META-LEARNING WITH COMPLEX TASKS

Weisen JIANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Meta-Learning aims at extracting knowledge from historical tasks to accelerate learning on new tasks. It has achieved promising performance in various applications and many researchers have developed algorithms to learn a meta-model that can be used as an initialization/regularization for task-specific finetuning algorithms. In this thesis, we focus on meta-learning with complex tasks, thus, task-specific models are diverse and *a simple meta-model* cannot represent all meta-knowledge.

First, we extend learning an efficient meta-regularization for *linear* models to *nonlinear* models by kernelized proximal regularization, allowing more powerful models like deep networks to deal with complex tasks. The inner problem is reformulated into a dual problem and a learnable proximal regularizer is introduced to the base learner. We propose a novel meta-learning algorithm to learn the proximal regularizer and establish its local/global convergence.

Second, we formulate the task-specific model parameters into *a subspace mixture* and propose a model-agnostic meta-learning algorithm to learn the subspace bases. Each subspace represents one type of meta-knowledge and *structural meta-knowledge* accelerates learning complex tasks more effectively than a simple meta-model. The proposed algorithm can be used for both linear and nonlinear models. Empirical results show that the proposed algorithm can discover the underlying subspace of task model parameters.

Third, we propose an effective and parameter-efficient meta-learning algorithm for language models. The proposed algorithm learns *a pool of multiple meta-prompts* to extract knowledge from meta-training tasks and then constructs *instance-dependent prompts* as weighted combinations of all the prompts in the pool by attention. Prompts in the pool are meta-parameters while the language model is frozen, thus very parameter-efficient. A novel soft verbalizer is proposed to reduce human effort in annotating words for labels.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Humans can easily learn new knowledge from a handful of examples and quickly adapt to unseen tasks. They leverage prior knowledge and experience from historical tasks to construct task-required knowledge when facing new tasks. Though deep networks have achieved great success in various applications [53, 129], they are data-hungry. Hence, a large number of training samples are required for a new task. This challenge remains a crucial bottleneck in making progress in machine learning algorithms and many research efforts attempt to use historical knowledge to improve data-efficiency in learning tasks.

Multitask learning (MTL) [176] learns common knowledge from several tasks by minimizing the weighted sum of losses on training data of each task. For the seen tasks, MTL has shown good performance on testing samples [177, 178, 20, 63, 89]. However, the learned MTL model is not guaranteed to generalize better and achieve faster learning on *unseen tasks*. Furthermore, MTL also suffers from the scalability issue when we have many tasks, leading to a heavy burden on computation and memory. For example, in 5-way 1-shot classification on the *mini-ImageNet* data [152], there are $\binom{64}{5} \approx 7 \times 10^6$ tasks in total.

Transfer learning [108, 165] finetunes a pretrained model on training data of a task to obtain a task-specific model. For example, for a *CIFAR-10* [75] classification task, we first pretrain a network (e.g., *ResNet-18* [53]) on *ImageNet* dataset [129], then use the pretrained network as initialization for gradient descent algorithms for minimizing the training loss on *CIFAR-10*. Transfer learning has been successfully used in image classification [115, 50], natural language processing [9, 34], and reinforcement learning [40, 182]. However, pretraining and finetuning are independent, thus, *the pretrained model may not be suitable for finetuning on tasks with limited examples*.

Recently, *meta-learning* (or *learning to learn*) [7, 145] provides a general framework to extract meta-knowledge from historical tasks to accelerate learning unseen tasks for

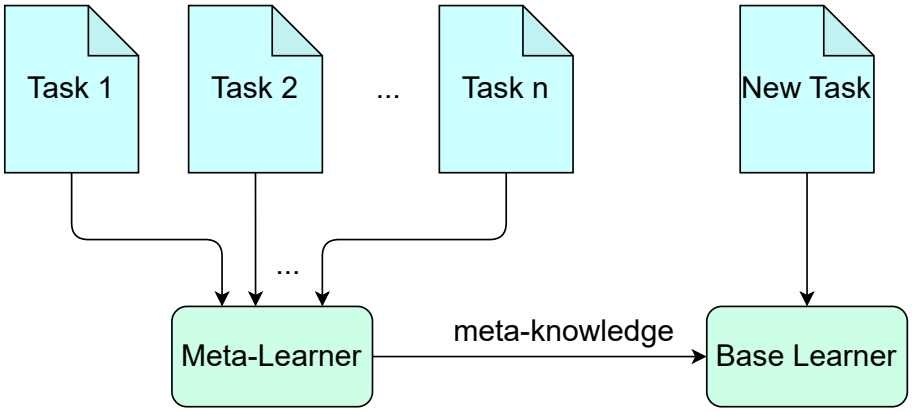


Figure 1.1: Illustration of meta-learning.

reducing the labor-intensive and time-consuming process of data. Figure 1.1 illustrates the procedure of meta-learning. Meta-learning algorithms usually operate in two levels. At each iteration, we sample a task; in the inner level, the base learner takes meta-knowledge (e.g., initialization, regularization, feature extractor) and training set to learn a task-specific model; in the outer level, the meta-learner takes the task-specific model and validation set to update the meta-knowledge. As the loss on validation set is a proxy measure of generalization ability, the meta-knowledge is explicitly tuned to learning new tasks with limited training data. At testing, the base learner uses the extracted meta-knowledge to achieve fast learning on unseen tasks. Meta-learning has been receiving increasing attention due to its diverse successful applications in few-shot learning [156, 35, 152, 140, 69], hyperparameter optimization [39], neural architecture search [91, 183], and reinforcement learning [122].

1.2 Category of Meta-Learning Algorithms

Popular meta-learning algorithms can be categorized into three groups: optimization-based, metric-based, and memory-based.

For *optimization-based* methods, the base learner performs gradient descent to minimize a task-specific loss, where meta-parameters in the optimization algorithm are meta-knowledge learned by the meta-learner. The meta-parameters can be initialization, regularization, learning rate, preconditioning matrix, sample weights. Meta-initialization and meta-regularization are two representative methods. MAML [35] is a pioneering meta-initialization method: the base learner takes a meta-initialization and performs several gradient updates on the training set to obtain a task-specific model, while the

meta-learner updates the meta-initialization by performing a gradient update on the validation set using the obtained task-specific model. As MAML is very general, many variants are proposed to improve its effectiveness (e.g., Meta-SGD[86], T-Nets [80], Meta-Curvature [109], WarpGrad [37]) and efficiency (e.g., FO-MAML [35], Reptile [104]). The bilevel structure is complex and challenging to understand MAML from a theoretical view. Recently, many efforts have been devoted to study its convergence [36, 32, 154, 155, 67, 66] and generalization [98, 180, 24, 25, 111, 112, 127, 33, 120, 15, 147]. In meta-regularization, the base learner minimizes a regularized loss of training data to obtain a task-specific model, while the meta-learner updates the learnable regularizer by minimizing the validation loss using the obtained model. Typical algorithms include iMAML [121], Meta-MinibatchProx [179], and regularization for linear models [23, 24, 25]. As real-world tasks are usually complex, **non-structured meta-learning** methods a single meta-initialization or meta-regularization may be insufficient for capturing meta-knowledge of all tasks. To deal with this issue, **structured meta-learning** methods [64, 74, 150, 180] propose to formulate meta-knowledge into structures like clusters, such that each task can choose a suitable cluster center as initialization.

Metric-based methods aim at meta-learning a good feature extractor to map inputs to an embedding space, where the base learner trains a simple but effective classifier with few samples. Convolutional neural networks (e.g., VGG [138], ResNet [53]) and Vision Transformers [30] are widely used in feature extraction. For classifier, we can use a nearest neighbor classifier (e.g., ProtoNet [140]), linear models (e.g., R2D2 [8]), and kernal classifier (e.g., MetaOptNet [79])).

Memory-based methods incorporate a memory structure to store meta-knowledge for accelerating learning future tasks. For example, an external memory (table or key-value pairs) is used in [132, 123, 102, 3]; hypernetworks (also called meta-networks) are external networks that contain knowledge of how to generate task parameters [101, 130, 139, 31, 103].

Learning meta-parameters for optimization-based algorithms is very general and has wide applications, thus, we focus on optimization-based meta-learning in this thesis.

1.3 Applications of Meta-Learning

1.3.1 Computer Vision (CV) Tasks

Meta-Learning has a wide variety of applications in computer vision regimes, including few-shot classification [152, 124], object detection [113], landmark prediction [47], few-shot object segmentation [135], few-shot image generation [172], and density estimation [126].

Few-shot classification (FSC) is the most common application of meta-learning and is used in the thesis to evaluate the performance of meta-learning algorithms. The task is to classify classes with limited samples per class, which is very challenging and many meta-learning methods are proposed to improve its performance. For example, metric-based meta-learning algorithms like ProtoNet are specialized to FSC.

The benchmark of FSC is more complex than that of traditional machine learning benchmark, which aims to evaluate the model from seen instances to unseen instances. However, in meta-learning, the FSC benchmark evaluates the generalization ability of the learned model from seen classes to unseen classes. Popular FSC benchmarks are *miniImageNet* [152, 124], *CIFAR-FS* [8], *Omniglot* [76], *Meta-Dataset* [167, 148].

1.3.2 Natural Language Processing (NLP) Tasks

Large language models have achieved great success recently and many pre-trained models are released for downstream tasks such as language understanding [29, 166, 141], machine translation [21, 49], and text classification [12, 81]. As finetuning the large models causes a heavy burden on computations, many parameter-efficient (PE) learning methods are proposed, e.g., prompt tuning and in-context learning. Prompt learning [12, 137, 27] freezes the pre-trained model and formulates the downstream task as a cloze-style masked language model (MLM) problem [26]. In-Context Learning (ICL) [99, 18] uses a pre-trained MLM to learn a new task by formatting training examples as a demonstration.

Similar to computer vision, collecting or designing many samples for training is infeasible. To deal with this issue, meta-learning is used to improve the data-efficiency of PE learning. For example, MetaPrompting [56] proposes to learn a good meta-initialization for the prompt vector, while MetaICL [99] finetunes the language model to make it

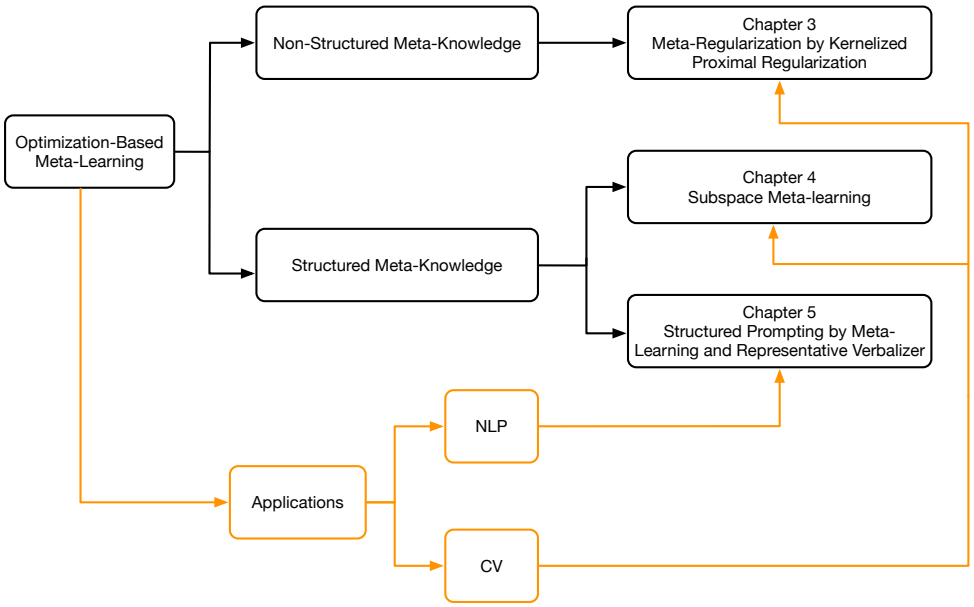


Figure 1.2: Outline.

more suitable for in-context learning.

