsilon$, where a and b are randomly sampled from $[-3, 3]$. The noise ϵ is sampled from $\mathcal{N}(0, 0.3^2)$. The experiment is carried out under the 5-shot setting: $m_i^{(t)} = 5$, and the validation set $S_i^{(v)}$ consists of $m_i^{(v)} = 50$ data points.

Similar to existing works in the literature (Finn et al., 2018), the base network to solve each regression task is a fully-connected network with 2 hidden layers, where each layer has 40 hidden units. Linear rectifier function is used as activation function, and no batch-normalisation is used. Gradient descent is used as the optimiser for the lower-level optimisation with learning rate fixed at 10^{-3} and iterated 5 times.

As shown in Figure 4.2, SImPa is able to vary the prediction variance, especially when there is more uncertainty in the training data, while MAML can only output a single value at each data point. For a quantitative comparison, we train many probabilistic meta-learning methods, including PLATIPUS (Finn et al., 2018), BMAML (Yoon et al., 2018) and ABML (Ravi and Beatson, 2019), in the same regression problem. Here, BMAML consists of 10 particles trained without Chaser Loss. As shown in Figure 4.3a, SImPa achieves much smaller MSE, comparing to MAML, PLATIPUS and ABML, and comparable NLL to the non-parametric BMAML when being evaluated on the same hold-out tasks.

To further evaluate the predictive uncertainty, we employ the reliability diagram based on the quantile calibration for regression (Song et al., 2019). The reliability diagram shows a correlation between predicted and actual probability. A perfectly calibrated model will have its predicted probability equal to the actual probability, and hence, align well with the diagonal $y = x$. The results in Figure 4.3b show that the model trained with SImPa achieves the best calibration among all the methods

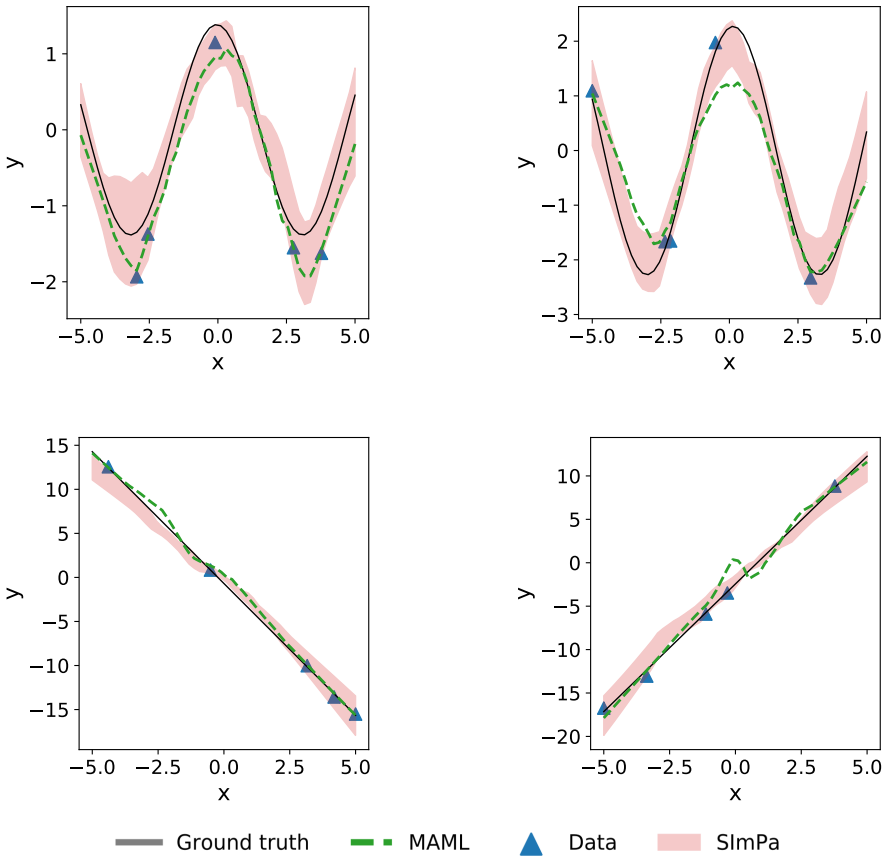
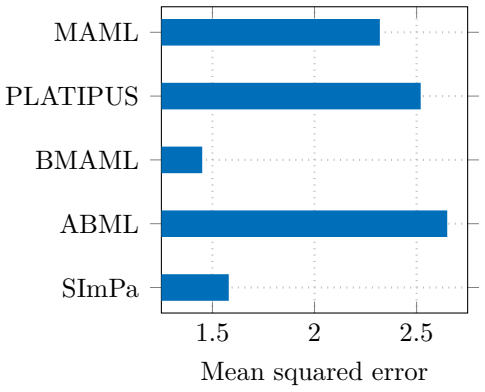
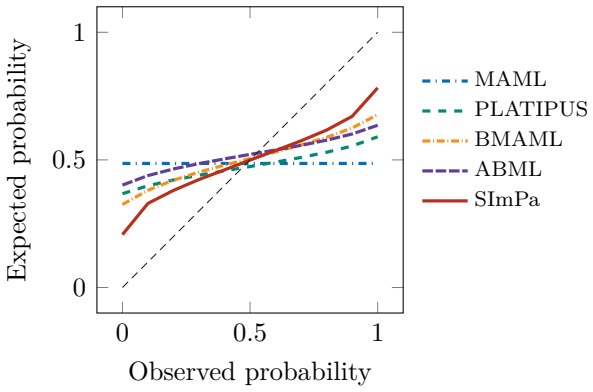


Figure 4.2: SImPa and MAML are compared in a regression problem when training is based on multi-modal data – half of the tasks are generated from sinusoidal functions, and the other half are from linear functions. The shaded area is the prediction made by SImPa $\pm$ 3$\times$ standard deviation.

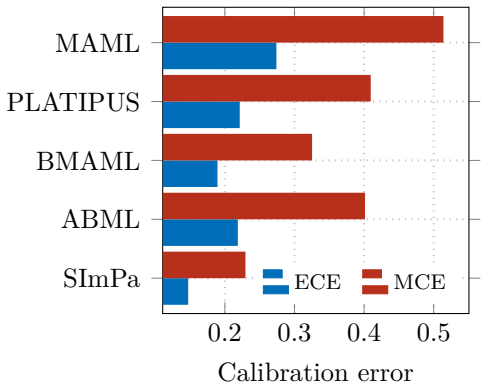
considered. Due to the nature of a deterministic approach, MAML (Finn et al., 2017) is represented as a horizontal line, resulting in a poorly calibrated model. The two probabilistic meta-learning methods, PLATIPUS and ABML, perform better than MAML; however, the averaged slopes of their performance curves are quite close to MAML, implying that their multivariate normal posteriors of task-specific model parameters have small covariance diagonal values. This may be caused by their exclusive reliance on less-expressive multivariate normal distributions with diagonal covariance matrices. The performance of BMAML is slightly better than PLATIPUS and ABML due to its non-parameteric modelling approach. In contrast, SImPa employs a much richer variational distribution $q(\mathbf{w}_i; \lambda_i)$ for task specific parameters, and therefore, produces a model with better calibration. For another quantitative comparison, we plot the expected calibration error (ECE) (C. Guo et al., 2017), which is the weighted average of the absolute errors measuring from the diagonal, and the maximum calibration error (MCE) (C. Guo et al., 2017), which returns the maximum of absolute errors in Figure 4.3c. Overall, SImPa outperforms all of the state-of-the-art methods in both ECE and MCE.



(a) MSE on hold-out tasks



(b) Reliability diagram



(c) ECE and MCE

Figure 4.3: Quantitative comparison between various probabilistic meta-learning approaches averaged over 1000 unseen tasks shows that SImPa has a comparable MSE error and the smallest calibration error.

4.5.2 Few-shot classification

We evaluate SImPa on the N-way k-shot setting, where a meta learner is trained on many related tasks containing N classes with k examples per class ($m_i^{(t)} = kN$). The evaluation is carried out by comparing the results of SImPa against the results of state-of-the-art methods on three popular few-shot learning benchmarking data sets: Omniglot (Lake et al., 2015), mini-ImageNet (Vinyals et al., 2016; Ravi and Larochelle, 2018) and tiered-ImageNet (Ren et al., 2018).

Omniglot dataset consists of 50 different alphabets with a total of 1,623 characters drawn online via Amazon's Mechanical Turk by 20 different people. Hence, Omniglot is often considered as a "transposed" MNIST since Omniglot has many classes, but each class has 20 images. We follow the original train-test split where 30 alphabets are used for training, while the other 20 alphabets are used for testing. To be consistent with previous evaluations, we pre-process by down-sampling all images to 28-by-28 pixels. No data augmentation, such as rotation, is used. Note that for the task formation, many existing meta-learning methods in the literature use

Table 4.1: The few-shot 5-way classification accuracy results (in percentage, with 95% confidence interval) of SImPa averaged over 1 million tasks on Omniglot (top), and 600 tasks on mini-ImageNet (middle-top and middle-bottom) and tiered-ImageNet (bottom) datasets. The bold numbers denote statistically significant best method according to t-test.

METHOD	1-SHOT	5-SHOT
Omniglot (Lake et al., 2015) - standard 4-block CNN		
MAML (Finn et al., 2017)	97.143 ± 0.005	
Prototypical nets (Snell et al., 2017)	96.359 ± 0.006	
BMAML (Yoon et al., 2018)	94.104 ± 0.008	
ABML (Ravi and Beatson, 2019)	97.281 ± 0.004	
SImPa	$\mathbf{98.352 \pm 0.005}$	
Mini-ImageNet (Ravi and Larochelle, 2018) - standard 4-block CNN		
Matching nets (Vinyals et al., 2016)	43.56 ± 0.84	55.31 ± 0.73
Meta-learner LSTM (Ravi and Larochelle, 2018)	43.44 ± 0.77	60.60 ± 0.71
MAML (Finn et al., 2017)	48.70 ± 1.84	63.15 ± 0.91
Prototypical nets (Snell et al., 2017)[1]	49.42 ± 0.78	$\mathbf{68.20 \pm 0.66}$
LLAMA (Grant et al., 2018)	49.40 ± 1.83	_
PLATIPUS (Finn et al., 2018)	50.13 ± 1.86	_
ABML (Ravi and Beatson, 2019)	45.00 ± 0.60	_
SImPa	$\mathbf{51.72 \pm 0.48}$	63.49 ± 0.40
Mini-ImageNet (Ravi and Larochelle, 2018) - non-standard network		
Relation nets (Sung et al., 2018)	50.44 ± 0.82	65.32 ± 0.70
VERSA (Gordon et al., 2019)	53.40 ± 1.82	67.37 ± 0.86
SNAIL (Mishra et al., 2018)	55.71 ± 0.99	68.88 ± 0.92
adaResNet (Munkhdalai et al., 2018)	56.88 ± 0.62	71.94 ± 0.57
TADAM (Oreshkin et al., 2018)	58.50 ± 0.30	76.70 ± 0.30
LEO (Rusu et al., 2019)	61.76 ± 0.08	77.59 ± 0.12
LGM-Net (H. Li et al., 2019)	$\mathbf{69.13 \pm 0.35}$	71.18 ± 0.68
SImPa[2]	62.85 ± 0.56	77.65 ± 0.50
Tiered-ImageNet (Ren et al., 2018) non-standard network		
MAML (Y. Liu et al., 2018)	51.67 ± 1.81	70.30 ± 0.08
Proto. Nets (Ren et al., 2018)	53.31 ± 0.89	72.69 ± 0.74
Relation Net (Y. Liu et al., 2018)	54.48 ± 0.93	71.32 ± 0.78
Trns. Prp. Nets (Y. Liu et al., 2018)	57.41 ± 0.94	71.55 ± 0.74
LEO (Rusu et al., 2019)	66.33 ± 0.05	81.44 ± 0.09
MetaOptNet (Lee et al., 2019)	65.81 ± 0.74	81.75 ± 0.53
SImPa[2]	$\mathbf{70.26 \pm 0.35}$	80.15 ± 0.28

non-standard train-test split where characters of all 50 alphabets are mixed, and randomly split. This splitting potentially forms easier tasks since knowing a character in an alphabet might help to classify other characters within that same alphabet. Moreover, the mixed and random split is different from evaluation to evaluation, making it challenging to fairly compare different meta-learning methods.

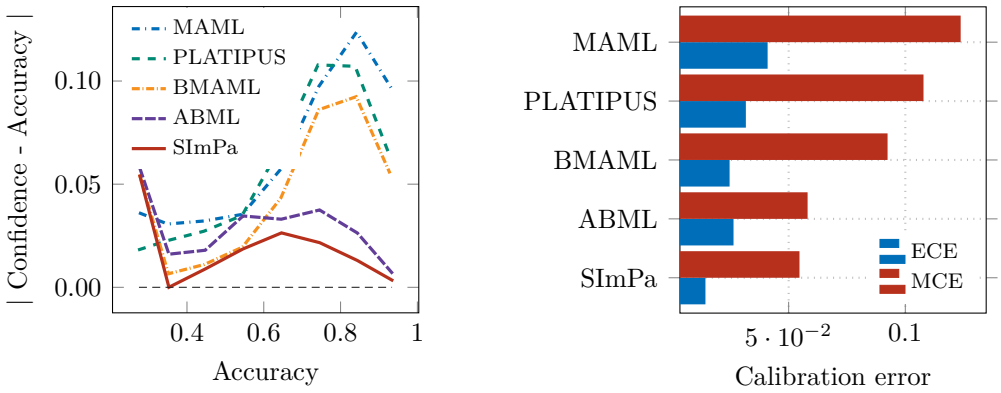
Mini-ImageNet (Vinyals et al., 2016) is another dataset used to evaluate classification performance between different meta-learning methods. The dataset consists of 100 classes, where each class contains 600 colour images taken from ImageNet (Russakovsky et al., 2015). We follow the standard train-test split which uses 64 classes for training, 16 classes for validation, and 20 classes for testing (Ravi and Larochelle, 2018). The images in the dataset are pre-processed by down-sampling to 84-by-84 pixels before any training is carried out. No data augmentation, such as image flipping or rotation, is used.

Tiered-ImageNet is one of the largest subsets of ImageNet, which consists of total 608 classes grouped into 34 high-level categories (Ren et al., 2018). Tiered-ImageNet is often used as a benchmark for large-scaled few-shot learning. We also follow the standard train-test split that consists of 20 categories for training, 6 categories for validation, and 8 categories for testing. In addition, our evaluation is carried out by employing the features extracted from a residual network trained on the data and classes from the training set (Rusu et al., 2019).

The evaluation is carried out in 2 cases: one with raw image data and the other with 640-dimensional image features extracted from a wide-residual network trained solely on training data (Rusu et al., 2019). In the "standard" case, the base network is the "standard" convolutional network consisting of 4 modules. Each module has a convolutional layer with 32 3-by-3 filters, followed by a batch normalisation, ReLU and 2-by-2 max-pooling. The output of the final module is flatten and connected to a fully connected layer to predict the label of the input image. In the "non-standard" case, the base network is a fully-connected network with 2 hidden layers. Each layer consists of 128 and 32 hidden units activated by rectified linear unit without any batch-normalisation.

We report the classification accuracy of SImPa on these three data sets in Table 4.1. For Omniglot, we use the published code to reproduce the results for some common meta-learning methods to fairly compare with SImPa. The accuracy averaged over more than 1 million testing tasks show that the proposed SImPa is better than competing meta-learning methods in the literature. For mini-ImageNet, SImPa achieves the best empirical results for the 1-shot setting when the base model is the "standard" CNN, and for the 5-shot setting when a different network architecture is used. SImPa shows the second best results for the 5-shot setting with the 4-layer CNN and the 1-shot setting with the different network architecture. Note that for the 5-shot setting using standard CNN, Prototypical networks need to train with a much higher "way" which is harder to learn, and might help the trained model to perform better on easier tasks with lower "way". For tiered-ImageNet, SImPa

[1]Trained on 30-way 1-shot setting
[2]Use extracted features (Rusu et al., 2019) as input



(a) Reliability diagram (b) ECE and MCE

Figure 4.4: Calibration of the "standard" 4-block CNN trained with different meta-learning methods on 5-way 1-shot classification tasks on mini-ImageNet.

outperforms the current state-of-the-art in 1-shot setting, while being comparable in 5-shot setting. To obtain a fairer comparison, we re-run MAML on the image data of mini-ImageNet using a ResNet10, which has about 5 million parameters (ours has about 8 millions parameters). However, MAML, with and without L2 regularisation, over-fits the training data (our best result for MAML was 89% accuracy on train, while only 42% on test). This known issue of overfitting when using larger networks in MAML was mentioned in the MAML paper (Finn et al., 2017, Section 5.2). We also try a similar model for ABML (Ravi and Beatson, 2019), but observed no improvement.

Similarly to the experiment for regression, we use reliability diagrams (C. Guo et al., 2017) to evaluate the predictive uncertainty. For a fair comparison, we re-implement several probabilistic meta-learning approaches, including MAML (Finn et al., 2017), PLATIPUS (Finn et al., 2018), BMAML (Yoon et al., 2018) and ABML (Ravi and Beatson, 2019), using the 4-block CNN as the base model, trained under the same setting, and plot their reliability chart. The performance curves in the reliability diagram show how well calibrated a model is when testing across many unseen tasks. A perfectly calibrated model will have its values overlapped with the identity function $y = x$, indicating that the probability associated with the label prediction is the same as the true probability. To ease the visualisation, we normalise the reliability chart by subtracting the predicted accuracy by its corresponding value on the diagonal $y = x$, as shown in Figure 4.4a. Hence, for the normalised reliability chart, the closer to $y = 0$, the better the calibration. Visually, the model trained with SImPa shows better calibration than the ones trained with other meta-learning methods. To further evaluate, we compute the expected calibration error (ECE) and maximum calibration error (MCE) (C. Guo et al., 2017) of the models trained with these methods. The results plotted in Figure 4.4b show that the model trained with

SImPa achieves the smallest ECE and MCE among all the methods considered in this comparison. The most competitive method to SImPa, regarding ECE and MCE, is ABML, but note that ABML has a worse classification accuracy than SImPa, as shown in Table 4.1 *(Top)* – see row "ABML (Ravi and Beatson, 2019)".

4.5.3 Discussion

As mentioned in Remark 4.2, the loss function ℓ is clipped to $[0, 1]$ to satisfy the assumption made in PAC-Bayes framework. We observed that it affects the training of SImPa, mostly at the early stages. The reason might be the imbalance between the clipped loss and the regularisation terms related to KL divergence, making the learning over-regularised with slow convergence. In the implementation, we first train SImPa without such regularisation terms for 1,000 tasks, and then add them back to the loss with such regularisation. This facilitates the training process, allowing the algorithm to converge faster.

As specified in Subsection 4.4.3, the usage of implicit distribution, and in particular the generator in (4.10) to generate the parameter for the neural network of interest, leads to an exponential increase in the number of learnable parameters. For example, in the classification of mini-ImageNet using the 4-convolutional module neural network, the number of parameter of the base network is 32,645, requiring us to have a generator with 16.7 million parameters. Such a large generator limits the scalability of SImPa, making it inapplicable for larger base networks. One workaround solution might be to utilise the architecture design proposed in Hypernetworks (Ha et al., 2017) that shares parameters to reduce the size of the generator. The trade-off is the slight decreasing of the generator expressiveness when generating the parameter for the base neural network.

Another limitation of SImPa is the need of GPU memory and training time comparing to other meta-learning approaches, such as MAML. In our implementation, the simplest baseline MAML needs 6 GPU-hours to train until convergence, while probabilistic baselines, such as ABML, BMAML and PLATIPUS, take about 30 GPU-hours to converge under the same setting. For SImPa, it requires more than 48 hours to converge. The reason of such long training time lies on the size of the meta-parameter, the need to train the ϕ network to estimate KL divergence, and the number of Monte Carlo samples used.

4.6 Summary

We introduce and formulate a new probabilistic algorithm for few-shot meta-learning. The proposed algorithm, SImPa, is based on PAC-Bayes framework which theoretic-

ally bounds the error induced on unseen tasks and unseen samples within each task. In addition, the proposed method employs a generative approach that implicitly models the posterior of task-specific model parameter $q(\mathbf{w}_i; \lambda_i)$, resulting in more expressive variational approximation compared to the usual variational methods using multivariate normal posteriors with diagonal covariance matrices, such as PLATIPUS (Finn et al., 2018) or ABML (Ravi and Beatson, 2019). The uncertainty, in the form of the learnt implicit distributions, can introduce more variability into the decision made by the model, resulting in well-calibrated and highly-accurate prediction. The algorithm can be combined with different base models that are trainable with gradient-based optimisation, and is applicable in regression and classification. We demonstrate that the algorithm has state-of-the-art calibration and prediction results on unseen data in a multi-modal 5-shot learning regression problem, and achieve state-of-the-art calibration and classification results on few-shot 5-way tasks on mini-ImageNet and tiered-ImageNet data sets.

Chapter 5

Probabilistic task modelling for meta-learning

The main content of this chapter is accepted to present at the Conference on Uncertainty in Artificial Intelligence (UAI) 2021:

> Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2021). 'Probabilistic task modelling for meta-learning'. In: *Conference on Uncertainty in Artificial Intelligence*.

Statement of Authorship

Title of Paper	Probabilistic task modelling for meta-learning
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Cuong Nguyen, Thanh-Toan Do and Gustavo Carneiro (2021). "Probabilistic task modelling for meta-learning". In Conference on Uncertainty in Artificial Intelligence.

Principal Author

Name of Principal Author (Candidate)	Cuong Nguyen		
Contribution to the Paper	- Developed the conception of the paper - Formulated the objective function of the proposed graphical model - Implemented the proposed method - Drafted and revised the paper		
Overall percentage (%)	70		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	20/10/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Thanh-Toan Do		
Contribution to the Paper	- Discussed and refined the idea of the paper - Verified the mathematical formulation - Suggested some ideas to overcome difficulties in the implementation - Wrote and revised the paper		
Signature		Date	20/10/2021

Name of Co-Author	Gustavo Carneiro		
Contribution to the Paper	- Discussed and refined the idea of the paper - Verified the correctness in the formulation of the paper - Suggested some ideas to overcome the difficulties in the implementation - Wrote and revised the paper		
Signature		Date	20/10/2021

Please cut and paste additional co-author panels here as required.

Abstract

We propose *probabilistic task modelling* – a generative probabilistic model for collections of tasks used in meta-learning. The proposed model combines variational auto-encoding and latent Dirichlet allocation to model each task as a mixture of Gaussian distribution in an embedding space. Such modelling provides an explicit representation of a task through its task-theme mixture. We present an efficient approximation inference technique based on variational inference method for empirical Bayes parameter estimation. We perform empirical evaluations to validate the *task uncertainty* and *task distance* produced by the proposed method through correlation diagrams of the prediction accuracy on testing tasks. We also carry out experiments of task selection in meta-learning to demonstrate how the task relatedness inferred from the proposed model help to facilitate meta-learning algorithms.

5.1 Introduction

The latest developments in machine learning have enabled the field to solve increasingly complex classification problems. Such complexity require high capacity models, which in turn need a massive amount of annotated data for training, resulting in an arduous, costly and even infeasible annotation process. This has, therefore, motivated the research of novel learning approaches, generally known as transfer learning, that exploit past experience (in the form of models learned from other training tasks) to quickly learn a new task using relatively small training sets.

Transfer-learning, and in particular, meta-learning, assumes the existent of a task environment where training and testing tasks are i.i.d. sampled from the same latent distribution. By modelling such environment through meta-parameters that are shared across all tasks, meta-learning can solve an unseen task more accurately and efficiently by learning how to solve many tasks generated from the same distribution, even if each task contains a limited number of training examples. Meta-learning has, therefore, progressed steadily with many remarkable state-of-the-art results in several few-shot learning benchmarks (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Yoon et al., 2018; Rusu et al., 2019; Allen et al., 2019). However, current development of meta-learning focuses on solving tasks without providing understanding on how tasks are generated or correlated, potentially leading to sub-optimal solutions. In fact, there is a large variation of prediction performance made by various meta-learning algorithms reported in (Dhillon et al., 2019, Figure 1) or shown in Figure 5.1, suggesting that not all testing tasks are equally related to training tasks. This motivates our work to model and represent tasks in a latent "task-theme" space. The new task representation allows further downstream works,

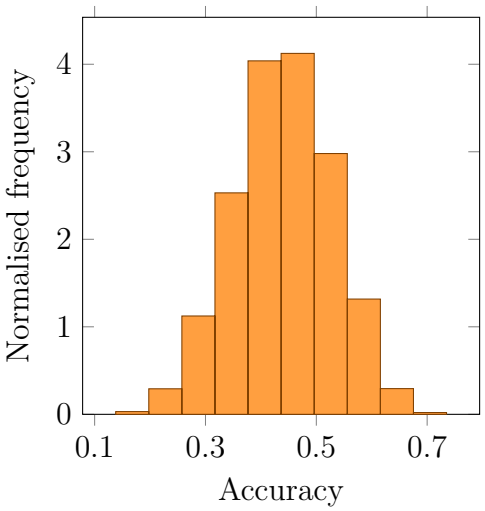


Figure 5.1: The results locally produced for MAML on 15,504 available 5-way 1-shot mini-ImageNet testing tasks vary from 20 to 70 percent accuracy, suggesting that not all testing tasks are equally related to training tasks.

such as task similarity or active task selection, to be developed to gain insights into, or even improve, the prediction performance of different meta-learning algorithms.

In this paper, we propose **probabilistic task modelling** (PTM) – a graphical model that combines variational auto-encoding (VAE) (Kingma and Welling, 2014b) and Gaussian latent Dirichlet allocation (LDA) (Das et al., 2015) – to **model tasks** used in meta-learning. Note that PTM itself is not a meta-learning method. With this modelling approach, the dataset associated with each task can be modelled as a mixture of finite Gaussian distributions, allowing to represent tasks in a latent "task-theme" simplex via its mixture coefficient vector. Such representation provides a convenient way to quantitatively measure "task uncertainty" or task relatedness, which can be utilised in active task selection for meta-learning.

5.2 Related Work

The proposed approach is closely related to Task2Vec (Achille et al., 2019) when modelling tasks for meta-learning. In Task2Vec, a task is represented by an embedding computed from the Fisher information matrix associated with the task-specific classifier. In PTM, a task is represented by the variational distribution of task-theme mixture, which is a part of the graphical model describing the task generation process. The two methods, therefore, differ at the modelling mechanism: Task2Vec follows a deterministic approach, while PTM is a probabilistic method. Such difference provides an advantage of PTM over Task2Vec, which includes modelling uncertainty into the task representation. In addition, PTM is more efficient than Task2Vec at inference when predicting task representation, since PTM only needs a single forward

pass, while Task2Vec requires to re-train or fine-tune the task-specific classifier and calculate the Fisher information matrix for the task that needs to be presented.

Our work is related to task similarity estimation, which has been intensively studied in the field of multi-task learning. Some remarkable examples in this area include task-clustering using k-nearest neighbours (Thrun and O'Sullivan, 1996), or modelling common prior between tasks as a mixture of distributions (Bakker and Heskes, 2003; Xue et al., 2007). Another approach is to formulate multi-task learning as a convex optimisation problem either to cluster tasks and utilise the clustering results to fast track the learning (Jacob et al., 2009), or to learn task relationship through task covariance matrices (Y. Zhang and Yeung, 2012). Other approaches provided theoretical guarantees when learning the similarity or relationship between tasks (Shui et al., 2019). Recently, the *taskonomy* project (Zamir et al., 2018) was conducted to carry out extensive experiments on 26 computer-vision tasks to empirically analyse the correlation between those tasks. Other works (Tran et al., 2019; C. V. Nguyen et al., 2020) take a slightly different approach by investigating the correlation of the label distributions between the tasks of interest to measure task similarity. One commonality among all studies above is their reliance on task-specific classifiers which are used to quantify task relatedness. In contrast, our proposed method explicitly models tasks without the help of any task-specific classifier, making it more efficient in training and prediction.

Our work is also connected to finite mixture models (Pritchard et al., 2000), such as the *latent Dirichlet allocation* (Blei et al., 2003), which analyses and summarises text data in topic modelling, or categorises natural scenes in computer vision (F.-F. Li and Perona, 2005). LDA assumes that each document within a given corpus can be represented as a mixture of finite categorical distributions, where each categorical distribution is a latent topic shared across all documents. Training an LDA model or its variants on a large text corpus is challenging, so several approximate inference techniques have been proposed, ranging from mean-field variational inference (VI) (Blei et al., 2003), collapsed Gibbs' sampling (Griffiths and Steyvers, 2004) and collapsed VI (Teh et al., 2007). Furthermore, several online inference methods have been developed to increase the training efficiency for large corpora (Canini et al., 2009; Hoffman et al., 2010; Foulds et al., 2013). Our work is slightly different from the modelling of the conventional LDA, where we do not use the data directly, but embed it into a latent space. In short, our approach is a combination of VAE (Kingma and Welling, 2014b) and LDA to model the dataset associated with a task. Our approach considers "word" as continuous data, instead of the discrete data represented by a bag-of-word vector generally used by LDA-based topic modelling methods. The resultant model in the embedding latent space is, therefore, similar to the Gaussian LDA (Das et al., 2015) for word embedding in

topic modelling.

5.3 Methodology

To relate **task modelling** to **topic modelling**, we consider **a task** as **a document**, and **a data point** as **a word**. Given these analogies, we can use LDA (Blei et al., 2003) – a popular topic model – to model tasks for meta-learning. However, simply applying LDA for task modelling would not scale well with high-dimensional data and large datasets. We, therefore, propose to employ the VAE (Kingma and Welling, 2014a) to reduce the dimension of the data, and use the inferred embeddings of data as words to model tasks. Due to the nature of VAE, the latent variables are often continuous, not discrete as the bag-of-words used in the conventional LDA. We, therefore, **replace** the *categorical* **word-topic** distributions **in LDA** by *Gaussian* **task-theme**[1] distributions. Given these assumptions, each task can be modelled as a mixture of K Gaussian task-themes, allowing to represent tasks by their inferred *task-theme mixture* vectors in the latent task-theme simplex as illustrated in Figure 5.2. Hence, it is beneficial to utilise this representation for further downstream tasks, such as measuring task similarity.

The graphical model of the proposed PTM is shown in Figure 5.3, where there are T tasks, and each task consists of N data points, denoted as $\mathbf{x}$. To simplify the formulation and analysis, N is assumed to be fixed across all tasks, but the extension of varying N is trivial and can be implemented straightforwardly. Under these assumptions, a task can be generated as follows:

- Initialise the Dirichlet prior for task-theme mixture: $\{\alpha_k\}_{k=1}^K$, where $\alpha \in \mathbb{R}_+$
- Initialise means and covariance matrices of K Gaussian task-themes $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, where $\boldsymbol{\mu}_k \in \mathbb{R}^D$, $\boldsymbol{\Sigma}_k \in \mathbb{R}^{D\times D}$ is positive definite matrix, and D is the dimension of the data embedding
- For task $\mathcal{T}_i$ in the collection of T tasks:
  - Choose a task-theme mixture vector: $\boldsymbol{\pi}_i \sim \text{Dirichlet}(\boldsymbol{\pi}; \boldsymbol{\alpha})$
  - For data point n-th of task $\mathcal{T}_i$:
    - Choose an task-theme assignment one-hot vector: $\mathbf{z}_{in} \sim \text{Categorical}(\mathbf{z}; \boldsymbol{\pi}_i)$
    - Choose an embedding of the data point: $\mathbf{u}_{in} \sim \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where: $z_{ink} = 1$
    - Generate the data point from a decoder h parameterised by θ: $\mathbf{x}_{in} = h(\mathbf{u}_{in}; \theta)$.

[1] "Task-theme" is inspired by F.-F. Li and Perona (2005)

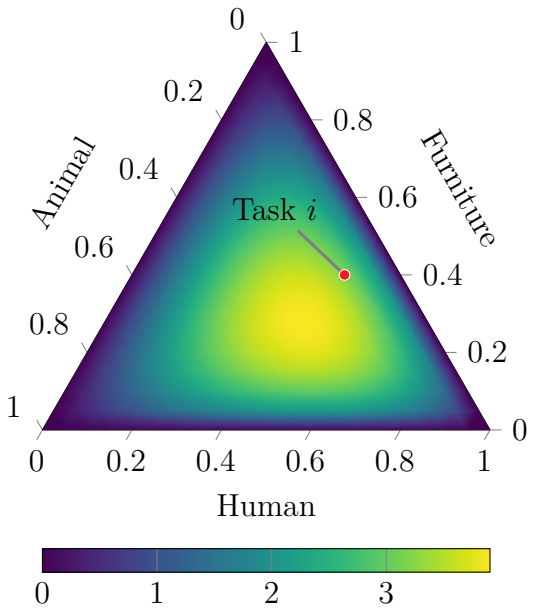


Figure 5.2: An example of a task-theme simplex where each task is represented by a 3-dimensional mixture vector.

To model tasks according to the task generation described above, we need to infer the task-agnostic (or meta) parameters of interest, namely $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}$ and θ. However, due to the complexity of the graphical model shown in Figure 5.3, the exact inference for the posterior $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta|\mathbf{x})$ is intractable, and therefore, the estimation must rely on approximate inference. For simplicity, maximum likelihood estimation (MLE) is used as the objective function:

$$\max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha},\theta} \ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta). \tag{5.1}$$

Although MLE simplifies the learning for the meta-parameters of interest, the log-likelihood in (5.1) is still difficult to evaluate due to the presense of the latent variables $\boldsymbol{\pi}_i$ and $\mathbf{z}_{in}$. One workaround solution is to find its lower-bound, and maximise the lower-bound instead of maximising the log-likelihood itself. This approach is analogous to the variational inference, which has been widely used to infer the latent parameters of VAE and LDA models.

Since the proposed PTM is a combination of VAE and LDA, the derivation for the lower-bound of the likelihood in (5.1) can be divided into two steps, where the first step is analogous to the lower bound of a VAE, and the second step is similar to the plain LDA model.

In the first step, the latent variable $\mathbf{u}$ is introduced, so that the log-likelihood $\ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta)$ can be bounded below by Jensen's inequality:

$$\ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta) \geq \mathsf{L}_{\text{VAE}}, \tag{5.2}$$

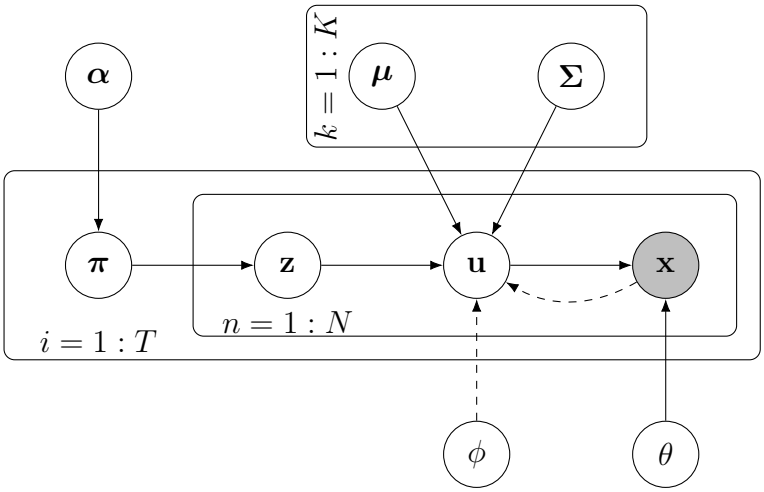


Figure 5.3: The graphical model used in task modelling. The solid arrows denote data generation, while the dashed arrows stand for inference. The boxes are "plates" representing replicates. The shading nodes denote observable variables, while the white nodes denote latent variables.

where the lower-bound is defined as:

$$\mathsf{L}_{\text{VAE}} = \mathbb{E}_{q(\mathbf{u})}\left[\ln p\left(\mathbf{x}|\mathbf{u},\theta\right) + \ln p\left(\mathbf{u}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha}\right) - \ln q(\mathbf{u})\right], \tag{5.3}$$

with $q(\mathbf{u})$ being the variational distribution for the latent variable $\mathbf{u}$.