1.4 Contributions and Organization

Figure 1.2 shows an overview of the organization of the thesis. The main contributions of each chapter are summarized as follows.

1. Chapter 3 learns a meta-regularization for nonlinear models. The content of this Chapter is mainly based on Jiang et al. [68].
    - We introduce nonlinearity to **meta-regularization** by kernelized proximal regularization.
    - We propose a novel meta-learning algorithm called MetaProx for learning the meta-regularization. For regression tasks, the base learner has an efficient closed-form solution.
    - We establish local and global convergence of the proposed algorithm.
    - Experiments on a variety of benchmark regression and classification datasets demonstrate MetaProx is better than the state-of-the-arts.
2. Chapter 4 formulates meta-knowledge into **a subspace mixture**. The content of this Chapter is mainly based on Jiang et al. [70].

- We formulate task model parameters into multiple subspaces (each subspace represents one type of meta-knowledge) and propose a model-agnostic algorithm MUSML to learn the subspace bases.
- We provide a theoretical analysis of the population risk, empirical risk, and generalization gap.
- Extensive experiments on regression and classification datasets demonstrate the effectiveness of learning a subspace mixture.

3. Chapter 5 studies applications of meta-learning in language models and learns **a prompt pool** for prompt tuning. The content of this Chapter is mainly based on Jiang et al. [71].

   - We use a prompt pool to extract meta-knowledge and construct instance-dependent prompts by attention.
   - We design a novel soft verbalizer called representative verbalizer, which builds label embeddings by averaging feature embeddings.
   - Combing meta-learning a prompt pool with a novel soft verbalizer, we propose a novel parameter-efficient meta-learning algorithm MetaPrompter.
   - Experimental results demonstrate the usefulness and parameter-efficiency of MetaPrompter. Moreover, the superiority of the proposed representative verbalizer over existing verbalizers is verified by empirical evaluations.

CHAPTER 2

Background

2.1 Formulation of Meta-Learning

In meta-learning, a collection $\mathcal{T}$ of tasks sampled from a task distribution $p(\tau)$ are used to learn a meta-parameter $\boldsymbol{\theta}$ and base learner's parameters $\{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{T}|}\}$. Each task τ contains a support (also called training) set $\mathcal{S}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_s\}$ and a query (also called validation) set $\mathcal{Q}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_q\}$, where $\mathbf{x} \in \mathbb{R}^d$ are the features and y the labels. For classification task, $\mathcal{Y}_\tau$ is the label set of τ. Let $f(\cdot; \mathbf{w})$ be a model parameterized by $\mathbf{w}$ and $\mathcal{L}(\mathcal{D}; \mathbf{w}) \equiv \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f(\mathbf{x}; \mathbf{w}), y)$ be the supervised loss on data set $\mathcal{D}$, where $\ell(\cdot, \cdot)$ is a loss function (e.g., cross-entropy loss for classification, mean squared loss for regression). In each meta-training iteration, a mini-batch $\mathcal{B}$ of tasks are randomly sampled from $\mathcal{T}$. The base learner takes a task τ from $\mathcal{B}$ and the meta-parameter $\boldsymbol{\theta}$ to build the model $f(\cdot; \mathbf{w}_\tau)$. After all tasks in the min-batch are processed by the base learner, the meta-learner updates $\boldsymbol{\theta}$ by minimizing the loss $\sum_{\tau\in\mathcal{B}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau)$ w.r.t. $\boldsymbol{\theta}$, and the iteration repeats. During meta-testing, given an unseen task $\tau' \sim p(\tau)$, a model $f(\cdot; \mathbf{w}_{\tau'})$ is similarly learned from $\mathcal{S}_{\tau'}$ and $\boldsymbol{\theta}$. Finally, its performance is evaluated on $\mathcal{Q}_{\tau'}$.

2.2 Representative Meta-Learning Algorithms

2.2.1 MAML

A pioneer work for learning an initialization is Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [35] . An illustration is shown in Figure 2.1.

Meta-Training. MAML operates in two optimization levels (let $\mathcal{B}_t$ be a mini-batch of tasks at iteration t):

- *Inner Level*: For each task $\tau \in \mathcal{B}_t$, the base learner takes its support set $\mathcal{S}_\tau$ and the meta-initialization $\boldsymbol{\theta}_{t-1}$ to build a task-specific model $\mathbf{w}_\tau^{(J)}$ by performing J

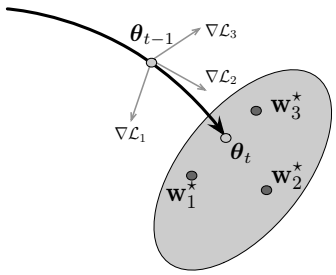


Figure 2.1: MAML.

gradient descent steps with initialization $\mathbf{w}_\tau^{(0)} = \boldsymbol{\theta}_{t-1}$ and step size $\alpha > 0$:

$$\mathbf{w}_\tau^{(j)} = \mathbf{w}_\tau^{(j-1)} - \alpha \nabla_{\mathbf{w}_\tau^{(j-1)}} \mathcal{L}\left(\mathcal{S}_\tau; \mathbf{w}_\tau^{(j-1)}\right), \ j = 1, \ldots, J. \tag{2.1}$$

Note that $\mathbf{w}_\tau^{(J)}$ is a function of $\boldsymbol{\theta}_t$.

- *Outer Level* : For each task $\tau \in \mathcal{B}_t$, the meta-learner takes its query set $\mathcal{Q}_\tau$ and the task-specific model $\mathbf{w}_\tau^{(J)}$ to compute the gradient of $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$ w.r.t. $\boldsymbol{\theta}_{t-1}$, which is called meta-gradient. The meta-initialization is updated as:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}\left(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}\right), \tag{2.2}$$

where $\eta_t > 0$ is step size.

By the chain rule, the meta-gradient $\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}) = \nabla_{\boldsymbol{\theta}_{t-1}} \mathbf{w}_\tau^{(J)} \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$ (particularly, $\nabla_{\boldsymbol{\theta}_{t-1}} \mathbf{w}_\tau^{(J)}$) requires back-propagating through the entire inner optimization path, incurring huge computations, especially for large models and a large J.

Meta-Testing. Given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, the base learner takes $\mathcal{S}_{\tau'}$ and $\boldsymbol{\theta}_T$ to build a task-specific model $\mathbf{w}_{\tau'}^{(J)}$, which is then used to predict the query sample in $\mathcal{Q}_{\tau'}$ and evaluate performance.

Several works improve MAML using preconditioning gradients in the inner loop, e.g., MetaSGD [86], Meta-Curvature [109], T-Nets [80], and WarpGrad [37]. Unlike MAML that updates all network parameters in the base learner, recent work [118, 105, 136] reveals that updating only part of network can improve both efficiency and effectiveness.

2.2.2 iMAML

Meta-regularization [23, 24, 121, 179, 25, 68] is another optimization-based meta-learning method, which assumes that task models are close to a prior model. The base learner

obtains task model by solving a regularized empirical risk minimization, while the prior model $\boldsymbol{\theta}$ in the regularization is learned by the meta-learner. Specifically, at iteration t, the base learner takes $\mathcal{S}_\tau$ and $\boldsymbol{\theta}_{t-1}$ to build task model $\hat{\mathbf{w}}_\tau$ by solving the regularized minimization:

$$\hat{\mathbf{w}}_\tau = \arg\min_{\mathbf{w}_\tau} \mathcal{L}(\mathcal{S}_\tau; \mathbf{w}_\tau) + \frac{\lambda}{2}\|\mathbf{w}_\tau - \boldsymbol{\theta}_{t-1}\|^2, \tag{2.3}$$

where $\lambda > 0$ is hyperparameter. Similar to meta-initialization, $\hat{\mathbf{w}}_\tau$ is a function of $\boldsymbol{\theta}_{t-1}$. As in MAML, meta-regularization methods (iMAML [121] and Denevi et al. [23]) update the regularizer by the meta-learner by minimizing the loss

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau). \tag{2.4}$$

By the chain rule, it follows that $\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau) = \nabla_{\boldsymbol{\theta}_{t-1}} \hat{\mathbf{w}}_\tau \nabla_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau)$. The second term $\nabla_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau)$ can be computed directly by auto-differentiation, but the first term $\nabla_{\boldsymbol{\theta}_{t-1}} \hat{\mathbf{w}}_\tau$ is more difficult as it is an implicit derivative. As $\hat{\mathbf{w}}_\tau$ is a minimizer to problem (2.3), it follows from the first-order optimal condition that

$$\nabla_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \lambda(\hat{\mathbf{w}}_\tau - \boldsymbol{\theta}_{t-1}) = \mathbf{0}. \tag{2.5}$$

By the implicit function theorem [128], it follows that $\nabla^2_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) \nabla_{\boldsymbol{\theta}_{t-1}} \hat{\mathbf{w}}_\tau + \lambda(\nabla_{\boldsymbol{\theta}_{t-1}} \hat{\mathbf{w}}_\tau - \mathbf{I}) = \mathbf{0}$, thus, the implicit derivative is

$$\nabla_{\boldsymbol{\theta}_{t-1}} \hat{\mathbf{w}}_\tau = \left(\frac{1}{\lambda} \nabla^2_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \mathbf{I}\right)^{-1}. \tag{2.6}$$

Compared with meta-initialization (e.g., MAML), iMAML has two advantages: (i) Computing the meta-gradients does not require to back-propagate through the inner optimization path. Hence, we can use many gradient updates in the base learner to obtain an accurate solution, which can address the short-horizon bias issue in one-step MAML. (ii) The meta-gradient does not depends on the optimization path, thus, the chosen optimizer in the base learner is flexible, e.g., AdaDelta [173], Adam [73], AdamW [97].

2.2.3 Prototypical Networks (ProtoNet)

ProtoNet [140] is a representative metric-based method. It employs a neural network to map inputs to an embedding space, then represents each class's prototype by the mean embeddings of the corresponding samples in the support set. For each sample in the

query set, its label prediction is based on the similarity between feature embedding and label embeddings. Let $\mathcal{S}_\tau^{(y)}$ be the subset of samples in $\mathcal{S}_\tau$ with label y, and $\mathsf{NN}(\cdot;\boldsymbol{\phi})$ be a feature extractor parameterized by $\boldsymbol{\phi}$. Specifically, the base learner builds class prototype as follows

$$\mathbf{p}_y = \frac{1}{|\mathcal{S}_\tau^{(y)}|} \sum_{(\mathbf{x}_i, y) \in \mathcal{S}_\tau^{(y)}} \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1}), \tag{2.7}$$

and the label prediction for $(\mathbf{x}, \cdot) \in \mathcal{Q}_\tau$ is $(y \in \mathcal{Y}_\tau)$

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\phi}_{t-1}) = \frac{\exp(-\kappa \operatorname{dist}(\mathbf{p}_y, \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})))}{\sum_{y' \in \mathcal{Y}_\tau} \exp(-\kappa \operatorname{dist}(\mathbf{p}_{y'}, \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})))}, \tag{2.8}$$

where κ is a temperature, and $\operatorname{dist}(\mathbf{z}_1, \mathbf{z}_2) = \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2$. The meta-learner updates meta-parameter as follows:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} + \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \frac{1}{|\mathcal{Q}_\tau|} \sum_{(\mathbf{x}_i, y_i) \in Q_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \log \mathbb{P}(y_i|\mathbf{x}_i; \boldsymbol{\phi}_{t-1}), \tag{2.9}$$

The framework of ProtoNet is simple and general, and the distance metric is flexible. Other possible *metrics* are cosine similarity [17, 44, 93], earth mover's distance [174], a metric learned by deep networks [143], a metric based on graph convolution blocks [42].

2.2.4 MetaOptNet

When the embedding space is high-dimensional, a trainable linear classifier is more expressive than the nearest neighbor learner ProtoNet [140]. MetaOptNet [79] is a representative algorithm. Let $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}) \in \mathbb{R}^e$ denote the feature embedding of $\mathbf{x}$, $\mathbf{Z}_\tau^{tr} \in \mathbb{R}^{|\mathcal{S}_\tau| \times e}$ is the embedding matrix of $\mathcal{S}_\tau$ (each row vector corresponds a feature embedding), and $\mathbf{Y}_\tau^{tr} \in \mathbb{R}^{|\mathcal{S}_\tau| \times C}$ is the label matrix (assume $|\mathcal{Y}_\tau| = C$).

The base learner takes $\mathcal{S}_\tau$ and $\boldsymbol{\phi}_{t-1}$ to construct task model $\mathbf{W}_\tau^\star$ by solving the following ridge regression problem:

$$\mathbf{W}_\tau^\star = \underset{\mathbf{W}_\tau \in \mathbb{R}^{e \times C}}{\arg\min} \frac{1}{2}\|\mathbf{Z}_\tau^{tr}\mathbf{W}_\tau - \mathbf{Y}_\tau^{tr}\|^2 + \frac{\lambda}{2}\|\mathbf{W}_\tau\|^2, \tag{2.10}$$

where $\lambda > 0$ is a hyperparameter. The above problem has a closed-form solution $\mathbf{W}_\tau^\star = \left(\mathbf{Z}_\tau^{tr\top}\mathbf{Z}_\tau^{tr} + \lambda\mathbf{I}\right)^{-1}\mathbf{Z}_\tau^{tr\top}\mathbf{Y}_\tau^\top$, which implicitly depends on $\boldsymbol{\phi}_{t-1}$ via $\mathbf{Z}_\tau^{tr}$. In the inner loop, different from MAML [35], R2D2 keeps the feature extractor frozen and only

learns the linear classifier. The meta-learner takes $\mathbf{W}_\tau^\star$ to makes prediction on query samples $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{Q}_\tau$ as

$$\hat{\mathbf{y}}_i = a_{t-1}\,\mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})^\top \mathbf{W}_\tau^\star + c_{t-1}, \tag{2.11}$$

where a_{t-1} and c_{t-1} are meta-parameters for scaling. Meta-parameters are updated as

$$(\boldsymbol{\phi}_t, a_t, c_t) = (\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1}) - \eta \nabla_{(\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1})} \frac{1}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{Q}_\tau} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i). \tag{2.12}$$

MetaOptNet extends the ridge regression in R2D2 to a general convex classifier, e.g., support vector machine (SVM). The base learner solves the dual problem:

$$\max_{\boldsymbol{\alpha}_{\tau,1}, \ldots, \boldsymbol{\alpha}_{\tau,C}} -\frac{1}{2} \sum_c \|\boldsymbol{\alpha}_{\tau,c}^\top \mathbf{Z}_\tau^{tr}\|^2 + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \alpha_{\tau, y_i, i} \tag{2.13}$$

$$\text{s.t.} \quad \alpha_{\tau, y_i, i} \leq \gamma, \alpha_{\tau, c, i} \leq 0 \text{ for } c \neq y_i, \mathbf{1}^\top \boldsymbol{\alpha}_{\tau, \cdot, i} = 0, \forall i. \tag{2.14}$$

Unlike ridge regression, the above quadratic program (QP) has no closed-form solution. As the optimization variable is of size $n_s \times C$, which is independent of the embedding dimension, the dual variable can be obtain with low cost by an iterative solver, e.g., projected gradient methods [13, 87]. With the dual solution $\boldsymbol{\alpha}_\tau^\star \in \mathbb{R}^{n_s \times C}$, the prediction of a query example $(\mathbf{x}, y)$ is $\hat{\mathbf{y}} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})^\top \mathbf{Z}_\tau^{tr\top} \boldsymbol{\alpha}_\tau^\star$. The meta-learner updates the meta-parameters as:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, y_i) \in Q_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \ell(\hat{\mathbf{y}}_i, y_i). \tag{2.15}$$

Note that $\boldsymbol{\alpha}_\tau^\star$ is a solution to the dual problem, implicitly depends on $\boldsymbol{\phi}_t$. Similar to iMAML [121], we can use implicit function theorem to compute the gradient of $\nabla_{\boldsymbol{\phi}_{t-1}} \boldsymbol{\alpha}_\tau^\star$. See the CVXPYLayers package [1] for an implementation to solve the dual problem and back-propagate gradients though the convex learner.

2.3 Prompt Learning for Language Models

2.3.1 Prompt Tuning

Recently, it is common to use a pre-trained MLM $\mathcal{M}(\cdot; \boldsymbol{\phi})$, with parameter $\boldsymbol{\phi}$, for various downstream tasks such as language understanding [29, 166, 141], machine translation [21, 49], and text classification [12, 81, 95]. Given a raw sentence represented as a

sequence of n tokens $(x_1, \ldots, x_n)$, the MLM takes $\mathbf{x} = (\texttt{[CLS]}, x_1, \ldots, x_n, \texttt{[SEP]})$ as input (where `[CLS]` is the start token and `[SEP]` is the separator), and encodes it to a sequence of hidden representations $(\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \ldots, \mathbf{h}_n, \mathbf{h}_{\texttt{[SEP]}})$. In standard fine-tuning [58, 26], an extra classifier (e.g., a fully connected layer with softmax normalization) is added on top of $\mathbf{h}_{\texttt{[CLS]}}$ to predict the label distribution. This classifier, together with $\boldsymbol{\phi}$, are tuned to maximize the probability of correct labels. As language models are large (e.g., 175 billion parameters in GPT-3 [12]), fine-tuning all parameters can cause a heavy burden on computation and memory.

On the other hand, prompt learning [12, 137, 27] freezes the pre-trained model and formulates the downstream task as a cloze-style MLM problem. For example, in topic classification, "`Topic is [MASK]`" can be used as the prompt, where `[MASK]` is a special token for prediction. The *discrete* tokens "`Topic is`" are also called anchor tokens. An input text $\mathbf{x}$ is wrapped with the prompt and mapped to an input embedding sequence $(\mathcal{E}(\mathbf{x}), \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]}))$, where $\mathcal{E}(\cdot)$ denotes input embedding. Designing a suitable prompt requires domain expertise and a good understanding of the downstream tasks [12, 131]. Thus, manually-designed prompts are likely to be sub-optimal.

Unlike discrete prompts, prompt tuning [81, 94] uses a *continuous* prompt $\boldsymbol{\theta} \in \mathbb{R}^{L_p \times d_i}$ (of length L_p) to directly wrap the input embedding sequence as $(\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{[MASK]}))$. This can be further combined with anchor tokens to form a *template* [94, 133, 27]:

$$\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}) = (\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]})).$$

The MLM then outputs the hidden embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}) \in \mathbb{R}^{d_o}$ of `[MASK]`, and infers the token to be filled at the `[MASK]` position.

A *verbalizer* [81, 27, 60] bridges the prediction at the `[MASK]` position and labels in prompt learning. Specifically, it is a *hand-crafted* mapping from each label y to a set of label-relevant tokens $\mathcal{V}_y$. For example, for $y = \texttt{SPORTS}$, we can have $\mathcal{V}_y = \{\texttt{sports}, \texttt{football}, \texttt{basketball}\}$. Prompt tuning then optimizes[1] $(\boldsymbol{\phi}, \boldsymbol{\theta})$ by maximizing the label probability:

$$\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{|\mathcal{V}_y|} \sum_{\texttt{w} \in \mathcal{V}_y} \mathbb{P}_{\mathcal{M}}(\texttt{[MASK]} = \texttt{w} | \mathbb{T}(\mathbf{x}; \boldsymbol{\theta})), \tag{2.16}$$

where $\mathbb{P}_{\mathcal{M}}(\texttt{[MASK]}|\mathbb{T}(\mathbf{x}; \boldsymbol{\theta}))$ is the probability distribution over vocabulary as predicted by the MLM at the `[MASK]` position.

[1] $\boldsymbol{\phi}$ can be fixed for parameter-efficiency in prompt learning.

The verbalizer is crucial to the performance of prompt learning [81, 27]. However, selecting label-relevant tokens requires intensive human labor. Recent works [51, 175, 22] propose *soft* verbalizers, which map each label to a *continuous* embedding and predict label distribution based on the similarity between feature embedding and label embeddings. WARP [51] and DART [175] obtain this label embedding by supervised learning, while ProtoVerb [22] uses contrastive learning [16, 146]. However, learning the embedding $\mathbf{v}_y \in \mathbb{R}^{d_o}$ for each label y can be challenging in the few-shot learning setting [41, 6, 52, 14, 56], as the number of samples per class is typically much smaller than d_o (e.g., $d_o = 768$ for BERT [26]).

2.3.2 MetaPrompting

As prompt tuning is sensitive to prompt initialization in few-shot tasks [81], meta-learning can be used to search for a good initialization. MetaPrompting [56] uses MAML to learn a meta-initialization for the task-specific prompts. At iteration t, the base learner takes a task τ and meta-parameter $(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$, and builds a task-specific model $(\boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J})$ by performing J gradient updates on the support set with step size $\alpha > 0$ and initialization $(\boldsymbol{\phi}_{t,0}, \boldsymbol{\theta}_{t,0}) \equiv (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$:

$$(\boldsymbol{\phi}_{t,j}, \boldsymbol{\theta}_{t,j}) = (\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1}) + \alpha \nabla_{(\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1})} \sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1}).$$

The meta-learner then updates the meta-initialization by maximizing the log-likelihood objective on the query set with step size $\eta > 0$:

$$(\boldsymbol{\phi}_{t}, \boldsymbol{\theta}_{t}) = (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}) + \eta \nabla_{(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})} \sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J}).$$

Though MetaPrompting achieves state-of-the-art performance in the few-shot classification experiments [56], it suffers from three problems. (i) When the tasks are complex, it is challenging to obtain good prompts for all tasks and samples from a single meta-initialization. (ii) MetaPrompting uses a hand-crafted verbalizer. However, selecting good label tokens for the hand-crafted verbalizer is labor-intensive and not scalable for a large label set. (iii) MetaPrompting requires expensive tuning the whole MLM.

CHAPTER 3

Meta-Regularization by Kernelized Proximal Regularization

3.1 Introduction

Humans can easily learn new tasks from a handful of examples using prior knowledge and experience. In contrast, deep networks are data-hungry, and a large number of training samples are required to mitigate overfitting. To reduce the labor-intensive and time-consuming process of data labeling, *meta-learning* (or *learning to learn*) [7, 145] aims to exact meta-knowledge from seen tasks to accelerate learning on unseen tasks. Recently, meta-learning has been receiving increasing attention due to its diverse successful applications in few-shot learning [156, 35, 152, 140], hyperparameter optimization [39], neural architecture search [91, 183], and reinforcement learning [122].

Many meta-learning algorithms operate on two levels. A base learner learns task-specific models in the inner loop, and a meta-learner learns the meta-parameter in the outer loop. A popular class of algorithms is based on meta-initialization [35, 104, 32, 149], such as the well-known MAML [35]. It learns a model initialization such that a good model for an unseen task can be learned from limited samples by a few gradient updates. However, computing the meta-gradient requires back-propagating through the entire inner optimization path, which is infeasible for large models and/or many gradient steps. During testing, it is common for MAML's base learner to perform many gradient steps to seek a more accurate solution [35]. However, for regression using a linear base learner and square loss, we will show that though the meta-learner can converge to the optimal meta-initialization, the base learner may overfit the training data at meta-testing.

Another class of meta-learning algorithms is based on meta-regularization [121, 179, 23, 24, 25], in which the base learner learns the task-specific model by minimizing the loss with a proximal regularizer (a biased regularizer from the meta-parameter). Denevi et al. [23] uses a linear model with efficient closed-form solution for the base learner.

However, extending to nonlinear base learners requires computing the meta-gradient using matrix inversion, which can be infeasible for deep networks [121].