Following the conventional VI for VAE (Kingma and Welling, 2014a), the variational distribution for the data embedding $\mathbf{u}$ is assumed to be a Gaussian distribution with diagonal covariance matrix:

$$q(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}; \mathbf{m}, \text{diag}\left(\mathbf{s}^2\right)\right), \tag{5.4}$$

where diag(.) denotes a diagonal matrix.

In addition, the parameters $\mathbf{m}$ and $\mathbf{s}$, which represent the distribution encoding the data $\mathbf{x}$, are modelled by a neural network (also known as an encoder) f parameterised by ϕ:

$$\begin{bmatrix}\mathbf{m}^\top & \mathbf{s}^\top\end{bmatrix}^\top = f(\mathbf{x};\phi). \tag{5.5}$$

Hence, instead of maximising the marginal log-likelihood in (5.1), the lower-bound in (5.3) is maximised, resulted in the alternative objective:

$$\max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha},\theta} \max_{\phi} \mathsf{L}_{\text{VAE}}. \tag{5.6}$$

One difficulty in maximising the lower-bound in (5.6) is the evaluation for the embedding prior $\ln p\left(\mathbf{u}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha}\right)$ in Eq. (5.3). In vanilla VAE, the embedding prior is often modelled as some standard distributions, such as Gaussian or Beta, resulting

in a tractable solution. In this paper, the prior is modelled as a Gaussian mixture model, making the solution intractable. However, since this prior is the marginal log-likelihood in the conventional LDA model, we can apply techniques developed for LDA methods to approximate this term. Here, we employ the VI approach in which the term is bounded below by Jensen's inequality:

$$\ln p\left(\mathbf{u}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}\right) \geq \mathsf{L}_{\text{LDA}}(\mathbf{u}, q(\mathbf{z}, \boldsymbol{\pi})), \tag{5.7}$$

where:

$$\begin{aligned}\mathsf{L}_{\text{LDA}}\left(\mathbf{u}, q(\mathbf{z}, \boldsymbol{\pi})\right) = \mathbb{E}_{q(\mathbf{z},\boldsymbol{\pi})}\left[\ln p(\mathbf{u}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \ln p(\mathbf{z}|\boldsymbol{\pi}) + \ln p(\boldsymbol{\pi}|\boldsymbol{\alpha})\right. \\ \left. - \ln q(\mathbf{z}) - \ln q(\boldsymbol{\pi})\right],\end{aligned} \tag{5.8}$$

with $q(\mathbf{z}, \boldsymbol{\pi})$ being the variational distribution for $\mathbf{z}$ and $\boldsymbol{\pi}$. This corresponds to the second step in the derivation.

Similar to LDA (Blei et al., 2003), the variational distribution $q(\mathbf{z}, \boldsymbol{\pi})$ is assumed to be fully factorised and followed the conjugate priors:

$$q(\mathbf{z}, \boldsymbol{\pi}) = \prod_{i=1}^{T} q(\boldsymbol{\pi}_i; \boldsymbol{\gamma}_i) \prod_{n=1}^{N} q(\mathbf{z}_{in}; \mathbf{r}_{in}), \tag{5.9}$$

where:

$$q(\boldsymbol{\pi}_i; \boldsymbol{\gamma}_i) = \text{Dirichlet}\left(\boldsymbol{\pi}_i; \boldsymbol{\gamma}_i\right) \tag{5.10}$$

$$q(\mathbf{z}_{in}; \mathbf{r}_{in}) = \text{Categorical}\left(\mathbf{z}_{in}; \mathbf{r}_{in}\right), \tag{5.11}$$

with $\mathbf{r}$ and $\boldsymbol{\gamma}$ being the parameters of the variational distribution $q(\mathbf{z}, \boldsymbol{\pi})$.

In practice, $q(\mathbf{z}, \boldsymbol{\pi})$ is obtained as the maximiser of the lower-bound $\mathsf{L}_{\text{LDA}}\left(\mathbf{u}, q(\mathbf{z}, \boldsymbol{\pi})\right)$ on the embedding data $\mathbf{u}$. It is, however, inapplicable in this case, since the data embedding $\mathbf{u}$ is used twice: one to optimise $q(\mathbf{z}, \boldsymbol{\pi})$, and the other is to optimise the objective in (5.6), which may result in overfitting. To avoid this issue, we employ the *empirical Bayes* approach relying on the train-test split method, where one half of data in a task, denoted as $\mathbf{u}^{(t)}$, is used to obtain $q(\mathbf{z}, \boldsymbol{\pi})$, while the other half, denoted as $\mathbf{u}^{(v)}$, is used for the optimisation in (5.6). This approach is analogous to the *empirical Bayes* meta-learning (Finn et al., 2017; C. Nguyen et al., 2020), where one part of data is used for task-adaptation (often known as "inner-loop"), while the other part is used to learn the meta-parameter (often known as "outer-loop").

Given this modelling approach, the objective function can be formally written as

a bi-level optimisation:

$$\begin{aligned}&\max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\alpha},\theta,\phi} \mathsf{L}\left(\mathbf{u}^{(v)}, q^*\left(\mathbf{z},\boldsymbol{\pi}\right)\right)\\ &\text{s.t.: } q^*\left(\mathbf{z},\boldsymbol{\pi}\right) = \arg\max_{q(\mathbf{z},\boldsymbol{\pi})} \mathbb{E}_{q(\mathbf{u}^{(t)};\phi)}\left[\mathsf{L}_{\text{LDA}}\left(\mathbf{u}^{(t)}, q(\mathbf{z},\boldsymbol{\pi})\right)\right],\end{aligned} \tag{5.12}$$

where

$$\begin{aligned}\mathsf{L}\left(\mathbf{u}^{(v)}, q^*\left(\mathbf{z},\boldsymbol{\pi}\right)\right) = \mathbb{E}_{q(\mathbf{u}^{(v)};\phi)}\Big[&\mathsf{L}_{\text{LDA}}\left(\mathbf{u}^{(v)}, q^*(\mathbf{z},\boldsymbol{\pi})\right) + \ln p\left(\mathbf{x}^{(v)}|\mathbf{u}^{(v)},\theta\right)\\ &- \ln q\left(\mathbf{u}^{(v)};\phi\right)\Big].\end{aligned} \tag{5.13}$$

Due to the assumptions made in Eqs. (5.4), (5.10) and (5.11), prior conjugate can be applied to simplify the evaluation for all the terms in (5.8) w.r.t. the variational distribution $q(.)$. Details of the evaluation can be referred to Appendix C.1. In addition, the optimisation for the meta-parameters in (5.12) is based on gradient ascent, and carried out in two steps, resulting in a process analogous to the expectation-maximisation (EM) algorithm. In the E-step (corresponding to the optimisation for the lower-level in (5.12)), the task-specific variational-parameters $\mathbf{r}$ and $\boldsymbol{\gamma}$ are iteratively updated, while holding the meta-parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta$ and ϕ fixed. In the M-step (corresponding to the optimisation for the upper-level), the meta-parameters are updated using the values of the task-specific variational-parameters obtained in the E-step. Note that the inference for the task-theme parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are similar to the estimation of Gaussian mixture model (Bishop, 2006, Chapter 9). Please refer to Appendix C.2 for more details on the optimisation.

Conventionally, the iterative updates in the E-step and M-step require a full pass through the entire collection of tasks. This is, however, very slow and even infeasible since T is often in the magnitude of millions. We, therefore, propose an online VI inspired by the online learning for LDA (Hoffman et al., 2010) to infer the meta-parameters. For each task $\mathcal{T}_i$, we perform the EM to obtain the "task-specific" parameters (denoted by a tilde on top of variables) that are locally optimal for that task. The "meta" parameters of interest are then updated as a weighted average between their previous values and the "task-specific" values:

$$\begin{aligned}\boldsymbol{\mu} &\leftarrow (1-\rho_i)\boldsymbol{\mu} + \rho_i\tilde{\boldsymbol{\mu}}\\ \boldsymbol{\Sigma} &\leftarrow (1-\rho_i)\boldsymbol{\Sigma} + \rho_i\tilde{\boldsymbol{\Sigma}}\\ \boldsymbol{\alpha} &\leftarrow \boldsymbol{\alpha} - \rho_i \underbrace{\tilde{\boldsymbol{\alpha}}_i}_{\mathbf{H}^{-1}\mathbf{g}},\end{aligned} \tag{5.14}$$

where $\rho_i = (\tau_0 + i)^{-\tau_1}$ with $\tau_0 \geq 0$ and $\tau_1 \in (0.5, 1]$ (Hoffman et al., 2010), $\mathbf{g}$ is the gradient of L_{LDA} w.r.t. $\boldsymbol{\alpha}$, and $\mathbf{H}$ is the Hessian matrix. The learning for the encoder ϕ and the decoder θ follows the conventional learning by stochastic gradient ascent.

Algorithm 4 Online probabilistic task modelling

procedure TRAINING
  Initialise LDA parameters: $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \alpha_k\}_{k=1}^K$
  Initialise encoder ϕ and decoder θ
  for each mini-batch of T_{mini} tasks **do**
    for $i = 1 : T_{\text{mini}}$ **do**
      Split data into $\{\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}\}$ and $\{\mathbf{x}_i^{(v)}, \mathbf{y}_i^{(v)}\}$
      $\mathbf{m}_i^{(t)}, \mathbf{s}_i^{(t)} \leftarrow f(\mathbf{x}_i^{(t)}; \phi)$
      $\mathbf{m}_i^{(v)}, \mathbf{s}_i^{(v)} \leftarrow f(\mathbf{x}_i^{(v)}; \phi)$
      $\boldsymbol{\gamma}, \mathbf{r} \leftarrow$ E-STEP$(\mathcal{N}(\mathbf{u}; \mathbf{m}_i^{(t)}, (\mathbf{s}_i^{(t)})^2\mathbf{I}))$
      Calculate $\mathsf{L}\left(\mathbf{u}_i^{(v)}, q_i^*(\mathbf{z}, \boldsymbol{\pi})\right)$ ▷ Eq. (5.13)
      Calculate "local" task-themes $\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i, \tilde{\boldsymbol{\alpha}}_i$
    end for
    $\overline{\mathsf{L}} = \frac{1}{T}\sum_{i=1}^T \mathsf{L}\left(\mathbf{u}_i^{(v)}, q_i^*(\mathbf{z}, \boldsymbol{\pi})\right)$
    $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha} \leftarrow$ online_LDA $\left(\tilde{\boldsymbol{\mu}}_{1:T}, \tilde{\boldsymbol{\Sigma}}_{1:T}, \tilde{\boldsymbol{\alpha}}_{1:T}\right)$
    $\theta, \phi \leftarrow$ SGD $\left(-\overline{\mathsf{L}}\right)$ ▷ gradient ascent
  end for
  return $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\alpha}, \theta, \phi$
end procedure

procedure E-STEP$(\mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{s}^2\mathbf{I}))$
  Initialise $\mathbf{r}, \boldsymbol{\gamma}$
  repeat
    calculate the un-normalised r_{ink} ▷ Eq. (C.10)
    normalise $\mathbf{r}_{in}$ such that $\sum_{k=1}^K r_{ink} = 1$
    calculate γ_{ik} ▷ Eq. (C.14)
  until $\frac{1}{K}$ |change in $\boldsymbol{\gamma}$| < threshold
  return $\boldsymbol{\gamma}, \mathbf{r}$
end procedure

The complete learning algorithm for the proposed probabilistic task modelling is shown in Algorithm 4.

Also, instead of updating the meta-parameters as in (5.14) when observing a single task, we use multiple or a mini-batch of tasks to reduce the effect of measurement noise. The mini-batch version requires a slight modification in the formulation presented above, where we calculate the average of all "task-specific" parameters for tasks in the same mini-batch, and use that as the "task-specific" value to update the corresponding "meta" parameters.

Although the "reconstruction" term $\ln p(\mathbf{x}^{(v)}|\mathbf{u}^{(v)}, \theta)$ in (5.12) is used to model the likelihood of un-labelled data, it can straightforwardly be extended to a labelled data pair $(\mathbf{x}^{(v)}, \mathbf{y}^{(v)})$ by introducing the parameter $\mathbf{w}$ of a classifier. In that case, the

"reconstruction" term can be expressed as:

$$\ln p(\mathbf{x}^{(v)}, \mathbf{y}^{(v)} | \mathbf{u}^{(v)}, \theta, \mathbf{w}) = \underbrace{\ln p(\mathbf{y}^{(v)} | \mathbf{u}^{(v)}, \mathbf{w})}_{\text{negative classification loss}} + \underbrace{\ln p(\mathbf{x}^{(v)} | \mathbf{u}^{(v)}, \theta)}_{\text{negative reconstruction loss}} . \quad (5.15)$$

In general, $\mathbf{w}$ can be either a task-specific parameter generated from an additional meta-parameter shared across all tasks – corresponding to *empirical Bayes* meta-learning (e.g. using train-test split to learn hyper-parameters) algorithms (Finn et al., 2017; C. Nguyen et al., 2020), or a meta-parameter itself – corresponding to *metric* meta-learning (Vinyals et al., 2016; Snell et al., 2017). For simplicity, we will use the latter approach relying on the prototypical network (Snell et al., 2017) with Euclidean distance on the data embedding $\mathbf{u}$, to calculate the classification loss on labelled data. This reduces the need to introduce an additional parameter $\mathbf{w}$ into our modelling.

Task representation

Given the inferred meta-parameters, including the task-themes $\{(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^K$, the Dirichlet prior $\{\boldsymbol{\alpha}_l\}_{l=1}^L$, the encoder ϕ and the decoder θ, we can embed the data of a task into a latent space, and calculate its variational Dirichlet posterior of the task-theme *mixing coefficients* $q(\boldsymbol{\pi}; \boldsymbol{\gamma}_i)$. The obtained distribution can be used represent the corresponding task in the latent task-theme simplex as illustrated in Figure 5.2. This new representation of tasks has two advantages comparing to the recently proposed task representation Task2Vec (Achille et al., 2019): (i) it explicitly models and represents tasks without the need of any pre-trained networks to use as a "probe" network, and (ii) it uses a probability distribution, instead of a vector as in Task2Vec, allowing to include modelling uncertainty when representing tasks. Given the probabilistic nature of PTM, we can use the entropy of the inferred task-theme mixture distribution $q(\boldsymbol{\pi}; \boldsymbol{\gamma}_i)$ as a measure of *task uncertainty*. In Subsection 5.4.1, we empirically show that this measure correlates to the generalisation or test performance.

In addition, the representation produced by PTM can be used to quantitatively analyse the similarity or *distance* between two tasks i and j through a divergence between $q(\boldsymbol{\pi}; \boldsymbol{\gamma}_i)$ and $q(\boldsymbol{\pi}; \boldsymbol{\gamma}_j)$. Commonly, symmetric distances, such as Jensen-Shannon divergence, Hellinger distance, or earth's mover distance are employed to calculate the divergence between distributions. However, it is argued that similarity should be represented as an asymmetric measure (Tversky, 1977). This is reasonable in the context of transfer learning, since knowledge gained from learning a difficult task might significantly facilitate the learning of an easy task, but the reverse might not always have the same level of effectiveness. In light of asymmetric distance, we

decide to use Kullback-Leibler (KL) divergence, denoted as KL[.‖.], to measure *task distance*. As KL $[P\|Q]$ is defined as the information lost when using a code optimised for Q to encode the samples of P, we, therefore, calculate KL $[q(\boldsymbol{\pi};\boldsymbol{\gamma}_{T+1})\|q(\boldsymbol{\pi};\boldsymbol{\gamma}_i)]$, where $i \in \{1,\ldots,T\}$, to assess how the training task $\mathcal{T}_i$ differs from the learning of the novel task $\mathcal{T}_{T+1}$.

5.4 Experiments

In this section, we empirically validate the two properties of PTM – *task uncertainty* and *task distance* – through task distance matrix and correlation diagrams. We also show two applications of the proposed approach used in active task selection for inductive and transductive life-long meta-learning. The experiments are based on the n-way k-shot tasks formed from Omniglot (Lake et al., 2015) and mini-ImageNet (Vinyals et al., 2016) – the two widely used datasets to evaluate the performance of meta-learning algorithms.

The Omniglot dataset consists of 1,623 different handwritten characters from 50 different alphabets, where each character was drawn in black and white by 20 different people. Instead of using random train-test split that mixes all characters, the original split (Lake et al., 2015) is used to yield finer-grained classification tasks. In addition to the task forming based on randomly mixing characters of many alphabets, the two-level hierarchy of alphabets and characters are utilised to increase the difficulty of the character classification. Note that no data augmentations, such as rotating images by multiples of 90 degrees, is used throughout the experiments. Also, all images are down-sampled to 64-by-64 pixel^2 to simplify the image reconstruct in the decoder.

The mini-ImageNet dataset comprises a small version of ImageNet, which contains 100 classes taken from ImageNet, and each class has 600 colour images. We follow the common train-test split that uses 64 classes for training, 16 classes for validation, and 20 classes for testing (Ravi and Larochelle, 2018). Similar to Omniglot, all images are also in 64-by-64 pixel^2.

The encoder used in the experiments consists of 4 convolutional modules, where each module has a convolutional layer with 4-by-4 filters and 2-by-2 stride, followed by a batch normalisation and a leaky rectified linear activation function with a slope of 0.01. The output of the last convolutional layer is flattened and connected to a fully connected layer to output the desired dimension for the latent variable $\mathbf{u}$. The decoder is designed similarly, except that the convolutional operator is replaced by the corresponding transposed convolution. For the Omniglot dataset, the number of filters within each convolutional layer of the encoder is 8, 16, 32, and 64, respectively, and the dimension of $\mathbf{u}$ is 64. For mini-ImageNet dataset, these numbers are 32,

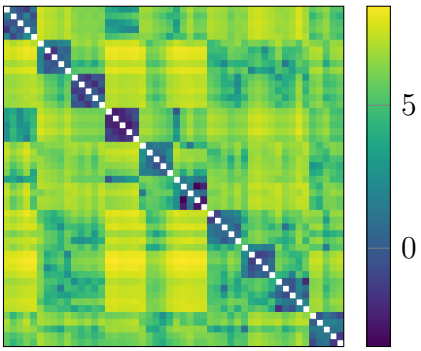


Figure 5.4: The matrix of log KL distances between Omniglot tasks shows that tasks that are generated from the same alphabet are closer together, denoted as the dark green blocks along the diagonal. The matrix is asymmetric due to the asymmetry of the KL divergence used as the task distance.

64, 128 and 256, and the dimension of $\mathbf{u}$ is 128. The reconstruction loss follows the negative log-likelihood of the continuous Bernoulli distribution (Loaiza-Ganem and Cunningham, 2019), which is often known as binary cross-entropy, while the classification loss is based on the prototypical network used in metric learning. The training subset of each task, $\mathbf{u}_i^{(t)}$, is used to calculate the class prototypes, and the classification loss is based on the soft-max function of the distances between the encoding of each input image to those prototypes (Snell et al., 2017). The optimiser used is Adam with the step size of 2×10^{-4} to optimise the parameters of the encoder and decoder after every mini-batch consisting of 20 tasks. For the LDA part, a total of $K = 8$ task-themes is used. The Dirichlet prior is assumed to be symmetric with a concentration $\alpha = 1.1$ across both datasets. The parameters of the learning rate used in the online LDA are $\rho_0 = 10^6$ and $\rho_1 = 0.5$. A total of 10^6 episodes are used to train PTM on both datasets. We note that setting $\alpha > 1$ enforces every task to be modelled as a mixture of many task-themes, avoiding the task-themes collapsing into a single task-theme during training. The phenomenon of task-theme collapse when $\alpha < 1$ is not observed in LDA, but in PTM due to the integration of VAE. At the beginning of training, the encoder is inadequate, producing mediocre embedding features. The resulting features, combined with $\alpha < 1$, makes a task more likely to be represented by a single task-theme. By learning solely from that task-theme, the encoder is pushed to bias further toward to that task-theme, making only one task-theme distribution updated, while leaving others unchanged. When $\alpha > 1$, all the task-themes contribute to the representation of a task, so they can be learnt along with the encoder.

5.4.1 Task distance matrix and correlation diagrams

Task distance matrix is used as one of the tools to qualitatively validate the prediction made by PTM. In particular, the hypothesis is that the PTM would predict small

distances for tasks that are close together. Since the "labels" specifying the closedness of tasks are unknown, we utilise the hierarchical structure of Omniglot dataset to form tasks. Each task is generated by firstly sampling an alphabet, and then choosing characters in that alphabet. Under this strategy, tasks formed from the same alphabet would have small distances comparing to tasks from different alphabets. Figure 5.4 shows the task distances between 50 testing tasks of Omniglot dataset, where each block of 5 tasks on rows and columns of the task distance matrix corresponds to a group of tasks sampled from the same alphabet. The result, especially the square 5-task-by-5-task blocks along the diagonal, agrees well with the hypothesis. Note that the distance matrix shown in Figure 5.4 is asymmetric due to the asymmetric nature of the KL divergence used to measure task distance.

We use a correlation diagram between prediction accuracy made by MAML and the *task entropy* produced by PTM as another verification. Since the *task entropy* denotes the uncertainty when modelling a task, we hypothesise that it proportionally relates to the difficulty when learning that task. To construct the correlation diagram, we firstly train a meta-learning model based on MAML using the training tasks of the two datasets, and evaluating the performance on 100 random testing tasks. Secondly, we calculate the *task entropy* for those 100 testing tasks. Finally, we plot the prediction accuracy and task entropy in Figures 5.5a and 5.5b. The results on both datasets show that the higher the task uncertainty, the worse the test performance. This observation, therefore, agrees with our hypothesis about *task entropy*.

We conduct another correlation diagram between training-testing task distance and the test performance to verify further the proposed PTM. Our hypothesis is the inverse proportion between training-testing task distance and prediction accuracy. A similar experiment as in task uncertainty is carried out with a modification in which the task uncertainty is replaced by the average KL divergence between all training tasks to each testing task. Due to the extremely large number of training tasks, e.g. more than 10^{12} unique 5-way tasks can be generated from both the two datasets, the calculation of the distance measure is infeasible. To make the training and testing tasks manageable, we randomly generate $10,000$ tasks for training, and 100 tasks for testing. This results in $1,000,000$ distances, which can be calculated in parallel with multiple computers. A testing task can be represented in the correlation diagram through its prediction accuracy and the average KL distance to training tasks, which is defined as:

$$\overline{\mathrm{KL}}(\boldsymbol{\gamma}_{T+1}) = \frac{1}{T}\sum_{i=1}^{T} \mathrm{KL}[q(\boldsymbol{\pi};\boldsymbol{\gamma}_{T+1}) \| q(\boldsymbol{\pi};\boldsymbol{\gamma}_i)].$$

The correlation diagrams for both datasets are then plotted in Figures 5.5c and 5.5d. The results agree well with our hypothesis, in which the further a testing task is

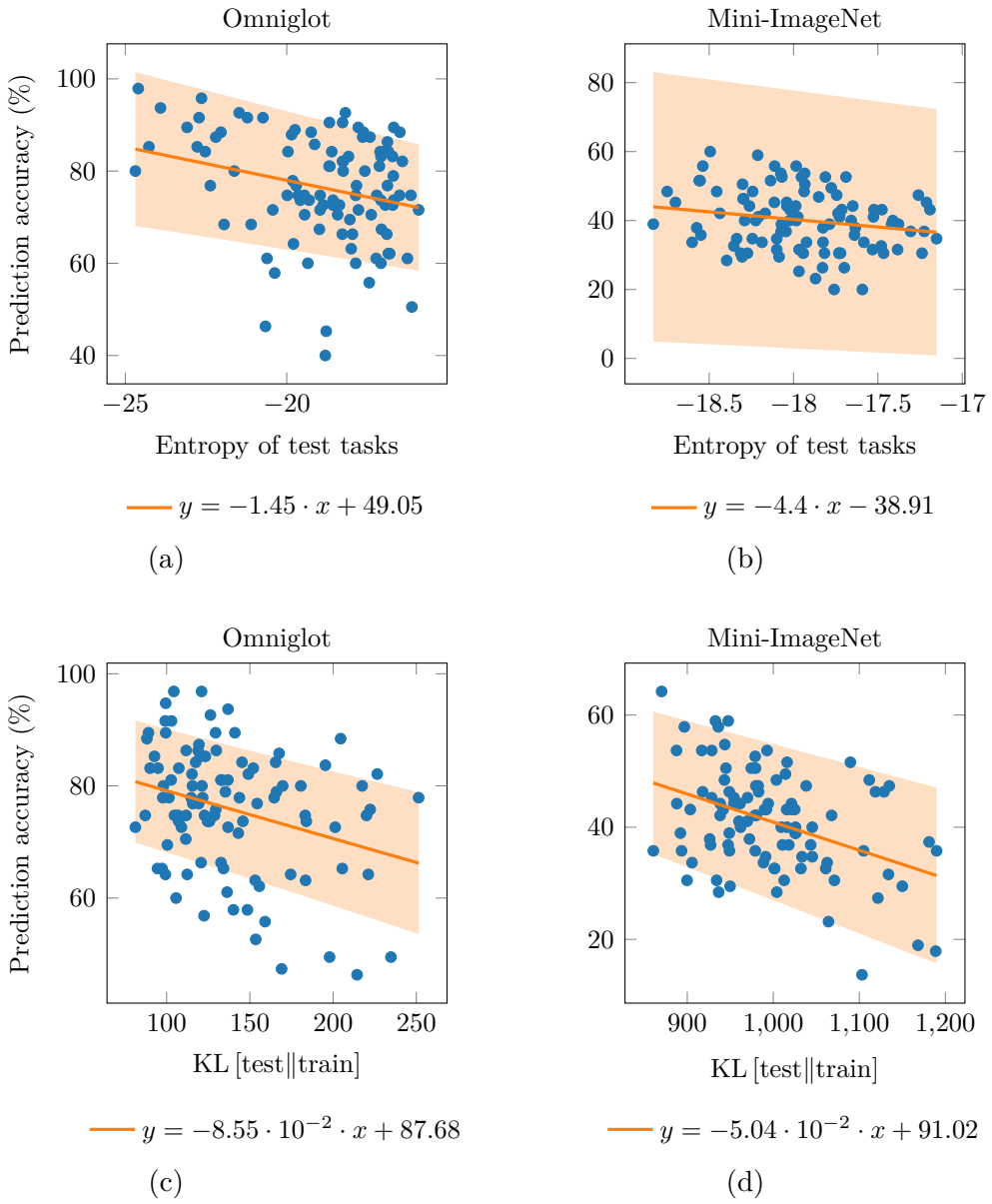


Figure 5.5: Correlation diagrams between prediction accuracy made by MAML on 100 5-way 1-shot testing tasks versus: (a) and (b) entropy of the inferred task-theme mixture distributions, and (c) and (d) the KL distances from testing to training tasks. The results show that largest the task entropy or distances, the worse the testing performance. The blue dots are the prediction made the MAML and PTM, the solid line is the mean of Bayesian Ridge regression, and the shaded areas correspond to ±1 standard deviation around the mean.

from the training tasks, the worse the prediction accuracy. This enables us to use the new representation produced by PTM to analyse task similarity.

5.4.2 Lifelong few-shot meta-learning

To further evaluate PTM, we conduct experiments following the lifelong learning framework (Ruvolo and Eaton, 2013) with slight modification where the supervised

tasks are replaced by 5-way 1-shot learning episodes. More precisely, the setting consists of a meta-learning model and a pool of T_{pool} tasks. At each time step, a task selected from the pool is used to update the meta-learning model, and discarded from the pool. A new task is then added to the pool to maintain T_{pool} tasks available for learning. The criterion for selecting a task to update the meta-learning model will depend on the objective of interest. Two common objectives often observed in practice are:

- **Induction:** the selected training task is expected to encourage the meta-learning model to be able to rapidly adapt to **any** future task,
- **Transduction:** the selected training task is targeted toward one or many **specific** testing tasks.

In the induction setting, the performance of the meta-learning model trained on tasks selected by PTM is compared with three baselines: Task2Vec (Achille et al., 2019), the "worst-case" approach (Collins et al., 2020) and random selection. For the PTM, the selection criteria is based on the task entropy specified in Section 5.3, where the training task with highest entropy is chosen for the learning. For Task2Vec, tasks with large embedding norm are reported as difficult to learn. Hence, we pick the one with the largest L1 norm produced by Task2Vec as the training task. Originally, Task2Vec requires fine-tuning a pre-trained network (known as probe network) on labelled data of a task. This fine-tuning step is, however, infeasible for few-shot learning due to the insufficient number of labelled data. We address this issue by training a MAML-based network to use as a probe network. When given few-shot data of a training task, the MAML-based probe network perform gradient update to adapt to that task. The task-specific embedding can, therefore, be calculated using the adapted probe network. We follow the Monte Carlo approach specified in the public code of Task2Vec to calculate the corresponding task embedding. For the "worst-case" approach, the training task that results in the highest loss for the current meta-learning model is selected. Due to this nature, the "worst-case" approach requires to evaluate all losses for each task in the pool at every time step, leading to an extensive computation and might not scale well when the number of tasks in the pool is large. For simplicity, we use MAML to train the meta-learning model of interest for each selection strategy.

The transduction setting follows a similar setup as the induction case, but the testing tasks, including the labelled and unlabelled data, are known during training. For PTM, the average KL distances between all testing tasks to each training task in the task pool are calculated, and the training task with smallest average distance is selected. For Task2Vec, the proposed cosine distance between normalised task embeddings is used to calculate the average distance between all testing tasks to

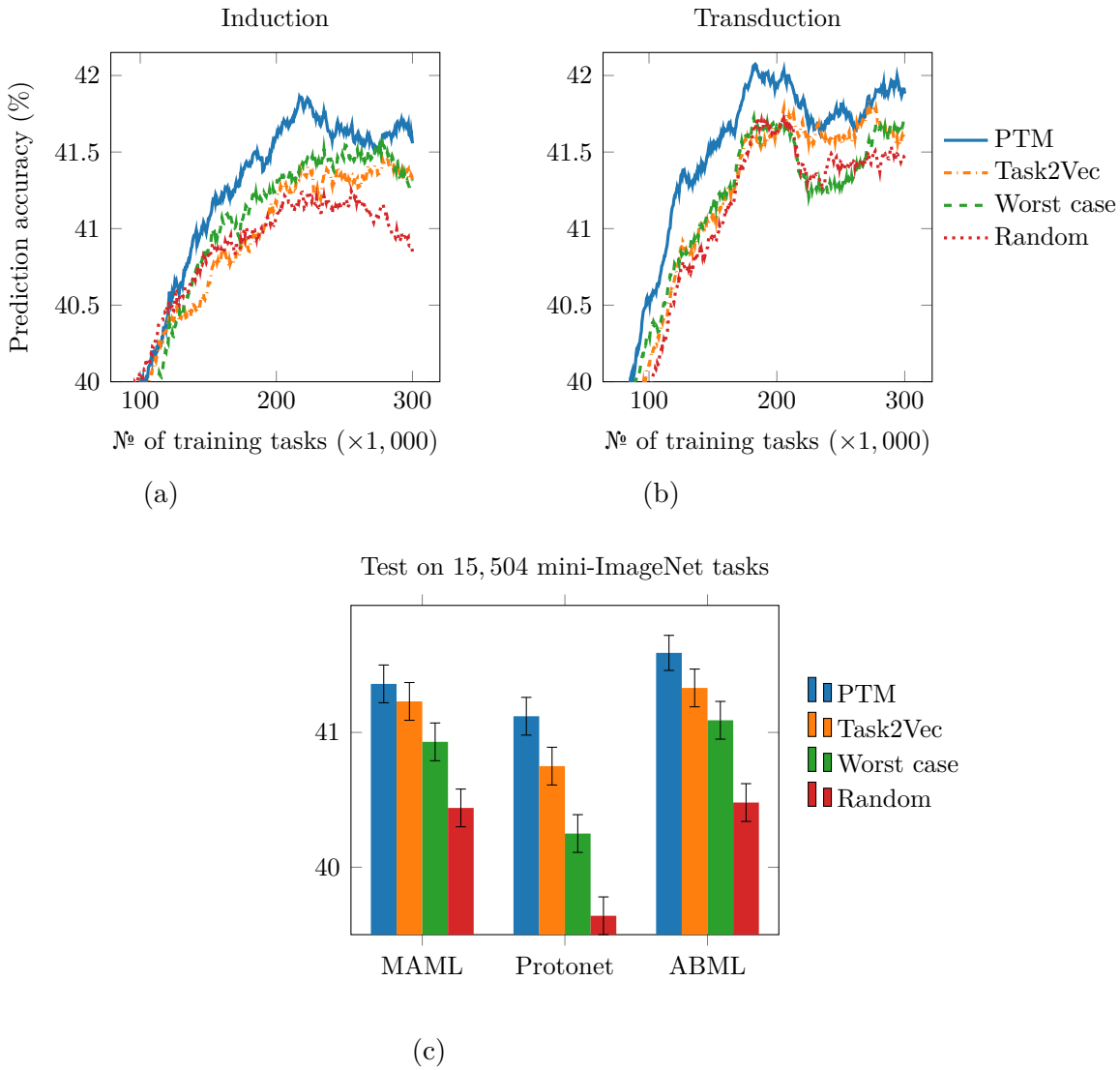


Figure 5.6: Exponential weighted moving average (EWMA) of prediction accuracy made by MAML following the lifelong learning for 100 random 5-way 1-shot tasks sampled from mini-ImageNet testing set: (a) inductive setting, and (b) transductive setting. The EWMA weight is set to 0.98 to smooth the noisy signal. (c) Prediction accuracy made by models trained on different task selection approaches on all 5-way 1-shot testing tasks generated from mini-ImageNet. The error bars correspond to 95 percent confident interval.

each training task (Achille et al., 2019). Similar to PTM, the training task with the smallest distance is prioritised for the learning. For the "worst-case" approach, the entropy of the prediction $\hat{y}$ on C-way testing tasks is used as the measure:

$$S_{T+1} = -\sum_{c=1}^{C} \hat{y}_c \ln \hat{y}_c,$$

and the task that contributed to the highest entropy at prediction is chosen (MacKay, 1992). The "worst-case" approach, therefore, requires T_{pool} trials at every time step. In each trial, the current meta-model is adapted to each training task in the pool,

and then the average prediction entropy on all testing tasks is calculated. This results in an extremely extensive computation.