To introduce nonlinearity to the base learner, a recent approach is to make use of the kernel trick. For example, R2D2 [8] and MetaOptNet [79] use deep kernels [160] in meta-learning for few-shot classification. Specifically, the deep network is learned in the meta-learner, while a base kernel is used in the base learner. Though they achieve state-of-the-art performance, their base learners use a Tikhonov regularizer rather than a learnable proximal regularizer as in meta-regularization methods.

As learning a meta-regularization has been shown to be effective in linear models for regression [23] and classification [24], in this chapter, we propose a kernel-based algorithm to meta-learn a proximal regularizer for a nonlinear base learner. By kernel extension, the learnable function in the proximal regularizer is a function in the reproducing kernel Hilbert space (RKHS) induced by the base kernel. The proposed algorithm is guaranteed to converge to a critical point of the meta-loss and its global convergence is also established. Experiments on various benchmark regression and classification datasets demonstrate the superiority of the proposed algorithm over the state-of-the-arts.

3.2 Meta-Initialization versus Meta-Regularization

In this section, we consider a simple regression setting with linear model and square loss. Each task τ is a linear regressor with parameter $\mathbf{w}_\tau^\star \in \mathbb{R}^d$. We assume that each input $\mathbf{x}$ is sampled from $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{x}}^2 \mathbf{I})$ and the output y is obtained as $\mathbf{x}^\top \mathbf{w}_\tau^\star + \xi$, where $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the random noise. We compare two representative meta-learning algorithms: (i) MAML [35], which is based on meta-initialization and performs one gradient descent step in the inner loop of the bilevel optimization problem; and (ii) learning around a common mean (denoted CommonMean) [23], which is based on meta-regularization. It learns the model parameters for task τ by minimizing the loss with a proximal regularizer around the meta-parameter $\boldsymbol{\theta}$:

$$\begin{aligned} \mathbf{w}_\tau^{(\text{prox})}(\boldsymbol{\theta}) &= \arg\min_{\mathbf{w}} \sum\nolimits_{(\mathbf{x}_i, y_i) \in S_\tau} \frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \frac{\lambda}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2 \\ &= (\lambda \mathbf{I} + \mathbf{X}_\tau^\top \mathbf{X}_\tau)^{-1}(\lambda \boldsymbol{\theta} + \mathbf{X}_\tau^\top \mathbf{y}_\tau), \end{aligned} \tag{3.1}$$

where $\mathbf{X}_\tau = [\mathbf{x}_1^\top; \ldots; \mathbf{x}_{n_s}^\top]$ is the sample matrix from S_τ (each column of $\mathbf{X}_\tau$ is an input vector), and $\mathbf{y}_\tau = [y_1; \ldots; y_{n_s}]$ is the corresponding label vector. Note that $\mathbf{w}_\tau^{(\text{prox})}$ is a function of $\boldsymbol{\theta}$. Algorithms 1 and 2 show MAML and CommonMean, respectively, for this problem.

Recently, Balcan et al. [4] study the convex online meta-learning setting and show that both approaches achieve the same average task regret. Here, we consider the offline setting. First, the following Proposition shows that both MAML and CommonMean converge to the same meta-parameter.

Proposition 3.2.1. *Let $\eta_t = 1/t$. Assume that $\gamma < 1/\sigma_{\mathbf{x}}^2$. Both $\boldsymbol{\psi}_t$ in MAML (with one inner gradient step) and $\boldsymbol{\theta}_t$ in CommonMean converge to $\bar{\mathbf{w}} = \mathbb{E}_\tau \mathbf{w}_\tau^\star$.*

Algorithm 1 MAML [35].

Require: step size γ and η_t, batch size b;
for $t = 1, 2, 3, \ldots$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $p(\tau)$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{w}_\tau^{(\text{gd})}(\boldsymbol{\psi}_t) = \boldsymbol{\psi}_t - \gamma \mathbf{X}_\tau^\top (\mathbf{X}_\tau \boldsymbol{\psi}_t - \mathbf{y}_\tau)$;
    $\mathbf{g}_\tau = \frac{1}{2} \sum_{(\mathbf{x},y) \in Q_\tau} \nabla_{\boldsymbol{\psi}_t} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{gd})}(\boldsymbol{\psi}_t) - y)^2$;
  end for
  meta-learner: $\boldsymbol{\psi}_{t+1} = \boldsymbol{\psi}_t - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for

Algorithm 2 CommonMean [23].

Require: hyperparameter λ, step size η_t, batch size b;
for $t = 1, 2, 3, \ldots$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $p(\tau)$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{w}_\tau^{(\text{prox})} = \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{X}_\tau \mathbf{w} - \mathbf{y}_\tau\|^2 + \frac{\lambda}{2} \|\mathbf{w} - \boldsymbol{\theta}_t\|^2$;
    $\mathbf{g}_\tau = \frac{1}{2} \sum_{(\mathbf{x},y) \in Q_\tau} \nabla_{\boldsymbol{\theta}_t} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{prox})} - y)^2$;
  end for
  meta-learner: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for

The following Proposition shows that $\bar{\mathbf{w}}$ in Proposition 3.2.1 is also the best $\boldsymbol{\psi}$ (for MAML) or $\boldsymbol{\theta}$ (for CommonMean) with the lowest population risk for this meta-learning problem.

Proposition 3.2.2. *Assume that $\gamma < 1/\sigma_{\mathbf{x}}^2$. We have $\bar{\mathbf{w}} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y) \in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(prox)} - y)^2 = \arg\min_{\boldsymbol{\psi}} \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y) \in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(gd)} - y)^2$.*

During meta-testing, we sample a task $\tau' \sim p(\tau)$ with parameter $\mathbf{w}^\star_{\tau'}$. Let $\mathbf{X}_{\tau'}$ be the sample matrix from $S_{\tau'}$, and $\mathbf{y}_{\tau'}$ be the corresponding label vector. We assume that $\mathbf{X}_{\tau'}$ is full rank and $n_s < d$. To simplify notations, we drop the subscript τ' in the following. Let the singular value decomposition of $\mathbf{X}$ be $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ (where $\mathbf{\Sigma} = \mathsf{diag}([\nu_1, \ldots, \nu_{n_s}])$), and $\mathbf{V}^\perp$ be $\mathbf{V}$'s orthogonal complement.

As only forward passes are needed, it is common for the base learner in MAML to perform multiple gradient steps [35]. With the convex loss and linear model here, the base learner can obtain a globally optimal solution $\mathbf{w}^{(\mathrm{gd}\infty)}$ directly (which is equivalent to taking infinite gradient steps). As is common in few-shot learning, the number of support samples is much smaller than feature dimensionality. Hence, $\mathbf{w}^{(\mathrm{gd}\infty)}$ is not unique but depends on the learned initialization. Let $\mathbf{w}^{(\mathrm{gd}\infty)}$ be written as $\mathbf{w}^{(\mathrm{gd}\infty)} = \mathbf{V}\mathbf{a}^{(\mathrm{gd}\infty)} + \mathbf{V}^\perp\mathbf{b}^{(\mathrm{gd}\infty)}$. For gradient descent, its update direction $\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top(\mathbf{X}\mathbf{w} - \mathbf{y})$ is always in the span of $\mathbf{V}$ and so $\mathbf{b}^{(\mathrm{gd}\infty)}$ **remains unchanged**.

Let $\mathbf{w}^\star$ and $\boldsymbol{\theta}$ be written as $\mathbf{w}^\star = \mathbf{V}\mathbf{a}^\star + \mathbf{V}^\perp\mathbf{b}^\star$ and $\boldsymbol{\theta} = \mathbf{V}\mathbf{a}_0 + \mathbf{V}^\perp\mathbf{b}_0$. Moreover, let $\tilde{\mathbf{a}} = \mathbf{a}_0 - \mathbf{a}^\star$ and $\tilde{\mathbf{b}} = \mathbf{b}_0 - \mathbf{b}^\star$.

Proposition 3.2.3 ([48]). *Assume that $\gamma < \min_{1\le j\le n_s} 1/\nu_j^2$. We have $\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(gd\infty)} - \mathbf{w}^\star\|^2 = \|\mathbf{b}^{(gd\infty)} - \mathbf{b}^\star\|^2 + \sum_{j=1}^{n_s}\left(\frac{\sigma_\xi}{\nu_j}\right)^2$, where the expectation is over the label noise vector $\boldsymbol{\xi}$.*

For CommonMean, using the Woodbury matrix identity, $\mathbf{w}^{(\mathrm{prox})} = \boldsymbol{\theta} + \mathbf{X}^\top(\lambda\mathbf{I} + \mathbf{X}\mathbf{X}^\top)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \mathbf{V}(\mathbf{a}_0 + \mathbf{\Sigma}\mathbf{U}^\top(\lambda\mathbf{I} + \mathbf{X}\mathbf{X}^\top)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})) + \mathbf{V}^\perp\mathbf{b}_0 \equiv \mathbf{V}\mathbf{a}^{(\mathrm{prox})} + \mathbf{V}^\perp\mathbf{b}^{(\mathrm{prox})}$, where $\mathbf{b}^{(\mathrm{prox})} = \mathbf{b}_0$. Assume that $\mathbf{w}^{(\mathrm{gd}\infty)}$ is initialized with $\boldsymbol{\theta}$. Since $\mathbf{b}^{(\mathrm{gd}\infty)}$ remains unchanged, $\mathbf{w}^{(\mathrm{prox})}$ and $\mathbf{w}^{(\mathrm{gd}\infty)}$ **only differ in the components lying in the column space of $\mathbf{V}$**.

Proposition 3.2.4. $\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(prox)} - \mathbf{w}^\star\|^2 = \|\tilde{\mathbf{b}}\|^2 + \sum_{j=1}^{n_s}\left(\frac{\lambda\tilde{\mathbf{a}}_j}{\lambda+\nu_j^2}\right)^2 + \sum_{j=1}^{n_s}\left(\frac{\sigma_\xi}{(\lambda/\nu_j)+\nu_j}\right)^2$, *where the expectation is over the label noise vector $\boldsymbol{\xi}$.*

As can be seen, when the labels are noise-free ($\sigma_\xi^2 = 0$), $\mathbf{w}^{(\mathrm{gd}\infty)}$ performs better than $\mathbf{w}^{(\mathrm{prox})}$. However, when the labels are noisy, as $n_s < d$, gradient descent always converges to zero training error and overfits the noisy labels. On the other hand, the estimation error of $\mathbf{w}^{(\mathrm{prox})}$ equals to that of $\mathbf{w}^{(\mathrm{gd}\infty)}$ when $\lambda = 0$. For $\lambda > 0$, it trades off between fitting the noisy labels (the last term in Proposition 3.2.4) and introducing an estimation bias of $\mathbf{a}^\star$ (the second term in Proposition 3.2.4).

3.3 The Proposed MetaProx

It is straightforward to use the dual formulation for the CommonMean algorithm. When the square loss is used as $\ell(\cdot,\cdot)$, it is easy to see that the dual variable has the closed-form solution

$$\boldsymbol{\alpha}_\tau = (\mathbf{I} + \lambda^{-1}\mathbf{X}_\tau\mathbf{X}_\tau^\top)^{-1}(\mathbf{y}_\tau - \mathbf{X}_\tau\boldsymbol{\theta}). \tag{3.2}$$

Compared with the primal formulation, we only need to invert a $n_s \times n_s$ matrix (instead of the $d \times d$ matrix $\lambda\mathbf{I} + \mathbf{X}_\tau^\top\mathbf{X}_\tau$). In meta-learning, usually $n_s \ll d$ (e.g., $n_s = 5$). From the dual solution $\boldsymbol{\alpha}_\tau$, the primal solution can be recovered as $\mathbf{w}_\tau = \boldsymbol{\theta} + \lambda^{-1}\mathbf{X}_\tau^\top\boldsymbol{\alpha}_\tau$. Given a query example $(\mathbf{x}, y) \in Q_\tau$, the model predicts $\hat{y} = \mathbf{x}^\top\mathbf{w}_\tau = \mathbf{x}^\top\boldsymbol{\theta} + \lambda^{-1}\mathbf{x}^\top\mathbf{X}_\tau^\top\boldsymbol{\alpha}_\tau$. The loss gradient is $\nabla_{\boldsymbol{\theta}}\ell(\hat{y}, y) = \nabla_1\ell(\hat{y}, y)\nabla_{\boldsymbol{\theta}}\hat{y}$, where $\nabla_1\ell(\hat{y}, y)$ denotes the gradient w.r.t. the first argument, and $\nabla_{\boldsymbol{\theta}}\hat{y} = \mathbf{x} + \lambda^{-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\alpha}_\tau)^\top\mathbf{X}_\tau\mathbf{x}$, $\nabla_{\boldsymbol{\theta}}\boldsymbol{\alpha}_\tau = -(\mathbf{I} + \lambda^{-1}\mathbf{X}_\tau\mathbf{X}_\tau^\top)^{-1}\mathbf{X}_\tau$. The complexity of computing $\nabla_{\boldsymbol{\theta}}\ell(\hat{y}, y)$ is thus very low ($\mathcal{O}(n_s^3 + n_s^2 d)$).

The dual formulation also allows introduction of nonlinearity with the kernel trick. Based on deep kernels [160], recent state-of-the-arts (R2D2 [8], MetaOptNet [79], and DKT [110]) propose to use a base kernel in the base learner and update the deep network in the meta-learner. However, their regularizers are not learnable. In this work, we propose learning a proximal regularizer for the base learner (Algorithm 3). Specifically, let $\mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})$ be a feature extractor parameterized by $\boldsymbol{\phi}$. An input $\mathbf{x}$ is mapped to $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})$ in an embedding space $\mathcal{E}$. $\mathbf{Z}_\tau \equiv [\mathbf{z}_1^\top; \ldots; \mathbf{z}_{n_s}^\top]$, where $\mathbf{z}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for $\mathbf{x}_i \in S_\tau$. $\mathcal{K}$ is a base kernel on $\mathcal{E} \times \mathcal{E}$, and $\mathcal{H}$ is the corresponding RKHS. The primal problem in the inner loop is:

$$\hat{f}_\tau = \operatorname*{arg\,min}_{f_\tau \in \mathcal{H}} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \frac{\lambda}{2}\|f_\tau - f_{\boldsymbol{\theta}_{t-1}}\|_{\mathcal{H}}^2. \tag{3.3}$$

By the representer theorem [134], the solution of (3.3) is $f_\tau = f_{\boldsymbol{\theta}} + \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \alpha_{\tau,i}\mathcal{K}_{\mathbf{z}_i}$, where $\mathcal{K}_{\mathbf{z}_i} = \mathcal{K}(\mathbf{z}_i, \cdot) \in \mathcal{K}$, and $\boldsymbol{\alpha}_\tau = [\alpha_{\tau,1}; \ldots; \alpha_{\tau,n_s}]$ is obtained from the convex program

$$\min_{\boldsymbol{\alpha}_\tau} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \boldsymbol{\alpha}_\tau^\top\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\boldsymbol{\alpha}_\tau, \tag{3.4}$$

where $\mathbf{Z}_\tau = [\mathbf{z}_1^\top; \ldots; \mathbf{z}_{n_s}^\top]$, and $\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$ is the kernel matrix. Note that the hyperparameter λ in (3.3) is absorbed into $\mathbf{z}$ as the network is learnable. With the square loss, the dual solution of (3.4) is $\boldsymbol{\alpha}_\tau = (\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1}(\mathbf{y}_\tau - f_{\boldsymbol{\theta}}(\mathbf{Z}_\tau))$, where $f_{\boldsymbol{\theta}}(\mathbf{Z}_\tau) = [f_{\boldsymbol{\theta}}(\mathbf{z}_1); \ldots; f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s})]$. For general loss functions, the dual problem has no

closed-form solution, but this has only n_s variables (which is usually small) and can be solved efficiently.

After the base learner has obtained the dual solution $\boldsymbol{\alpha}_\tau$, the meta-learner updates $f_{\boldsymbol{\theta}}$ and network parameter $\boldsymbol{\phi}$ by one gradient descent step on the validation loss $\sum_{(\mathbf{x},y)\in Q_\tau} \ell(\hat{y}, y)$, where $\hat{y} \equiv f_\tau(\mathbf{z}) = f_{\boldsymbol{\theta}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}_\tau$. By chain rule, $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \ell(\hat{y}, y) = \nabla_1 \ell(\hat{y}, y) \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \hat{y}$. The first component $\nabla_1 \ell(\hat{y}, y)$ can be computed directly and the second component is

$$\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \hat{y} = \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z}) + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}))^\top \boldsymbol{\alpha}_\tau + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau)^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}). \tag{3.5}$$

Both $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z})$ and $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})$ can be obtained by direct differentiation. By the chain rule, $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau = \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathbf{p}$, where $\mathbf{p} = \left[f_{\boldsymbol{\theta}}(\mathbf{z}_1); \ldots; f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s}); \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_1); \ldots; \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{n_s})\right] \in \mathbb{R}^{n_s + n_s^2}$ is the input to the dual problem. $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathbf{p}$ can be directly computed. When the square loss is used, $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau = -(\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1} \left[\mathbf{I} \;\middle|\; \mathbf{I} \otimes \boldsymbol{\alpha}_\tau^\top\right]$. For a general loss, $\boldsymbol{\alpha}_\tau$ is obtained by solving the convex program. Hence, $\boldsymbol{\alpha}_\tau$ depends implicitly on $\mathbf{p}$ and $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau$ can be obtained by implicit differentiation. Denote the dual objective in (3.4) by $g(\mathbf{p}, \boldsymbol{\alpha})$. By the implicit function theorem [128], $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau = -\left(\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)\right)^{-1} \frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)$, where $\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = \sum_{(\mathbf{x}_i, y_i)\in S_\tau} \nabla_1^2 \ell(f_\tau(\mathbf{z}_i), y_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i)^\top + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$, $\frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = \left[\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\mathbf{D} \;\middle|\; (\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\mathbf{D}) \otimes \boldsymbol{\alpha}_\tau^\top + \mathbf{v}^\top \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\alpha}_\tau^\top\right]$, $\mathbf{D} = \operatorname{diag}([\nabla_1^2 \ell(f_\tau(\mathbf{z}_1), y_1); \ldots; \nabla_1^2 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})])$, $\mathbf{v} = [\nabla_1 \ell(f_\tau(\mathbf{z}_1), y_1); \ldots; \nabla_1 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})]$, where $\otimes$ is the Kronecker product. The whole procedure, called MetaProx, is shown in Algorithm 3. Let $n_{\boldsymbol{\phi}}$ and $n_{\boldsymbol{\theta}}$ be the numbers of parameters in $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. Computing $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \ell(\hat{y}, y)$ takes $\mathcal{O}(n_s^3 + n_s^2(n_{\boldsymbol{\theta}} + n_{\boldsymbol{\phi}}))$ time, which is linear in the number of meta-parameters. This is lower than the other meta-learning algorithms (e.g., MAML [35] with single step takes $\mathcal{O}(n_\phi^2)$ time, iMAML [121]: $\mathcal{O}(n_\phi^3)$, CommonMean [23]: $\mathcal{O}(d^3)$).

The proposed MetaProx has several advantages:

1. After kernel extension, $f_{\boldsymbol{\theta}}$ is a function in $\mathcal{H}$. For nonlinear kernels (e.g., RBF kernel, cosine kernel), $f_{\boldsymbol{\theta}}$ is nonlinear, thus, MetaProx learns a meta-regularization for a *nonlinear* base learner.

2. $f_{\boldsymbol{\theta}}$ in the base learner is *learnable*. By setting $f_{\boldsymbol{\theta}} = \mathbf{0}$, MetaProx recovers the state-of-the-art MetaOptNet [79].

3. For square loss, $\boldsymbol{\alpha}_\tau = (\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1}(\mathbf{y}_\tau - f_{\boldsymbol{\theta}_{t-1}}(\mathbf{Z}_\tau))$ has an efficient closed-form solution. For general losses, the dual problem is convex and can be solved

Algorithm 3 MetaProx.

Require: stepsize η_t, batch size b, feature extractor $\mathsf{NN}(\cdot;\boldsymbol{\phi})$, base kernel $\mathcal{K}$;

for $t = 1, 2, \cdots, T$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{z}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for each $(\mathbf{x}_i, y_i) \in S_\tau$;
    $f_\tau(\mathbf{z}; \boldsymbol{\alpha}) \equiv f_{\boldsymbol{\theta}_{t-1}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}$ denote the task model w.r.t. dual variables;
    $\boldsymbol{\alpha}_\tau = \arg\min_{\boldsymbol{\alpha}} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i; \boldsymbol{\alpha}), y_i) + \boldsymbol{\alpha}^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\boldsymbol{\alpha}$;
  end for
  meta-learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{g}_\tau = \sum_{(\mathbf{x},y) \in Q_\tau} \nabla_{(\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1})} \ell(\hat{y}, y)$, where $\hat{y} = f_\tau(\mathbf{z}; \boldsymbol{\alpha}_\tau)$ and $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})$;
  end for
  $(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) = (\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1}) - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for
return $(\boldsymbol{\theta}_T, \boldsymbol{\phi}_T)$.

efficiently, as the size of $\boldsymbol{\alpha}$ is very small (only n_s). Though MetaProx still requires matrix inversion in computing meta-gradients, the size is only $n_s \times n_s$, much smaller than $n_{\boldsymbol{\phi}} \times n_{\boldsymbol{\phi}}$ in iMAML [2].

3.4 Theoretical Analysis

Let $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{(\mathbf{x},y) \in Q_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y)$ be the empirical loss of generalization measure $\mathbf{E}_{\tau \sim p(\tau)} \sum_{(\mathbf{x},y) \in Q_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y)$, where $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})$. With the linear kernel and square loss, the dual solution (3.2) is affine in the meta-parameter, and so is the primal solution $\mathbf{w}_\tau = \boldsymbol{\theta} + \lambda^{-1} \mathbf{X}_\tau^\top \boldsymbol{\alpha}_\tau$. Thus, the meta-loss $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is convex and convergence follows from convex optimization [11, 23]. After introducing nonlinearity, the meta-loss is no longer convex. The following introduces Lipschitz-smoothness assumptions, which have been commonly used in stochastic non-convex optimization [43, 125] and meta-learning in non-convex settings [32, 179].

Assumption 3.4.1 (Smoothness)**.** (i) The deep network $\mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})$ is Lipschitz-smooth, i.e., $\|\nabla_{\boldsymbol{\phi}} \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}) - \nabla_{\boldsymbol{\phi}} \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}')\| \leq \eta_1 \|\boldsymbol{\phi} - \boldsymbol{\phi}'\|$ with a Lipschitz constant $\eta_1 > 0$; (ii) the kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}')$ is Lipschitz-smooth w.r.t. $(\mathbf{z}, \mathbf{z}')$; (iii) $f_{\boldsymbol{\theta}}(\mathbf{z})$ is Lipschitz-smooth w.r.t. $(\boldsymbol{\theta}, \mathbf{z})$; (iv) $\nabla_1^2 \ell(\hat{y}, y)$ is Lipschitz w.r.t. $\hat{y}$, i.e., $|\nabla_1^2 \ell(\hat{y}, y) - \nabla_1^2 \ell(\hat{y}', y)| \leq \eta_2 |\hat{y} - \hat{y}'|$ with a Lipschitz constant η_2; (v) $\mathbb{E}_{\tau \sim \mathcal{T}} \|\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \sum_{(\mathbf{x},y) \in Q_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y) - \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})\|^2 = \sigma_{\mathbf{g}}^2$, where $\tau \sim \mathcal{T}$ denotes uniformly sample a task from $\mathcal{T}$.