Four MAML-based meta-learning models are initialised identically and trained on the tasks selected from a pool of $T_{\text{pool}} = 200$ tasks according to the four criteria mentioned above. Figures 5.6a and 5.6b show the testing results on 100 random mini-ImageNet tasks after every 500 time steps. Note that the plotted results are smoothed by the exponential weighted moving average with a weight of 0.98 to ease the visualisation. In general, PTM, Task2Vec and "worst-case" can generalise better than random task selection. In addition, the model trained with tasks chosen by PTM performs slightly better than Task2Vec and the "worst-case" approach in both settings. This observation might be explained based on the designated purpose of Task2Vec and the "worst-case" approach. Task2Vec requires a sufficient number of labelled data to fine-tune its probe network to calculate task embedding. Hence, it might not work well in few-shot learning. For the "worst-case", tasks are selected according to a measure based on the current meta-model without taking task relatedness into account. PTM, however, has a weakness in active selection since the approach only focuses on task uncertainty or task similarity without considering the current state of the meta-learning model. Nevertheless, PTM still provides a good selection criterion comparing to Task2Vec and the "worst-case" approaches. Note that although the active task selection is able to select the best task within the pool, there might be the case where all remaining tasks in the pool are uninformative, resulting in overfitting as observed in Figure 5.6a. However, for simplicity, no additional mechanism is integrated to decide whether to learn from the selected task, or simply discarded from the pool. We believe that adding L2 regularisation or applying early stopping based on a validation set of tasks will help with this overfitting issue.

To further compare, we implement two additional meta-learning algorithms: Prototypical Networks (Snell et al., 2017) and Amortised Bayesian meta-learning (ABML) (Ravi and Beatson, 2019) and show results for the induction setting on all available testing 5-way 1-shot tasks of mini-ImageNet in Figure 5.6c. Again, the prediction accuracy made by the model trained on tasks selected by PTM outperforms other baselines, especially the random one by a large margin.

5.5 Summary

We propose a generative approach based on variational auto-encoding and LDA adopted in topic modelling to model tasks used in meta-learning. Under this modelling approach, the dataset associated with a task can be expressed as a mixture model of finite Gaussian distributions, where each task differs at the mixture coefficients. An online VI method is presented to infer the parameters of the Gaussian

task-theme distributions. The obtained model allows us to represent a task by its variational distribution of mixture coefficient in a latent task-theme simplex, enabling the quantification of either the task uncertainty or task similarity for active task selection.

Chapter 6

Task weighting for meta-learning using trajectory optimisation

The content of this chapter is submitted to the International Conference on Machine Learning (ICML) 2022.

Note that in this chapter, we no longer use the notations defined in Chapter 1. Instead, we follow the ones used in trajectory optimisation which are explicitly defined in the following parts of this chapter.

Statement of Authorship

Title of Paper	
Publication Status	☐ Published ☐ Accepted for Publication ☑ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	

Principal Author

Name of Principal Author (Candidate)			
Contribution to the Paper			
Overall percentage (%)			
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author			
Contribution to the Paper			
Signature		Date	

Name of Co-Author			
Contribution to the Paper			
Signature		Date	20/10/2021

Please cut and paste additional co-author panels here as required.

Abstract

Developing meta-learning algorithms that are un-biased toward a subset of training tasks often requires hand-designed criteria to weight tasks, potentially resulting in sub-optimal solutions. In this paper, we introduce a new principled and fully automated task-weighting algorithm for meta-learning methods. By considering the weights of tasks within the same mini-batch as an action, and the meta-parameter of interest as the system state, we cast the task-weighting meta-learning problem to a trajectory optimisation and employ the iterative linear quadratic regulator to optimise for the action or the weights of tasks. We theoretically show that the proposed algorithm converges to an ϵ-stationary point, and empirically demonstrate that the proposed approach out-performs common hand-engineering weighting methods in two few-shot learning benchmarks.

6.1 Introduction

Meta-learning has been studied from the early 1990s (Schmidhuber, 1987; Naik and Mammone, 1992; Thrun and L. Pratt, 1998) and recently gained a renewed interest with the use of deep learning methods that achieves remarkable state-of-art results in several few-shot learning benchmarks (Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017; Nichol et al., 2018; Ravi and Beatson, 2019; Allen et al., 2019; Khodak et al., 2019; Baik et al., 2020; Flennerhag et al., 2020). However, the majority of existing meta-learning algorithms simply minimise the average loss evaluated on validation subsets of training tasks, implicitly assuming that all training tasks are evenly distributed. This assumption is hardly true in practice, and potentially biases the trained meta-learning models toward tasks observed more frequently during training, and consequently, resulting in a large variation of performance when evaluating on different subsets of testing tasks (Dhillon et al., 2019, Figure 1).

One way to address such issue is to exploit the diversity of training tasks, so that the trained meta-learning models can generalise to a wider range of testing tasks. In fact, various studies in task relatedness or task similarity have shown that learning from certain tasks may facilitate the generalisation of meta-learning models (Thrun and O'Sullivan, 1996; Zamir et al., 2018; Achille et al., 2019; C. Nguyen et al., 2021). This suggests the design of a re-weighting mechanism to diversify the contribution of each training task when learning a meta-learning model of interest. Existing re-weighting methods mostly rely on either hand-crafted criteria to determine those weights (Collins et al., 2020), or additional validation tasks to learn the re-weighting factors of interest (Xu et al., 2021). Such ad-hoc development of re-weighting mechanisms motivates us to design a more principled approach to

re-weight tasks for meta-learning.

In this paper, we present a new principled and fully automated task-weighting algorithm, called **t**rajectory **o**ptimisation based task **w**eighting for meta-learning (TOW). We note that TOW is not a meta-learning method, but a task weighting framework that can be integrated into existing meta-learning algorithms to circumvent the problematic assumption about the even distribution of training tasks. Our contributions can be summarised as follows:

- We propose to cast the task-weighting problem in meta-learning to a finite-horizon discrete-time trajectory optimisation with state denoted by the meta-parameter and action by the re-weight factors of tasks, and solve such problem using the iterative linear quadratic regulator.

- We prove that under the conditions of boundedness and smoothness of the loss function used, TOW converges to a particular ϵ-stationary point.

- We demonstrate TOW's functionality by incorporating it into two common meta-learning algorithms, namely MAML (Finn et al., 2017) and Prototypical Networks (Snell et al., 2017), and showing that TOW enables meta-learning methods to converge with fewer training tasks and achieves higher prediction accuracy than some common task re-weighting mechanisms in the literature.

6.2 Background

6.2.1 Trajectory optimisation

Given continuous state $\mathbf{x} \in \mathbb{R}^D$ and action $\mathbf{u} \in \mathbb{R}^M$, the objective of a trajectory optimisation is to find an optimal sequence of actions $\{\mathbf{u}_t^*\}_{t=1}^T$ that minimises the total cost:

$$\min_{\{\mathbf{u}_t\}_{t=1}^T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \quad \text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{6.1}$$

where $c(\mathbf{x}_t, \mathbf{u}_t)$ and $f(\mathbf{x}_t, \mathbf{u}_t)$ are the cost function and the state-transition dynamics at time step t, respectively. These functions are stationary and assumed to be twice differentiable. In addition, the initial state $\mathbf{x}_1$ is given, and *trajectory optimisation* means finding the optimal sequence of actions $\{\mathbf{u}_t^*\}_{t=1}^T$ for a particular $\mathbf{x}_1$, not for all possible initial states.

In trajectory optimisation, the finite-horizon discrete-time problem shown in (6.1) can be solved approximately by iterative methods, such as differential dynamic programming (DDP) (Jacobson and Mayne, 1970) or iterative linear quadratic regulator (iLQR) (Todorov and W. Li, 2005; Tassa et al., 2012). These methods rely on a local approximation of the state-transition dynamics and cost function

using Taylor series about a nominal trajectory $\{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t\}_{t=1}^{T}$. In DDP, both the state-transition dynamics and cost function are approximated to the second order of their Taylor series, while in iLQR – a "simplified" version of DDP, the state-transition dynamics is approximated up to the first order. In a loose sense, DDP is analogy to the Newton's method, while iLQR is analogous to a quasi-Newton's method.

The main idea of iLQR is to cast a general non-linear trajectory optimisation problem shown in (6.1) to a linear quadratic problem (LQP) in which the state-transition dynamics is linear and the cost function is quadratic. The approximate LQP can then be solved exactly by the linear quadratic regulator (LQR) (Anderson and Moore, 2007). Subsequently, the newly obtained trajectory is used as the nominal trajectory for the next iteration. This process is repeated until the cost function is converged. The detailed derivation of iLQR can be found in Appendix D.1. Further details of iLQR can be referred to (Todorov and W. Li, 2005; Tassa et al., 2012). One of the major contributions of this paper is that we provide the proof of convergence for iLQR adopted from DDP (Sidney Yakowitz and Rutherford, 1984) in Appendix D.2, which to the best of our knowledge has not been done before.

6.2.2 Meta-learning

The setting of the meta-learning considered in this paper follows the *task environment* (Baxter, 2000), where tasks are i.i.d. sampled from an unknown distribution $p(\mathcal{T})$ over a family of tasks. Each task $\mathcal{T}_i$ is associated with two data subsets: training (or support) subset $\mathcal{S}_i^{(s)} = \{\mathbf{s}_{ij}^{(s)}, y_{ij}^{(s)}\}_{j=1}^{m_i^{(s)}}$, where $\mathbf{s}_{ij}^{(s)}$ denotes a training input and $y_{ij}^{(s)}$ denotes the corresponding training label, $i \in \{1, \ldots, M\}$, and validation (or query) subset $\mathcal{S}_i^{(q)}$ which is similarly defined. For training tasks $\{\mathcal{T}\}_{i=1}^{M}$, both data subsets have labels, while for testing tasks $\mathcal{T}_{M+1}$, only the data in $\mathcal{S}_{M+1}^{(s)}$ is labelled. The aim is to learn a meta-parameter $\mathbf{x} \in \mathbb{R}^D$ shared across all tasks, so that $\mathbf{x}$ can be efficiently fine-tuned on $\mathcal{S}_i^{(s)}$ to produce a task-specific model that can predict accurately the unlabelled data in $\mathcal{S}_i^{(q)}$. One of the simplest forms of meta-learning is analogous to an extension of hyper-parameter optimisation in single-task learning, where the shared meta-parameter $\mathbf{x}$ is learnt from many tasks. The objective of meta-learning can be expressed as:

$$\min_{\mathbf{x}} \frac{1}{M} \mathbf{1}_M^\top \boldsymbol{\ell}(\mathbf{x}), \tag{6.2}$$

where $\mathbf{1}_M$ is an M-dimensional vector with all elements equal to 1, and $\boldsymbol{\ell}(\mathbf{x}) \in \mathbb{R}^M$ is a vector containing M validation losses induced by evaluating the meta-parameter $\mathbf{x}$ on each data subset $\mathcal{S}_i^{(q)}$ of M training tasks. Each element of $\boldsymbol{\ell}(\mathbf{x})$ can be expressed

as:

$$\boldsymbol{\ell}_i(\mathbf{x}) = \frac{1}{m_i^{(q)}} \sum_{j=1}^{m_i^{(q)}} \ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right), \forall i \in \{1, \dots, M\}, \tag{6.3}$$

where $\ell(.)$ is the loss function that is non-negative and twice differentiable, $\phi(\mathbf{x})$ is the parameter fine-tuned on task $\mathcal{T}_i$:

$$\phi_i(\mathbf{x}) = \mathbf{x} - \frac{\gamma}{m_i^{(s)}} \sum_{k=1}^{m_i^{(s)}} \boldsymbol{\nabla}_{\mathbf{x}} \left[\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \mathbf{x}\right)\right], \tag{6.4}$$

and γ is the step size or learning rate for the task adaptation step (also known as inner-loop).

Note that the gradient-based task adaptation step in (6.4) is a special case of meta-learning in which $\mathbf{x}$ is considered as the initialisation of the neural network of interest (Finn et al., 2017). In metric-based meta-learning (Snell et al., 2017), the task adaptation step in (6.4) is slightly different where the class prototypes of training data are embedded into a latent space by the meta-model, and the validation loss is based on the distance between the class prototypes to each data-point in $\mathcal{S}_i^{(q)}$. There are also other extensions of (6.2) using probabilistic approaches (Yoon et al., 2018; Ravi and Beatson, 2019; C. Nguyen et al., 2020). Nevertheless, our approach proposed in Section 6.3 can be integrated into any of these meta-learning algorithms with a slight modification.

6.2.3 Task-weighting meta-learning

The minimisation of the average validation loss over M tasks in (6.2) implicitly implies that those tasks are evenly distributed. This assumption is, however, hardly true in practice, and consequently, makes the trained meta-model perform poorly for testing tasks that are rarely observed (similar to class imbalance problems in single-task learning). To address such issue, a task-weighting factor is introduced to diversify the contribution of each training task, allowing the trained meta-model to generalise better to unseen tasks even if those tasks are rare. The objective of such meta-learning problem can be written as:

$$\mathbf{x}^* = f(\mathbf{x}, \mathbf{u}) = \arg\min_{\mathbf{x}} \mathbf{u}^\top \boldsymbol{\ell}(\mathbf{x}) \quad \text{s.t. } \mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^M, \tag{6.5}$$

where $\mathbf{u}$ is an M-dimensional vector that re-weights the influence of M training tasks, and $\mathcal{U}$ is the set of feasible weights, i.e., as defined by some weighting criterion.

Note that the task-weighting problem in Eq. (6.5) is carried out at the meta level (or "outer-loop"). It is, therefore, different from some recent meta-learning methods (Khodak et al., 2019; Baik et al., 2020; Flennerhag et al., 2020) that

design different learning strategies for $\phi_i(\mathbf{x})$ at the "inner-loop" to estimate the meta-parameters with the same outer-loop objective shown in Eq. (6.2).

The objective in (6.5) is more flexible than (6.2), since it allows one to select different weighting criteria to train the meta-learning model of interest. The most widely-used weighting criterion is **uniform**: $\mathbf{u}_i = 1/M, \forall i \in \{1, \dots, M\}$, making the objective in (6.5) resemble the one in (6.2). Another popular criterion is to select **difficult** tasks – tasks that have the largest validation losses – for training to optimise the performance on the worst-case scenarios (Collins et al., 2020). However, such difficult tasks may not always be preferred when outliers and noise are present. That leads to another weighting approach which favours the **most familiar** data-points in single-task learning (Kumar et al., 2010; Bengio et al., 2009; Wang et al., 2017) – often referred as *curriculum learning*. The two latter task-weighting approaches can be considered as the "exploration" and "exploitation" strategies used in reinforcement learning (RL), respectively. Similar to the exploration and exploitation dilemma in RL, we hypothesise that the optimality for task weighting is formed by a balance between these two approaches. In the following section, we propose a principled approach to automate re-weighting tasks through an optimisation on a sequence of many mini-batches rather than relying on manually-designed criteria as the previous papers.

6.3 Methodology

6.3.1 Task-weighting as a trajectory optimisation

In practice, the optimisation in (6.5) is often carried out using a gradient-based optimiser where the next meta-parameter $\mathbf{x}^*$ is obtained from the current meta-parameter $\mathbf{x}$ and its corresponding $\mathbf{u}$ via the function f. Such update can be considered as a state-transition dynamics where the meta-parameter $\mathbf{x}$ is the state and the weighting vector $\mathbf{u}$ is the action. Given this observation, we explicitly replace the weighting criterion in (6.5) by a trajectory optimisation to formulate the task-weighting meta-learning problem as follows:

$$\begin{aligned} \mathbf{x}^*_{t+1} &= f(\mathbf{x}^*_t, \mathbf{u}^*_t)\ \forall t \in \{1, \dots, T\} \\ \text{s.t. } \{\mathbf{u}^*_t\}_{t=1}^T &= \arg\min_{\{\mathbf{u}\}_{t=1}^T} \sum_{t=1}^T c(\mathbf{x}_t, \mathbf{u}_t) \\ &\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{x}^*_1 &= \mathbf{x}_1, \end{aligned} \tag{6.6}$$

where $f(.,.)$ corresponds to the formulation of an optimiser such as stochastic gradient descent (SGD) (Robbins and Monro, 1951) or Adam (Kingma and Ba, 2014), $c(.,.)$ is a cost function representing the weighting criteria, $\mathbf{x}_1$ is the initialisation of the meta-learning parameter, and the subscript denotes the time step.

To solve for an optimal re-weighting vector $\mathbf{u}$ in the constraint of (6.6), the cost function needs to be defined. Since our interest is the convergence speed and the generalisation of the learnt meta-model, we define the cost function as an un-discounted sum of uniformly-weighted validation losses of tasks belonging to a sequence of T mini-batches plus a penalisation on the action $\mathbf{u}$. For simplicity, the penalty on the action $\mathbf{u}$ is assumed to follow a Gaussian prior with mean μ_u and precision β_u. In particular, the cost function can be expressed as:

$$c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{1}_M^\top \boldsymbol{\ell}(\mathbf{x}_t) + \frac{\beta_u}{2} \|\mathbf{u}_t - \mu_u \mathbf{1}_M\|^2, \tag{6.7}$$

where $\|.\|$ denotes the L2-norm.

Note that the action $\mathbf{u}_t$ is not necessarily normalised to 1. We argue that imposing such constraint might not work well in some cases, for example, a mini-batch containing all familiar tasks, and another one containing all unfamiliar tasks. Our hypothesis is to have small weights for familiar tasks in the former mini-batch, while setting large weights for unfamiliar tasks in the latter mini-batch to diversify the learning. Normalising $\mathbf{u}_t$ to 1 will, however, be undesirable since the contribution of the tasks in both mini-batches would be the same, making the meta-learning model even biased further toward the familiar tasks in the first mini-batch. Hence, we allow the weights to be determined automatically by the optimisation in (6.6) with a Gaussian prior.

In general, the constraint in (6.6) cannot be solved exactly, but approximately using iterative methods such as DDP or iLQR. Given the state-transition dynamics f follows the formulation of a first-order gradient-based optimiser (refer to Eq. (D.30) in Appendix D.3.1 and Eq. (D.37) in Appendix D.3.2 for the explicit form of f using SGD and Adam, respectively), f consists of the first derivatives of the weighted loss $\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)$ w.r.t. $\mathbf{x}$. Hence, applying DDP will result in an intractable solution since DDP requires the second derivatives of f, corresponding to the third derivatives of the weighted loss $\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)$. In contrast, iLQR needs only the first derivatives of f, which corresponds to the second derivatives of the weighted loss $\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)$. Although this means that iLQR no longer exhibits the quadratic convergence rate as DDP, in the context of meta-learning, the significant reduction in computation out-weights the speed of convergence for the task weighting vector $\mathbf{u}$. In this paper, we use iLQR to solve the constraint in (6.6). The locally-optimal actions obtained is then used to re-weight the tasks in each mini-batch to train the meta-learning model of interest.

The approximation using Taylor's series on the state-transition dynamics and cost function is shown in Appendices D.3 and D.4, respectively. This approximation leads to the calculation of two Hessian matrices: one for the sum of weighted loss, $\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)$, in the dynamics, denoted as $\mathbf{F}_{\mathbf{x}_t}$, and the other for the sum of non-weighted loss, $\mathbf{1}_M^\top \boldsymbol{\ell}(\mathbf{x}_t)$, in the cost function, denoted as $\mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t}$. In addition, while performing recursive backward iLQR, we need to calculate another temporary Hessian matrix of the *cost-to-go* in (D.1) (referred to Appendix D.1), denoted as $\mathbf{V}_t$. Naively calculating these Hessian matrices comes at the quadratic complexity $\mathcal{O}(D^2)$ in terms of running time and storage, resulting in an intractable solution for large-scaled models. To address such issue, the two Hessian matrices $\mathbf{F}_{\mathbf{x}_t}$ and $\mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t}$ may be approximated by their diagonals which can be efficiently computed using the Hutchinson's method (Bekas et al., 2007). However, as the size of the model increases, using a few samples from the uniform Rademacher distribution produces noisy estimations of the Hessian diagonals, resulting in a poor approximation (Yao et al., 2021). Instead of calculating the Hessian diagonals, we use the Gauss-Newton diagonals as replacements. As the Gauss-Newton matrix is known to be a good approximation of the Hessian matrix (Martens, 2010; Botev et al., 2017), this, therefore, results in a good approximation for the Hessian operator. In addition, Gauss-Newton diagonals can be efficiently calculated using a single backward pass (Dangel et al., 2020). For the matrix $\mathbf{V}_t$, we approximate it by its diagonal matrix. We also provide some results using full matrix $\mathbf{V}_t$ in Appendix D.7. Nevertheless, these approximation increases the tractability of our proposed method, allowing to implement the proposed method for very large models, such as deep neural networks.

The whole procedure for the proposed approach can be described as follows: first, a meta-parameter $\mathbf{x}_1$ is initialised as the initial state, and then, iLQR is employed to solve the constraint in (6.6) to determine a locally-optimal action $\{\mathbf{u}_t^*\}_{t=1}^T$ about an arbitrary-but-feasible trajectory $\{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t\}_{t=1}^T$ with $\hat{\mathbf{x}}_1 = \mathbf{x}_1$. The obtained weighting vectors $\{\mathbf{u}_t^*\}_{t=1}^T$ are then used to weight tasks in each mini-batch to train the meta-parameter $\mathbf{x}_{t+1}^*$ in (6.6). The newly calculated state at the end of the T time steps, $\mathbf{x}_{T+1}^*$, is then used as the initial state for the next iteration. This process is repeated until the weighted validation loss $\mathbf{u}^\top \boldsymbol{\ell}(\mathbf{x})$ converges to a local minima. The complete algorithm of the proposed task-weighting meta-learning approach is outlined in Algorithm 5.

To simplify the implementation and convergence analysis, we select the nominal actions that coincide with the uniform weighting, meaning that: $\hat{\mathbf{u}}_{ti} = {}^1\!/_M, \forall t \in \{1, \ldots, T\}, i \in \{1, \ldots, M\}$. In addition, we constrain that all elements of the weighting vector or action $\mathbf{u}$ are non-negative since each task would either contribute more or less or even not contribute to the learning for $\mathbf{x}$. This constraint is incorporated into the stopping condition for iLQR shown in step 18 of Algorithm 5. If there is at

Algorithm 5 Task-weighting for meta-learning

procedure TRAIN
  define total loss J in Eq. (D.13)
  define IBACKWARD( ) in Algorithm 7 (Appendix D.5)
  initialise $\mathbf{x}_1$
  while $\mathbf{x}$ is not converged **do**
    get T mini-batches, each consists of M tasks
    generate a random sequence of action $\{\hat{\mathbf{u}}_t\}_{t=1}^T$
    obtain the corresponding state $\{\hat{\mathbf{x}}_t\}_{t=1}^T$
    while iLQR cost is not converged **do**
      $\{\mathbf{K}_t, \mathbf{k}_t\}_{t=1}^T, \theta_1 \leftarrow \text{IBACKWARD}(\{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t\}_{t=1}^T)$
      $\varepsilon = 2$
      repeat ▷ Backtracking line search
        $\varepsilon \leftarrow \frac{1}{2}\varepsilon$
        for $t = 1 : T$ **do** ▷ Forward pass
          $\mathbf{u}_t = \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \varepsilon\mathbf{k}_t + \hat{\mathbf{u}}_t$
          $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$
        end for
      until $J(\mathbf{u}_{1:N}) - J(\hat{\mathbf{u}}_{1:N}) \leq \frac{1}{2}\varepsilon\theta_1$ and $\mathbf{u}_{ti} \geq 0$
      $\{\hat{\mathbf{x}}_t\}_{t=1}^T \leftarrow \{\mathbf{x}_t\}_{t=1}^T$ ▷ Update nominal state
      $\{\hat{\mathbf{u}}_t\}_{t=1}^T \leftarrow \{\mathbf{u}_t\}_{t=1}^T$
    end while
    $\mathbf{x}_1 \leftarrow \mathbf{x}_T$ ▷ Update meta-parameter
  end while
  return $\mathbf{x}_1$
end procedure

least one element $\mathbf{u}_{ti}, t \in \{1, \ldots, T\}, i \in \{1, \ldots, M\}$ being negative, the backtracking line search will iterate one more time with ε decaying toward 0, forcing $\mathbf{u}_t$ to stay close to the nominal $\hat{\mathbf{u}}_t$. Thus, in the worst-case, ε is reduced to 0, making $\mathbf{u}_t$ coincide with $\hat{\mathbf{u}}_t$, which is the uniform weighting.

The downside of TOW is the overhead due to the linearisation and quadraticisation for the state-transition dynamics and cost function, and the calculation to obtain the controller $\mathbf{K}_t$ and $\mathbf{k}_t$ shown in Algorithm 5. If $\mathcal{O}(T_0)$ is the time complexity to train a meta-learning method following a uniform weighting strategy, then the time complexity required by TOW will consist of the following:

- nominal trajectory: $\mathcal{O}(T_0)$
- linearisation and quadraticisation using Gauss-Newton matrices: $\mathcal{O}(n_{\text{iLQR}} m_0 \eta D)$
- iLQR backward: $\mathcal{O}(n_{\text{iLQR}} MD)$
- iLQR forward with back-tracking line search: $\mathcal{O}(n_{\text{iLQR}} n_{\text{ls}} T_0)$,

where n_{iLQR} is the number of iterations in iLQR, $m_0 = m_i^{(q)}, i \in \{1, \ldots, M\}$, is the total number of validation samples within a task, η is the number of arithmetic operations in the model of interest, and n_{ls} is the number of back-tracking line search.

Thus, the final complexity of TOW is: $\mathcal{O}((n_{\text{iLQR}} n_{\text{ls}} + 1)T_0 + n_{\text{iLQR}}(m_0\eta + M)D))$ comparing to $\mathcal{O}(T_0)$ in the conventional meta-learning.

6.3.2 Convergence analysis

This subsection proves that the training process for MAML using TOW to weight tasks converges to an ϵ-stationary point where ϵ is greater than some positive constant. In other words, we prove an inequality similar to the following:

$$\exists \epsilon > 0 : \|\boldsymbol{\nabla}_{\mathbf{x}} \mathbf{u}_{T_{\text{iter}}} \boldsymbol{\ell}\left(\mathbf{x}_{T_{\text{iter}}}\right)\| \leq \epsilon. \tag{6.8}$$

Before analysing the convergence of TOW, we state a lemma bounding the norm of the weighting vector (or action) $\mathbf{u}_t$ obtained from iLQR:

Lemma 6.1

If $\mathbf{u}_t$ is a stationary action of a nominal action $\hat{\mathbf{u}}_t$ obtained from iLQR, then:

$$\exists \delta > 0 : \|\mathbf{u}_t - \hat{\mathbf{u}}_t\| \leq \delta.$$

To analyse the convergence of a general non-convex function, one typically assumes the boundedness and Lipschitz continuity of the loss function and its first and second derivatives as shown in Assumptions 6.1, 6.2 and 6.3, respectively (Collins et al., 2020; Fallah et al., 2020).

Assumption 6.1

The loss function of interest ℓ mentioned in (6.3) is B-bounded and L-Lipschitz.

Assumption 6.2

The gradient $\boldsymbol{\nabla}_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x})$ is S-Lipschitz.

Assumption 6.3

The Hessian $\boldsymbol{\nabla}^2_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x})$ is ρ-Lipschitz.

These assumptions are used to bound the gradient of the "true" validation loss of task $\mathcal{T}_i$, which is defined as follows:

$$\bar{\boldsymbol{\ell}}_i(\mathbf{x}) = \mathbb{E}_{\mathcal{D}_i^{(q)}} \left[\ell \left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi(\mathbf{x}) \right) \right], \tag{6.9}$$

where $\mathbb{E}_{\mathcal{D}_i^{(q)}}$ indicates the expectation over all data pairs $\{(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)})\}_{j=1}^{+\infty}$ sampled from the true probability distribution $\mathcal{D}_i^{(q)}$.

Lemma 6.2

If the conditions in Assumptions 6.1, 6.2 and 6.3 hold, then $\bar{\boldsymbol{\ell}}_i(\mathbf{x})$ defined in Eq. (6.9) is $\widetilde{S}$-smooth, where: $\widetilde{S} = S(1+\gamma S)^2 + \gamma\rho L$.

In addition, we assume that the variance of the loss function ℓ evaluated on different data points is bounded.

Assumption 6.4

The variance of the gradient $\boldsymbol{\nabla}_{\mathbf{x}}\ell$ is σ^2-bounded.

Such assumption leads to the boundedness of the variance of the weighted validation loss as shown in Lemma 6.3.

Lemma 6.3

If Assumption 6.4 holds, then the variance of $\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{u}_t^\top\boldsymbol{\ell}(\mathbf{x}_t)$ is bounded by $\widetilde{\sigma}^2 = \sigma^2\left(\delta + M^{-0.5}\right)^2$.

Given the above assumptions and lemmas, the convergence of TOW can be shown in Theorem 6.4. Further details on the proof is referred to Appendix D.6.

Theorem 6.4

If Assumptions 6.1 - 6.4 hold, the learning rate $\alpha < {}^2/\widetilde{S}(\delta\sqrt{M}+1)$, and $\mathbf{z}$ is randomly sampled from $\{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}$ returned by Algorithm 5, then:

$$\mathbb{E}_{\mathbf{z}\sim\{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}}\left[\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\left\|\boldsymbol{\nabla}_{\mathbf{z}}\mathbf{u}_t^\top\bar{\boldsymbol{\ell}}_{1:M}(\mathbf{z})\right\|^2\right]\right] \leq \epsilon_0 + \frac{\kappa}{T_{\text{iter}}},$$

where:

$$\epsilon_0 = \frac{4\delta B\sqrt{M} + \alpha^2\widetilde{\sigma}^2\widetilde{S}\left(\delta\sqrt{M}+1\right)}{\alpha\left[2-\alpha\widetilde{S}\left(\delta\sqrt{M}+1\right)\right]} > 0 \tag{6.10}$$

$$\kappa = \frac{2\mathbf{u}_1^\top\bar{\boldsymbol{\ell}}_{1:M}(\mathbf{x}_1)}{\alpha\left[2-\alpha\widetilde{S}\left(\delta\sqrt{M}+1\right)\right]}, \tag{6.11}$$

with T_{iter} as the number of gradient-update for the meta-parameter (or the number of mini-batches of tasks used), and $\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}$ as the expectation taken over all data sampled from t mini-batches $\{\mathcal{D}_i^{(q)}\}_{i=1}^t$, each $\mathcal{D}_i^{(q)}$ has M tasks.

Theorem 6.4 shows that the expectation of squared gradient norm of the weighted validation loss is upper-bounded by a monotonically reducing function w.r.t. the number of iterations T_{iter}. This implies that Algorithm 5 converges in expectation to an ϵ_0-stationary point.

6.4 Related work

Our work directly relates to re-weighting tasks in meta-learning. One notable recent work is TR-MAML (Collins et al., 2020) which places higher weights on tasks with larger validation losses to optimise performance for worst-case scenarios. However, when the number of training tasks is very large, e.g. there will be $\binom{1000}{5} \approx 8.25 \times 10^{12}$ 5-way classification tasks formed from 1000 characters in Omniglot dataset (Lake et al., 2015), learning weight for each training task is intractable. TR-MAML circumvents such issue by clustering tasks into a small number of clusters based on some ad-hoc intuition and learn the weight for each cluster. This, however, reduces the practicability of TR-MAML. Another work, α-MAML (Cai et al., 2020), provides an upper-bound on the distance between the weighted risk evaluated on training tasks to the expected risk on testing tasks. The re-weight factors can then be obtained to minimise that upper-bound, reducing the variance between training and testing tasks. In reinforcement learning (RL), MWL-MAML (Xu et al., 2021) is recently proposed to employ meta-learning to learn the local optimal re-weight factor of each trajectory using a few gradient descent steps. The downside of MWL-MAML is the need of validation trajectories (or validation tasks in meta-learning) that are representative enough to learn those weights. Furthermore, TR-MAML, α-MAML and MWL-MAML rely on a single mini-batch of tasks to determine the weights without considering the effect of sequence of mini-batches when training a meta-model, potentially rendering sub-optimal solutions. In contrast, our proposed method does not need to cluster tasks nor require additional set of validation tasks. In addition, our proposed method automates the calculation of task-weighting through an optimisation over a sequence of mini-batches, allowing to obtain better local-optimal solutions outside of a single mini-batch of tasks.

Our work is also similar to task-weighting in multi-task learning (Zhao Chen et al., 2018; Sener and Koltun, 2018; M. Guo et al., 2018; L. Liu et al., 2021) where the goal is to obtain an optimal re-weighting vector $\mathbf{u}$ for all tasks. Such modelling can, therefore, work well with a small number of tasks, but potentially fall short when the number of tasks is very large, e.g. in the magnitude of 10^{12} 5-way classification training tasks in Omniglot, due to the poor scalability of the computational and storage complexities of that modelling. In comparison, our proposed approach does not explicitly learn the weighting vector for all training tasks, but determines the weighting vector for tasks in current and some following mini-batches via a trajectory optimisation technique. In a loose sense, the multi-task learning approaches can be considered as an analogy to a "batch" learning setting w.r.t. the weighting vector $\mathbf{u}$, while ours is analogous to an "online" learning setting which can scale well to the number of training tasks.

This paper is motivated from the observation of large variation in terms of prediction performance made by meta-learning algorithms on various testing tasks (Dhillon et al., 2019, Figure 1), implying that the trained meta-model may be biased toward certain training tasks. Such observation may be rooted in task relatedness or task similarity which is a growing research topic in the field of transfer learning. Existing works include task-clustering using k-nearest neighbours (Thrun and O'Sullivan, 1996) or using convex optimisation (Jacob et al., 2009), learning task relationship through task covariance matrices (Y. Zhang and Yeung, 2012), or theoretical guarantees to learn similarity between tasks (Shui et al., 2019). Recently, a large-scale empirical study, known as Taskonomy (Zamir et al., 2018), investigated the relationship between 26 computer vision tasks. Another promising direction to quantify task similarity is to employ task representation, notably Task2Vec (Achille et al., 2019), which is based on Fisher Information matrix to embed tasks into a latent space. One commonality among those studies is that learning from certain training tasks may be beneficial to generalise to unseen tasks. This suggests the design of a mechanism to re-weight the contribution of each training task to improve the performance of the meta-model of interest.

Furthermore, our work is related to finite-horizon discrete-time trajectory optimisation or open-loop optimal control which has been well studied in the field of control and robotics. The objective is to minimise a cost function that depends on the states and actions in many consecutive time steps given the state-transition dynamics. Exact solution can be obtained for the simplest problem where the cost is quadratic and the dynamics is linear using linear quadratic regulator (Anderson and Moore, 2007). For a general non-linear problem, approximate solutions can be found via iterative approaches, such as differential dynamic programming (Jacobson and Mayne, 1970; Murray and SJ Yakowitz, 1984; Sidney Yakowitz and Rutherford, 1984) and iterative LQR (iLQR) (Todorov and W. Li, 2005; Tassa et al., 2012).

6.5 Experiments

In this section, we empirically compare the performance of the proposed trajectory optimisation task weighting (TOW) approach with three baselines: one with uniform weighting, denoted as *uniform*, one with higher weights on difficult tasks (or tasks with higher losses), denoted as *exploration*, and the other one with higher weights on easier tasks (or tasks with lower losses), denoted as *exploitation*. The experiments are based on n-way k-shot classification setting used in few-shot learning with tasks formed from Omniglot (Lake et al., 2015) and mini-ImageNet (Vinyals et al., 2016) – the two most widely used datasets to evaluate the performance of meta-learning algorithms.

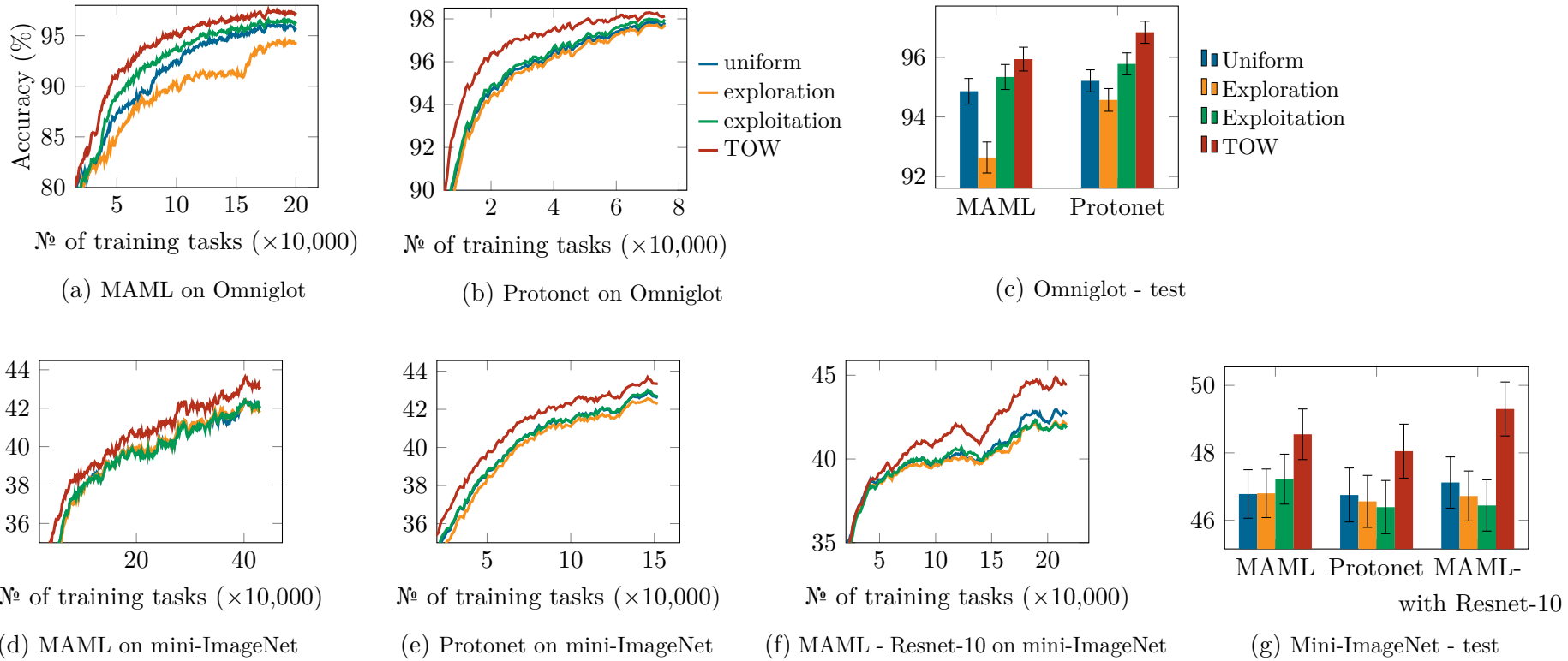
Naively implementing the two baselines, *exploration* and *exploitation*, will easily lead to trivial solutions where only the task with largest or smallest loss within a mini-batch is selected. Thus, only one task in each mini-batch is used for learning, and consequently, making the learning noisy and unstable. We, therefore, introduce a prior, denoted as $p(\mathbf{u})$, as a regularisation to prevent many tasks within the same mini-batch from being discarded. The objective to determine the weights for these two baselines can be written as follows:

$$\mathbf{u}^* = \begin{cases} \arg\min_{\mathbf{u}} -\mathbf{u}^\top \boldsymbol{\ell}(\mathbf{x}) - \ln p(\mathbf{u}) & \text{for } \textit{exploration} \\ \arg\min_{\mathbf{u}} \mathbf{u}^\top \boldsymbol{\ell}(\mathbf{x}) - \ln p(\mathbf{u}) & \text{for } \textit{exploitation}. \end{cases} \tag{6.12}$$

In general, the prior $p(\mathbf{u})$ can be any distribution that has support in $(0, +\infty)$ such as Beta, Gamma or Cauchy distribution. For simplicity, $p(\mathbf{u})$ is selected as a Dirichlet distribution with a concentration $\kappa > 1$ to constrain the weight vector within a probability simplex. One can then use a non-linear optimisation solver to solve (6.12) to obtain an optimal $\mathbf{u}^*$ for one of the two baselines. In the implementation, we use Sequential Least SQuares Programming (SLSQP) to obtain $\mathbf{u}^*$. Note that the definition of the *exploration* baseline above resembles TR-MAML (Collins et al., 2020), but is applicable for common few-shot learning benchmarks where the number of tasks is large. Similarly, the *exploitation* is an analogy to robust Bayesian data re-weighting (Wang et al., 2017) or *curriculum learning* in single-task learning.

For Omniglot dataset, we follow the original train-test split (Lake et al., 2015) with 30 alphabets used for training and 20 alphabets used for testing. For mini-ImageNet, we evaluate on the standard train-test split with 64 classes for training, 16 classes for validation and 20 for testing (Ravi and Larochelle, 2018). The base model used is the 4 CNN module network that is widely used in few-shot image classification (Vinyals et al., 2016; Finn et al., 2017). Two common meta-learning algorithms considered in this section include MAML (Finn et al., 2017) and Prototypical Networks (Snell et al., 2017) with Euclidean distances.

For all experiments, the learning rate γ of task adaptation (also known as inner-loop) shown in Eq. (6.4) is 0.1 for Omniglot and 0.01 for mini-ImageNet with 5 gradient updates. The learning rate for the meta-parameters, α, is set at 10^{-4} for all the setting. The mini-batch size is $M = 5$ and 10 tasks. For the Dirichlet concentration of the prior in the *exploration* and *exploitation* baselines, we try three values of $\kappa \in \{0.2, 1.2, 5\}$, and found that a too small value of κ leads to noisy learning since only the easiest or hardest task is selected, while too large value of κ makes both the baselines identical to uniform weighting. Hence, we select $\kappa = 1.2$ that balances between these two strategies. Note that $\kappa = 1$ results in a random prior, leading to a trivial solution. For the trajectory optimiser iLQR, the state-transition



(a) MAML on Omniglot (b) Protonet on Omniglot (c) Omniglot - test

(d) MAML on mini-ImageNet (e) Protonet on mini-ImageNet (f) MAML - Resnet-10 on mini-ImageNet (g) Mini-ImageNet - test

Figure 6.1: Validation accuracy exponential moving average (with smoothing factor 0.1) of different task-weighting strategies evaluated on: (a) and (b) Omniglot, and (d), (e) and (f) mini-ImageNet. The column plots show testing accuracy on: (c) Omniglot and (g) mini-ImageNet.

Table 6.1: Running time of different task-weighting methods based on MAML (unit in GPU-hour evaluated on an NVIDIA A6000).

	Omniglot	**mini-ImageNet**	
		CNN	Resnet-10
Exploration	1.55	5.24	7.18
Exploitation	1.55	5.24	7.18
Uniform	1.35	5.03	7.16
TOW	7.50	38.12	67.78

dynamics f follows the formula of Adam optimiser since Adam provides a less noisy training as in SGD. The nominal trajectory is, as mentioned in Subsection 6.3.1, selected with uniform actions: $\hat{\mathbf{u}}_{tj} = 1/M, \forall j \in \{1, \dots, M\}, t \in \{1, \dots, T\}$. The number of iteration used is iLQR is 2, and the number of time steps (or number of mini-batches) is $T = 5$ and 10. The parameters of the prior on the action $\mathbf{u}_t$ are $\mu_u = 1/M$ and $\beta_u = 10$. As we do not observe any major difference between different configuration of M and T used in this experiment, we report the result for the case $M = 10$ and $T = 5$.

Figures 6.1a, 6.1b, 6.1d and 6.1e plot the testing accuracy evaluated on 100 validation tasks drawn from Omniglot and mini-ImageNet following the 5-way 1-shot setting. The results show that TOW can achieve higher performance comparing to the three baselines given different datasets and meta-learning methods. Furthermore, we carry out an experiment with Resnet-10 (He et al., 2016) on mini-ImageNet to demonstrate the scalability of TOW and have a similar observation in Figure 6.1f. We note that the validation accuracy curves of Resnet-10 fluctuates due to our injected dropout to regularise the network from overfitting. For the evaluation, we follow the standard setting in few-shot learning by measuring the prediction accuracy on 1,000 and 600 testing tasks formed from Omniglot and mini-ImageNet, respectively (Vinyals et al., 2016; Finn et al., 2017). The results in Figures 6.1c and 6.1g show that TOW can be at least 2% more accurate than the best baseline among Uniform, Exploration, and Exploitation. Despite the promising results, the downside of TOW is the overhead caused by approximating the cost and state-transition dynamics over T mini-batches of tasks to determine the locally-optimal $\{\mathbf{u}_t^*\}_{t=1}^T$. As shown in Table 6.1, TOW is about 7 to 9 times slower than the three baselines. We also provide a visualisation of the weights $\mathbf{u}_t$ in Appendix D.8.

6.6 Discussion

We propose a principled approach based on trajectory optimisation to mitigate the issue of non-uniform distribution of training tasks in meta-learning. The idea is to

model the training process in meta-learning by trajectory optimisation with state as meta-parameter and action as the weights of training tasks. The local optimal weights obtained from iLQR – a trajectory optimiser are then used to re-weight tasks to train the meta-parameter of interest. We demonstrate that the proposed approach converges with less number of training tasks and has a final prediction accuracy that out-performs some common hand-crafted task-weighting baselines.

Our proposed method also has some limitations that could be addressed in future work. TOW relies on iLQR which is not ideal for large-scale systems with high dimensional state space such as deep neural networks. Despite the approximation of Hessian matrices to use only diagonals as mentioned in Subsection 6.3.1, the linearisation of the state-transition dynamics and quadraticisation of the cost function are still time-consuming, and consequently, reduce TOW's efficiency. Future work might find a faster approximation to optimise the running time for TOW.

Furthermore, our method is local in nature due to the Taylor's series approximation about a nominal trajectory used in iLQR. One way to improve further is to define a "global" or "stationary" policy $\pi_\theta(\mathbf{x}_t, \mathbf{u}_t)$, which is similar to Guided Policy Search (Levine and Koltun, 2013; Levine and Koltun, 2014). This policy can then be trained on multiple local optimal trajectories obtained from iLQR. While this approach may offer a superior generalisation for the policy, scalability is an issue since the policy needs to process the high-dimensional state $\mathbf{x}_t$. As a result, a very large model may be required to implement such policy.

Chapter 7

Conclusion

In this thesis, we have explicitly formulated the meta-learning problem as an extension of hyper-parameter optimisation, where the hyper-parameter of interest is shared across several tasks. Such formulation allows us to analyse some major issues in meta-learning research, including the potential overfitting due to the small number of training data within each task and the ability to generalise to unseen tasks. In addition, we have pointed out that the structure of task distribution and how to distribute the influence of each task in the learning of the meta-parameter of interest could benefit meta-learning, but have not been thoroughly explored yet.

We have, therefore, carried several studies to investigate and analyse such issues. Firstly, we proposed to integrate variational inference that takes epistemic uncertainty – the uncertainty due to modelling – into account to introduce a new probabilistic meta-learning algorithm to address the overfitting issue. We showed that the proposed algorithm can achieve the smallest calibration errors compared to some common meta-learning methods, meaning that the meta-model trained with our proposed algorithm is less fragile. Secondly, we employed PAC-Bayes framework to derive an upper-bound of the generalisation errors evaluated on unseen tasks drawn from the same task distribution. Such upper-bound can then be used as a new loss function to train different meta-learning methods. We demonstrated that the meta-learning models trained with the new proposed loss function can achieve state-of-the-art results in both regression and classification. Next, we explored meta-learning by modelling tasks via a graphical model based on a mixture of Gaussian distribution. Such modelling allows us to quantify task similarity or relatedness, which can be used in active task selection to optimise the training of meta-learning models. Lastly, we adopt trajectory optimisation in optimal control to propose a principled way to re-weight the contribution of each task to the learning of a meta-learning model. We show that such automatic re-weighting mechanism can converge faster in terms of number of training tasks and achieve better testing accuracy. In general, all the studies in this thesis contribute to the reliability improvement for meta-learning and

the optimisation for the prediction performance made by meta-learning models by exploiting the insightful intuition extracted from the training task distribution.

Limitations and future work

Despite the promising results presented, the studies carried out in this thesis have some limitations which need to be addressed in future work.

The first limitation is related to the use of a simplified variational distribution in Chapter 3, and in particular, multivariate normal distributions with diagonal covariance matrices, potentially hampering the modelling expressiveness. Although it has been partly addressed by implicit distributions – a "hyper" neural network outputs the "base" neural network that performs regression or classification – in Chapter 4, such implicit modelling suffers from the curse of dimensionality as seen in GAN models, making the proposed methods less applicable for large models. Thus, future works should focus on solving the scalability issue when approximating the posterior of the model of interest.

The second limitation is also a scalability issue but related to the task re-weighting method – TOW – presented in Chapter 6. At the present, TOW relies on iLQR – a quasi-Newton like method which has a quadratic time and storage complexity in terms of the dimension of the model used. Although the Hessian matrices in iLQR have been approximated by the diagonals of their corresponding Gauss-Newton matrices, such approximation is still rough, potentially resulting in a sub-optimal solution. In addition, calculating the approximated diagonal matrices is still time consuming, making TOW lag behind some common task re-weighting baselines in terms of running time and consequently, reduce its applicability, especially for large-scale problems. Thus, future work needs to investigate the bottle-neck computation of TOW and explore further the approximation to optimise TOW.

The last but not least limitation is the assumption of mono-modality of the task environment used in meta-learning. Conventionally, the task environment is often assumed to consist of only one family of tasks, e.g. either regression or classification. In other words, the label space of tasks sampled from such environment is the same from task-to-task, e.g. $\mathcal{Y} \subseteq \mathbb{R}$ in regression and $\mathcal{Y} \subseteq \mathbb{N}$ in classification. This, however, quite limited since it does not utilise the knowledge extracted from a variety of available task families, such as a mixture of image classification and segmentation task families. To the best of our knowledge, no research has been carried out in the direction of multiple task family setting, but only some empirical studies that show the knowledge transferability from one task family to another (Zamir et al., 2018). Thus, future works would be to investigate the feasibility of multi-modality of task environment and develop novel meta-learning algorithms that solves different tasks sampled from such complex setting.

Bibliography

Achille, Alessandro, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto and Pietro Perona (2019). 'TASK2VEC: Task embedding for meta-learning'. In: *International Conference on Computer Vision*, pp. 6430–6439.

Allen, Kelsey, Evan Shelhamer, Hanul Shin and Joshua Tenenbaum (2019). 'Infinite mixture prototypes for few-shot learning'. In: *International Conference on Machine Learning*, pp. 232–241.

Alquier, Pierre and Benjamin Guedj (2018). 'Simpler PAC-Bayesian bounds for hostile data'. In: *Machine Learning* 107.5, pp. 887–902.

Alquier, Pierre, Massimiliano Pontil et al. (2017). 'Regret bounds for lifelong learning'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 261–269.

Alquier, Pierre, James Ridgway and Nicolas Chopin (2016). 'On the properties of variational approximations of Gibbs posteriors'. In: *Journal of Machine Learning Research* 17.1, pp. 8374–8414.

Amit, Ron and Ron Meir (2018). 'Meta-learning by adjusting priors based on extended PAC-Bayes theory'. In: *International Conference on Machine Learning*, pp. 205–214.

Anderson, Brian DO and John B Moore (2007). *Optimal control: linear quadratic methods*. Courier Corporation.

Andrychowicz, Marcin, Misha Denil, Sergio Gómez Colmenarejo, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford and Nando de Freitas (2016). 'Learning to learn by gradient descent by gradient descent'. In: *Advances in Neural Information Processing Systems*, pp. 3988–3996.

Bai, Yu, Minshuo Chen, Pan Zhou, Tuo Zhao, Jason Lee, Sham Kakade, Huan Wang and Caiming Xiong (2021). 'How important is the train-validation split in meta-learning?' In: *International Conference on Machine Learning*, pp. 543–553.

Baik, Sungyong, Myungsub Choi, Janghoon Choi, Heewon Kim and Kyoung Mu Lee (2020). 'Meta-learning with adaptive hyperparameters'. In: *Advances in Neural Information Processing Systems*.

Bakker, Bart and Tom Heskes (2003). 'Task clustering and gating for Bayesian multitask learning'. In: *Journal of Machine Learning Research* 4.May, pp. 83–99.

Banerjee, Arindam (2006). 'On Bayesian bounds'. In: *International Conference on Machine learning*, pp. 81–88.

Baxter, Jonathan (2000). 'A model of inductive bias learning'. In: *Journal of Artificial Intelligence Research* 12, pp. 149–198.

Bekas, Costas, Effrosyni Kokiopoulou and Yousef Saad (2007). 'An estimator for the diagonal of a matrix'. In: *Applied Numerical Mathematics* 57.11-12, pp. 1214–1229.

Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira and Jennifer Wortman Vaughan (2010). 'A theory of learning from different domains'. In: *Machine Learning* 79.1-2, pp. 151–175.

Ben-David, Shai, John Blitzer, Koby Crammer, Fernando Pereira et al. (2007). 'Analysis of representations for domain adaptation'. In: *Advances in Neural Information Processing Systems* 19, p. 137.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert and Jason Weston (2009). 'Curriculum learning'. In: *International Conference on Machine Learning*, pp. 41–48.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer.

Blei, David M, Andrew Y Ng and Michael I Jordan (2003). 'Latent Dirichlet allocation'. In: *Journal of Machine Learning Research* 3.Jan, pp. 993–1022.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu and Daan Wierstra (2015). 'Weight uncertainty in neural networks'. In: *International Conference on Machine Learning*, pp. 1613–1622.

Botev, Aleksandar, Hippolyt Ritter and David Barber (2017). 'Practical Gauss-Newton optimisation for deep learning'. In: *International Conference on Machine Learning*, pp. 557–565.

Bridle, John S and Stephen J Cox (1991). 'Recnorm: Simultaneous normalisation and classification applied to speech recognition'. In: *Advances in Neural Information Processing Systems*, pp. 234–240.

Bubeck, Sébastien et al. (2015). 'Convex Optimization: Algorithms and Complexity'. In: *Foundations and Trends® in Machine Learning* 8.3-4, pp. 231–357.

Cai, Diana, Rishit Sheth, Lester Mackey and Nicolo Fusi (2020). 'Weighted meta-learning'. In: *ICML Workshop on Automated Machine Learning*.

Canini, Kevin, Lei Shi and Thomas Griffiths (2009). 'Online inference of topics with latent Dirichlet allocation'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 65–72.

Caruana, Rich (1997). 'Multitask learning'. In: *Machine Learning* 28.1, pp. 41–75.

Catoni, Olivier (2004). *Statistical learning theory and stochastic optimization: Ecole d'Eté de Probabilités de Saint-Flour XXXI-2001*. Springer.

Chen, Wei-Yu, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang and Jia-Bin Huang (2019). 'A closer look at few-shot classification'. In: *International Conference on Learning Representations*.

Chen, Zhao, Vijay Badrinarayanan, Chen-Yu Lee and Andrew Rabinovich (2018). 'Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks'. In: *International Conference on Machine Learning*, pp. 794–803.

Chen, Zhiyuan and Bing Liu (2018). 'Lifelong machine learning'. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12.3, pp. 1–207.

Collins, Liam, Aryan Mokhtari and Sanjay Shakkottai (2020). 'Task-robust model-agnostic meta-learning'. In: *Advances in Neural Information Processing Systems*.

Dangel, Felix, Frederik Kunstner and Philipp Hennig (2020). 'BackPACK: Packing more into Backprop'. In: *International Conference on Learning Representations*.

Das, Rajarshi, Manzil Zaheer and Chris Dyer (2015). 'Gaussian LDA for topic models with word embeddings'. In: *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 795–804.

Daume III, Hal and Daniel Marcu (2006). 'Domain adaptation for statistical classifiers'. In: *Journal of Artificial Intelligence Research* 26, pp. 101–126.

Denevi, Giulia, Carlo Ciliberto, Riccardo Grazzi and Massimiliano Pontil (2019a). 'Learning-to-learn stochastic gradient descent with biased regularization'. In: *International Conference on Machine Learning*, pp. 1566–1575.

Denevi, Giulia, Carlo Ciliberto, Dimitris Stamos and Massimiliano Pontil (2018). 'Learning to learn around a common mean'. In: *Advances in Neural Information Processing Systems*. Vol. 31.

Denevi, Giulia, Massimiliano Pontil and Carlo Ciliberto (2020). 'The advantage of conditional meta-learning for biased regularization and fine tuning'. In: *Advances in Neural Information Processing Systems* 33.

Denevi, Giulia, Dimitris Stamos, Carlo Ciliberto and Massimiliano Pontil (2019b). 'Online-within-online meta-learning'. In: *Advances in Neural Information Processing Systems*. Vol. 32, pp. 1–11.

Dhillon, Guneet S, Pratik Chaudhari, Avinash Ravichandran and Stefano Soatto (2019). 'A baseline for few-shot image classification'. In: *International Conference on Learning Representations*.

Diggle, Peter J and Richard J Gratton (1984). 'Monte Carlo methods of inference for implicit statistical models'. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 46.2, pp. 193–212.

Ding, Nan, Xi Chen, Tomer Levinboim, Sebastian Goodman and Radu Soricut (2021). 'Bridging the Gap Between Practice and PAC-Bayes Theory in Few-Shot Meta-Learning'. In: *Advances in Neural Information Processing Systems*.

Domke, Justin (2012). 'Generic methods for optimization-based modeling'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 318–326.

Edwards, Harrison and Amos Storkey (2017). 'Towards a neural statistician'. In: *International Conference on Learning Representations*.

Fallah, Alireza, Aryan Mokhtari and Asuman Ozdaglar (2020). 'On the convergence theory of gradient-based model-agnostic meta-learning algorithms'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 1082–1092.

Finn, Chelsea, Pieter Abbeel and Sergey Levine (2017). 'Model-agnostic meta-Learning for fast adaptation of deep networks'. In: *International Conference on Machine Learning*, pp. 1126–1135.

Finn, Chelsea, Kelvin Xu and Sergey Levine (2018). 'Probabilistic model-agnostic meta-learning'. In: *Advances in Neural Information Processing Systems*, pp. 9537–9548.

Flennerhag, Sebastian, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin and Raia Hadsell (2020). 'Meta-learning with warped gradient descent'. In: *International Conference on Learning Representations*.

Foulds, James, Levi Boyles, Christopher DuBois, Padhraic Smyth and Max Welling (2013). 'Stochastic collapsed variational Bayesian inference for latent Dirichlet allocation'. In: *International Conference on Knowledge Discovery and Data Mining*, pp. 446–454.

French, Robert M (1999). 'Catastrophic forgetting in connectionist networks'. In: *Trends in Cognitive Sciences* 3.4, pp. 128–135.

Garnelo, Marta, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami and Yee Whye Teh (2018). 'Neural processes'. In: *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Germain, Pascal, Francis Bach, Alexandre Lacoste and Simon Lacoste-Julien (2016). 'PAC-Bayesian theory meets Bayesian inference'. In: *Advances in Neural Information Processing Systems*, pp. 1884–1892.

Glorot, Xavier and Yoshua Bengio (2010). 'Understanding the difficulty of training deep feedforward neural networks'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 249–256.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014). 'Generative adversarial nets'. In: *Advances in Neural Information Processing Systems*.

Gordon, Jonathan, John Bronskill, Matthias Bauer, Sebastian Nowozin and Richard Turner (2019). 'Meta-learning probabilistic inference for prediction'. In: *International Conference on Learning Representations*.

Grant, Erin, Chelsea Finn, Sergey Levine, Trevor Darrell and Thomas Griffiths (2018). 'Recasting gradient-based meta-learning as hierarchical Bayes'. In: *International Conference on Learning Representations*.

Graves, Alex, Abdel-rahman Mohamed and Geoffrey Hinton (2013). 'Speech recognition with deep recurrent neural networks'. In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 6645–6649.

Griffiths, Thomas L and Mark Steyvers (2004). 'Finding scientific topics'. In: *Proceedings of the National Academy of Sciences* 101.suppl 1, pp. 5228–5235.

Guo, Chuan, Geoff Pleiss, Yu Sun and Kilian Q. Weinberger (2017). 'On calibration of modern neural networks'. In: *International Conference on Machine Learning*.

Guo, Michelle, Albert Haque, De-An Huang, Serena Yeung and Li Fei-Fei (Sept. 2018). 'Dynamic Task Prioritization for Multitask Learning'. In: *European Conference on Computer Vision*.

Ha, David, Andrew Dai and Quoc V Le (2017). 'Hypernetworks'. In: *International Conference on Learning Representations*.

Havaei, Mohammad, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin and Hugo Larochelle (2017). 'Brain tumor segmentation with deep neural networks'. In: *Medical Image Analysis* 35, pp. 18–31.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren and Jian Sun (2016). 'Deep residual learning for image recognition'. In: *Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Heckman, James J (1979). 'Sample selection bias as a specification error'. In: *Econometrica: Journal of The Econometric Society*, pp. 153–161.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed and Alexander Lerchner (2016). 'Beta-VAE: Learning basic visual concepts with a constrained variational framework'. In: *International Conference on Learning Representations*.

Hinton, Geoffrey et al. (2012). 'Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups'. In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Hoffman, Matthew, Francis R Bach and David M Blei (2010). 'Online learning for latent Dirichlet allocation'. In: *Advances in Neural Information Processing Systems*, pp. 856–864.

Hospedales, Timothy M, Antreas Antoniou, Paul Micaelli and Amos J Storkey (2021). 'Meta-learning in neural networks: A survey'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ioffe, Sergey and Christian Szegedy (2015). 'Batch normalization: accelerating deep network training by reducing internal covariate shift'. In: *International Conference on Machine Learning*, pp. 448–456.

Jacob, Laurent, Jean-philippe Vert and Francis R Bach (2009). 'Clustered multi-task learning: A convex formulation'. In: *Advances in Neural Information Processing Systems*, pp. 745–752.

Jacobson, David H and David Q. Mayne (1970). *Differential dynamic programming*. Elsevier.

Japkowicz, Nathalie and Shaju Stephen (2002). 'The class imbalance problem: A systematic study'. In: *Intelligent Data Analysis* 6.5, pp. 429–449.

Johnson, Rie and Tong Zhang (2013). 'Accelerating stochastic gradient descent using predictive variance reduction'. In: *Advances in Neural Information Processing Systems* 26, pp. 315–323.

Kaiser, Łukasz, Ofir Nachum, Aurko Roy and Samy Bengio (2017). 'Learning to remember rare events'. In: *International Conference on Learning Representations*.

Khodak, Mikhail, Maria-Florina F Balcan and Ameet S Talwalkar (2019). 'Adaptive gradient-based meta-learning methods'. In: *Advances in Neural Information Processing Systems* 32, pp. 5917–5928.

Kingma, Diederik P and Jimmy Ba (2014). 'Adam: A method for stochastic optimization'. In: *International Conference on Learning Representations*.

— (2015). 'Adam: A method for stochastic optimization'. In: *International Conference on Learning Representations*.

Kingma, Diederik P and Max Welling (2014a). 'Auto-encoding variational Bayes'. In: *International Conference on Learning Representations*.

— (2014b). 'Auto-encoding variational Bayes'. In: *International Conference on Learning Representations*.

Koch, Gregory, Richard Zemel and Ruslan Salakhutdinov (2015). 'Siamese neural networks for one-shot image recognition'. In: *ICML Deep Learning Workshop*. Vol. 2.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E Hinton (2012). 'Imagenet classification with deep convolutional neural networks'. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Kumar, M Pawan, Benjamin Packer and Daphne Koller (2010). 'Self-Paced learning for latent variable models.' In: *Advances in Neural Information Processing Systems*. Vol. 1, p. 2.

Lake, Brenden M, Ruslan Salakhutdinov and Joshua B Tenenbaum (2015). 'Human-level concept learning through probabilistic program induction'. In: *Science* 350.6266.

Lee, Kwonjoon, Subhransu Maji, Avinash Ravichandran and Stefano Soatto (2019). 'Meta-learning with differentiable convex optimization'. In: *Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665.

Levine, Sergey and Vladlen Koltun (2013). 'Guided policy search'. In: *International Conference on Machine Learning*, pp. 1–9.

— (2014). 'Learning complex neural network policies with trajectory optimization'. In: *International Conference on Machine Learning*, pp. 829–837.

Li, Da, Yongxin Yang, Yi-Zhe Song and Timothy M Hospedales (2018). 'Learning to generalize: Meta-learning for domain generalization'. In: *AAAI Conference on Artificial Intelligence*.

Li, Fei-Fei, Rob Fergus and Pietro Perona (2006). 'One-shot learning of object categories'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4, pp. 594–611.

Li, Fei-Fei and Pietro Perona (2005). 'A Bayesian hierarchical model for learning natural scene categories'. In: *Conference on Computer Vision and Pattern Recognition*. Vol. 2, pp. 524–531.

Li, Huaiyu, Weiming Dong, Xing Mei, Chongyang Ma, Feiyue Huang and Bao-Gang Hu (2019). 'LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning'. In: *International Conference on Machine Learning*, pp. 3825–3834.

Li, Ke and Jitendra Malik (2017). 'Learning to optimize'. In: *International Conference on Learning Representations*.

Li, Yiying, Yongxin Yang, Wei Zhou and Timothy Hospedales (2019). 'Feature-critic networks for heterogeneous domain generalization'. In: *International Conference on Machine Learning*, pp. 3915–3924.

Li, Zhenguo, Fengwei Zhou, Fei Chen and Hang Li (2017). 'Meta-sgd: Learning to learn quickly for few-shot learning'. In: *arXiv preprint arXiv:1707.09835*.

Liu, Liyang, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao and Wayne Zhang (2021). 'Towards Impartial Multi-task Learning'. In: *International Conference on Learning Representations*.

Liu, Yanbin, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang and Yi Yang (2018). 'Transductive propagation network for few-shot learning'. In: *International Conference on Learning Representations*.

Loaiza-Ganem, Gabriel and John P Cunningham (2019). 'The continuous Bernoulli: fixing a pervasive error in variational autoencoders'. In: *Advances in Neural Information Processing Systems*.

Lorraine, Jonathan, Paul Vicol and David Duvenaud (2020). 'Optimizing millions of hyperparameters by implicit differentiation'. In: *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552.

MacKay, David JC (1992). 'The evidence framework applied to classification networks'. In: *Neural computation* 4.5, pp. 720–736.

Martens, James (2010). 'Deep learning via hessian-free optimization.' In: *International Conference on Machine Learning*. Vol. 27, pp. 735–742.

Maurer, Andreas and Tommi Jaakkola (2005). 'Algorithmic stability and meta-learning.' In: *Journal of Machine Learning Research* 6.6.

Maurer, Andreas, Massimiliano Pontil and Bernardino Romera-Paredes (2016). 'The benefit of multitask representation learning'. In: *Journal of Machine Learning Research* 17.81, pp. 1–32.

McAllester, David A (1999). 'PAC-Bayesian model averaging'. In: *Conference on Computational Learning Theory*. Vol. 99, pp. 164–170.

Minka, Thomas (2000). *Estimating a Dirichlet distribution*.

Mishra, Nikhil, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel (2018). 'A simple neural attentive meta-learner'. In: *International Conference on Learning Representations*.

Munkhdalai, Tsendsuren and Hong Yu (2017). 'Meta networks'. In: *International Conference on Machine Learning*.

Munkhdalai, Tsendsuren, Xingdi Yuan, Soroush Mehri and Adam Trischler (2018). 'Rapid adaptation with conditionally shifted neurons'. In: *International Conference on Machine Learning*, pp. 3661–3670.

Murray, DM and SJ Yakowitz (1984). 'Differential dynamic programming and Newton's method for discrete optimal control problems'. In: *Journal of Optimization Theory and Applications* 43.3, pp. 395–414.

Nagabandi, Anusha, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine and Chelsea Finn (2019). 'Learning to adapt in dynamic, real-world environments through meta-reinforcement learning'. In: *International Conference on Learning Representations*.

Naik, Devang K and RJ Mammone (1992). 'Meta-neural networks that learn by learning'. In: *International Joint Conference on Neural Networks*. Vol. 1. IEEE, pp. 437–442.

Nguyen, Cuong, Thanh-Toan Do and Gustavo Carneiro (2020). 'Uncertainty in model-agnostic meta-learning using variational inference'. In: *Winter Conference on Applications of Computer Vision*, pp. 3090–3100.

— (2021). 'Probabilistic task modelling for meta-learning'. In: *Conference on Uncertainty in Artificial Intelligence*.

— (2022). 'PAC-Bayes meta-learning with implicit task-specific posteriors'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. DOI: 10.1109/TPAMI.2022.3147798.

Nguyen, Cuong V, Tal Hassner, Cedric Archambeau and Matthias Seeger (2020). 'LEEP: A new measure to evaluate transferability of learned representations'. In: *International Conference on Machine Learning*.

Nichol, Alex, Joshua Achiam and John Schulman (2018). 'On first-order meta-learning algorithms'. In: *arXiv preprint arXiv:1803.02999*.

Oreshkin, Boris N, Alexandre Lacoste and Pau Rodriguez (2018). 'TADAM: Task dependent adaptive metric for improved few-shot learning'. In: *Advances in Neural Information Processing Systems*, pp. 719–729.

Parisi, German I, Ronald Kemker, Jose L Part, Christopher Kanan and Stefan Wermter (2019). 'Continual lifelong learning with neural networks: A review'. In: *Neural Networks* 113, pp. 54–71.

Pearlmutter, Barak A. (1994). 'Fast exact multiplication by the Hessian'. In: *Neural Computation* 6, pp. 147–160.

Pentina, Anastasia and Christoph Lampert (2014). 'A PAC-Bayesian bound for lifelong learning'. In: *International Conference on Machine Learning*, pp. 991–999.

Polak, Elijah (1971). *Computational methods in optimization: a unified approach*. Vol. 77. Academic press.

Pratt, Lorien Y, Jack Mostow, Candace A Kamm and Ace A Kamm (1991). 'Direct transfer of learned information among neural networks.' In: *AAAI Conference on Artificial Intelligence*. Vol. 91, pp. 584–589.