Theorem 3.4.1. *Let the step size be* $\eta_t = \min(1/\sqrt{T}, 1/2\eta_{meta})$. *Algorithm 3 satisfies*

$$\min_{1\le t\le T} \mathbb{E}\|\nabla_{(\boldsymbol{\theta}_t,\boldsymbol{\phi}_t)}\mathcal{L}_{meta}(\boldsymbol{\theta}_t,\boldsymbol{\phi}_t)\|^2 = \mathcal{O}\left(\frac{\sigma_{\mathbf{g}}^2}{\sqrt{T}}\right),$$

where the expectation is taken over the random training samples.

This rate is the same as MAML [32, 67] and Meta-MinibatchProx [179]. For MAML with $J > 1$ gradient steps, [67] assumes that the step size in the inner loop is of the order $1/J$. This slows down inner loop learning when J is large. On the other hand, MetaProx does not have this restriction, as its meta-gradient depends only on the last iterate rather than all iterates along the trajectory.

Next, we study the global convergence of MetaProx. Prior work [36, 179] focus on the case where $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta},\boldsymbol{\phi})$ is strongly convex in $(\boldsymbol{\theta},\boldsymbol{\phi})$. This can be restrictive in deep learning. We instead only require $\ell(\hat{y}, y)$ to be strongly convex in $\hat{y}$. This assumption is easily met by commonly-used loss functions such as the square loss and logistic loss with a compact domain. A recent work [154] studies the global convergence of MAML in over-parameterized neural networks. Over-parameterization is closely related to the assumption of uniform conditioning [61, 78, 90].

Assumption 3.4.2 (Uniform conditioning [90])**.** A multivariable function $\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi})$ is μ-uniformly conditioning if its tangent kernel [61] satisfies

$$\min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \lambda_{\min}(\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi})\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}^{\top}\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi})) \ge \mu > 0,$$

where $\lambda_{\min}(\cdot)$ is the smallest eigenvalue of the matrix argument.

Assume that the loss $\ell(\cdot,\cdot)$ is ρ-strongly convex w.r.t. the first argument and Assumption 3.4.1 holds. Let $\mathbf{x}_{\tau,j}$ be the jth query example of task τ, $\mathbf{z}_{\tau,j}$ be its embedding, and $\hat{y}_{\tau,j} = f_{\boldsymbol{\theta}}(\mathbf{z}_{\tau,j}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{\tau,j})^\top \boldsymbol{\alpha}_\tau$ be its prediction, where $\boldsymbol{\alpha}_\tau$ is the dual solution. Let $\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi}) = \left[\hat{y}_{\tau_1,1}; \ldots; \hat{y}_{\tau_1,n_q}; \ldots; \hat{y}_{\tau_{|\mathcal{T}|},1}; \ldots; \hat{y}_{\tau_{|\mathcal{T}|},n_q}\right]$ be an auxiliary function which maps the meta-parameter to predictions on all query examples. The following Theorem shows that the proposed algorithm converges to a global minimum of the empirical risk $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta},\boldsymbol{\phi})$ at the rate of $\mathcal{O}(\sigma_{\mathbf{g}}^2/\sqrt{T})$. The rate is improved to exponential if the meta-learner adopts full gradient descent.

Theorem 3.4.2. *Assume that $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is uniform conditioning. (i) Let $\eta_t = \min(1/\sqrt{T}, 1/2\eta_{meta})$. Algorithm 3 satisfies $\min_{1\leq t\leq T} \mathbb{E}\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}\left(\sigma_g^2/\sqrt{T}\right)$, where the expectation is taken over the random training samples. (ii) Let $\eta_t = \eta < \min(1/2\eta_{meta}, 4|\mathcal{T}|/\rho\mu)$ and $\mathcal{B}_t = \mathcal{T}$. Algorithm 3 satisfies $\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}((1 - \eta\rho\mu/4|\mathcal{T}|)^t)$.*

3.5 Experiments on Few-shot Regression

Data sets. Experiments are performed on three data sets.

(i) *Sine*. This is the sinusoid regression problem in [35]. Samples x's are uniformly sampled from $[-5, 5]$. Each task τ learns a sine function $y = a_\tau \sin(x + b_\tau) + \xi$, where $a_\tau \in [0.1, 5]$, $b_\tau \in [0, \pi]$, and $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the label noise. We consider both $\sigma_\xi^2 = 0$ (noise-free) and $\sigma_\xi^2 = 1$. In addition to the 5-shot setting in [35], we also evaluate on the more challenging 2-shot setting. We randomly generate a meta-training set of 8000 tasks, a meta-validation set of 1000 tasks for early stopping, and a meta-testing set of 2000 tasks for performance evaluation.

(ii) *Sale*. This is a real-world dataset from [144], which contains weekly purchased quantities of 811 products over 52 weeks. For each product (task), a sample is to predict the sales quantity for the current week from sales quantities in the previous 5 weeks. Thus, each product contains 47 samples. We evaluate on the 5-shot and 1-shot settings. We randomly split the tasks into a meta-training set of 600 tasks, a meta-validation set of 100 tasks, and a meta-testing set of 111 tasks.

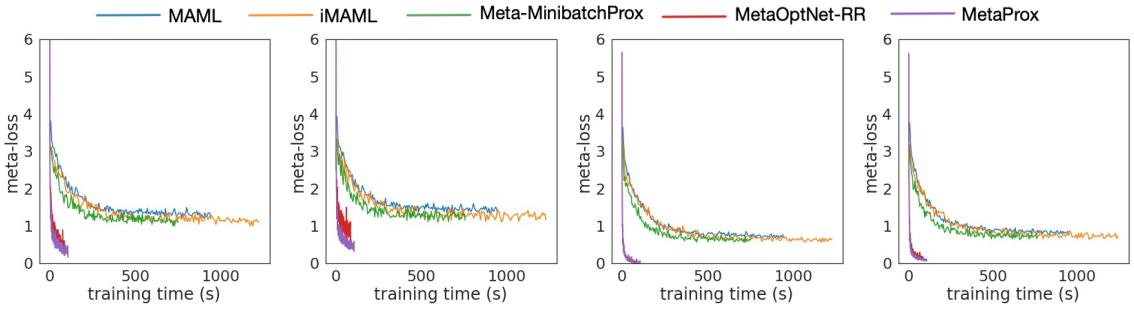
(iii) *QMUL*, which is a multiview face dataset [45] from Queen Mary University of London. This consists of grayscale face images of 37 people (32 for meta-training and 5 for meta-testing). We follow the setting in [110] and evaluate the model on 10-shot regression. Each person has 133 facial images covering a viewsphere of ± 90 in yaw and ± 30 in tilt at 10 increment. A task is a trajectory taken from the discrete manifold for images from the same person. The regression goal is to predict the tilt given an image. In the *in-range* setting, meta-training tasks are sampled from the entire manifold. In the more challenging *out-of-range* setting, meta-training tasks are sampled from the sub-manifold with yaw in $[-90, 0]$. In both settings, meta-testing tasks are sampled from the entire manifold. We randomly sample 2400 tasks for meta-training, and 500

tasks for meta-testing. As in [110], we do not use a meta-validation set since the dataset is small.

Network Architecture. For *Sine* and *Sale*, we use the network in [35], which is a small multilayer perceptron with two hidden layers of size 40 and ReLU activation. For *QMUL*, we use the three-layer convolutional neural network in [110]. The embeddings are always from the last hidden layer. We use a simple linear kernel as base kernel, and $f_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\theta}^\top \mathbf{z}$.

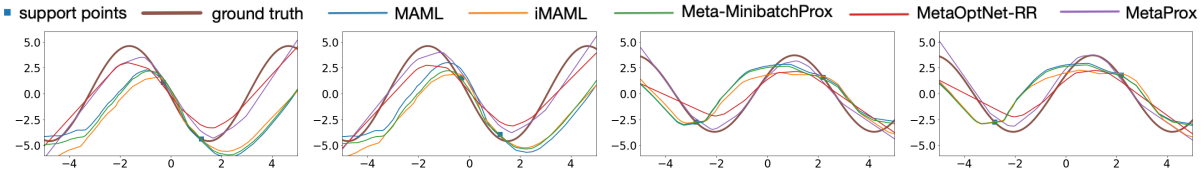
Implementation Details. We use the Adam optimizer [73] with a learning rate of 0.001. Each mini-batch has 16 tasks. For *Sine* and *Sale*, the model ($\boldsymbol{\phi}$ and $f_{\boldsymbol{\theta}}$) is meta-trained for $40,000$ iterations. To prevent overfitting on the meta-training set, we evaluate the meta-validation performance every 500 iterations, and stop training when the loss on the meta-validation set has no significant improvement for 10 consecutive evaluations. For *QMUL*, we follow [110] and meta-train the model for 100 iterations. We repeat each experiment 30 times. For performance evaluation, we use the average mean squared error (MSE) on the meta-testing set.

Baselines. On *Sine* and *Sale*, we compare MetaProx with CommonMean [23], MAML [35], MetaOptNet-RR [79], Meta-MinibatchProx [179], and iMAML [121]. CommonMean is a linear model, and MetaOptNet-RR is equivalent to MetaProx when $f_{\boldsymbol{\theta}} = 0$. Following [35], we set the number of inner gradient steps for MAML to 1 during meta-training, and 20 during meta-validation and meta-testing. Both Meta-MinibatchProx [179] and iMAML [121] are meta-regularization approaches. For *QMUL*, we compare MetaProx with the baselines reported in [110] (namely, DKT [110], Feature Transfer [28], and MAML). As further baselines, we also compare with Meta-MinibatchProx and MetaOptNet-RR to evaluate the improvement of MetaProx due to the learnable $f_{\boldsymbol{\theta}}$.



(a) 2-shot, $\sigma_\xi^2 = 0$. (b) 2-shot, $\sigma_\xi^2 = 1$. (c) 5-shot, $\sigma_\xi^2 = 0$. (d) 5-shot, $\sigma_\xi^2 = 1$.

Figure 3.1: Convergence curves for few-shot sinusoid regression.



(a) task $\tau_1, \sigma_\xi^2 = 0$. (b) task $\tau_1, \sigma_\xi^2 = 1$. (c) task $\tau_2, \sigma_\xi^2 = 0$. (d) task $\tau_2, \sigma_\xi^2 = 1$.

(e) task $\tau_1, \sigma_\xi^2 = 0$. (f) task $\tau_1, \sigma_\xi^2 = 1$. (g) task $\tau_2, \sigma_\xi^2 = 0$. (h) task $\tau_2, \sigma_\xi^2 = 1$.

Figure 3.2: Sinusoid regression: Two meta-testing tasks τ_1 and τ_2 with different σ_ξ's in 2-shot ((a) –(d)) and 5-shot ((e)–(h)) settings.

Table 3.1: Average MSE (with 95% confidence intervals) of few-shot regression on the *Sine* and *Sale* datasets. (The confidence intervals in *Sale* experiments are ± 0.001 for all methods)

	Sine (2-shot)		*Sine* (5-shot)		*Sale*	
	noise-free	noisy	noise-free	noisy	1-shot	5-shot
CommonMean [23]	4.58 ± 0.07	4.59 ± 0.07	4.29 ± 0.06	4.31 ± 0.06	0.090	0.074
MAML [35]	1.24 ± 0.12	1.91 ± 0.13	0.41 ± 0.03	1.15 ± 0.05	0.069	0.063
iMAML [121]	1.12 ± 0.11	1.84 ± 0.10	0.38 ± 0.02	1.02 ± 0.05	0.068	0.063
Meta-MinibatchProx [179]	1.15 ± 0.08	1.87 ± 0.09	0.37 ± 0.02	1.01 ± 0.03	0.081	0.064
MetaOptNet-RR [79]	0.18 ± 0.01	0.79 ± 0.01	0.01 ± 0.00	0.19 ± 0.01	0.088	0.068
MetaProx	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.43 \pm 0.01}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.13 \pm 0.01}$	**0.061**	**0.060**

Results on Sine. Figure 3.1 shows the convergence curves of MetaProx and the baselines. We do not show the convergence of CommonMean, as it does not use a neural network backbone as the other methods. As can be seen, MetaProx converges much faster and better than the non-kernel-based methods (MAML, iMAML and Meta-MinibatchProx). In the 2-shot settings, MetaProx converges to a loss smaller than that of MetaOptNet-RR.