Pritchard, Jonathan K, Matthew Stephens and Peter Donnelly (2000). 'Inference of population structure using multilocus genotype data'. In: *Genetics* 155.2, pp. 945–959.

Rajeswaran, Aravind, Chelsea Finn, Sham Kakade and Sergey Levine (2019). 'Meta-learning with implicit gradients'. In: *Advances in Neural Information Processing Systems*.

Ravi, Sachin and Alex Beatson (2019). 'Amortized Bayesian meta-Learning'. In: *International Conference on Learning Representations*.

Ravi, Sachin and Hugo Larochelle (2018). 'Optimization as a model for few-shot learning'. In: *International Conference on Learning Representations*.

Ren, Mengye, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle and Richard S Zemel (2018). 'Meta-learning for semi-supervised few-shot classification'. In: *International Conference on Learning Representations*.

Rendell, Larry A, Raj Sheshu and David K Tcheng (1987). 'Layered concept-learning and dynamically variable bias management.' In: *International Joint Conference on Artificial Intelligence*, pp. 308–314.

Robbins, Herbert and Sutton Monro (1951). 'A stochastic approximation method'. In: *The Annals of Mathematical Statistics*, pp. 400–407.

Rosenstein, Michael T, Zvika Marx, Leslie Pack Kaelbling and Thomas G Dietterich (2005). 'To transfer or not to transfer'. In: *NIPS 2005 workshop on transfer learning*. Vol. 898, pp. 1–4.

Rothfuss, Jonas, Vincent Fortuin, Martin Josifoski and Andreas Krause (2021). 'PACOH: Bayes-optimal meta-learning with PAC-guarantees'. In: *International Conference on Machine Learning*, pp. 9116–9126.

Russakovsky, Olga et al. (2015). 'ImageNet large scale visual recognition challenge'. In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Rusu, Andrei A, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero and Raia Hadsell (2019). 'Meta-learning with latent embedding optimization'. In: *International Conference on Learning Representations*.

Ruvolo, Paul and Eric Eaton (2013). 'Active task selection for lifelong machine learning'. In: *AAAI Conference on Artificial Intelligence*. Vol. 27.

Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra and Timothy Lillicrap (2016). 'Meta-learning with memory-augmented neural networks'. In: *International Conference on Machine Learning*, pp. 1842–1850.

Schmidhuber, Jürgen (1987). 'Evolutionary principles in self-referential learning (On learning how to learn: the meta-meta-... hook)'. Diploma thesis. Technische Universität München.

Sener, Ozan and Vladlen Koltun (2018). 'Multi-Task Learning as Multi-Objective Optimization'. In: *Advances in Neural Information Processing Systems*.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Al-Shedivat, Maruan, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch and Pieter Abbeel (2018). 'Continuous adaptation via meta-learning in nonstationary and competitive environments'. In: *International Conference on Learning Representations*.

Shimodaira, Hidetoshi (2000). 'Improving predictive inference under covariate shift by weighting the log-likelihood function'. In: *Journal of Statistical Planning and Inference* 90.2, pp. 227–244.

Shui, Changjian, Mahdieh Abbasi, Louis-Emile Robitaille, Boyu Wang and Christian Gagné (2019). 'A principled approach for learning task similarity in multitask learning'. In: *International Joint Conference on Artificial Intelligence*, pp. 3446–3452.

Simonyan, Karen and Andrew Zisserman (2015). 'Very deep convolutional networks for large-scale image recognition'. In: *International Conference on Learning Representations*.

Snell, Jake, Kevin Swersky and Richard Zemel (2017). 'Prototypical networks for few-shot learning'. In: *Advances in Neural Information Processing Systems*, pp. 4077–4087.

Song, Hao, Tom Diethe, Meelis Kull and Peter Flach (June 2019). 'Distribution calibration for regression'. In: *International Conference on Machine Learning*, pp. 5897–5906.

Sung, Flood, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr and Timothy M. Hospedales (2018). 'Learning to compare: relation network for few-shot learning'. In: *Conference on Computer Vision and Pattern Recognition*.

Tassa, Yuval, Tom Erez and Emanuel Todorov (2012). 'Synthesis and stabilization of complex behaviors through online trajectory optimization'. In: *International Conference on Intelligent Robots and Systems*. IEEE, pp. 4906–4913.

Teh, Yee W, David Newman and Max Welling (2007). 'A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation'. In: *Advances in Neural Information Processing Systems*, pp. 1353–1360.

Thrun, Sebastian and Joseph O'Sullivan (1996). 'Discovering structure in multiple learning tasks: The TC algorithm'. In: *International Conference on Machine Learning*. Vol. 96, pp. 489–497.

Thrun, Sebastian and Lorien Pratt (1998). *Learning to learn*. Springer Science & Business Media.

Todorov, Emanuel and Weiwei Li (2005). 'A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems'. In: *American Control Conference*. IEEE, pp. 300–306.

Tran, Anh T, Cuong V Nguyen and Tal Hassner (2019). 'Transferability and hardness of supervised classification tasks'. In: *International Conference on Computer Vision*, pp. 1395–1405.

Tversky, Amos (1977). 'Features of similarity.' In: *Psychological Review* 84.4, p. 327.

Utgoff, Paul E (1986). 'Shift of bias for inductive concept learning'. In: *Machine learning: An Artificial Intelligence Approach* 2, pp. 107–148.

Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, Daan Wierstra et al. (2016). 'Matching networks for one shot learning'. In: *Advances in Neural Information Processing Systems*. Vol. 29, pp. 3630–3638.

Wang, Yixin, Alp Kucukelbir and David M Blei (2017). 'Robust probabilistic modeling with bayesian data reweighting'. In: *International Conference on Machine Learning*, pp. 3646–3655.

Xu, Zhixiong, Xiliang Chen, Wei Tang, Jun Lai and Lei Cao (2021). 'Meta weight learning via model-agnostic meta-learning'. In: *Neurocomputing* 432, pp. 124–132.

Xue, Ya, Xuejun Liao, Lawrence Carin and Balaji Krishnapuram (2007). 'Multi-task learning for classification with Dirichlet process priors'. In: *Journal of Machine Learning Research* 8.Jan, pp. 35–63.

Yakowitz, Sidney and Brian Rutherford (1984). 'Computational aspects of discrete-time optimal control'. In: *Applied Mathematics and Computation* 15.1, pp. 29–45.

Yao, Zhewei, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer and Michael Mahoney (2021). 'ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning'. In: *AAAI Conference on Artificial Intelligence*, pp. 10665–10673.

Yoon, Jaesik, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio and Sungjin Ahn (2018). 'Bayesian model-agnostic meta-learning'. In: *Advances in Neural Information Processing Systems*, pp. 7332–7342.

Yosinski, Jason, Jeff Clune, Yoshua Bengio and Hod Lipson (2014). 'How transferable are features in deep neural networks?' In: *Advances in Neural Information Processing Systems*.

Zamir, Amir R, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik and Silvio Savarese (2018). 'Taskonomy: Disentangling task transfer learning'. In: *Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722.

Zhang, Ruixiang, Tong Che, Zoubin Ghahramani, Yoshua Bengio and Yangqiu Song (2018). 'MetaGAN: An adversarial approach to few-shot learning'. In: *Advances in Neural Information Processing Systems*, pp. 2371–2380.

Zhang, Yu and Dit-Yan Yeung (2012). 'A convex formulation for learning task relationships in multi-task learning'. In: *Conference on Uncertainty in Artificial Intelligence*.

Appendix A

Variational inference meta-learning

A.1 Multi-modal regression from sinusoidal and linear task distribution

A.1.1 Training configuration

As mentioned in Subsection 3.5.1, the experiment is carried out on a multi-modal structured data, where a half of tasks are generated from sinusoidal functions, while the other half of tasks are from linear functions. The sinusoidal functions are in the form of $A\sin(x+\varphi)$, where the amplitude A and the phase φ are uniformly sampled from [0.1, 5] and [0, π], respectively. The linear functions are in the form of $ax+b$, where the slope a and the intercept b are sampled from the uniform distribution on [-3, 3]. The input x is uniformly sampled from [-5, 5]. In addition, a Gaussian noise with zero-ed mean and a standard deviation of 0.3 is added to the output.

The model used in this experiment is a 3-hidden fully connected neural network with 100 hidden units per each hidden layer. Output from each layer is activated by ReLU without batch normalisation. The optimisation for the objective function in (3.5) is carried out by Adam. Note that for regression, there is we do not place any weighting factor for the KL divergence term of VFE. Please refer to Table A.1 for the details of hyperparameters used.

A.1.2 Additional results

We also implement many Bayesian meta-learning methods, such as PLATIPUS, BMAML and Amortised Meta-learner, to compare with VAMPIRE using reliability diagram. We train all the methods of interest in the same setting used for VAMPIRE to obtain a fair comparison. The mean-squared error (MSE) of each method after

Table A.1: Hyper-parameters used in the regression experiments on multi-modal structured data.

Hyper-parameters	**Notation**	**Value**
Learning rate for variational parameters	α	0.001
Number of gradient updates for variational parameters		5
Number of Monte Carlo samples for variational parameters	L_t	128
Number of tasks before updating meta-parameters	T	10
Learning rate for meta-parameters	γ	0.001
Number of Monte Carlo samples for meta-parameters	L_v	128

training can be referred to Table A.2. Please note that for probabilistic methods, MSE is the average value across many Monte Carlo samples or particles sampled from the posterior distribution of model parameters.

Table A.2: Mean squared error of many meta-learning methods after being trained in the same setting are tested on 1000 tasks.

Method	**MSE**
MAML	1.96
PLATIPUS	1.86
BMAML	1.12
Amortised Meta-learner	2.32
VAMPIRE	2.24

A.2 Classification experiments

This section describes the detailed setup to train and validate the few-shot learning on Omniglot and mini-ImageNet presented in Subsection 3.5.2. Following the notation used in Subsection 3.4.1, each task or episode i has N classes, where the support set $S_i^{(t)}$ has k samples per class, and the query set $S_i^{(v)}$ has 15 samples per class. This is to be consistent with the previous works in the literature (Ravi and Larochelle, 2018; Finn et al., 2017). The training is carried out by using Adam to minimise the cross-entropy loss of the softmax output. The learning rate of the meta-parameters θ is set to be $\gamma = 10^{-3}$ across all trainings, and decayed by a factor of 0.99 after every 10,000 tasks. Other hyperparameters used are specified in Table A.3. We select the number of ensemble models L_t and L_v to fit into the memory of one Nvidia 1080 Ti GPU. Higher values of L_t and L_v are desirable to achieve a better Monte Carlo approximation.

For the experiments using extracted features (Rusu et al., 2019) presented in Table A.4 for mini-ImageNet, and the bottom part of Table 3.2 for tiered-ImageNet,

Table A.3: Hyper-parameters used in the few-shot classification presented in Section 3.5.

Description	**Notation**	**Omniglot** 5-way	**Omniglot** 20-way	**mini-ImageNet** 5-way
Number tasks per meta-update	T	32	16	2
Number of ensemble models (train)	L_t (train)	1	1	10
Number of ensemble models (train)	L_v (train)	1	1	10
Number of ensemble models (test)	L_t (test)	10	10	10
Number of ensemble models (test)	L_v (test)	10	10	10
Learning rate for $\mathbf{w}_i$	α	0.1	0.1	0.01
Learning rate for θ	γ	10^{-3}	10^{-3}	10^{-3}
Number of inner gradient updates		5	5	5

we used a 2-hidden fully connected layer with 128 and 32 hidden units. The learning rate α is set as 0.01 and 5 gradient updates were carried out. The learning rate for meta-parameters was $\gamma = 0.001$.

Both the experiments for classification re-weight the KL divergence term of VFE by a factor of 0.1.

Table A.4: Accuracy for 5-way classification on mini-ImageNet tasks (in percentage) of many methods which uses extra parameters, deeper network architectures or different training settings.

	Mini-ImageNet (Ravi and Larochelle, 2018) 1-shot	5-shot
Non-standard CNN		
Relation nets (Sung et al., 2018)	50.44 ± 0.82	65.32 ± 0.70
VERSA (Gordon et al., 2019)	53.40 ± 1.82	67.37 ± 0.86
SNAIL (Mishra et al., 2018)	55.71 ± 0.99	68.88 ± 0.92
adaResNet (Munkhdalai et al., 2018)	56.88 ± 0.62	71.94 ± 0.57
TADAM (Oreshkin et al., 2018)	58.5 ± 0.30	76.7 ± 0.30
LEO (Rusu et al., 2019)	61.76 ± 0.08	77.59 ± 0.12
MetaOptNet (Lee et al., 2019)	**64.09 ± 0.62**	**80.00 ± 0.45**
VAMPIRE	62.16 ± 0.24	76.72 ± 0.37

A.2.1 Model calibration for classification - ECE and MCE

We provide the results of model calibration, in particular, ECE and MCE in the numeric form. We also include the 95% confidence interval in Table A.5, although they are extremely small due to the large number of unseen tasks.

Table A.5: Results of ECE and MCE of several meta-learning methods that are tested in 5-way 1-shot setting over 15504 unseen tasks sampled from mini-ImageNet dataset.

Method	**ECE**	**MCE**
MAML	0.0410 ± 0.005	0.124
PLATIPUS	0.032 ± 0.005	0.108
BMAML	0.025 ± 0.006	0.092
Amortised Meta-learner	0.026 ± 0.003	0.058
VAMPIRE	0.008 ± 0.002	0.038

A.3 Pseudo-code for evaluation

Algorithm 6 VAMPIRE testing

1: **Require:** a new task $\mathcal{T}_{T+1}$, θ, L_t, L_v, and α
2: $\lambda_{T+1} \leftarrow \boldsymbol{\theta}$
3: sample $\hat{\mathbf{w}}_{T+1}^{(l)} \sim q(\mathbf{w}_{T+1}|\lambda_{T+1})$, where $l_t = 1 : L_t$
4: approximate $\mathsf{L}(S_{T+1}^{(t)}, \lambda_{T+1})$ in (3.10) via Monte Carlo
5: update: $\lambda_{T+1}^{*} \leftarrow \lambda_{T+1} - \alpha \boldsymbol{\nabla}_{\lambda_{T+1}} \mathsf{L}(S_{T+1}^{(t)}, \lambda_{T+1})$
6: draw L_v ensemble model parameters $\hat{\mathbf{w}}_i^{(l_v)} \sim q(\mathbf{w}_i; \lambda_{T+1}^{*})$
7: compute prediction $\hat{\mathbf{y}}_{(T+1),j}^{(v)(l_v)} = h(\mathbf{x}_{(T+1),j}; \hat{\mathbf{w}}_i^{(l_v)})$

Appendix B

PAC-Bayes meta-learning

B.1 Proof of PAC-Bayes upper-bound for meta-learning

The derivation is divided into three steps. The first two steps are to derive the PAC-Bayes bound for the generalisation errors induced by the unseen queried examples within each task (Appendix B.1.1) and the unseen tasks (Appendix B.1.2). The novel bound is then constructed by combining the results obtained in the first two steps and presented in Theorem 4.2.

B.1.1 PAC-Bayes upper-bound of the validation loss for a single task

This subsection presents a PAC-Bayes upper-bound of the validation loss for task $\mathcal{T}_i$, and in particular, $\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right]$. Note that this loss is different from the left hand-side term of Theorem 4.1 used for single-task learning (presented in Subsection 4.3.4) at the additional expectation over the posterior $q(\theta;\psi)$.

Lemma B.1

If $S_i^{(v)}$ consists of $m_i^{(v)}$ input $\mathbf{x}_{ik}^{(v)}$ sampled from a data probability distribution $\mathcal{D}_i$ and labelled by f_i, $\mathcal{H}$ is a hypothesis class with each hypothesis h parameterised by $\mathbf{w}_i$, $\ell : \mathcal{H} \times \mathcal{Y} \to [0,1]$ is a loss function, $q(\mathbf{w}_i;\lambda_i^*)$ is a "posterior" over the hypothesis parameter and $p(\mathbf{w}_i)$ is a prior, then the following holds with the

probability at least $1-\varepsilon_i$:

$$\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ij}^{(v)},y_{ij}^{(v)};\mathbf{w}_i\right)\right] \leq \frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ik}^{(v)},y_{ik}^{(v)};\mathbf{w}_i\right)\right]+R_i$$

where: $\varepsilon_i \in (0,1]$, and

$$R_i=\sqrt{\frac{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\frac{\ln m_i^{(v)}}{\varepsilon_i}}{2\left(m_i^{(v)}-1\right)}} \tag{B.1}$$

Proof. To simplify the proof, let $\Delta\mathsf{L}$ be the difference between the true loss and empirical loss:

$$\Delta\mathsf{L}=\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\left[\ell\left(\mathbf{x}_{ij}^{(v)},y_{ij}^{(v)};\mathbf{w}_i\right)\right]-\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\ell\left(\mathbf{x}_{ik}^{(v)},y_{ik}^{(v)};\mathbf{w}_i\right).$$

Given the Fubini's theorem to interchange the expectations, the problem can be re-written as:

$$\Pr\left(\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right]\leq\sqrt{\frac{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\frac{\ln m_i^{(v)}}{\varepsilon_i}}{2\left(m_i^{(v)}-1\right)}}\right)\geq 1-\varepsilon_i. \tag{B.2}$$

We are now proceeding to prove the lemma. First, applying Lemma 4.3 (presented in Subsection 4.4.3 and proved in Appendix B.2) with $2(m_i^{(v)}-1)\Delta\mathsf{L}^2$ as $\phi(h)$ gives:

$$2\left(m_i^{(v)}-1\right)\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}^2\right]\leq\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]+\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]. \tag{B.3}$$

The next steps are to lower-bound the left-hand side term in (B.3), while upper-bounding the last term of the right-hand side in (B.3).

To lower-bound the left-hand side term of (B.3), Jensen's inequality is applied on the convex function x^2 to obtain:

$$2\left(m_i^{(v)}-1\right)\left(\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right]\right)^2\leq 2\left(m_i^{(v)}-1\right)\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}^2\right]. \tag{B.4}$$

The results in (B.3) and (B.4) lead to:

$$\sqrt{2\left(m_i^{(v)}-1\right)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right]\leq\sqrt{\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]+\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]}. \tag{B.5}$$

Taking expectation over $q(\theta;\psi)$ on both sides gives:

$$\sqrt{2\left(m_i^{(v)}-1\right)}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right] \\ \leq \mathbb{E}_{q(\theta;\psi)}\left[\sqrt{\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]+\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]}\right]. \tag{B.6}$$

Note that $\sqrt{x}$ is a concave function. Hence, one can apply Jensen's inequality on the right-hand side term to obtain an upper-bound. This results in:

$$\sqrt{2\left(m_i^{(v)}-1\right)}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right] \\ \leq \sqrt{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\mathbb{E}_{q(\theta;\psi)}\left[\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\right]}. \tag{B.7}$$

To upper-bound the last term in (B.7), Lemma B.4 (presented in Appendix B.2) is then used. To do that, $\Delta\mathsf{L}$ is required to satisfied the assumption of Lemma B.4. This can be done by applying the Hoeffding's inequality. Consider each loss value $\ell(\mathbf{x}_{ik}^{(v)},y_{ik};\mathbf{w}_i)\in[0,1]$ as an i.i.d. random variable where its true mean and empirical mean are $\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}[\ell(\mathbf{x}_{ij}^{(v)},y_{ij}^{(v)};\mathbf{w}_i)]$ and $\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\ell(\mathbf{x}_{ik}^{(v)},y_{ik}^{(v)};\mathbf{w}_i)$, respectively. Hence, applying Hoeffding's inequality gives:

$$\Pr\left(|\Delta L|\geq\epsilon\right)\leq e^{-2m_i^{(v)}\epsilon^2},\forall\epsilon\geq 0 \tag{B.8}$$

Therefore, this allows to apply Lemma B.4 to upper-bound the last term in the right-hand side of (B.7) to obtain:

$$\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\leq m_i^{(v)}. \tag{B.9}$$

Taking the expectation w.r.t. the prior $p(\mathbf{w}_i)$on both sides gives:

$$\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\leq m_i^{(v)}. \tag{B.10}$$

Note that the two expectations on the left-hand side term are interchanged due to Fubini's theorem. Taking the logarithm on both side, and applying Jensen's inequality to lower-bound the left-hand side term give:

$$\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\leq\ln\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\leq\ln m_i^{(v)}. \tag{B.11}$$

Taking the expectation over the distribution $q(\theta;\psi)$ on both sides and applying Fubini's theorem to interchange the two expectations on the left-hand side give:

$$\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\mathbb{E}_{q(\theta;\psi)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\leq\ln m_i^{(v)}. \tag{B.12}$$

The lower-bound (or the term in the left-hand side of the above inequality) can be lower-bounded further by applying Markov's inequality:

$$\Pr\left(\mathbb{E}_{q(\theta;\psi)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\geq\epsilon\right)\leq\frac{\mathbb{E}_{(\mathcal{D}_i^{(v)},f)}\mathbb{E}_{q(\theta;\psi)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]}{\epsilon},\tag{B.13}$$

where $\epsilon>0$.

This implies that:

$$\Pr\left(\mathbb{E}_{q(\theta;\psi)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\geq\epsilon\right)\leq\frac{\ln m_i^{(v)}}{\epsilon},\forall\epsilon>0.\tag{B.14}$$

Equivalently, one can write the above inequality as:

$$\Pr\left(\mathbb{E}_{q(\theta;\psi)}\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2\left(m_i^{(v)}-1\right)\Delta\mathsf{L}^2}\right]\leq\epsilon\right)\geq 1-\frac{\ln m_i^{(v)}}{\epsilon},\forall\epsilon>0.\tag{B.15}$$

Hence, adding an expectation of a KL divergence on both sides of the inequality inside the probability function and taking square root gives:

$$\begin{aligned}\Pr\Bigg(&\sqrt{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\mathbb{E}_{q(\theta;\psi)}\left[\ln\mathbb{E}_{p(\mathbf{w}_i)}\left[e^{2(m_i^{(v)}-1)\Delta\mathsf{L}^2}\right]\right]}\leq\\&\leq\sqrt{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\epsilon}\Bigg)\geq 1-\frac{\ln m_i^{(v)}}{\epsilon},\forall\epsilon>0.\end{aligned}\tag{B.16}$$

The results in (B.7) and (B.16) lead to:

$$\begin{aligned}&\Pr\left(\sqrt{2\left(m_i^{(v)}-1\right)}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right]\leq\sqrt{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\epsilon}\right)\\&\geq 1-\frac{\ln m_i^{(v)}}{\epsilon},\ \forall\epsilon>0.\end{aligned}\tag{B.17}$$

Setting $\varepsilon_i=\frac{\ln m_i^{(v)}}{\epsilon}$ and dividing both sides of the inequality inside the probability function by $\sqrt{2\left(m_i^{(v)}-1\right)}$ give:

$$\Pr\left(\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\Delta\mathsf{L}\right]\leq\sqrt{\frac{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i^*)||p(\mathbf{w}_i)\right]\right]+\frac{\ln m_i^{(v)}}{\varepsilon_i}}{2\left(m_i^{(v)}-1\right)}}\right)\geq 1-\varepsilon_i.\tag{B.18}$$

□

B.1.2 PAC-Bayes upper-bound for unseen tasks

The PAC-Bayes upper-bound on the generalisation loss for unseen tasks can be obtained as a corollary of Theorem 4.1 (presented in Subsection 4.3.4). In particular, if:

- the loss function is defined as $\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}[\ell(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i)]$,
- data generation is the task environment $p(\mathcal{D}, f)$,
- dataset consists of $T > 1$ tasks i.i.d. sampled from the task environment,
- hypothesis is θ,
- posterior is $q(\theta; \psi)$,
- and prior is $p(\theta)$,

then one can apply Theorem 4.1 to obtain a PAC-Bayes upper-bound for unseen tasks as shown in Corollary B.2.

Corollary B.2

$$\Pr\Bigg(\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{p(\mathcal{D},f)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right] \leq \\ \leq \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right] + R_0\Bigg) \geq 1 - \varepsilon_0,$$

where: $\varepsilon_0 \in (0, 1]$, and

$$R_0 = \sqrt{\frac{\mathrm{KL}\left[q(\theta;\psi)||p(\theta)\right] + \frac{\ln T}{\varepsilon_0}}{2(T-1)}}. \tag{B.19}$$

B.1.3 PAC-Bayes upper-bound for meta-learning

Theorem 4.2

Given T tasks sampled from the same task environment $p(\mathcal{D}, f)$, where each task has an associated pair of datasets $(S_i^{(t)}, S_i^{(v)})$ with samples generated from the task-specific data generation model $(\mathcal{D}_i^{(t)}, \mathcal{D}_i^{(v)}, f_i)$, then for a bounded loss function $\ell : \mathcal{W} \times \mathcal{Y} \to [0, 1]$ and any distributions $q(\theta; \psi)$ of meta-parameter θ and $q(\mathbf{w}_i; \lambda_i)$ of task-specific parameter $\mathbf{w}_i$, the following holds with the probability

at least $1-\varepsilon, \forall \varepsilon \in (0,1]$:

$$\begin{aligned}
&\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{p(\mathcal{D},f)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right] \\
&\leq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda)}\left[\ell\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}; \mathbf{w}_i\right)\right] \\
&\quad + \sqrt{\frac{\mathbb{E}_{q(\theta;\psi)}\left[\mathrm{KL}\left[q(\mathbf{w}_i;\lambda_i)||p(\mathbf{w}_i)\right]\right] + \frac{T^2}{(T-1)\varepsilon}\ln m_i^{(v)}}{2\left(m_i^{(v)}-1\right)}} \\
&\quad + \sqrt{\frac{\mathrm{KL}\left[q(\theta;\psi)||p(\theta)\right] + \frac{T\ln T}{\varepsilon}}{2(T-1)}},
\end{aligned}$$

where $p(\mathbf{w}_i), \forall i \in \{1,\ldots,T\}$ is the prior of task-specific parameter $\mathbf{w}_i$ and $p(\theta)$ is the prior of meta-parameter θ.

Proof. First, the upper-bound for the unseen examples of a single-task obtained from Lemma B.1 (presented in Appendix B.1.1) is extended for T training tasks by using Lemma B.7 (presented in Appendix B.2) with the following substitution:

- $X_i := \mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right]$
- $Y_i := \frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}; \mathbf{w}_i\right)\right] + R_i$

to obtain:

$$\begin{aligned}
&\Pr\left(\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right]\right. \\
&\quad \left.\leq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}; \mathbf{w}_i\right)\right] + R_i\right) \geq 1 - \sum_{i=1}^{T}\varepsilon_i. \quad \text{(B.20)}
\end{aligned}$$

Given Corollary B.2 (presented in Appendix B.1.2) and the result in (B.20), one can apply Corollary B.8 (presented in Appendix B.2) to obtain the following:

$$\begin{aligned}
&\Pr\left(\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{(\mathcal{D},f)}\mathbb{E}_{(\mathcal{D}_i^{(v)},f_i)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ij}^{(v)}, y_{ij}^{(v)}; \mathbf{w}_i\right)\right]\right. \\
&\quad \left.\leq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{m_i^{(v)}}\sum_{k=1}^{m_i^{(v)}}\mathbb{E}_{q(\theta;\psi)}\mathbb{E}_{q(\mathbf{w}_i;\lambda_i^*)}\left[\ell\left(\mathbf{x}_{ik}^{(v)}, y_{ik}^{(v)}; \mathbf{w}_i\right)\right] + R_i + R_0\right) \geq 1 - \sum_{j=0}^{T}\varepsilon_j. \quad \text{(B.21)}
\end{aligned}$$

Setting $\varepsilon_0 = \frac{\varepsilon}{T}$ and $\varepsilon_i = \frac{(T-1)\varepsilon}{T^2}, \forall i \in \{1,\ldots,T\}$ completes the proof. □

B.2 Auxiliary lemmas

Lemma 4.3: Compression lemma (Banerjee, 2006)

For any measurable function $\phi(h)$ on a set of predictors under consideration $\mathcal{H}$, and any distributions P and Q on $\mathcal{H}$, the following holds:

$$\mathbb{E}_Q\left[\phi(h)\right] - \ln \mathbb{E}_P\left[e^{\phi(h)}\right] \leq \mathrm{KL}\left[Q\|P\right].$$

Further,

$$\sup_{\phi} \mathbb{E}_Q\left[\phi(h)\right] - \ln \mathbb{E}_P\left[e^{\phi(h)}\right] = \mathrm{KL}\left[Q\|P\right].$$

Proof. For a measurable function $\phi(h)$:

$$\mathbb{E}_Q\left[\phi(h)\right] = \mathbb{E}_Q\left[\ln\left(e^{\phi(h)}\frac{Q(h)}{P(h)}\frac{P(h)}{Q(h)}\right)\right] = \mathrm{KL}\left[Q||P\right] + \mathbb{E}_Q\left[\ln\left(e^{\phi(h)}\frac{P(h)}{Q(h)}\right)\right]. \tag{B.22}$$

Applying Jensen's inequality on the last term gives:

$$\mathbb{E}_Q\left[\phi(h)\right] \leq \mathrm{KL}\left[Q||P\right] + \ln \mathbb{E}_Q\left[e^{\phi(h)}\frac{P(h)}{Q(h)}\right] = \mathrm{KL}\left[Q||P\right] + \ln \mathbb{E}_P\left[e^{\phi(h)}\right]. \tag{B.23}$$

Re-arrange the term proves the first part of the lemma, which is the lower-bound of the KL divergence.

To prove the second part of the lemma, we simply show that there exists a function $\phi(h)$ that makes the lower-bound achieve the maximal value which is the KL divergence. Let:

$$\phi(h) = \ln \frac{Q(h)}{P(h)}, \tag{B.24}$$

then the lower-bound can be expressed as:

$$\mathbb{E}_Q\left[\phi(h)\right] - \ln \mathbb{E}_P\left[e^{\phi(h)}\right] = \mathrm{KL}\left[Q||P\right] - \ln \mathbb{E}_P\left[\frac{Q(h)}{P(h)}\right] = \mathrm{KL}\left[Q||P\right]. \tag{B.25}$$

That completes the proof. □

Lemma B.4: Exercise 31.1 in (Shalev-Shwartz and Ben-David, 2014)

If X is a random variable that satisfies: $\Pr\left(X \geq \epsilon\right) \leq e^{-2m\epsilon^2}, \forall \epsilon \geq 0$, then:

$$\mathbb{E}\left[e^{2(m-1)X^2}\right] \leq m.$$

Proof. For simplicity, let $Y = e^{2(m-1)X^2}$. Since $X^2 \in [0, +\infty)$, then $Y \in [1, +\infty)$. Thus, we can write the value of Y as the following:

$$Y = \int_0^Y dt = \int_1^{+\infty} \mathbb{1}\left(Y \geq t\right) dt + 1,$$

where $\mathbb{1}(A)$ is the indicator function of event A. The equality is equivalent to the area of a rectangle in which its height is 1 and its width is Y.

One important property of indicator function is that:

$$\mathbb{E}\left[\mathbb{1}\left(Y \geq t\right)\right] = \Pr\left(Y \geq t\right).$$

With the above representation, we can express the expectation of interest as:

$$\begin{aligned}
\mathbb{E}\left[Y\right] &= \mathbb{E}\left[\int_1^{+\infty} \mathbb{1}\left(Y \geq t\right) dt\right] + 1 \\
&= \int_1^{+\infty} \mathbb{E}\left[\mathbb{1}\left(Y \geq t\right)\right] dt + 1 \quad \text{(Fubini's theorem)} \\
&= \int_1^{+\infty} \Pr\left(Y \geq t\right) dt + 1.
\end{aligned}$$

Or:

$$\mathbb{E}\left[e^{2(m-1)X^2}\right] = \int_1^{+\infty} \Pr\left(e^{2(m-1)X^2} \geq x\right) dx + 1.$$

We will change the variable from x to ϵ to utilise the given inequality $\Pr\left(X \geq \epsilon\right) \leq e^{-2m\epsilon^2}, \forall \epsilon \geq 0$. Let:

$$x = e^{2(m-1)\epsilon^2},$$

where $\epsilon \geq 0$. This leads to the followings:

$$\epsilon = \sqrt{\frac{\ln x}{2(m-1)}}, \tag{B.26}$$

and

$$dx = 4(m-1)\epsilon e^{2(m-1)\epsilon^2} d\epsilon.$$

And since $x \in [1, +\infty)$, $\epsilon \in [0, +\infty)$ due to Eq. (B.26). The expectation of interest can then be written in term of the changed variable ϵ as follows:

$$\begin{aligned}
\mathbb{E}\left[e^{2(m-1)X^2}\right] &= \int_0^{+\infty} \Pr\left(e^{2(m-1)X^2} \geq e^{2(m-1)\epsilon^2}\right) 4(m-1)\epsilon e^{2(m-1)\epsilon^2} d\epsilon + 1 \\
&= \int_0^{+\infty} \underbrace{\Pr\left(X \geq \epsilon\right)}_{\leq e^{-2m\epsilon^2}} 4(m-1)\epsilon e^{2(m-1)\epsilon^2} d\epsilon + 1 \\
&\leq 4(m-1) \underbrace{\int_0^{+\infty} \epsilon e^{-2\epsilon^2} d\epsilon}_{1/4} + 1 = m.
\end{aligned}$$

□

Lemma B.5

For $i = 1 : n$, if X_i and Y_i are random variables, then:

$$p\left(\sum_{i=1}^{n} X_i \leq \sum_{i=1}^{n} Y_i\right) \geq p\left(\bigcap_{i=1}^{n} (X_i \leq Y_i)\right).$$

Proof. The proof is quite direct:

$$X_i \leq Y_i \implies \sum_{i=1}^{n} X_i \leq \sum_{i=1}^{n} Y_i. \tag{B.27}$$

Hence, applying the probability for implication completes the proof. □

Lemma B.6

For n events A_i with $i = 1 : n$, the following holds:

$$p\left(\bigcap_{i=1}^{n} A_i\right) \geq \left(\sum_{i=1}^{n} p(A_i)\right) - (n-1), \forall n \geq 2.$$

Proof. Proof can be done by induction.

For $n = 2$:

$$p(A_1 \cap A_2) = p(A_1) + p(A_2) - p(A_1 \cup A_2) \geq p(A_1) + p(A_2) - 1.$$

Suppose that it is true for case n:

$$p\left(\bigcap_{i=1}^{n} A_i\right) \geq \left(\sum_{i=1}^{n} p(A_i)\right) - (n-1).$$

We prove that this is also true for case $(n+1)$:

$$\begin{aligned}
p\left(\bigcap_{i=1}^{n+1} A_i\right) &= p\left(\bigcap_{i=1}^{n} A_i\right) + p(A_{n+1}) - p\left(\left(\bigcap_{i=1}^{n} A_i\right) \bigcup A_{n+1}\right) \\
&\geq p\left(\bigcap_{i=1}^{n} A_i\right) + p(A_{n+1}) - 1 \\
&\geq \left(\sum_{i=1}^{n} p(A_i)\right) - (n-1) + p(A_{n+1}) - 1 \\
&\quad \text{(assumption of induction for case } n\text{)} \\
&\geq \left(\sum_{i=1}^{n+1} p(A_i)\right) - ((n+1)-1).
\end{aligned}$$

It is, therefore, true for $(n+1)$, and hence, the proof. □

Lemma B.7

Let X_i and Y_i are random variables with $i = 1 : n$. If $p(X_i \leq Y_i) \geq 1 - \delta_i$ with $\delta_i \in (0, 1]$, then:

$$p\left(\sum_{i=1}^{n} X_i \leq \sum_{i=1}^{n} Y_i\right) \geq 1 - \sum_{i=1}^{n} \delta_i.$$

Proof. Applying Lemmas B.5 and B.6 for the left-hand side term of this lemma gives:

$$\begin{aligned} p\left(\sum_{i=1}^{n} X_i \leq \sum_{i=1}^{n} Y_i\right) &\geq p\left(\bigcap_{i=1}^{n} (X_i \leq Y_i)\right) \quad \text{(Lemma B.5)} \\ &\geq \sum_{i=1}^{n} p\left((X_i \leq Y_i)\right) - (n-1) \quad \text{(Lemma B.6)} \\ &\geq \sum_{i=1}^{n} (1 - \delta_i) - (n-1) \\ &= 1 - \sum_{i=1}^{n} \delta_i. \end{aligned} \tag{B.28}$$

□

Corollary B.8

If $p(a \leq b) \geq 1 - \delta_1$ and $p(b \leq c) \geq 1 - \delta_2$ with $\delta_1, \delta_2 \in (0, 1]$, then:

$$p(a \leq c) \geq 1 - \delta_1 - \delta_2.$$

B.3 Complexity analysis

In this section, we analyse the running time complexity of different meta-learning algorithms related to SImPa per one gradient update for the parameter of interest. These methods include:

- point estimate such as MAML (Finn et al., 2017),
- probabilistic modelling based on variational inference where the posterior is approximated by a multivariate normal distribution with a diagonal covariance matrix (Ravi and Beatson, 2019; C. Nguyen et al., 2020),
- SImPa.

For simplicity, we assume that the number of samples within each training subset $S_i^{(t)}$ is the same across all tasks: $m_0^{(t)} = m_i^{(t)}, \forall i \in \{1, \ldots, T\}$, and so is the validation subset. In addition, as all the methods mentioned are implemented with their first-order versions, the analysis also relies on such assumption. Furthermore, given

that such algorithms are implemented with automatic differentiation, the running time complexity of a back-propagation is linear w.r.t. the number of the model's parameters.

To ease the analysis, we re-state the definition of some notations and define some new ones as shown in Table B.1.

B.3.1 Deterministic point estimate meta-learning (MAML)

Lower-level optimisation for each task

The back-propagation of a single gradient update is $\mathcal{O}(m_0^{(t)}n)$. This is then repeated N_{lower} times. Hence, the total complexity to adapt to each task is $\mathcal{O}(N_{\text{lower}}m_0^{(t)}n)$.

Upper-level optimisation

Given the task-specific parameter $\mathbf{w}_i$ obtained in the lower-level of (4.3), one can calculate the gradient of the validation loss on each task w.r.t. the meta-parameter similarly to back-propagation. The complexity is, therefore, $\mathcal{O}(m_0^{(v)}n)$. And since there are T tasks in total, the complexity of this step is $\mathcal{O}(m_0^{(v)}Tn)$.

The total complexity of the whole algorithm is: $\mathcal{O}((N_{\text{lower}}m_0^{(t)} + m_0^{(v)}T)n)$.

B.3.2 Probabilistic meta-learning with multivariate normal distributions

These methods have similar complexity as the deterministic point estimate method, except the association of Monte Carlo sampling to draw $\mathbf{w}_i$ from $q(\mathbf{w}_i; \lambda_i)$ when evaluating the training and validation losses.

Lower-level optimisation for each task

The optimisation for the lower-level in (4.3) now has 2 steps:

- Sampling for task-specific parameter: $\mathbf{w}_i \sim q(\mathbf{w}_i; \lambda_i)$, resulting in a complexity of $\mathcal{O}(n)$
- Back-propagation with complexity of $\mathcal{O}(m_0^{(t)}n)$.

These steps are repeated from $m_{\mathbf{w}}$ samples, and iterated N_{lower}, resulting in a total complexity of $\mathcal{O}(m_{\mathbf{w}}N_{\text{lower}}(m_0^{(t)} + 1)n)$.

Table B.1: Notations used in the running time complexity analysis.

Notations	**Description**
n	the number of base model parameters $= \lvert\mathbf{w}_i\rvert$
n'	the number of the hyper-parameters $= \lvert\boldsymbol{\mu}_\theta\rvert = \lvert\theta\rvert$
N_{lower}	the number of gradient updates to minimise lower-level function in (4.3)
N_ϕ	the number of gradient updates to learn the task-specific ϕ-net
$m_{\mathbf{w}}$	the number of Monte Carlo samples drawn from task-specific posterior $q(\mathbf{w}_i; \lambda_i)$
m_θ	the number of Monte Carlo samples drawn from hyper-meta posterior $q(\theta; \psi)$
$m_0^{(t)}$	the number of samples in the training subset of each task
$m_0^{(v)}$	the number of samples in the validation subset of each task
m_ϕ	the number of samples generated to train the ϕ network in SImPa
T	the number of tasks within a mini-batch to update the meta-parameter of interest

Upper-level optimisation

Similarly, the upper-level is also affected by the Monte Carlo sampling of task-specific parameter $\mathbf{w}_i$ from $q(\mathbf{w}_i; \lambda_i)$. This results in a complexity of $\mathcal{O}(m_{\mathbf{w}} T(m_0^{(v)} + 1)n)$.

The total running time complexity is, therefore:

$$\mathcal{O}(m_{\mathbf{w}}(N_{\text{lower}}(m_0^{(t)} + 1) + T(m_0^{(v)} + 1))n).$$

B.3.3 SImPa

The forward pass of SImPa is more complicated than the deterministic point estimate and the probabilistic modelling mentioned above. The reason is that the parameter of interest is not the meta-parameter θ, but the hyper-parameter ψ or $\boldsymbol{\mu}_\theta$, which is one level higher in the hierarchical structure.

In addition, we assume that the generator is much larger than the base model:

$$n' \gg n$$

Lower-level optimisation

The optimisation for the lower-level is carried out in 2 main steps. The first step is to train the ϕ network which consists of:

- Sample m_ϕ latent noise vectors $\{\mathbf{z}_\kappa\}_{\kappa=1}^{m_\phi}$: $\mathcal{O}(m_\phi z)$ with z being assumed to be much less than n'
- Forward pass to generate $\mathbf{w}_i$ from the generator represented by the implicit distribution $q(\mathbf{w}_i; \lambda_i)$: $\mathcal{O}(m_\phi n')$
- Sample $\mathbf{w}_i$ from the prior $p(\mathbf{w})$: $\mathcal{O}(m_\phi n)$
- Back-propagate to train the ϕ network: $\mathcal{O}(2m_\phi n')$

This process is repeated N_ϕ times, resulting in a total complexity of $\mathcal{O}(3N_\phi m_\phi n')$ to train ϕ-network.

Given the trained ϕ-network for a particular task, the second step is to adapt the meta-parameter to the task-specific parameter:

- Generate $\mathbf{w}_i$ from the generator: $\mathcal{O}(n')$
- Back-propagation to train the task-specific generator: $\mathcal{O}(m_0^{(t)} n')$.

Table B.2: Running time complexity per one gradient update of different meta-learning methods.

Method	**Complexity**	
	General	In practice[1]
Deterministic	$\mathcal{O}((N_{\text{lower}} m_0^{(t)} + m_0^{(v)} T)n)$	$\mathcal{O}(100n)$
Probabilistic	$\mathcal{O}(m_{\mathbf{w}}(N_{\text{lower}}(m_0^{(t)} + 1) + T(m_0^{(v)} + 1))n)$	$\mathcal{O}(440n)$
SImPa	$\mathcal{O}(m_\theta(N_{\text{lower}}(m_{\mathbf{w}}(m_0^{(t)} + 1) + 3N_\phi m_\phi) + T(m_0^{(v)} + 1))n')$	$\mathcal{O}(125n')$

This is repeated $m_{\mathbf{w}}$ times, resulting in a complexity of $\mathcal{O}(m_{\mathbf{w}}(m_0^{(t)} + 1)n'$.

And again, this whole process of training both the ϕ-network and the generator is repeated N_{lower} times. Thus, the total complexity is $\mathcal{O}(N_{\text{lower}}(m_{\mathbf{w}}(m_0^{(t)} + 1) + 3N_\phi m_\phi)n')$.

Upper-level optimisation

Given the task-specific generator obtained in the optimisation of the lower-level in (4.3), the gradient of the hyper-meta-parameter in the upper-level can be calculated in 2 steps:

- Sample θ from $q(\theta; \psi)$: $\mathcal{O}(n')$
- Backward to calculate the gradient: $\mathcal{O}(m_0^{(v)} n')$

This is done for T tasks, resulting in a complexity of $\mathcal{O}(T(m_0^{(v)} + 1)n')$ per a single θ.

The optimisation in both the lower- and upper-levels is then repeated m_θ times for m_θ samples of θ. Thus, the total complexity of SImPa is: $\mathcal{O}(m_\theta(N_{\text{lower}}(m_{\mathbf{w}}(m_0^{(t)} + 1) + 3N_\phi m_\phi) + T(m_0^{(v)} + 1))n')$.

To ease the analysis, we specify the running time complexity of the three methods in Table B.2.

To make it even easier to compare, we also add a "practical" setting into Table B.2. This setting is the one done in our experiments with the following hyper-parameter values: $N_{\text{lower}} = 5$, $N_\phi = 1$, $m_0^{(t)} = 5$ and $m_0^{(v)} = 15$, $m_\theta = m_{\mathbf{w}} = 1$ for SImPa and $m_{\mathbf{w}} = 4$ for probabilistic methods that are based on multivariate normal distributions with diagonal covariance matrices, $T = 5$, and $N_\phi = 256$.

We note that despite the difference of the running time complexity, in the implementation using GPU, some operations, such as matrix multiplication, are implemented efficiently in parallel or vectorisation. Hence, the difference of running time in practice might not be the same as the one analysed in this appendix.

[1] assume the sampling of N_ϕ samples is parallel in GPU

Appendix C

Probabilistic task modelling for meta-learning

C.1 Calculation of each term in the ELBO

As described in Section 5.3, the variational distributions for $\mathbf{u}, \mathbf{z}$ and $\boldsymbol{\pi}$ are:

$$q(\mathbf{u}_{in}; \phi) = \mathcal{N}\left(\mathbf{u}_{in}; \mathbf{m}_{in}, (\mathbf{s}_{in})^2 \mathbf{I}\right) \tag{5.4}$$

$$q(\boldsymbol{\pi}_i; \boldsymbol{\gamma}_i) = \text{Dirichlet}\left(\boldsymbol{\pi}_i; \boldsymbol{\gamma}_i\right) \tag{5.10}$$

$$q(\mathbf{z}_{in}; \mathbf{r}_{in}) = \text{Categorical}\left(\mathbf{z}_{in}; \mathbf{r}_{in}\right). \tag{5.11}$$

C.1.1 $\mathbb{E}_{q(\mathbf{u}_i; \boldsymbol{\mu}_{u_i}, \boldsymbol{\Sigma}_{u_i})} \mathbb{E}_{q(\mathbf{z}_i, \boldsymbol{\pi}_i)} \left[\ln p(\mathbf{u}_i | \mathbf{z}_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right]$

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{z}_i, \boldsymbol{\pi}_i)} \left[\ln p(\mathbf{u}_i | \mathbf{z}_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right] &= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{ink} \ln p(\mathbf{u}_{in} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{ink} \ln \mathcal{N}(\mathbf{u}_{in} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).
\end{aligned} \tag{C.1}$$

Hence:

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{u}_i; \boldsymbol{\mu}_{u_i}, \boldsymbol{\Sigma}_{u_i})} \mathbb{E}_{q(\mathbf{z}_i, \boldsymbol{\pi}_i)} \left[\ln p(\mathbf{u}_i | \mathbf{z}_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right] \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{ink} \underbrace{\mathbb{E}_{q(\mathbf{u}_i; \boldsymbol{\mu}_{u_i}, \boldsymbol{\Sigma}_{u_i})} \left[\ln \mathcal{N}(\mathbf{u}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]}_{\text{cross-entropy between 2 Gaussians}} \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{ink} \left[-\frac{1}{2} \text{tr}(\boldsymbol{\Sigma}_k^{-1} \boldsymbol{\Sigma}_{u_{in}}) + \ln \mathcal{N}(\boldsymbol{\mu}_{u_{in}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right].
\end{aligned} \tag{C.2}$$

C.1.2 $\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln p(\mathbf{z}_i|\boldsymbol{\pi}_i)\right]$

$$\begin{aligned}\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln p(\mathbf{z}_i|\boldsymbol{\pi}_i)\right] &= \sum_{n=1}^{N}\sum_{k=1}^{K} r_{ink}\int \mathrm{Dir}_K(\boldsymbol{\pi}_i;\boldsymbol{\gamma}_i)\ln\pi_{ik}\,\mathrm{d}\pi_{ik} \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} r_{ink}\ln\tilde{\pi}_{ik}, \end{aligned} \tag{C.3}$$

where:

$$\boxed{\ln\tilde{\pi}_{ik} = \psi(\gamma_{ik}) - \psi\left(\sum_{j=1}^{K}\gamma_{ij}\right).} \tag{C.4}$$

C.1.3 $\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln p(\boldsymbol{\pi}_i|\boldsymbol{\alpha})\right]$

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln p(\boldsymbol{\pi}_i|\boldsymbol{\alpha})\right] \\ &= \mathbb{E}_{\mathrm{Dir}(\boldsymbol{\pi}_i;\boldsymbol{\gamma}_i)}\left[\ln\Gamma\left(\sum_{j=1}^{K}\alpha_j\right) - \left[\sum_{k=1}^{K}\ln\Gamma(\alpha_k) - (\alpha_k-1)\ln\pi_{ik}\right]\right] \\ &= \left[\ln\Gamma\left(\sum_{j=1}^{K}\alpha_j\right) - \sum_{k=1}^{K}\ln\Gamma(\alpha_k)\right] + \sum_{k=1}^{K}(\alpha_k-1)\ln\tilde{\pi}_{ik}. \end{aligned} \tag{C.5}$$

C.1.4 $\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln q(\mathbf{z}_i|\mathbf{r}_i)\right]$

$$\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln q(\mathbf{z}_i|\mathbf{r}_i)\right] = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{ink}\ln r_{ink}. \tag{C.6}$$

C.1.5 $\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln q(\boldsymbol{\pi}_i|\boldsymbol{\gamma}_i)\right]$

$$\mathbb{E}_{q(\mathbf{u}_i;\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})}\mathbb{E}_{q(\mathbf{z}_i,\boldsymbol{\pi}_i)}\left[\ln q(\boldsymbol{\pi}_i|\boldsymbol{\gamma}_i)\right] = \ln\Gamma\left(\sum_{j=1}^{K}\gamma_{ij}\right) - \sum_{k=1}^{K}\left[\ln\Gamma(\gamma_{ik}) - (\gamma_{ik}-1)\ln\tilde{\pi}_{ik}\right]. \tag{C.7}$$

C.2 Maximisation of the ELBO

Since the ELBO can be evaluated as shown in Appendix C.1, we can maximise the ELBO w.r.t. "task-specific" variational parameters by taking derivative, setting it to zero and solving for the parameters of interest.

C.2.1 Variational categorical distribution

Note that:

$$\sum_{k=1}^{K} r_{ink} = 1. \tag{C.8}$$

The derivative of L_i with respect to r_{ink} can be expressed as:

$$\frac{\partial \mathsf{L}}{\partial r_{ink}} = -\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}}) + \ln \mathcal{N}(\boldsymbol{\mu}_{u_{in}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \ln \tilde{\pi}_{ik} - \ln r_{ink} - 1 + \lambda, \tag{C.9}$$

where: λ is the Lagrange multiplier and $\ln \tilde{\pi}_{ik}$ is defined in Eq. (C.4). Setting the derivative to zero and solving for r_{ink} give:

$$\boxed{r_{ink} \propto \exp\left[-\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}}) + \ln \mathcal{N}(\boldsymbol{\mu}_{u_{in}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \ln \tilde{\pi}_{ik}\right].} \tag{C.10}$$

C.2.2 Variational Dirichlet distribution

The lower-bound related to γ_{ik} can be written as:

$$\begin{aligned}
\mathsf{L} &= \sum_{k=1}^{K}\sum_{n=1}^{N} r_{ink} \ln \tilde{\pi}_{ik} + \sum_{k=1}^{K}(\alpha_k - 1)\ln \tilde{\pi}_{ik} - \ln \Gamma\left(\sum_{j=1}^{K}\gamma_{ij}\right) \\
&\quad + \sum_{k=1}^{K}\left[\ln \Gamma(\gamma_{ik}) - (\gamma_{ik} - 1)\ln \tilde{\pi}_{ik}\right] \\
&= -\ln \Gamma\left(\sum_{j=1}^{K}\gamma_{ij}\right) + \sum_{k=1}^{K}\ln \tilde{\pi}_{ik}\left(\alpha_k - \gamma_{ik} + \sum_{n=1}^{N} r_{ink}\right) + \ln \Gamma(\gamma_{ik}) \\
&= -\ln \Gamma\left(\sum_{j=1}^{K}\gamma_{ij}\right) + \sum_{k=1}^{K}\left[\psi(\gamma_{ik}) - \psi\left(\sum_{j=1}^{K}\gamma_{ij}\right)\right]\left(\alpha_k - \gamma_{ik} + \sum_{n=1}^{N} r_{ink}\right) \\
&\quad + \ln \Gamma(\gamma_{ik}).
\end{aligned} \tag{C.11}$$

Hence, the lower-bound related to γ_{ik} is:

$$\begin{aligned}
\mathsf{L}[\gamma_{ik}] = &-\ln \Gamma\left(\sum_{j=1}^{K}\gamma_{ij}\right) + \psi(\gamma_{ik})\left(\alpha_k - \gamma_{ik} + \sum_{n=1}^{N} r_{ink}\right) \\
&- \psi\left(\sum_{j=1}^{K}\gamma_{ij}\right)\left(\sum_{j=1}^{K}\alpha_j - \gamma_{ij} + \sum_{n=1}^{N} r_{inj}\right) + \ln \Gamma(\gamma_{ik})
\end{aligned} \tag{C.12}$$

Taking derivative w.r.t. γ_{ik} gives:

$$\begin{aligned}
\frac{\partial \mathsf{L}}{\partial \gamma_{ik}} &= -\psi\left(\sum_{j=1}^{K}\gamma_{ij}\right) + \Psi(\gamma_{ik})\left(\alpha_k - \gamma_{ik} + \sum_{n=1}^{N} r_{ink}\right) - \psi(\gamma_{ik}) \\
&\quad - \Psi\left(\sum_{j=1}^{K}\right)\left(\sum_{j=1}^{K}\alpha_j - \gamma_{ij} + \sum_{n=1}^{N} r_{inj}\right) + \psi\left(\sum_{j=1}^{K}\gamma_{ij}\right) + \psi(\gamma_{ik}) \\
&= \Psi(\gamma_{ik})\left(\alpha_k - \gamma_{ik} + \sum_{n=1}^{N} r_{ink}\right) - \Psi\left(\sum_{j=1}^{K}\right)\sum_{j=1}^{K}\alpha_j - \gamma_{ij} + \sum_{n=1}^{N} r_{inj},
\end{aligned} \tag{C.13}$$

where $\Psi(.)$ is the trigamma function.

Setting the derivative to zero yields a maximum at:

$$\boxed{\gamma_{ik} = \alpha_k + N_{ik},} \tag{C.14}$$

where:

$$N_{ik} = \sum_{n=1}^{N} r_{ink}. \tag{C.15}$$

C.2.3 Maximum likelihood for the task-theme $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$

The terms in the objective function relating to μ_k can be written as:

$$\begin{aligned}\mathsf{L}[\mu_k] &= \sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink} \ln \mathcal{N}\left(\boldsymbol{\mu}_{u_{in}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \\ &= -\frac{1}{2}\sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink} \left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)^\top \boldsymbol{\Sigma}_k^{-1} \left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)\end{aligned} \tag{C.16}$$

Taking gradient w.r.t. $\boldsymbol{\mu}_k$ gives:

$$\boldsymbol{\nabla}_{\boldsymbol{\mu}_k}\mathsf{L} = \sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k). \tag{C.17}$$

Setting the gradient to zero yields a maximum at:

$$\boxed{\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\boldsymbol{\mu}_{u_{in}}}{\sum_{i=1}^{T} N_{ik}}.} \tag{C.18}$$

The terms in the objective function relating to $\boldsymbol{\Sigma}_k$ is given as:

$$\begin{aligned}\mathsf{L} &= \sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\left[-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}}) + \ln \mathcal{N}(\boldsymbol{\mu}_{u_{in}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \\ &= -\frac{1}{2}\sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\Big[\mathrm{tr}(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}}) + d\ln(2\pi) + \ln|\boldsymbol{\Sigma}_k| \\ &\quad + \left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)^\top \boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)\Big].\end{aligned} \tag{C.19}$$

Taking gradient w.r.t. $\boldsymbol{\Sigma}_k$ gives:

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\Sigma}_k}\mathsf{L} &= -\frac{1}{2}\sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\Big[-\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}}\boldsymbol{\Sigma}_k^{-1} + \boldsymbol{\Sigma}_k^{-1} \\ &\quad -\boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)\left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)^\top \boldsymbol{\Sigma}_k^{-1}\Big] \\ &= \frac{1}{2}\sum_{i=1}^{T}\sum_{n=1}^{N} r_{ink}\left[\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{u_{in}} - \mathbf{I} + \boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)\left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k\right)^\top\right]\boldsymbol{\Sigma}_k^{-1}.\end{aligned} \tag{C.20}$$

Setting the gradient to zero gives:

$$\boxed{\boldsymbol{\Sigma}_k = \frac{1}{\sum_{i=1}^{T} N_{ik}} \sum_{i=1}^{T} \sum_{n=1}^{N} r_{ink} \left[\boldsymbol{\Sigma}_{u_{in}} + \left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k \right) \left(\boldsymbol{\mu}_{u_{in}} - \boldsymbol{\mu}_k \right)^{\top} \right].} \tag{C.21}$$

C.2.4 Maximum likelihood for $\boldsymbol{\alpha}$

The lower-bound with terms relating to α_k can be expressed as:

$$\mathsf{L} = T \left[\ln \Gamma \left(\sum_{j=1}^{K} \alpha_j \right) - \sum_{k=1}^{K} \ln \Gamma(\alpha_k) \right] + \sum_{i=1}^{T} \sum_{k=1}^{K} (\alpha_k - 1) \left[\psi(\gamma_{ik}) - \psi \left(\sum_{j=1}^{K} \gamma_{ij} \right) \right]. \tag{C.22}$$

Taking derivative w.r.t. α_k gives:

$$g_k = \frac{\partial \mathsf{L}}{\partial \alpha_k} = T \left[\psi \left(\sum_{j=1}^{K} \alpha_j \right) - \psi(\alpha_k) \right] + \sum_{i=1}^{T} \left[\psi(\gamma_{ik}) - \psi \left(\sum_{j=1}^{K} \gamma_{ij} \right) \right]. \tag{C.23}$$

The second derivative is, therefore, obtained as:

$$\frac{\partial^2 \mathsf{L}}{\partial \alpha_k \partial \alpha_{k'}} = T \left[\Psi \left(\sum_{j=1}^{K} \alpha_j \right) - \delta(k - k') \Psi(\alpha_k) \right]. \tag{C.24}$$

The Hessian can be written in matrix form (Minka, 2000) as:

$$\mathbf{H} = \mathbf{Q} + \mathbf{1}\mathbf{1}^T a \tag{C.25}$$

$$q_{kk'} = -T \delta(k - k') \Psi(\alpha_k) \tag{C.26}$$

$$a = T \Psi \left(\sum_{j=1}^{K} \alpha_j \right). \tag{C.27}$$

One Newton step is therefore:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \mathbf{H}^{-1} \mathbf{g} \tag{C.28}$$

$$(\mathbf{H}^{-1} \mathbf{g})_k = \frac{g_k - b}{q_{kk}}, \tag{C.29}$$

where:

$$b = \frac{\sum_{j=1}^{K} g_j / q_{jj}}{1/a + \sum_{j=1}^{K} 1/q_{jj}}. \tag{C.30}$$

Appendix D

Task weighting for meta-learning

D.1 Derivation of iLQR

This section presents the derivation for one iteration of iLQR adopted from the original iLQR papers (Todorov and W. Li, 2005; Tassa et al., 2012). One difference is at the backtracking line-search where we employ a similar acceptance criterion as in DDP.

For brevity, the finite-horizon discrete-time optimal control problem in (6.1) is restated:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}_t} \mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t) \quad \text{s.t. } \mathbf{u}_t \in \mathcal{U}_t, \tag{6.5}$$

where $\mathcal{U}_t$ is set of feasible weighting at time step t, corresponding to the weighting criterion.

Let Q_t be the *cost-to-go* defined as:

$$Q_t(\mathbf{x}_{t:T}, \mathbf{u}_{t:T}) = \sum_{j=t}^{T} c(\mathbf{x}_j, \mathbf{u}_j), \tag{D.1}$$

and V_t be the *value* function at time step t which is the cost-to-go given the local optimal action sequence:

$$V_t(\mathbf{x}_t) = \min_{\mathbf{u}_{t:T}} Q_t(\mathbf{x}_{t:T}, \mathbf{u}_{t:T}). \tag{D.2}$$

According to the Dynamic Programming Principle, the objective function which is the minimisation over an entire sequence of actions can be reduced to a sequence of minimisation over a single action, proceeding backwards in time:

$$V_t(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + V\left(\mathbf{x}_{t+1}\right)\right] = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + V\left(f(\mathbf{x}_t, \mathbf{u}_t)\right)\right]. \tag{D.3}$$

By applying the Taylor's series to the first order for the state-transition dynamics

f and to the second order for the cost function c about a nominal trajectory $\{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t\}$, the term inside the minimisation in (D.3) can be approximated to:

$$c(\mathbf{x}_t, \mathbf{u}_t) + V\left(f(\mathbf{x}_t, \mathbf{u}_t)\right) \approx \frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}^\top \begin{bmatrix}0 & \mathbf{q}_{\mathbf{x}_t}^\top & \mathbf{q}_{\mathbf{u}_t}^\top\\ \mathbf{q}_{\mathbf{x}_t} & \mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} & \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t}\\ \mathbf{q}_{\mathbf{u}_t} & \mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t} & \mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_t\\ \delta\mathbf{u}_t\end{bmatrix}, \tag{D.4}$$

where $\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$, $\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$, and

$$\begin{aligned}
\mathbf{q}_{\mathbf{x}_t} &= \mathbf{c}_{\mathbf{x}_t} + \mathbf{F}_{\mathbf{x}_t}^\top \mathbf{v}_{t+1} & \mathbf{q}_{\mathbf{u}_t} &= \mathbf{c}_{\mathbf{u}_t} + \mathbf{F}_{\mathbf{u}_t}^\top \mathbf{v}_{t+1}\\
\mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} &= \mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t} + \mathbf{F}_{\mathbf{x}_t}^\top \mathbf{V}_{t+1}\mathbf{F}_{\mathbf{x}_t} & \mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t} &= \mathbf{C}_{\mathbf{u}_t,\mathbf{u}_t} + \mathbf{F}_{\mathbf{u}_t}^\top \mathbf{V}_{t+1}\mathbf{F}_{\mathbf{u}_t}\\
\mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t} &= \mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t}^\top = \mathbf{C}_{\mathbf{x}_t,\mathbf{u}_t} + \mathbf{F}_{\mathbf{x}_t}^\top \mathbf{V}_{t+1}\mathbf{F}_{\mathbf{u}_t},
\end{aligned} \tag{D.5}$$

with $\mathbf{F}_{\mathbf{x}_t}$ and $\mathbf{F}_{\mathbf{u}_t}$ are the first derivatives of the state-transtition dynamics f w.r.t. the variable in the subscripts, $\mathbf{c}_{(.)}$ and $\mathbf{C}_{(.)}$ are the first and second derivatives of the cost function c w.r.t. the variables in the subscripts, and $\mathbf{v}_{t+1}$ and $\mathbf{V}_{t+1}$ are the first and second derivatives of the value function V w.r.t. the state $\mathbf{x}_{t+1}$.

Minimising the quadratic problem in (D.4) w.r.t. $\delta\mathbf{u}_t$ results in:

$$\delta\mathbf{u}_t^* = \mathbf{K}_t\delta\mathbf{x}_t + \mathbf{k}_t, \tag{D.6}$$

where:

$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1}\mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t} \qquad \mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1}\mathbf{q}_{\mathbf{u}_t}, \tag{D.7}$$

Substituting the action $\delta\mathbf{u}_t$ obtained in (D.6) into (D.4) gives a quadratic model:

$$V_t(\mathbf{x}_t) \approx \frac{1}{2}\delta\mathbf{x}_t^\top\mathbf{V}_t\delta\mathbf{x}_t + \delta\mathbf{x}_t^\top\mathbf{v}_t + \text{const.}, \tag{D.8}$$

where $\mathbf{V}_t$ and $\mathbf{v}_t$ are the second and first derivatives of the value function V_t, which can be expressed as:

$$\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t}\mathbf{K}_t \qquad \mathbf{v}_t = \mathbf{q}_{\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t}\mathbf{k}_t. \tag{D.9}$$

Hence, one can recursively calculate the local quadratic model $\{\mathbf{V}_t, \mathbf{v}_t\}$ of the value function V_t, and the linear controller $\{\mathbf{K}_t, \mathbf{k}_t\}$ backward through time. Once it is completed, a new trajectory can be calculated using the forward pass with the general non-linear state-transition dynamics f as follows:

$$\begin{aligned}
\hat{\mathbf{x}}_1 &= \mathbf{x}_1\\
\mathbf{u}_t &= \mathbf{K}_t\left(\mathbf{x}_t - \hat{\mathbf{x}}_t\right) + \mathbf{k}_t + \hat{\mathbf{u}}_t\\
\mathbf{x}_{t+1} &= f\left(\mathbf{x}_t, \mathbf{u}_t\right).
\end{aligned} \tag{D.10}$$

Although the trajectory obtained in (D.10) converges to a local minimum for the approximate model of the value function V_t, it does not guarantee the convergence for general non-linear models such as the one in (6.1). The reason is that the new trajectory might deviate too far from the nominal trajectory, resulting in a poor Taylor approximation of the true model. To overcome, a backtracking linear-search with parameter $\varepsilon \in (0, 1]$ is introduced:

$$\mathbf{u}_t = \mathbf{K}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t) + \varepsilon \mathbf{k}_t + \hat{\mathbf{u}}_t. \tag{D.11}$$

The criterion to accept the trajectory produced in the iteration with backtracking line-search is similar to the one in DDP:

$$J(\mathbf{u}_{1:T}) - J(\hat{\mathbf{u}}_{1:T}) < \frac{1}{2}\varepsilon\theta_1, \tag{D.12}$$

where

$$J(\mathbf{u}_{1:T}) = \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \tag{D.13}$$

$$\theta_t = \theta_{t+1} - \mathbf{q}_{\mathbf{u}_t}^{\top} \mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t} \mathbf{q}_{\mathbf{u}_t}. \tag{D.14}$$

Hence, in the worst case when the new trajectory strays too far from the model's region of validity, then $\varepsilon \to 0$ and the trajectory is the same as the nominal trajectory. The procedure of one iLQR iteration is outlined in Algorithm 7 in Appendix D.5. In addition, the convergence proof for iLQR adapted from DDP (Sidney Yakowitz and Rutherford, 1984) is provided in Appendix D.2 to complete the analysis.

D.2 Convergence of iLQR

To prove the convergence of iLQR algorithm, we rely on Theorem 3 in (Polak, 1971, p. 14), which is re-stated in Theorem D.1 in Appendix D.2.1

D.2.1 Auxiliary to prove convergence

Definition D.1: Algorithm model

Given $a : \mathcal{T} \to \mathcal{T}$ is an algorithm, and $c : \mathcal{T} \to \mathcal{R}$ is a function used as stopping criterion, the algorithm model can be described as:

1. Compute a $z_0 \in \mathcal{T}$.
2. Set $i = 0$.

3. Compute $a(z_i)$.

4. Set $z_{i+1} = a(z_i)$.

5. If $c(z_{i+1}) \geq c(z_i)$, stop;[a] else set $i = i + 1$ and go to step 3.

[a] A direct test for determining if z_i is desirable may be substituted for the test $c(z_{i+1}) \geq c(z_i)$.

Theorem D.1 will show what such an algorithm will compute.

Theorem D.1: (Polak, 1971, Theorem 3, p. 14)

Suppose that:

(i) $c(.)$ is either continuous at all non-desirable points $z \in \mathcal{T}$, or else $c(z)$ is bounded from below for $z \in \mathcal{T}$;

(ii) for every $z \in \mathcal{T}$ which is not desirable, there exist an $\varepsilon(z) > 0$ and a $\delta(z) < 0$ such that:

$$c(a(z')) - c(z') \leq \delta(z) < 0, \forall z' \in B(z, \varepsilon(z)) = \{z \in \mathcal{T} : \|z' - z\|_{\mathcal{B}} \leq \varepsilon(z)\}. \tag{D.15}$$

Then, either the sequence $\{z_i\}$ constructed by algorithm defined in Definition D.1 is finite and its next to last element is desirable, or else it is infinite and every accumulation point of $\{z_i\}$ is desirable.

D.2.2 Proof of iLQR convergence

Since iLQR is an algorithm model as described in Definition D.1, to prove its convergence property, one needs to assert that the 2 conditions in Theorem D.1 are met.

First condition is satisfied since the cost $J(\mathbf{u}_{1:T})$ is continuous, and twice differentiable. We will prove that iLQR satisfies the second condition of Theorem D.1.

From (D.11) and (D.14), the sequence of actions obtained from iLQR $\mathbf{u}_{1:T}$ and θ_1 depend on the nominal trajectory. Hence, we explicitly denote them as $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ and $\theta_1(\hat{\mathbf{u}})$. In addition, since the loss function $\ell(.)$ is assumed to be twice differentiable, the cost function and state-transition dynamics are continuous. Hence, both $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ and $\theta_1(\hat{\mathbf{u}})$ are also continuous.

The convergence proof for iLQR is presented in Appendix D.2.2. The proof requires some auxiliary lemmas in Appendix D.2.2.

Auxiliary lemmas

Lemma D.2

If $\mathbf{u}(\varepsilon, \bar{\mathbf{u}})$ is a non-stationary sequence of actions, then $\theta_1 < 0$.

Proof. It is apparent from the derivation of iLQR that $\mathbf{q}_{\mathbf{u}_t} = 0, \forall t \in \{1, \dots, T\}$ if and only if $\hat{\mathbf{u}}$ is a stationary sequence of actions. Hence, by negation, if $\hat{\mathbf{u}}$ is a non-stationary sequence of actions, $\mathbf{q}_{\mathbf{u}_t} \neq 0$ for some time steps $t \in \{1, \dots, T\}$.