Figure 3.2 shows the learned functions on 2 meta-testing tasks (τ_1 with ($a = 4.6, b = 3.2$) and τ_2 with ($a = 3.7, b = 0.5$)) in the 5-shot setting and more challenging 2-shot setting. As can be seen, MetaProx always fits the target curve well. Though MAML, iMAML and Meta-MinibatchProx can fit the support samples, it performs worse in regions far from the support samples. This is especially noticeable in the 2-shot setting.

Table 3.1 shows the MSE on the meta-testing set. Obviously, CommonMean (a linear model) fails in this nonlinear regression task. MetaProx and MetaOptNet-RR significantly outperform the other baselines. MetaProx (with the learned f_θ) performs better than MetaOptNet-RR, particularly when the data is noisy.

Table 3.2: Average MSE (with 95% confidence intervals) of few-shot regression on *QMUL* (10-shot). Results of the first four methods are from [110].

	in-range	out-of-range
Feature Transfer [28]	0.22 ± 0.03	0.18 ± 0.01
MAML [35]	0.21 ± 0.01	0.18 ± 0.02
DKT + RBF [110]	0.12 ± 0.04	0.14 ± 0.03
DKT + Spectral [110]	0.10 ± 0.02	0.11 ± 0.02
Meta-MinibatchProx [179]	0.171 ± 0.022	0.193 ± 0.025
MetaOptNet-RR [79]	0.021 ± 0.007	0.039 ± 0.009
MetaProx	$\mathbf{0.012 \pm 0.003}$	$\mathbf{0.020 \pm 0.005}$

Results on Sale. As can be seen from Table 3.1, the linear model (CommonMean) performs poorly as expected. MetaProx again outperforms the other baselines, particularly in the more challenging 1-shot setting.

Results on QMUL. Table 3.2 shows that MetaProx achieves the lowest MSE and the kernel methods (DKT+RBF, DKT+Spectral, MetaOptNet-RR, and MetaProx) perform better than non-kernel-based methods (Feature Transfer, MAML, and Meta-MinibatchProx). MetaProx with the learnable $f_{\boldsymbol{\theta}}$ reduces the errors of MetaOptNet-RR by half.

3.6 Experiments on Few-shot Classification

Datasets. We use the standard 5-way K-shot setting ($K = 1$ or 5) on the *mini-ImageNet* [152] dataset, which consists of 100 randomly chosen classes from *ILSVRC*-2012 [129]. Each class contains 600 84×84 images. We use the commonly-used split in [124]: the 100 classes are randomly split into 64 for meta-training, 16 for meta-validation, and 20 for meta-testing.

Network Architecture. For the network backbone, we use the *Conv4* in [35, 152] and *ResNet-12* in [79]. As the cosine similarity is more effective than ℓ_2 distance in few-shot classification [17], we adopt the cosine kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}') = \cos(\mathbf{z}, \mathbf{z}')$ as base kernel, where $\mathbf{z}$ is the embedding of sample $\mathbf{x}$ extracted from the last hidden layer as in regression. For each $c = 1, \ldots, 5$, $f_{\boldsymbol{\theta}^{(c)}} = [\mathcal{K}_{\mathbf{q}^{(1)}}; \ldots; \mathcal{K}_{\mathbf{q}^{(5)}}]^\top \boldsymbol{\theta}^{(c)}$ is a weighted prototype classifier on the embedding space, where $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(5)}$ are the class centroids computed from S_τ, and the weights $\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(5)}\}$ are meta-parameters.

Baselines. We compare MetaProx with the state-of-the-arts: (i) meta-initialization: MAML [35] and its variants FOMAML [35], and REPTILE [104]; (ii) meta-regularization:

iMAML [121] and Meta-MinibatchProx [179]; and (iii) metric learning: ANIL [118], R2D2 [8], ProtoNet [140], and MetaOptNet [79] with SVM using the linear kernel and cosine kernel.

Implementation Details. The entire model is trained end-to-end. $\ell(\hat{\mathbf{y}}, y)$ is the cross-entropy loss. The CVXPYLayers package [1] is used to solve the dual problem. We train the model for $80,000$ iterations, and each mini-batch has 4 tasks. We use the Adam optimizer [73] with an initial learning rate of 0.001, which is then reduced by half every $2,500$ iterations. To prevent overfitting, we evaluate the meta-validation performance every 500 iterations, and stop training when the meta-validation accuracy has no significant improvement for 10 consecutive evaluations. We report the classification accuracy averaged over 600 tasks randomly sampled from the meta-testing set.

Results. Tables 3.3 and 3.4 show the results for *Conv4* and *ResNet-12*, respectively. As can be seen, MetaProx is always the best. Compared with MetaOptNet-SVM, MetaProx is better due to the learnable regularizer.

Table 3.3: Accuracies (with 95% confidence intervals) of 5-way few-shot classification on *mini-ImageNet* using *Conv4*. † means that the result is obtained by rerunning the code with our setup here. Other results from their original publications (Result on the 5-shot setting is not reported in iMAML [121]).

	1-shot	5-shot
MAML [35]	48.7 ± 1.8	63.1 ± 0.9
FOMAML [35]	48.1 ± 1.8	63.2 ± 0.9
REPTILE [104]	50.0 ± 0.3	66.0 ± 0.6
iMAML [121]	49.0 ± 1.8	–
Meta-MinibatchProx [179]	50.8 ± 0.9	67.4 ± 0.9
ANIL [118]	46.7 ± 0.4	61.5 ± 0.5
R2D2 [8]	49.5 ± 0.2	65.4 ± 0.3
ProtoNet [140]	49.4 ± 0.8	68.2 ± 0.7
MetaOptNet-SVM(lin)† [79]	49.8 ± 0.9	66.9 ± 0.7
MetaOptNet-SVM(cos)† [79]	50.1 ± 0.9	67.2 ± 0.6
MetaProx	$\mathbf{52.4} \pm 1.0$	$\mathbf{68.8} \pm 0.8$

3.7 Conclusion

In this chapter, we propose MetaProx, an effective meta-regularization algorithm for meta-learning. MetaProx combines deep kernel and meta-regularization. By reformulating the inner problem in the dual space, a learnable proximal regularizer is introduced to the base learner. The meta-parameters in the regularizer and network are updated

Table 3.4: Accuracies (with 95% confidence intervals) of 5-way few-shot classification on *mini-ImageNet* using *ResNet-12*. † means that the result is obtained by rerunning the code with our setup here.

	1-shot	5-shot
FOMAML† [35]	57.41 ± 0.71	72.12 ± 0.54
ANIL† [118]	59.66 ± 0.68	73.28 ± 0.49
ProtoNet [140]	59.25 ± 0.64	75.60 ± 0.48
MetaOptNet-SVM(lin)† [79]	62.31 ± 0.64	78.21 ± 0.42
MetaOptNet-SVM(cos)† [79]	62.75 ± 0.42	78.68 ± 0.24
MetaProx	$\mathbf{63.82 \pm 0.23}$	$\mathbf{79.12 \pm 0.18}$

by the meta-learner. We also establish convergence of MetaProx. Extensive experiments on standard datasets for regression and classification verify the effectiveness of learning a proximal regularizer. Furthermore, MetaProx is more efficient than existing non-kernel-based methods.

CHAPTER 4

Subspace Learning for Effective Meta-Learning

4.1 Introduction

Typical meta-learning algorithms [35, 23, 121, 179] learn a globally-shared meta-model for all tasks. For example, the Model-Agnostic Meta-Learning (MAML) algorithm [35] learns a meta-initialization such that a good model for an unseen task can be fine-tuned from limited samples by a few gradient updates. MetaProx proposed in Chapter 3 seeks a common meta-regularization for all task models. However, when the task environments are heterogeneous, task model parameters are diverse and a single meta-model may not be sufficient to capture all the meta-knowledge.

To tackle this issue, a variety of methods have been proposed to learn structured meta-knowledge by exploring the task structure [65, 167, 168, 180, 150]. For example, Jerfel et al. [65] formulate the task distribution as a mixture of hierarchical Bayesian models, and update the components (i.e., initializations) using an Expectation Maximization procedure. TSA-MAML [180] first trains task models using vanilla MAML. Tasks are grouped into clusters by k-means clustering, and cluster centroids form group-specific initializations.

Alternatively, task model parameters can be formulated into a subspace. In the linear regression setting where task model vectors are sampled from a low-dimensional subspace, recent attempts [74, 150] use a moment-based estimator to recover the subspace based on the property that the column space of the sample covariance matrix recovers the underlying subspace. However, for nonlinear models such as deep networks, this nice property no longer holds and the moment-based methods cannot be generalized.

In this chapter, we propose a model-agnostic algorithm called MUSML (MUltiple Subspaces for Meta-Learning). Each subspace represents one type of meta-knowledge, and subspace bases are treated as meta-parameters. For each task, the base learner builds a task model from each subspace. The meta-learner then updates the subspace bases by minimizing a weighted validation loss of the task models. We theoretically establish upper bounds on the population risk, empirical risk and generalization gap. All these

bounds depend on the complexity of the subspace mixture (number of component subspaces and subspace dimensionality). Experiments on various datasets verify the effectiveness of the proposed MUSML.

Our major contributions are four-fold: (i) We formulate task model parameters into a subspace mixture and propose a novel algorithm to learn the subspace bases. (ii) The proposed MUSML is model-agnostic, and can be used on linear and nonlinear models. (iii) We theoretically study the generalization properties of the learned subspaces. (iv) We perform extensive experiments on synthetic and real-world datasets. Results on the synthetic dataset confirm that MUSML is able to discover the underlying subspaces of task model parameters. Results on the real-world datasets demonstrate superiority of MUSML over the state-of-the-arts.

4.2 Learning Multiple Subspaces for Meta-Learning

4.2.1 Linear Regression Tasks

We first focus on the linear setting, where the task is linear regression and all task parameters lie in one single (linear) subspace. The following proposition shows that the underlying subspace can be recovered using a moment-based estimator.

Proposition 4.2.1. *[74, 150]. Assume that $p(\tau)$ is a distribution of linear regression tasks. Each task τ is associated with a $\mathbf{w}_\tau^\star \in \mathbb{R}^d$, and its samples are generated as $y = \mathbf{x}^\top \mathbf{w}_\tau^\star + \xi$, where $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the noise. Then, $\mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{(\mathbf{x},y)\sim\tau,(\mathbf{x}',y')\sim\tau} yy'\mathbf{x}\mathbf{x}'^\top = \mathbb{E}_{\tau\sim p(\tau)}\mathbf{w}_\tau^\star\mathbf{w}_\tau^{\star\top}$.*

Proposition 4.2.1 shows $\hat{\mathbf{S}} \equiv \frac{1}{|\mathcal{B}|}\sum_{\tau\in\mathcal{B}} y_\tau y'_\tau \mathbf{x}_\tau \mathbf{x}'^\top_\tau$ is an unbiased estimator of $\mathbb{E}_{\tau\sim p(\tau)}\mathbf{w}_\tau^\star\mathbf{w}_\tau^{\star\top}$, where $(\mathbf{x}_\tau, y_\tau)$ and $(\mathbf{x}'_\tau, y'_\tau)$ are two samples drawn from τ, and $\mathcal{B}$ is a collection of tasks. Hence, the column space of $\hat{\mathbf{S}}$ recovers the column space of $\mathbb{E}_{\tau\sim p(\tau)}\mathbf{w}_\tau^\star\mathbf{w}_\tau^{\star\top}$ (i.e., the underlying subspace) when the number of tasks is sufficient.