Also note that, if $\mathbf{u}(\varepsilon, \bar{\mathbf{u}})$ is non-stationary, then $\hat{\mathbf{u}}$ is also non-stationary. In addition, if the Hessian matrix w.r.t. action $\mathbf{u}$ is assumed to be positive-definite (PSD), then from the construction, $\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}$ and its inverse are also PSD.

The PSD of $\mathbf{Q}^{-1}_{\mathbf{u}_t, \mathbf{u}_t}$ combining with (D.14) leads to the fact that $\theta_1(\hat{\mathbf{u}}) < 0$ for any non-stationary sequence of actions $\hat{\mathbf{u}}$. □

Lemma D.3

If $\mathbf{u}(\varepsilon, \bar{\mathbf{u}})$ is a non-stationary sequence of actions, then:

$$J(\mathbf{u}(\varepsilon, \bar{\mathbf{u}})) - J(\bar{\mathbf{u}}) = \varepsilon \theta_1(\bar{\mathbf{u}}) + \mathcal{O}(\varepsilon^2). \tag{D.16}$$

Proof. If we define ΔQ_t as the resulting incremental change in the cost-to-go Q from the perturbation $\Delta \mathbf{u}_t$ using Taylor expansion, then:

$$\Delta Q_t = Q_t(\mathbf{x}_t, \mathbf{u}_t + \Delta \mathbf{u}_t) - Q_t(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{q}_{\mathbf{u}_t}^\top \Delta \mathbf{u}_t + \mathcal{O}(\|\Delta \mathbf{u}_t\|^2). \tag{D.17}$$

Note that:

$$\begin{aligned} J(\mathbf{u}(\varepsilon, \hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) &= J(\mathbf{u}_{1:N} + \Delta \mathbf{u}_{1:N}) - J(\mathbf{u}_{1:N}) \\ &= \sum_{t=1}^{T} \Delta Q_t = \sum_{t=1}^{N} \mathbf{q}_{\mathbf{u}_t}^\top \Delta \mathbf{u}_t + \mathcal{O}\left(\|\Delta \mathbf{u}_{1:N}\|^2\right). \end{aligned} \tag{D.18}$$

Taking the derivative w.r.t. ε gives:

$$\frac{\partial J(\mathbf{u}(\varepsilon))}{\partial \varepsilon} = \sum_{t=1}^{T} \mathbf{q}_{\mathbf{u}_t}^\top \frac{\mathrm{d}\mathbf{u}_t}{\mathrm{d}\varepsilon}. \tag{D.19}$$

From (D.11), $\mathbf{u}_t$ is a linear function w.r.t. ε, resulting in:

$$\frac{\mathrm{d}\mathbf{u}_t}{\mathrm{d}\varepsilon} = \mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{q}_{\mathbf{u}_t}. \tag{D.20}$$

Hence:

$$\frac{\partial J(\mathbf{u}(\varepsilon))}{\partial \varepsilon} = -\sum_{t=1}^{T} \mathbf{q}_{\mathbf{u}_t}^\top \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{q}_{\mathbf{u}_t} = \theta_1(\hat{\mathbf{u}}), \tag{D.21}$$

which implies (D.16). □

Lemma D.4

If $\mathbf{u}(\varepsilon, \bar{\mathbf{u}})$ is a non-stationary sequence of actions, then there exists ε_1 such that:

$$J(\mathbf{u}(\varepsilon, \bar{\mathbf{u}})) - J(\hat{\mathbf{u}}) \leq \eta\varepsilon\theta_1(\hat{\mathbf{u}}), \forall\varepsilon \in [0, \varepsilon_1], \forall\eta \in (0.5, 1). \tag{D.22}$$

Proof. The result in Lemma D.3 can be re-written as:

$$J(\mathbf{u}(\varepsilon, \hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) = \eta\varepsilon\theta_1(\hat{\mathbf{u}}) + (1-\eta)\varepsilon\theta_1(\hat{\mathbf{u}}) + \mathcal{O}(\varepsilon^2), \forall\eta \in (0.5, 1). \tag{D.23}$$

Hence, there must exist a value $\varepsilon = \varepsilon_1 \geq 0$ such that: $(1-\eta)\theta_1(\hat{\mathbf{u}}) + \mathcal{O}(\varepsilon^2) < 0$. This leads to:

$$J(\mathbf{u}(\varepsilon, \hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) \leq \eta\varepsilon_1\theta_1(\hat{\mathbf{u}}) \leq \eta\varepsilon\theta_1(\hat{\mathbf{u}}), \forall\varepsilon \in [0, \varepsilon_1]. \tag{D.24}$$

Note that the second inequality holds due to the negativity of θ_1 proved in Lemma D.2. This completes the proof. □

Lemma D.5

Given that θ_1 and any action sequence $\mathbf{u}$ are continuous, and for any $\delta > 0$ such that $\|\hat{\mathbf{u}} - \mathbf{u}\| < \delta$, there exists $\varepsilon_2 > 0$ such that:

$$\theta_1(\mathbf{u}) - \varepsilon_2 < \theta_1(\hat{\mathbf{u}}) < \theta_1(\mathbf{u}) + \varepsilon_2.$$

Proof. Since θ_1 is assumed to be continuous, and the variable $\mathbf{u}$ is also continuous, we can employ the definition of continuity of function θ_1 at $\mathbf{u}$ to obtain the following:

$$\forall\varepsilon_2 > 0 \implies \exists\delta > 0 : |\theta_1(\hat{\mathbf{u}}) - \theta_1(\mathbf{u})| < \varepsilon_2, \forall\hat{\mathbf{u}} \in \{\hat{\mathbf{u}} \in \mathcal{U} : \|\hat{\mathbf{u}} - \mathbf{u}\| < \delta\}. \tag{D.25}$$

The negation form of the statement in (D.25) can be expressed as:

$$\forall\delta > 0 \implies \exists\varepsilon_2 > 0 : |\theta_1(\hat{\mathbf{u}}) - \theta_1(\mathbf{u})| < \varepsilon_2, \forall\hat{\mathbf{u}} \in \{\hat{\mathbf{u}} \in \mathcal{U} : \|\hat{\mathbf{u}} - \mathbf{u}\| < \delta\}. \tag{D.26}$$

Solving the inequation above completes the proof. □

Convergence of iLQR

Theorem D.6

Let the state-transition dynamics f and the cost function c have continuous second partial derivatives w.r.t the continuous state $\mathbf{x}$ and action $\mathbf{u}$. If $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ denotes the successive application of iLQR, then any accumulation point of $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ is stationary w.r.t the finite-horizon cost $J(\mathbf{u}_{1:T})$.

Proof. We first determine the condition that $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ calculated in (D.11) is the successor of iLQR at $\hat{\mathbf{u}}$. According to iLQR algorithm, $\mathbf{u}(\varepsilon, \hat{\mathbf{u}}) = \text{iLQR}(\hat{\mathbf{u}})$ only if:

$$J(\mathbf{u}(\varepsilon, \hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) \leq \frac{\varepsilon}{2}\theta_1(\hat{\mathbf{u}}). \tag{D.27}$$

Note that from Lemma D.4:

$$J(\mathbf{u}(\varepsilon, \hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) \leq \eta\varepsilon\theta_1(\hat{\mathbf{u}}) < \frac{\varepsilon}{2}\theta_1(\hat{\mathbf{u}}), \forall\eta \in (0.5, 1), \forall\varepsilon \in [0, \varepsilon_1] \tag{D.28}$$

which indicates that $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ is a successor of iLQR when $\varepsilon \in [0, \varepsilon_1]$.

If $\mathbf{u}(\varepsilon, \hat{\mathbf{u}})$ is a successor of iLQR, the acceptance criterion combining with the result in Lemma D.5 leads to:

$$J(\text{iLQR}(\hat{\mathbf{u}})) - J(\hat{\mathbf{u}}) \leq \frac{\varepsilon}{2}\theta_1(\hat{\mathbf{u}}) < \frac{\varepsilon}{2}\left[\theta_1(\mathbf{u}) + \varepsilon_2\right], \forall\hat{\mathbf{u}} \in \{\hat{\mathbf{u}} \in \mathcal{U} : \|\hat{\mathbf{u}} - \mathbf{u}\| < \delta\}. \tag{D.29}$$

Note that if δ is set to be small enough, then ε_2 is also very small, resulting in $\theta_1(\mathbf{u}) + \varepsilon_2 < 0$. If we set $\delta(\hat{\mathbf{u}}) = \delta$ and $\varepsilon(\hat{\mathbf{u}}) = \frac{\varepsilon}{2}\left[\theta_1(\mathbf{u}) + \varepsilon_2\right]$, then iLQR satisfies the second condition in Theorem D.1. □

D.3 Linearisation of state-transition dynamics

D.3.1 Stochastic gradient descent

The transition dynamics is given as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t - \alpha\nabla_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]. \tag{D.30}$$

Applying Taylor's expansion to the first order around a state-action pair $(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$ gives:

$$\begin{aligned}\mathbf{x}_{t+1} = \hat{\mathbf{x}}_{t+1} + \left(\mathbf{I}_D - \alpha\,\nabla^2_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\substack{\mathbf{x}_t=\hat{\mathbf{x}}_t \\ \mathbf{u}_t=\hat{\mathbf{u}}_t}}\right)(\mathbf{x}_t - \hat{\mathbf{x}}_t) \\ - \alpha\,\nabla^\top_{\mathbf{x}_t}\left[\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}(\mathbf{u}_t - \hat{\mathbf{u}}_t).\end{aligned} \tag{D.31}$$

It can also be written as:

$$\delta\mathbf{x}_{t+1} = \left(\mathbf{I}_D - \alpha\, \nabla^2_{\mathbf{x}_t} \left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\substack{\mathbf{x}_t=\hat{\mathbf{x}}_t \\ \mathbf{u}_t=\hat{\mathbf{u}}_t}}\right) \delta\mathbf{x}_t + \left(-\alpha\, \nabla^\top_{\mathbf{x}_t} \left[\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}\right) \delta\mathbf{u}_t. \quad \text{(D.32)}$$

Hence, the coefficient matrices of the Taylor's series for the state-transition dynamics following the SGD update can be expressed as:

$$\boxed{\begin{aligned} \mathbf{F}_{\mathbf{x}_t} &= \mathbf{I}_D - \alpha\, \nabla^2_{\mathbf{x}_t} \left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\substack{\mathbf{x}_t=\hat{\mathbf{x}}_t \\ \mathbf{u}_t=\hat{\mathbf{u}}_t}} && \text{(D.33a)} \\ \mathbf{F}_{\mathbf{u}_t} &= -\alpha\, \nabla^\top_{\mathbf{x}_t} \left[\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}. && \text{(D.33b)} \end{aligned}}$$

D.3.2 Adam

The gradient-based update for the parameter of interest using Adam keeps track of the mean and variance:

$$\begin{cases} \mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{J}_t^\top \mathbf{u}_t \\ \mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\left(\mathbf{J}_t^\top \mathbf{u}_t\right) \odot \left(\mathbf{J}_t^\top \mathbf{u}_t\right), \end{cases} \quad \text{(D.34)}$$

where:

- $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$,
- $\mathbf{J}_t$ is the Jacobian matrix of the validation losses for tasks in a minibatch t-th:

$$\mathbf{J}_t = \nabla_{\mathbf{x}_t} \left[\boldsymbol{\ell}(\mathbf{x}_t)\right]. \quad \text{(D.35)}$$

- $\odot$ is the elementwise multiplication.

The corrected-bias estimators are then defined as:

$$\begin{cases} \hat{\mathbf{m}}_t &= \dfrac{\mathbf{m}_t}{1-\beta_1^t} \\ \hat{\mathbf{v}}_t &= \dfrac{\mathbf{v}_t}{1-\beta_2^t}. \end{cases} \quad \text{(D.36)}$$

The update or state-transition dynamics is then given as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}. \quad \text{(D.37)}$$

The update for a new state of the model parameter can also be written by substitution:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \frac{\mathbf{m}_t}{1-\beta_1^t} \frac{1}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}} + \epsilon}. \quad \text{(D.38)}$$

The approximation using Taylor's expansion up to the first order will result with the following matrices:

$$\boxed{\begin{aligned}\mathbf{F}_{\mathbf{x}_t} &= \mathbf{I}_D - \frac{\alpha}{1-\beta_1^t}\,\nabla_{\mathbf{x}_t}\left[\frac{\mathbf{m}_t}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon}\right]\Bigg|_{\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t} && \text{(D.39a)}\\ \mathbf{F}_{\mathbf{u}_t} &= -\frac{\alpha}{1-\beta_1^t}\,\nabla_{\mathbf{u}_t}\left[\frac{\mathbf{m}_t}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon}\right]\Bigg|_{\hat{\mathbf{x}}_t,\hat{\mathbf{u}}_t}. && \text{(D.39b)}\end{aligned}}$$

For simplicity, we assume that $\mathbf{m}_{t-1}$ and $\mathbf{v}_{t-1}$ are constant w.r.t. both $\mathbf{x}_{t-1}$ and $\mathbf{u}_{t-1}$.

The gradient w.r.t. $\mathbf{u}_t$ on $\mathbf{m}_t$ and $\mathbf{v}_t$ can be expressed as:

$$\begin{aligned}\nabla_{\mathbf{u}_t}[\mathbf{m}_t] &= (1-\beta_1)\mathbf{J}_t^\top\\ \nabla_{\mathbf{u}_t}[\mathbf{v}_t] &= 2(1-\beta_2)\mathbf{J}_t^\top \odot \left(\mathbf{J}_t^\top \mathbf{u}_t\right).\end{aligned} \tag{D.40}$$

Note that given two functions f (a notation for a general function, not the state-transition dynamic) and g

$$\begin{aligned}\nabla_{\mathbf{u}_t}\left[\frac{f}{g}\right] &= \nabla_{\mathbf{u}_t}\left[f \odot \frac{1}{g}\right]\\ &= \nabla_{\mathbf{u}_t}[f] \odot \frac{1}{g} + f \odot \nabla_{\mathbf{u}_t}\left[\frac{1}{g}\right]\\ &= \nabla_{\mathbf{u}_t}[f] \odot \frac{1}{g} - f \odot \frac{1}{g^2} \odot \nabla_{\mathbf{u}_t}[g].\end{aligned} \tag{D.41}$$

Hence:

$$\nabla_{\mathbf{u}_t}\left[\frac{\mathbf{m}_t}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon}\right] = \frac{\nabla_{\mathbf{u}_t}[\mathbf{m}_t]}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon} - \frac{\mathbf{m}_t}{\left(\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon\right)^2} \odot \frac{\nabla_{\mathbf{u}_t}[\mathbf{v}_t]}{2\sqrt{(1-\beta_2^t)\mathbf{v}_t}}. \tag{D.42}$$

To calculate the gradient w.r.t. $\mathbf{x}_t$, the update of Adam is rewritten as:

$$\begin{cases}\mathbf{m}_t &= \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\nabla_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\\ \mathbf{v}_t &= \beta_2\mathbf{v}_{t-1} + (1-\beta_2)\nabla_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right] \odot \nabla_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right].\end{cases} \tag{D.43}$$

The gradient w.r.t. $\mathbf{x}_t$ on $\mathbf{m}_t$ and $\mathbf{v}_t$ can be expressed as:

$$\begin{aligned}\nabla_{\mathbf{x}_t}[\mathbf{m}_t] &= (1-\beta_1)\nabla^2_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\\ \nabla_{\mathbf{x}_t}[\mathbf{v}_t] &= 2(1-\beta_2)\nabla_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right] \odot \nabla^2_{\mathbf{x}_t}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right].\end{aligned} \tag{D.44}$$

Hence:

$$\nabla_{\mathbf{x}_t}\left[\frac{\mathbf{m}_t}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon}\right]=\frac{\nabla_{\mathbf{x}_t}\left[\mathbf{m}_t\right]}{\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon}-\frac{\mathbf{m}_t}{\left(\sqrt{\frac{\mathbf{v}_t}{1-\beta_2^t}}+\epsilon\right)^2}\odot\frac{\nabla_{\mathbf{x}_t}\left[\mathbf{v}_t\right]}{2\sqrt{(1-\beta_2^t)\mathbf{v}_t}}. \tag{D.45}$$

D.4 Quadraticise cost function w.r.t. state $\mathbf{x}_t$

The cost function consists of two terms: validation loss on the query subset and penalisation of the action $\mathbf{u}_t$.

D.4.1 Quadraticise validation loss

The loss term in (6.7) can be approximated to second order as:

$$\begin{aligned}\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\approx\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}&+(\mathbf{x}_t-\hat{\mathbf{x}}_t)^\top\nabla_{\mathbf{x}_t}\left[\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}\\&+\frac{1}{2}(\mathbf{x}_t-\hat{\mathbf{x}}_t)^\top\nabla_{\mathbf{x}_t}^2\left[\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t}(\mathbf{x}_t-\hat{\mathbf{x}}_t).\end{aligned} \tag{D.46}$$

D.4.2 Quadraticise the penalisation of the action $\mathbf{u}_t$

The penalisation term is already in the quadratic form. Here, it is rewritten to follow the Taylor's series form.

$$\begin{aligned}&\frac{\beta_u}{2}(\mathbf{u}_t-\mu_u\mathbf{1}_M)^\top(\mathbf{u}_t-\mu_u\mathbf{1}_M)\\&=\frac{\beta_u}{2}\left[(\mathbf{u}_t-\hat{\mathbf{u}}_t)+\hat{\mathbf{u}}_t-\mu_u\mathbf{1}_M\right]^\top\left[(\mathbf{u}_t-\hat{\mathbf{u}}_t)+\hat{\mathbf{u}}_t-\mu_u\mathbf{1}_M\right]\\&=\frac{1}{2}(\mathbf{u}_t-\hat{\mathbf{u}}_t)^\top(\beta_u\mathbf{I}_M)(\mathbf{u}_t-\hat{\mathbf{u}}_t)+(\mathbf{u}_t-\hat{\mathbf{u}}_t)^\top\beta_u(\hat{\mathbf{u}}_t-\mu_u\mathbf{1}_M).\end{aligned} \tag{D.47}$$

Given the locally-quadratic approximation of the cost function in (D.46) and (D.47), the coefficient matrices and vector of the quadratic form of the cost function can be written as:

$$\boxed{\begin{aligned}\mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t}&=\nabla_{\mathbf{x}_t}^2\left[\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t} && \text{(D.48a)}\\ \mathbf{C}_{\mathbf{x}_t,\mathbf{u}_t}&=\mathbf{C}_{\mathbf{u}_t,\mathbf{x}_t}=\mathbf{0} && \text{(D.48b)}\\ \mathbf{C}_{\mathbf{u}_t,\mathbf{u}_t}&=\beta_u\mathbf{I}_M && \text{(D.48c)}\\ \mathbf{c}_{\mathbf{x}_t}&=\nabla_{\mathbf{x}_t}\left[\mathbf{1}_M^\top\boldsymbol{\ell}(\mathbf{x}_t)\right]\Big|_{\mathbf{x}_t=\hat{\mathbf{x}}_t} && \text{(D.48d)}\\ \mathbf{c}_{\mathbf{u}_t}&=\beta_u(\hat{\mathbf{u}}_t-\mu_u\mathbf{1}_M). && \text{(D.48e)}\end{aligned}}$$

D.5 Trajectory optimisation algorithm(s)

Algorithm 7 Implementation of iLQR backward

procedure IBACKWARD($\{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t\}_{t=1}^{T}$)
  $\mathbf{V}_{T+1} = \mathbf{0}$, and $\mathbf{v}_{T+1} = \mathbf{0}$ ▷ Quadratic matrix and vector of cost-to-go
  $\theta_{T+1} = 0$ ▷ Stopping criterion for a nominal trajectory
  for $t = T : 1$ **do** ▷ Backward through time
    $\mathbf{F}_{\mathbf{x}_t}, \mathbf{F}_{\mathbf{u}_t}$ = linearise dynamics ▷ see Appendix D.3
    $\mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t}, \mathbf{C}_{\mathbf{x}_t,\mathbf{u}_t}, \mathbf{C}_{\mathbf{u}_t,\mathbf{x}_t}, \mathbf{C}_{\mathbf{u}_t,\mathbf{u}_t}, \mathbf{c}_{\mathbf{x}_t}, \mathbf{c}_{\mathbf{u}_t}$ = quadraticise cost function ▷ see Appendix D.4
    $\mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} = \mathbf{C}_{\mathbf{x}_t,\mathbf{x}_t} + \mathbf{F}_{\mathbf{x}_t}^{\top}\mathbf{V}_{t+1}\mathbf{F}_{\mathbf{x}_t}$ ▷ 2nd derivatives of cost-to-go
    $\mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t} = \mathbf{F}_{\mathbf{x}_t}^{\top}\mathbf{V}_{t+1}\mathbf{F}_{\mathbf{u}_t}$
    $\mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t} = \mathbf{F}_{\mathbf{u}_t}^{\top}\mathbf{V}_{t+1}\mathbf{F}_{\mathbf{x}_t}$
    $\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t} = \mathbf{C}_{\mathbf{u}_t,\mathbf{u}_t} + \mathbf{F}_{\mathbf{u}_t}^{\top}\mathbf{V}_{t+1}\mathbf{F}_{\mathbf{u}_t}$
    $\mathbf{q}_{\mathbf{x}_t} = \mathbf{c}_{\mathbf{x}_t} + \mathbf{F}_{\mathbf{x}_t}^{\top}\mathbf{v}_{t+1}$ ▷ 1st derivatives of cost-to-go
    $\mathbf{q}_{\mathbf{u}_t} = \mathbf{c}_{\mathbf{u}_t} + \mathbf{F}_{\mathbf{u}_t}^{\top}\mathbf{v}_{t+1}$
    $\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1}\mathbf{Q}_{\mathbf{u}_t,\mathbf{x}_t}$ ▷ Linear controller
    $\mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}^{-1}\mathbf{q}_{\mathbf{u}_t}$
    $\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t,\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t}\mathbf{K}_t$ ▷ 2nd derivatives of value function
    $\mathbf{v}_t = \mathbf{q}_{\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t,\mathbf{u}_t}\mathbf{k}_t$ ▷ 1st derivatives of value function
    $\theta_t = \theta_{t+1} - \mathbf{q}_{\mathbf{u}_t}^{\top}\mathbf{Q}_{\mathbf{u}_t,\mathbf{u}_t}\mathbf{q}_{\mathbf{u}_t}$
  end for
  return $\{\mathbf{K}_t, \mathbf{k}_t\}_{t=1}^{T}, \theta_1$
end procedure

D.6 Convergence analysis for TOW

D.6.1 Notations

The following notations are used throughout the paper:

- $||\mathbf{x}||$ is the L2 norm of a vector $\mathbf{x} \in \mathbb{R}^D$, e.g. $\sqrt{\mathbf{x}^{\top}\mathbf{x}}$
- $||\mathbf{A}||$ is the matrix norm of a matrix $\mathbf{A} \in \mathbb{R}^{M \times D}$ induced by a vector norm:

$$||\mathbf{A}|| = \sup \frac{||\mathbf{A}\mathbf{x}||}{||\mathbf{x}||}, \forall \mathbf{x} \in \mathbf{R}^D : ||\mathbf{x}|| \neq 0.$$

Given the vector and matrix norm, two common inequalities used in this section are:

- Triangle inequality: $||\mathbf{A}+\mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||, \forall \mathbf{A}, \mathbf{B} \in \mathbb{R}^{M \times D}$
- Sub-multiplication: $||\mathbf{A}\mathbf{x}|| \leq ||\mathbf{A}||||\mathbf{x}||$.

D.6.2 Assumptions on boundedness and smoothness

Assumption 6.1

The loss function of interest ℓ mentioned in (6.3) is B-bounded and L-Lipschitz.

Formally, the loss function ℓ satisfies the following conditions:

- Boundedness: $\exists B > 0 : \forall \mathbf{x} \in \mathbb{R}^D, |\ell(\mathbf{s}_{ij}, y_{ij}; \mathbf{x})| \leq B$,
- Lipschitz continuity:

$$\exists L > 0 : \forall \widetilde{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}^D, |\ell(\mathbf{s}_{ij}, y_{ij}; \widetilde{\mathbf{x}}) - \ell(\mathbf{s}_{ij}, y_{ij}; \overline{\mathbf{x}})| \leq L\|\widetilde{\mathbf{x}} - \overline{\mathbf{x}}\|.$$

This smoothness assumption for the loss function ℓ also implies that it has an L-Lipschitz continuous gradient (see Lemma D.12):

$$\|\boldsymbol{\nabla}_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x})\| \leq L. \tag{D.49}$$

Assumption 6.2

The gradient $\boldsymbol{\nabla}_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x})$ is S-Lipschitz.

This assumption means that:

$$\exists S > 0 : \forall \widetilde{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}^D, \|\boldsymbol{\nabla}_{\mathbf{x}} \ell(\mathbf{s}_{ij}, y_{ij}; \widetilde{\mathbf{x}}) - \boldsymbol{\nabla}_{\mathbf{x}} \ell(\mathbf{s}_{ij}, y_{ij}; \overline{\mathbf{x}})\| \leq S\|\widetilde{\mathbf{x}} - \overline{\mathbf{x}}\|.$$

In addition, such assumption also leads to the followings:

$$-S\mathbf{I} \preceq \boldsymbol{\nabla}^2_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x}) \preceq S\mathbf{I}, \tag{D.50}$$

where $\mathbf{I}$ is the identity matrix.

Assumption 6.3

The Hessian $\boldsymbol{\nabla}^2_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x})$ is ρ-Lipschitz.

This implies that:

$$\exists \rho > 0 : \forall \widetilde{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}^D, \left\|\boldsymbol{\nabla}^2_{\mathbf{x}} \ell(\mathbf{s}_{ij}, y_{ij}; \widetilde{\mathbf{x}}) - \boldsymbol{\nabla}^2_{\mathbf{x}} \ell(\mathbf{s}_{ij}, y_{ij}; \overline{\mathbf{x}})\right\| \leq \rho\|\widetilde{\mathbf{x}} - \overline{\mathbf{x}}\|.$$

Assumption 6.4

The variance of the gradient $\nabla_{\mathbf{x}}\ell$ is σ^2-bounded.

This implies that:

$$\exists\sigma > 0 : \forall \mathbf{x} \in \mathbb{R}^D, \mathbb{E}_{(\mathbf{s}_{ij},y)\sim\mathcal{D}_i}\left[\left\|\nabla_{\mathbf{x}}\ell(\mathbf{s}_{ij}, y_{ij}; \mathbf{x}) - \mathbb{E}_{(\mathbf{s}_{ij},y)\sim\mathcal{D}_i}\left[\nabla_{\mathbf{x}}\ell(\mathbf{s}_{ij}, y_{ij}; \mathbf{x})\right]\right\|^2\right] \leq \sigma^2.$$

D.6.3 Auxiliary lemmas

Boundedness of action (or weighting vector) u

Lemma 6.1

If $\mathbf{u}_t$ is a stationary action of a nominal action $\hat{\mathbf{u}}_t$ obtained from iLQR, then:

$$\exists\delta > 0 : \|\mathbf{u}_t - \hat{\mathbf{u}}_t\| \leq \delta.$$

Proof. According to the procedure of iLQR shown in (D.11):

$$\mathbf{u}_t - \hat{\mathbf{u}}_t = \mathbf{K}_t\left(\mathbf{x}_t - \hat{\mathbf{x}}_t\right) + \varepsilon\mathbf{k}_t. \tag{D.51}$$

Since the matrix $\mathbf{K}$, vectors $\mathbf{k}$, $\mathbf{x}$ and $\hat{\mathbf{x}}$ are well-defined, the norm of $\mathbf{u} - \hat{\mathbf{u}}$ is also well-defined.

In addition, δ is implicitly related to the Gaussian prior $\mathcal{N}(\mathbf{u}_t; \mu_u\mathbf{1}, {}^1\!/_{\beta_u}\mathbf{I}_M)$. A larger value of β_u in (6.7) would result in a smaller value for δ. □

Corollary D.8

If $\mathbf{u}_{t_1}$ and $\mathbf{u}_{t_2}$ are stationary actions of two nominal actions $\hat{\mathbf{u}}_{t_1} = \hat{\mathbf{u}}_{t_2} = {}^1\!/_M\,\mathbf{1}$ obtained from iLQR at time steps t_1 and t_2, respectively, then:

$$\|\mathbf{u}_{t_1} - \mathbf{u}_{t_2}\| \leq 2\delta \qquad \text{(bounded L2 norm of difference)} \tag{D.52a}$$

$$\|\mathbf{u}_{t_1}\| \leq \delta + \frac{1}{\sqrt{M}} \qquad \text{(bounded L2 norm)} \tag{D.52b}$$

$$\mathbf{1}^\top\mathbf{u}_{t_1} \leq \delta\sqrt{M} + 1. \qquad \text{(bounded L1 norm)} \tag{D.52c}$$

Proof. The first inequality can be proved by simply applying triangle inequality on the L2 norm and employing the result in Lemma 6.1:

$$\begin{aligned}\|\mathbf{u}_{t_1} - \mathbf{u}_{t_2}\| = \left\|(\mathbf{u}_{t_1} - \frac{1}{M}\mathbf{1}) + (\frac{1}{M}\mathbf{1} - \mathbf{u}_{t_2})\right\| &\leq \left\|\mathbf{u}_{t_1} - \frac{1}{M}\mathbf{1}\right\| + \left\|\mathbf{u}_{t_2} - \frac{1}{M}\mathbf{1}\right\| \\ &\leq \|\mathbf{u}_{t_1} - \hat{\mathbf{u}}_{t_1}\| + \|\mathbf{u}_{t_2} - \hat{\mathbf{u}}_{t_2}\| \\ &\leq 2\delta. \end{aligned} \tag{D.53}$$

The second inequality can similarly be proved using triangle inequality:

$$\|\mathbf{u}_{t_1}\| = \left\|(\mathbf{u}_{t_1} - \frac{1}{M}\mathbf{1}) + \frac{1}{M}\mathbf{1}\right\| \leq \left\|\mathbf{u}_{t_1} - \frac{1}{M}\mathbf{1}\right\| + \left\|\frac{1}{M}\mathbf{1}\right\| \leq \delta + 1. \tag{D.54}$$

The last inequality can be proved using Cauchy-Schwarz inequality:

$$\mathbf{1}^\top \mathbf{u}_{t_1} \leq |\mathbf{1}^\top \mathbf{u}_{t_1}| \leq \sqrt{M}||\mathbf{u}_{t_1}|| \leq \delta\sqrt{M} + 1. \tag{D.55}$$

□

Boundedness and smoothness of validation loss

In this subsubsection, we prove Lemma 6.2 about the boundedness and smoothness of validation loss. To make the subsubsectionself-contained, we re-state the definition of the task-specific parameter $\phi_i(\mathbf{x}$ and the true validation loss $\bar{\boldsymbol{\ell}}_i(\mathbf{x})$ as follows:

$$\phi_i(\mathbf{x}) = \mathbf{x} - \frac{\gamma}{m_i^{(s)}} \sum_{k=1}^{m_i^{(s)}} \nabla_{\mathbf{x}} \left[\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \mathbf{x}\right)\right] \tag{6.4}$$

$$\bar{\boldsymbol{\ell}}_i(\mathbf{x}) = \mathbb{E}_{(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}) \sim \mathcal{D}_i^{(q)}} \left[\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi(\mathbf{x})\right)\right]. \tag{6.9}$$

Lemma 6.2 and its proof are shown as follows:

Lemma 6.2

If the conditions in Assumptions 6.1, 6.2 and 6.3 hold, then $\bar{\boldsymbol{\ell}}_i(\mathbf{x})$ defined in Eq. (6.9) is $\widetilde{S}$-smooth, where: $\widetilde{S} = S(1+\gamma S)^2 + \gamma\rho L$.

Proof. Before starting the proof, we abuse the notation of gradient of the loss function at a point $\mathbf{x} = \mathbf{v}$ as follows:

$$\nabla_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{v}) = \nabla_{\mathbf{x}} \ell(\mathbf{s}, y; \mathbf{x}) \Big|_{\mathbf{x}=\mathbf{v}.} \tag{D.56}$$

Given the definition of the true validation loss in Eq. (6.9), its gradient w.r.t. $\mathbf{x}$ can be calculated using chain rule as follows:

$$\begin{aligned}\nabla_{\mathbf{x}} \bar{\boldsymbol{\ell}}_i(\mathbf{x}) &= \nabla_{\mathbf{x}} \mathbb{E}_{(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}) \sim \mathcal{D}_i^{(q)}} \left[\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right)\right] \\ &= \mathbb{E}_{(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}) \sim \mathcal{D}_i^{(q)}} \left[\nabla_{\mathbf{x}} \phi_i(\mathbf{x}) \times \nabla_{\mathbf{x}} \ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right)\right].\end{aligned} \tag{D.57}$$

Note that we abuse the notation and use $\mathbb{E}_{(\mathbf{s}_{ij}^{(s)}, y_{ij}^{(s)}) \sim \mathcal{S}_i^{(s)}}$ to indicate the average evaluated on all data points in set $\mathcal{S}_i$.

Since $\phi_i(\mathbf{x})$ defined in Eq. (6.4) does not depends on validation (or query) samples,

we can, therefore, rewrite the above gradient as:

$$\begin{aligned}\nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\mathbf{x}) &= \nabla_{\mathbf{x}}\phi_i(\mathbf{x}) \times \mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right)\right] \\ &= \left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \mathbf{x}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right)\right].\end{aligned} \tag{D.58}$$

In the following, we omit sample $(\mathbf{s}, y)$ from the expectation to simplify the notations. In particular, the above gradient can be re-written as:

$$\nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\mathbf{x}) = \left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \mathbf{x}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \phi_i(\mathbf{x})\right)\right]. \tag{D.59}$$

Thus, we can calculate the difference of the gradient evaluated on the same task $\mathcal{T}_i$ but with two different meta-parameters:

$$\begin{aligned}&\left\|\nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\bar{\mathbf{x}}) - \nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\widetilde{\mathbf{x}})\right\| \\ &= \Bigg\|\left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\bar{\mathbf{x}})\right)\right] \\ &\quad - \left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \widetilde{\mathbf{x}}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right]\Bigg\| \\ &= \Bigg\|\left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\bar{\mathbf{x}})\right)\right] \\ &\quad - \Bigg\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right] + \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right] \\ &\qquad -\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \widetilde{\mathbf{x}}\right)\right]\Bigg\} \times \mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right]\Bigg\| \\ &= \Bigg\|\left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\bar{\mathbf{x}})\right)\right. \\ &\quad \left.-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right] - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right) - \nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \widetilde{\mathbf{x}}\right)\right] \\ &\quad \times\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right]\Bigg\|.\end{aligned} \tag{D.60}$$

Applying the triangle inequality gives:

$$\begin{aligned}&\left\|\nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\bar{\mathbf{x}}) - \nabla_{\mathbf{x}}\bar{\boldsymbol{\ell}}_i(\widetilde{\mathbf{x}})\right\| \\ &\quad \leq \Bigg\|\left\{\mathbf{I} - \gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]\right\}\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\bar{\mathbf{x}})\right)\right. \\ &\qquad \left.-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right]\Bigg\| + \Bigg\|\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right. \\ &\qquad \left.-\nabla_{\mathbf{x}}^2\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \widetilde{\mathbf{x}}\right)\right]\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\widetilde{\mathbf{x}})\right)\right]\Bigg\|.\end{aligned} \tag{D.61}$$

Next, we upper-bound the two terms in the right-hand side of Ineq. (D.61). The

first term can be upper-bounded as:

$$\begin{aligned}\text{First term} &= \Big\|\Big\{\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\Big\}\\ &\quad\times\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\bar{\mathbf{x}})\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\widetilde{\mathbf{x}})\right)\right]\Big\|\\ &\leq \left\|\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\right\|\\ &\quad\times\left\|\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\bar{\mathbf{x}})\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\widetilde{\mathbf{x}})\right)\right]\right\| \qquad \text{(D.62)}\end{aligned}$$

Applying Jensen's inequality on the L2 norm of the expectation in the right-hand side of the above inequality to bring the expectation outside of the L2 norm, then employing the smoothness of ℓ in Assumption 6.2 to obtain the following:

$$\begin{aligned}\text{First term} &\leq \left\|\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\right\|\\ &\quad\times\mathbb{E}_{\mathcal{D}_i^{(q)}}\left\|\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\bar{\mathbf{x}})\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(q)},y_{ik}^{(q)};\phi_i(\widetilde{\mathbf{x}})\right)\right\|\\ &\leq \left\|\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\right\|\times S\left\|\phi_i(\bar{\mathbf{x}})-\phi_i(\widetilde{\mathbf{x}})\right\|. \qquad \text{(D.63)}\end{aligned}$$

Given the definition of $\phi(\mathbf{x})$ in Eq. (6.4), we can obtain the following:

$$\begin{aligned}\left\|\phi_i(\bar{\mathbf{x}})-\phi_i(\widetilde{\mathbf{x}})\right\| &= \left\|(\bar{\mathbf{x}}-\widetilde{\mathbf{x}})-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\bar{\mathbf{x}}\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\widetilde{\mathbf{x}}\right)\right]\right\|\\ &\leq \left\|\bar{\mathbf{x}}-\widetilde{\mathbf{x}}\right\|+\gamma\left\|\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\bar{\mathbf{x}}\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\widetilde{\mathbf{x}}\right)\right]\right\|\\ &\quad\text{(triangle inequality)}\\ &\leq \left\|\bar{\mathbf{x}}-\widetilde{\mathbf{x}}\right\|+\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left\|\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\bar{\mathbf{x}}\right)-\nabla_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ij}^{(s)};\widetilde{\mathbf{x}}\right)\right\|\\ &\quad\text{(Jensen's inequality)}\\ &\leq \left\|\bar{\mathbf{x}}-\widetilde{\mathbf{x}}\right\|+\gamma S\left\|\bar{\mathbf{x}}-\widetilde{\mathbf{x}}\right\| \ \text{(Assumption 6.2)}\end{aligned} \qquad \text{(D.64)}$$

Thus, one can upper-bound further Ineq. (D.63) as follows:

$$\text{First term}\leq S(1+\gamma S)\left\|\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\right\|\times\left\|\bar{\mathbf{x}}-\widetilde{\mathbf{x}}\right\|. \qquad \text{(D.65)}$$

If $\{\lambda_d\}_{d=1}^D$ are the eigenvalues of the Hessian matrix $\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]$, then due to Lemma D.11, the eigenvalues of $\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]$ are $\{1-\gamma\lambda_d\}_{d=1}^D$. In addition, since $\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]$ is symmetric and positive semi-definite, its norm equals to the largest eigenvalue (refer to Lemma D.13):

$$\left\|\mathbf{I}-\gamma\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla^2_{\mathbf{x}}\ell\left(\mathbf{s}_{ik}^{(s)},y_{ik}^{(s)};\bar{\mathbf{x}}\right)\right]\right\|=\max_d|1-\gamma\lambda_d|\leq 1+\gamma\max_d|\lambda_d|. \qquad \text{(D.66)}$$

According to Assumption 6.2, one can imply that the eigenvalues of the Hessian matrix $\mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]$ are smaller than S (Bubeck et al., 2015, section 3.2, page 266). This results in:

$$\left\|\mathbf{I} - \gamma \mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right)\right]\right\| \leq 1 + \gamma S. \tag{D.67}$$

Therefore, the first term on the right-hand side of (D.61) is upper-bounded by:

$$\text{First term} \leq S\left(1 + \gamma S\right)^2 \left\|\bar{\mathbf{x}} - \tilde{\mathbf{x}}\right\|. \tag{D.68}$$

The second term in the right-hand side of (D.61) can be upper-bounded as:

$$\begin{aligned}
\text{Second term} = & \Big\|\gamma \mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right) - \nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \tilde{\mathbf{x}}\right)\right] \\
& \times \mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}} \ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\tilde{\mathbf{x}})\right)\right]\Big\| \\
\leq & \Big\|\gamma \mathbb{E}_{\mathcal{S}_i^{(s)}}\left[\nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right) - \nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \tilde{\mathbf{x}}\right)\right]\Big\| \\
& \times \Big\|\mathbb{E}_{\mathcal{D}_i^{(q)}}\left[\nabla_{\mathbf{x}} \ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\tilde{\mathbf{x}})\right)\right]\Big\| \\
\leq & \gamma \mathbb{E}_{\mathcal{S}_i^{(s)}}\left\|\nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \bar{\mathbf{x}}\right) - \nabla_{\mathbf{x}}^2 \ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \tilde{\mathbf{x}}\right)\right\| \\
& \times \mathbb{E}_{\mathcal{D}_i^{(q)}}\left\|\nabla_{\mathbf{x}} \ell\left(\mathbf{s}_{ik}^{(q)}, y_{ik}^{(q)}; \phi_i(\tilde{\mathbf{x}})\right)\right\| \text{ (Jensen's inequality)} \\
\leq & \gamma \rho L \left\|\bar{\mathbf{x}} - \tilde{\mathbf{x}}\right\| \text{ (Eq. (D.49) and Assumption 6.3).}
\end{aligned} \tag{D.69}$$

Combining the results in (D.61), (D.68) and (D.69) gives:

$$\left\|\nabla_{\mathbf{x}} \bar{\boldsymbol{\ell}}_i\left(\bar{\mathbf{x}}\right) - \nabla_{\mathbf{x}} \bar{\boldsymbol{\ell}}_i\left(\tilde{\mathbf{x}}\right)\right\| \leq \left[S(1 + \gamma S)^2 + \gamma \rho L\right] \left\|\bar{\mathbf{x}} - \tilde{\mathbf{x}}\right\|. \tag{D.70}$$

This completes the proof. □

Lemma 6.3

If Assumption 6.4 holds, then the variance of $\nabla_{\mathbf{x}} \mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)$ is bounded by $\tilde{\sigma}^2 = \sigma^2\left(\delta + M^{-0.5}\right)^2$.

Proof. We use the following well-known inequality for variance as a part of the proof.

If $X_i, \forall i \in \{1, \ldots, n\}$ are random variables with finite variance: $\operatorname{Var}\left(X_i\right) < +\infty$, then:

$$\operatorname{Var}\left(\sum_{i=1}^{n} X_i\right) \leq n \sum_{i=1}^{n} \operatorname{Var}\left(X_i\right).$$

Note that $\boldsymbol{\ell}_i(\mathbf{x}_t)$ in (6.3) is the empirical expected values of loss ℓ evaluated on

task i-th. Hence, applying the above inequality for variance gives:

$$\begin{aligned}\text{Var}\left(\boldsymbol{\nabla}_{\mathbf{x}}\boldsymbol{\ell}_i(\mathbf{x}_t)\right) &= \text{Var}\left(\frac{1}{m_q}\sum_{j=1}^{m_q}\boldsymbol{\nabla}_{\mathbf{x}}\ell\left(\mathbf{s}_{ij}^{(q)}, y_{ij}^{(q)}; \mathbf{x} - \frac{\gamma}{m_s}\sum_{k=1}^{m_s}\boldsymbol{\nabla}_{\mathbf{x}}\left[\ell\left(\mathbf{s}_{ik}^{(s)}, y_{ik}^{(s)}; \mathbf{x}_t\right)\right]\right)\right) \\ &\leq \frac{1}{m_q}\sum_{j=1}^{m_q}\text{Var}\left(\boldsymbol{\nabla}_{\mathbf{x}}\ell\right) \leq \sigma^2.\end{aligned} \tag{D.71}$$

Hence, the variance of $\mathbf{u}_{ti}\boldsymbol{\nabla}_{\mathbf{x}}\bar{\boldsymbol{\ell}}(\mathbf{x}_t)$ is bounded by $\mathbf{u}_{ti}^2\sigma^2$. This leads to:

$$\begin{aligned}\text{Var}\left(\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{u}_t^\top\bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right) &= \text{Var}\left(\sum_{i=1}^{M}\mathbf{u}_{ti}\boldsymbol{\nabla}_{\mathbf{x}}\boldsymbol{\ell}_i(\mathbf{x}_t)\right) \\ &= \sum_{i=1}^{M}\mathbf{u}_{ti}^2\text{Var}\left(\boldsymbol{\nabla}_{\mathbf{x}}\boldsymbol{\ell}_i(\mathbf{x}_t)\right) \\ &\leq \sigma^2\sum_{i=1}^{M}\mathbf{u}_{ti}^2 = \sigma^2\|\mathbf{u}_t\|^2 \\ &\leq \sigma^2\left(\delta + M^{-0.5}\right)^2 \quad \text{(Corollary D.8)}.\end{aligned} \tag{D.72}$$

□

D.6.4 Convergence of TOW

Theorem 6.4

If Assumptions 6.1 - 6.4 hold, the learning rate $\alpha < {}^2/\widetilde{S}(\delta\sqrt{M}+1)$, and $\mathbf{z}$ is randomly sampled from $\{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}$ returned by Algorithm 5, then:

$$\mathbb{E}_{\mathbf{z}\sim\{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}}\left[\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\left\|\boldsymbol{\nabla}_{\mathbf{z}}\mathbf{u}_t^\top\bar{\boldsymbol{\ell}}_{1:M}(\mathbf{z})\right\|^2\right]\right] \leq \epsilon_0 + \frac{\kappa}{T_{\text{iter}}},$$

where:

$$\epsilon_0 = \frac{4\delta B\sqrt{M} + \alpha^2\tilde{\sigma}^2\widetilde{S}\left(\delta\sqrt{M}+1\right)}{\alpha\left[2 - \alpha\widetilde{S}\left(\delta\sqrt{M}+1\right)\right]} > 0 \tag{6.10}$$

$$\kappa = \frac{2\mathbf{u}_1^\top\bar{\boldsymbol{\ell}}_{1:M}(\mathbf{x}_1)}{\alpha\left[2 - \alpha\widetilde{S}\left(\delta\sqrt{M}+1\right)\right]}, \tag{6.11}$$

with T_{iter} as the number of gradient-update for the meta-parameter (or the number of mini-batches of tasks used), and $\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}$ as the expectation taken over all data sampled from t mini-batches $\{\mathcal{D}_i^{(q)}\}_{i=1}^t$, each $\mathcal{D}_i^{(q)}$ has M tasks.

Proof. According to Eq. 6.9, $\bar{\boldsymbol{\ell}}_i(\mathbf{x}) \in \mathbb{R}$ is the expected validation loss of task i-th using meta-parameter $\mathbf{x}$. In addition, the notation $\mathcal{D}_{1:M}^{(q)}$ indicates the data probability that generates query data pairs for M tasks in a mini-batch.

For convenience, we also denote $\bar{\boldsymbol{\ell}}(x) \in \mathbb{R}^M$ is the vector consisting of expected validation losses on M tasks in a mini-batch of tasks:

$$\bar{\boldsymbol{\ell}}(x) = \begin{bmatrix} \bar{\boldsymbol{\ell}}_1(x) & \bar{\boldsymbol{\ell}}_2(x) & \dots & \bar{\boldsymbol{\ell}}_M(x) \end{bmatrix}^\top. \tag{D.73}$$

From Lemma 6.2, the validation loss $\bar{\boldsymbol{\ell}}_i(\mathbf{x})$ is $\widetilde{S}$-smooth, or its gradient is $\widetilde{S}$-Lipschitz continuous. Hence, applying Taylor's theorem gives:

$$\bar{\boldsymbol{\ell}}_i(\mathbf{x}_{t+1}) \leq \bar{\boldsymbol{\ell}}_i(\mathbf{x}_t) + \boldsymbol{\nabla}_{\mathbf{x}}^\top \bar{\boldsymbol{\ell}}_i(\mathbf{x}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{\widetilde{S}}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2. \tag{D.74}$$

Note that $\mathbf{u}_{ti}$ is constrained to be non-negative as mentioned in Subsection 6.3.1 or in step 18 of Algorithm 5. Hence, one can multiply both sides by $\mathbf{u}_{ti} \geq 0, t \in \{1, \dots, T\}, i \in \{1, \dots, M\}$ and sum side-by-side to obtain:

$$\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_{t+1}) \leq \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_t) + \boldsymbol{\nabla}_{\mathbf{x}}^\top \left[\mathbf{u}_t \bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right](\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{\widetilde{S}}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \mathbf{1}_M^\top \mathbf{u}_t. \tag{D.75}$$

By conditioning on $\mathbf{x}_t$ and taking the expectation over all data sampled from $\{\mathcal{D}_i^{(q)}\}_{i=1}^M$, which is used to calculate $\mathbf{x}_{t+1}$, we can obtain the following:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_{t+1}) \,|\mathbf{x}_t\right] \leq \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\Big[& \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_t) + \boldsymbol{\nabla}_{\mathbf{x}}^\top \left[\mathbf{u}_t \bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right](\mathbf{x}_{t+1} - \mathbf{x}_t) \\ & + \frac{\widetilde{S}}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \mathbf{1}_M^\top \mathbf{u}_t \;\Big|\; \mathbf{x}_t\Big]. \end{aligned} \tag{D.76}$$

Note that for simplicity, we assume that the state-transition dynamics is SGD:

$$\mathbf{x}_{t+1} - \mathbf{x}_t = -\alpha \boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]. \tag{D.77}$$

Thus, one can simplify further to obtain:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_{t+1}) \,|\mathbf{x}_t\right] \leq \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\Big[& \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}(\mathbf{x}_t) - \alpha \boldsymbol{\nabla}_{\mathbf{x}}^\top \left[\mathbf{u}_t \bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right] \boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right] \\ & + \frac{\alpha^2 \widetilde{S}}{2}\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\|^2 \mathbf{1}_M^\top \mathbf{u}_t \;\Big|\; \mathbf{x}_t\Big]. \end{aligned} \tag{D.78}$$

Note that one can use Eq. 6.9 to imply that: $\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t \bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right] = \mathbb{E}_{\mathcal{S}_{1:M}^{(q)} \sim \mathcal{D}_{1:M}^{(q)}\, m_{1:M}^{(q)}}\left[\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\right]$, which also implies:

$$\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t \bar{\boldsymbol{\ell}}(\mathbf{x}_t)\right] = \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top \boldsymbol{\ell}(\mathbf{x}_t)\right]\right]. \tag{D.79}$$

Thus, the above inequality can be written as:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right] &\leq \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \alpha\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2 \\ &\quad + \frac{\alpha^2\widetilde{S}}{2}\left[\operatorname{Var}\left[\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top\boldsymbol{\ell}\left(\mathbf{x}_t\right)\right]\right] + \left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2\right]\mathbf{1}_M^\top\mathbf{u}_t \\ &\quad\quad \text{(since } \mathbb{E}[X^2] = \operatorname{Var}[X] + (\mathbb{E}[X])^2 \text{)} \\ &\leq \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \alpha\left(1 - \frac{\alpha\widetilde{S}}{2}\mathbf{1}_M^\top\mathbf{u}_t\right)\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2 \\ &\quad + \frac{\alpha^2\widetilde{S}}{2}\operatorname{Var}\left[\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top\boldsymbol{\ell}\left(\mathbf{x}_t\right)\right]\right]\mathbf{1}_M^\top\mathbf{u}_t.\end{aligned} \tag{D.80}$$

Since:

$$\begin{cases}\mathbf{1}_M^\top\mathbf{u}_t & \leq \delta\sqrt{M}+1 \quad \text{(Corollary D.8 in Appendix D.6.3)} \\ \operatorname{Var}\left[\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^\top\boldsymbol{\ell}\left(\mathbf{x}_t\right)\right]\right] & \leq \tilde{\sigma}^2 \quad \text{(Lemma 6.3)}\end{cases}$$

then:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right] &\leq \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \alpha\left[1 - \frac{\alpha\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right]\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2 \\ &\quad + \frac{\alpha^2\tilde{\sigma}^2\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right).\end{aligned} \tag{D.81}$$

Re-arranging the gradient norm of the weighted validation loss to the left-hand side gives:

$$\begin{aligned}\alpha\left[1 - \frac{\alpha\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right]\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2 &\leq \mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right] \\ &\quad + \frac{\alpha^2\tilde{\sigma}^2\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right).\end{aligned} \tag{D.82}$$

The right-hand-side, excluding the constant term at the end, can be written as:

$$\begin{aligned}\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right] &= \left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right) - \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_{t+1}^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\right] \\ &\quad + \left[\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_{t+1}^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\right. \\ &\quad \left. - \mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^\top \bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\right]\end{aligned} \tag{D.83}$$

The second part in the right-hand side of the above expression can be upper-

bounded as:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_{t+1}^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]-\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_t^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\\
&=\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\left(\mathbf{u}_{t+1}-\mathbf{u}_t\right)^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\\
&\leq\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\left\|\mathbf{u}_{t+1}-\mathbf{u}_t\right\|\sqrt{\sum_{i=1}^{M}\bar{\boldsymbol{\ell}}_i^2\left(\mathbf{x}_{t+1}\right)}\,\middle|\,\mathbf{x}_t\right]\quad\text{(Cauchy-Schwarz inequality)}\\
&\leq B\sqrt{M}\,\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\left\|\mathbf{u}_{t+1}-\mathbf{u}_t\right\||\mathbf{x}_t\right]\quad(\ell\text{ is }B\text{-bounded, and hence, }\bar{\boldsymbol{\ell}}_i\text{ is }B\text{-bounded})\\
&\leq 2\delta B\sqrt{M}\quad\text{(Corollary D.8)}.
\end{aligned}\tag{D.84}$$

Hence, one can upper-bound further the right-hand side of (D.82), resulting in:

$$\begin{aligned}
\alpha\left[1-\frac{\alpha\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right]\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2\leq\mathbf{u}_t^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)-\mathbb{E}_{\mathcal{D}_{1:M}^{(q)}}\left[\mathbf{u}_{t+1}^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)|\mathbf{x}_t\right]\\
+2\delta B\sqrt{M}+\frac{\alpha^2\widetilde{\sigma}^2\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right).
\end{aligned}\tag{D.85}$$

Note that the recursive gradient update is:

$$\begin{aligned}
\mathbf{x}_{t+1}&=\mathbf{x}_t-\alpha\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t^{\top}\boldsymbol{\ell}\left(\mathbf{x}_t\right)\right]\\
\mathbf{x}_t&=\mathbf{x}_{t-1}-\alpha\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_{t-1}^{\top}\boldsymbol{\ell}\left(\mathbf{x}_{t-1}\right)\right]\\
&\vdots\\
\mathbf{x}_2&=\mathbf{x}_1-\alpha\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_1^{\top}\boldsymbol{\ell}\left(\mathbf{x}_1\right)\right].
\end{aligned}$$

We take the expectation over all the mini-batches used at time step $t, t-1, \ldots, 1$ to remove the conditioning in (D.85). We denote this expectation as $\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}$, where the superscript t indicates the power, meaning that the expectation is carried out over t mini-batches that are used to calculate the state $\mathbf{x}_{t+1}$ from $\mathbf{x}_1$. This results in:

$$\begin{aligned}
&\alpha\left[1-\frac{\alpha\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right]\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2\right]\\
&\leq\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\mathbf{u}_t^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]-\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\mathbf{u}_{t+1}^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{t+1}\right)\middle|\mathbf{x}_1\right]+2\delta B\sqrt{M}+\frac{\alpha^2\widetilde{\sigma}^2\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right).
\end{aligned}\tag{D.86}$$

Summing (D.86) from $t=1$ to T_{iter} gives:

$$\begin{aligned}
&\alpha\left[1-\frac{\alpha\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right]\sum_{t=1}^{T_{\text{iter}}}\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}}\left[\left\|\boldsymbol{\nabla}_{\mathbf{x}}\left[\mathbf{u}_t\bar{\boldsymbol{\ell}}\left(\mathbf{x}_t\right)\right]\right\|^2\right]\\
&\leq\mathbf{u}_1^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_1\right)-\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,T_{\text{iter}}}}\left[\mathbf{u}_{T_{\text{iter}}+1}^{\top}\bar{\boldsymbol{\ell}}\left(\mathbf{x}_{T_{\text{iter}}+1}\right)\right]+T_{\text{iter}}\left[2\delta B\sqrt{M}+\frac{\alpha^2\widetilde{\sigma}^2\widetilde{S}}{2}\left(\delta\sqrt{M}+1\right)\right].
\end{aligned}\tag{D.87}$$

Note that $\mathbf{u}_{ti} \geq 0 \ \forall t \in \{1, \ldots, T_{\text{iter}}, i \in \{1, \ldots, M\}$, and $\bar{\boldsymbol{\ell}}_i \geq 0$ due to the non-negativity of the loss function ℓ assumed in (6.3). Hence, we can upper-bound further the right-hand side to:

$$\begin{aligned}
&\alpha \left[1 - \frac{\alpha \widetilde{S}}{2} \left(\delta \sqrt{M} + 1\right)\right] \sum_{t=1}^{T_{\text{iter}}} \mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}} \left[\left\| \boldsymbol{\nabla}_{\mathbf{x}} \left[\mathbf{u}_t \bar{\boldsymbol{\ell}} \left(\mathbf{x}_t\right)\right]\right\|^2\right] \\
&\leq \mathbf{u}_1^\top \bar{\boldsymbol{\ell}} \left(\mathbf{x}_1\right) + T_{\text{iter}} \left[2\delta B \sqrt{M} + \frac{\alpha^2 \widetilde{\sigma}^2 \widetilde{S}}{2} \left(\delta \sqrt{M} + 1\right)\right].
\end{aligned} \tag{D.88}$$

If the learning rate α is selected such that: $1 - {}^{\alpha \widetilde{S}}/_2 \left(\delta \sqrt{M} + 1\right) > 0$, then dividing both sides by $\alpha T_{\text{iter}} \left[1 - {}^{\alpha \widetilde{S}}/_2 \times \left(\delta \sqrt{M} + 1\right)\right]$ gives:

$$\begin{aligned}
&\frac{1}{T_{\text{iter}}} \sum_{t=1}^{T_{\text{iter}}} \mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}} \left[\left\| \boldsymbol{\nabla}_{\mathbf{x}} \left[\mathbf{u}_t \bar{\boldsymbol{\ell}} \left(\mathbf{x}_t\right)\right]\right\|^2\right] \\
&\leq \frac{2\mathbf{u}_1^\top \bar{\boldsymbol{\ell}} \left(\mathbf{x}_1\right) + T_{\text{iter}} \left[4\delta B \sqrt{M} + \alpha^2 \widetilde{\sigma}^2 \widetilde{S} \left(\delta \sqrt{M} + 1\right)\right]}{\alpha T_{\text{iter}} \left[2 - \alpha \widetilde{S} \left(\delta \sqrt{M} + 1\right)\right]}.
\end{aligned} \tag{D.89}$$

Next, we use a similar trick in the SVRG paper (Johnson and T. Zhang, 2013) to make the left-hand side term useful. The idea is to output some randomly chosen $\mathbf{x}_t$ rather than outputing $\mathbf{x}_{T_{\text{iter}}}$. For simplicity, let $\mathbf{z} = \mathbf{x}_t$ with a uniform probability for $t \in \{1, \ldots, T_{\text{iter}}\}$. The expectation of the gradient norm in this case can be expressed as:

$$\mathbb{E}_{\mathbf{z} \sim \{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}} \left[\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}} \left[\left\| \boldsymbol{\nabla}_{\mathbf{x}} \left[\mathbf{u}_t \bar{\boldsymbol{\ell}} \left(\mathbf{x}_t\right)\right]\right\|^2\right]\right] = \frac{1}{T_{\text{iter}}} \sum_{t=1}^{T_{\text{iter}}} \mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}} \left[\left\| \boldsymbol{\nabla}_{\mathbf{x}} \left[\mathbf{u}_t \bar{\boldsymbol{\ell}} \left(\mathbf{x}_t\right)\right]\right\|^2\right]. \tag{D.90}$$

This combining with (D.89) leads to:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z} \sim \{\mathbf{x}_t\}_{t=1}^{T_{\text{iter}}}} \left[\mathbb{E}_{\mathcal{D}_{1:M}^{(q)\,t}} \left[\left\| \boldsymbol{\nabla}_{\mathbf{z}} \left[\mathbf{u}_t \bar{\boldsymbol{\ell}} \left(\mathbf{z}\right)\right]\right\|^2\right]\right] \\
&\leq \frac{2\mathbf{u}_1^\top \bar{\boldsymbol{\ell}} \left(\mathbf{x}_1\right) + T_{\text{iter}} \left[4\delta B \sqrt{M} + \alpha^2 \widetilde{\sigma}^2 \widetilde{S} \left(\delta \sqrt{M} + 1\right)\right]}{\alpha T_{\text{iter}} \left[2 - \alpha \widetilde{S} \left(\delta \sqrt{M} + 1\right)\right]}.
\end{aligned} \tag{D.91}$$

□

D.6.5 Miscellaneous lemmas

Lemma D.11

If λ is an eigenvalue of matrix $\mathbf{A}$, then $\lambda - 1$ is an eigenvalue of matrix $\mathbf{A} - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Proof. According to the definition of eigenvalue, λ is an eigenvalue of $\mathbf{A}$ if:

$$\det\left(\mathbf{A}-\lambda\mathbf{I}\right)=0.$$

Rewriting the above equation gives:

$$\det\left(\left(\mathbf{A}-\mathbf{I}\right)-\left(\lambda-1\right)\mathbf{I}\right)=0. \tag{D.92}$$

Hence, $\lambda-1$ is an eigenvalue of $\mathbf{A}-\mathbf{I}$. □

Lemma D.12: Adpated from https://math.stackexchange.com/a/4303207/274798

If a function $f:\mathbb{R}^n\to\mathbb{R}^m$, where $n,m\in\mathbb{N}$, is differentiable and L-Lipschitz, then its gradient norm is bounded by L.

Proof. According to the definition of vector or matrix norm:

$$\begin{aligned}\|\boldsymbol{\nabla}f(x)\|&=\sup_v\frac{\|\boldsymbol{\nabla}f(x)\,v\|}{\|v\|},\forall x,v\in\mathbb{R}^n:\|v\|\neq 0\\&=\sup_v\lim_{\lambda\to 0}\frac{|\lambda|\|\boldsymbol{\nabla}f(x)\,v\|}{|\lambda|\|v\|},\ \lambda\in\mathbb{R},\lambda\neq 0\\&=\sup_v\lim_{\lambda\to 0}\frac{\|\boldsymbol{\nabla}f(x)\,(\lambda v)\|}{\|\lambda v\|}.\end{aligned} \tag{D.93}$$

Applying the triangle inequality gives:

$$\begin{aligned}\|\boldsymbol{\nabla}f(x)\|&\leq\sup_v\lim_{\lambda\to 0}\frac{\|f(x+\lambda v)-f(x)-\boldsymbol{\nabla}f(x)\,(\lambda v)\|}{\|\lambda v\|}+\frac{\|f(x+\lambda v)-f(x)\|}{\|\lambda v\|}\\&\leq\sup_v\lim_{\lambda\to 0}\frac{\|f(x+\lambda v)-f(x)-\boldsymbol{\nabla}f(x)\,(\lambda v)\|}{\|\lambda v\|}+\frac{L\|x+\lambda v-x\|}{\|\lambda v\|}\\&\qquad(f\text{ is }L\text{-Lipschitz})\\&\leq L.\end{aligned} \tag{D.94}$$

The first term equals to 0 since it corresponds to the definition of Fréchet derivative. □

Lemma D.13: Adapted from https://math.stackexchange.com/a/2193914/274798

If $A \in \mathbb{R}^{n \times n}$ is a positive definite symmetric matrix, then its largest eigenvalue is

$$\lambda_{\max} = \max \left\{ \frac{\|A\,x\|}{\|x\|} : x \neq 0 \right\}. \tag{D.95}$$

Proof. According to the spectral theorem, if A is a real and symmetric matrix, then there exists an orthonormal basis consisting of eigenvectors of A, denoted as $v_1, v_2, \ldots, v_n$. Without loss of generality, let's assume that the corresponding eigenvalues are: $\lambda_1, \lambda_2, \ldots, \lambda_n$.

For any non-zero vector x, we can represent it in this orthonormal basis as:

$$x = \sum_{i=1}^{n} c_i v_i, \;\; c_i \in \mathbb{R}. \tag{D.96}$$

Then, the function of interest can be calculated as follows:

$$\begin{aligned} \frac{\|A\,x\|}{\|x\|} &= \frac{\|\sum_{i=1}^n c_i \lambda_i v_i\|}{\sqrt{\sum_{i=1}^n c_i^2}} \\ &= \sqrt{\frac{\sum_{i=1}^n c_i^2 \lambda_i^2}{\sum_{i=1}^n c_i^2}} \quad (\{v_i\}_{i=1}^n \text{ is an orthonormal basis}) \\ &\leq \max_{i \in \{1,\ldots,n\}} \lambda_i. \end{aligned} \tag{D.97}$$

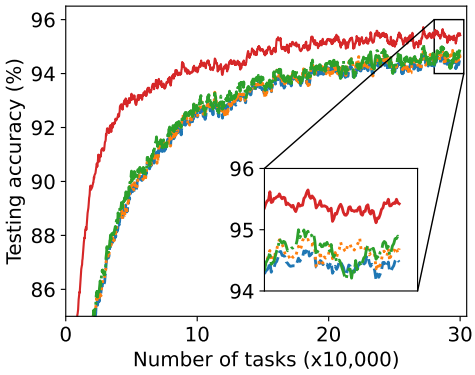
The equality occurs when the vector x equals to the corresponding eigenvector of the eigenvalue $\max_{i \in \{1,\ldots,n\}} \lambda_i$. □

D.7 Results with full matrix $\mathbf{V}_t$

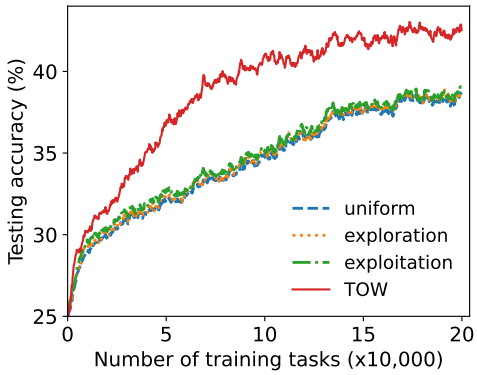
In this Appendix, we provide additional results of prediction accuracy on tasks formed from the evaluation sets of Omniglot and mini-ImageNet. These results are based on the naive implementation of iLQR where the auxiliary Hessian matrix $\mathbf{V}_t$ of the *cost-to-go* in (D.1) is exact without any approximation. We note that due to the quadratic complexity of running time, we cannot train the one on mini-ImageNet until convergence. The running time using the full matrix $\mathbf{V}_t$ is 160 GPU-hour for Omniglot and 184 GPU-hour for mini-ImageNet.

D.8 Visualisation of weight values

To further understand how the weight $\mathbf{u}_t$ varies along the training process, we conduct a study to monitor the weights $\mathbf{u}_t$ of a set of pre-defined Omniglot tasks. In the first setting, we fix $T = 5$ mini-batches of tasks, each consisting of $M = 5$ tasks



(a) MAML on Omniglot



(b) MAML on mini-ImageNet

Figure D.1: Additional results of prediction accuracy on 100 random validation tasks using MAML when the matrix $\mathbf{V}_t$ is exact without any approximation.

formed from the *same* alphabet. In the second case, the configuration is similar, but each mini-batch contains $M = 5$ tasks formed from 5 *different* alphabets. For each case, we also run with tasks drawn from training and testing sets. Our desire is to observe how the weight changes and whether there is a difference between training and testing tasks. Figure D.2 shows the variation of the weights of the tasks of interest with a common trend among all the tasks considered, in which the weights are large at the beginning and gradually reduce when training progresses. This is, indeed, expected since most tasks are unfamiliar to the model at the early state, and gradually becomes more familiar. In addition, we observe that the weights for testing tasks are slightly larger than the ones for training tasks.

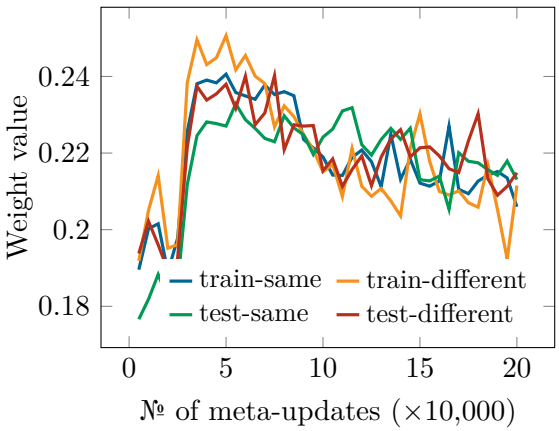


Figure D.2: Visualisation of the weight values where tasks are drawn from the same Omniglot alphabet (either training or testing set); the notation *same* means that all tasks in a mini-batch are formed from one alphabet, while *different* indicates the mini-batch consists of tasks formed from different alphabets.

We also conduct the same ablation study for mini-ImageNet tasks. Since in mini-ImageNet, we do not have any information regarding to the hierarchical structure

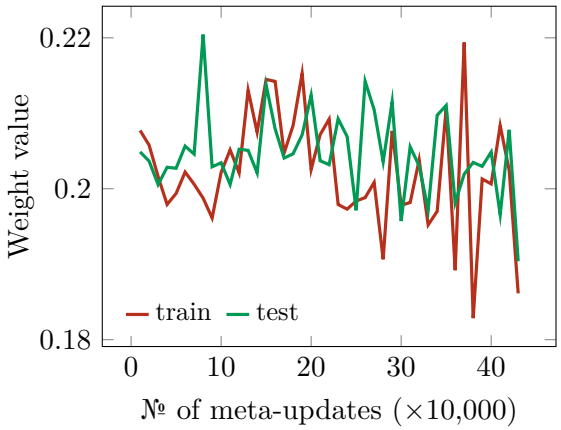


Figure D.3: Visualisation of the weight values associated with mini-ImageNet tasks.

of classes, we carry out the experiment on training and testing tasks only. One could, of course, utilise the word-net structure to categorise classes into "alphabet" as Omniglot, but for the sake of simplicity, we proceed without including such information. Figure D.3 shows the weight values of some tasks at each training checkpoint, which also has a similar but noisier trend as the ones in Omniglot.