4.2.2 The Proposed MUSML

While Proposition 4.2.1 can be used to recover the column space in a linear meta-learning setting, extension to the nonlinear setting (such as deep networks) is difficult. To address this problem, we propose a model-agnostic algorithm called MUSML (MUltiple

Subspaces for Meta-Learning). We assume that the model parameters $\mathbf{w}_\tau$'s lie in K subspaces $\{\mathbb{S}_1, \ldots, \mathbb{S}_K\}$, which can be seen as an approximation to a nonlinear manifold. For simplicity, we assume that all K subspaces have the same dimensionality m (this can be easily extended to the case where the subspaces have different dimensionalities). Let $\mathbf{S}_k \in \mathbb{R}^{d\times m}$ be a basis of $\mathbb{S}_k$. $\{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ are then the meta-parameters to be learned.

The proposed procedure is shown in Algorithm 6. Given a task τ, the base learner searches for the model parameter $\mathbf{w}_\tau$ over all subspaces with fixed $\mathbf{S}_k$ (steps 4-11). In each subspace $\mathbb{S}_k$, we search for the best linear combination $\mathbf{v}^\star_{\tau,k}$ of the subspace's basis to form $\mathbf{w}_\tau$:

$$\mathbf{v}^\star_{\tau,k} = \arg\min_{\mathbf{v}_\tau \in \mathbb{R}^m} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_\tau). \tag{4.1}$$

$\mathbf{S}_k \mathbf{v}^\star_{\tau,k}$ is then the task model parameter corresponding to the kth subspace. When $\ell(f(\mathbf{x}; \mathbf{w}), y)$ is convex in $\mathbf{w}$, it is easy to verify that $\mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_\tau)$ is also convex in $\mathbf{v}_\tau$. Hence, problem (4.1) can be solved as a convex program [11]. However, for nonlinear models such as deep networks, the loss function in (4.1) is nonconvex, and thus finding $\mathbf{v}^\star_{\tau,k}$ is computationally intractable. Instead, we seek an approximate minimizer $\mathbf{v}_{\tau,k}$ by performing T_{in} gradient descent steps from an initialization $\mathbf{v}^{(0)}_{\tau,k}$, i.e.,

$$\mathbf{v}^{(t'+1)}_{\tau,k} = \mathbf{v}^{(t')}_{\tau,k} - \alpha \nabla_{\mathbf{v}^{(t')}_{\tau,k}} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}^{(t')}_{\tau,k}),$$

where $\alpha > 0$ is the step size and $\mathbf{v}_{\tau,k} \equiv \mathbf{v}^{(T_{in})}_{\tau,k}$.

At meta-training, one can assign τ to the subspace with the best training set performance. However, this is inefficient for learning meta-parameters since only one subspace is updated at each step. Similar to DARTS [91], we relax the categorical choice to a softmax selection over all candidate subspaces. The relaxed operation is differentiable and all subspace bases can then be updated simultaneously, which accelerates learning. Let $o_{\tau,k} = \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_{\tau,k})$ be the training loss for task τ when the kth subspace (where $k = 1, \ldots, K$) is used to construct its task model. The meta-learner updates $\{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ by performing one gradient update on the weighted validation loss (steps 13-14):

$$\sum_{k=1}^{K} \frac{\exp(-o_{\tau,k}/\gamma)}{\sum_{k'=1}^{K} \exp(-o_{\tau,k'}/\gamma)} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{S}_k \mathbf{v}_{\tau,k}), \tag{4.2}$$

where $\gamma > 0$ is the temperature. When γ is close to 0, the softmax selection becomes one-hot; whereas when γ increases to ∞, the selection becomes uniform. In practice,

Algorithm 4 MUltiple Subspaces for Meta-Learning (MUSML).

Require: stepsize α, $\{\eta_t\}$; number of inner gradient steps T_{in}, number of subspaces K, subspace dimension m, temperature $\{\gamma_t\}$; initialization $\mathbf{v}^{(0)}$;

for $t = 0, 1, \ldots, T-1$ **do**
  sample a task τ with $\mathcal{S}_\tau$ and $\mathcal{Q}_\tau$;
  base learner:
  for $k = 1, \ldots, K$ **do**
    initialize $\mathbf{v}_{\tau,k}^{(0)} = \mathbf{v}^{(0)}$;
    for $t' = 0, 1, \ldots, T_{in} - 1$ **do**
      $\mathbf{v}_{\tau,k}^{(t'+1)} = \mathbf{v}_{\tau,k}^{(t')} - \alpha \nabla_{\mathbf{v}_{\tau,k}^{(t')}} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_{k,t} \mathbf{v}_{\tau,k}^{(t')})$;
    end for
    $\mathbf{v}_{\tau,k} \equiv \mathbf{v}_{\tau,k}^{(T_{in})}$;
    $o_{\tau,k} = \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_{k,t} \mathbf{v}_{\tau,k})$;
  end for
  meta-learner:
  $\mathcal{L}_{vl} = \sum_{k=1}^{K} \frac{\exp(-o_{\tau,k}/\gamma_t)}{\sum_{k'=1}^{K} \exp(-o_{\tau,k'}/\gamma_t)} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{S}_{k,t} \mathbf{v}_{\tau,k})$;
  $\{\mathbf{S}_{1,t+1}, \ldots, \mathbf{S}_{k,t+1}\} = \{\mathbf{S}_{1,t}, \ldots, \mathbf{S}_{k,t}\} - \eta_t \nabla_{\{\mathbf{S}_{1,t}, \ldots, \mathbf{S}_{k,t}\}} \mathcal{L}_{vl}$;
end for
Return $\mathbf{S}_{1,T}, \ldots, \mathbf{S}_{K,T}$.

we start at a high temperature and anneal to a small but nonzero temperature as in [62, 19, 181]. Note that $\{o_{\tau,k} : k = 1, \ldots, K\}$ depend on the bases and $\nabla_{\{\mathbf{S}_1, \ldots, \mathbf{S}_k\}} o_{\tau,k}$ can be computed by auto-differentiation.

At meta-testing, for each testing task τ', we assign τ' to the subspace with the lowest training loss, i.e., $\mathbf{w}_{\tau'} = \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau', k_{\tau'}}$, where $k_{\tau'} \equiv \arg\min_{1 \le k \le K} \mathcal{L}(\mathcal{S}_{\tau'}; \mathbf{S}_k \mathbf{v}_{\tau',k})$ is the chosen subspace index.

4.3 Theoretical Analysis

In this section, we study the generalization performance of the learned subspace bases $\mathcal{S} \equiv \{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ at meta-testing. The following assumptions on smoothness and compactness are standard in meta-learning [5, 46, 32] and bilevel optimization [39, 5]. The boundedness assumption on the loss function is widely used in analyzing generalization of meta-learning algorithms [98, 111, 2] and traditional machine learning algorithms [10].

Assumption 4.3.1. (i) $\ell(f(\mathbf{x}; \mathbf{w}), y)$ and $\nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}), y)$ are ϱ-Lipschitz and β-Lipschitz in $\mathbf{w}$, respectively;[1] (ii) $\{\mathbf{v}_{\tau,k} : \tau \sim p(\tau), k = 1, \ldots, K\}$ and column vectors of $\mathbf{S}_k$

[1] In other words, $\|\ell(f(\mathbf{x}; \mathbf{w}), y) - \ell(f(\mathbf{x}; \mathbf{w}'), y)\| \le \varrho \|\mathbf{w} - \mathbf{w}'\|$, and $\|\nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}), y) -$

$(k = 1, \ldots, K)$ are in a compact set, and their ℓ_2-norms are upper bounded by a constant $\rho > 0$. (iii) $\ell(\cdot, \cdot)$ is upper bounded by a constant $\nu > 0$.

Let $\mathcal{R}(\mathcal{S}) \equiv \mathbb{E}_{\tau'} \mathbb{E}_{\mathcal{D}_{\tau'}^{tr}} \mathbb{E}_{(\mathbf{x},y)\sim\tau'} \ell(f(\mathbf{x}; \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau', k_{\tau'}}), y)$ be the expected population risk, and $\hat{\mathcal{R}}(\mathcal{S}) \equiv \mathbb{E}_{\tau'} \mathbb{E}_{\mathcal{D}_{\tau'}^{tr}} \mathcal{L}(\mathcal{D}_{\tau'}^{tr}; \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau', k_{\tau'}})$ be the expected empirical risk averaged over all tasks. The following Theorem characterizes the generalization gap, i.e., the gap between the population and empirical risks.

Theorem 4.3.1. *With Assumption 4.3.1, we have*

$$\mathcal{R}(\mathcal{S}) \leq \hat{\mathcal{R}}(\mathcal{S}) + K \sqrt{\frac{\nu^2 + 12\rho\nu(1 + m\alpha\delta)^{T_{in}}}{2n_s}}, \tag{4.3}$$

where $\delta = \beta\rho^2 > 0$.

The proof is based on the connection between generalization and stability [10]. The dependence on n_s in (4.3) agrees with their Theorem 11, as the stability constant in Algorithm 6 is of the order $\mathcal{O}(1/n_s)$. From (4.3), one can observe that increasing the subspace complexity (i.e., m or K) increases the upper bound of $\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S})$.

For notation simplicity, throughout this section, we omit the superscript τ'. Let $\mathbf{z} \equiv (\mathbf{x}, y)$ be the samples, $\ell(\mathbf{z}; \mathbf{w}) \equiv \ell(f(\mathbf{x}; \mathbf{w}), y)$, and $\nabla_{\mathbf{w}} \ell(f(\mathbf{z}; \mathbf{S}\mathbf{v})) \equiv \nabla_{\mathbf{w}} \ell(f(\mathbf{z}; \mathbf{w}))\,|_{\mathbf{w}=\mathbf{S}\mathbf{v}}$.

We first show the stability constant (in Lemma 9 of [10]) in Algorithm 4 is of the order $\mathcal{O}(1/n_s)$.

Let $\{\mathbf{z}_i' : i = 1, \ldots, n_s\}$ be another n_s samples from τ. Let $\mathcal{S}_\tau^{(i)}$ be another training set which differs from $\mathcal{S}_\tau$ only in the ith sample (i.e., $\mathcal{S}_\tau^{(i)} \equiv (\mathcal{S}_\tau - \{\mathbf{z}_i\}) \cup \{\mathbf{z}_i'\}$). We let $\mathbf{v}_{\tau,k,i}$ (resp. $\mathbf{v}_{\tau,k}$) be the task model obtained from the base learner when using training set $\mathcal{S}_\tau^{(i)}$ (resp. $\mathcal{S}_\tau$).

Let $\mathbf{w}_{\tau'}^\star \equiv \arg\min_{\mathbf{w}_{\tau'}} \mathbb{E}_{(\mathbf{x},y)\sim\tau'} \ell(f(\mathbf{x}; \mathbf{w}_{\tau'}), y)$ be the optimal task model for task τ', and $\mathcal{R}^\star \equiv \mathbb{E}_{\tau'} \mathbb{E}_{(\mathbf{x},y)\sim\tau'} \ell(f(\mathbf{x}; \mathbf{w}_{\tau'}^\star), y)$ be the minimum expected loss averaged over all tasks. The following Theorem provides an upper bound on the expected excess risk [180] $\mathcal{R}(\mathcal{S}) - \mathcal{R}^\star$, which compares the performance of the learned task model with that of the optimal model.

$\nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}'), y)\| \leq \beta \|\mathbf{w} - \mathbf{w}'\|$.

Theorem 4.3.2. *With Assumption 4.3.1, we have*

$$\mathcal{R}(\mathcal{S}) - \mathcal{R}^\star \leq \rho\sqrt{m}\, \mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}} \|\mathbf{v}_{\tau',k_{\tau'}} - \mathbf{v}^\star_{\tau',k_{\tau'}}\| + \varrho\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}} \mathrm{dist}(\mathbf{w}^\star_{\tau'}, \mathbb{S}_{k_{\tau'}}) + K\sqrt{\frac{\nu^2 + 12\varrho\nu(1+m\alpha\delta)^{T_{in}}}{2n_s}},$$

where $\mathrm{dist}(\mathbf{w}^\star_{\tau'}, \mathbb{S}_{k_{\tau'}}) \equiv \min_{\mathbf{w}\in\mathbb{S}_{k_{\tau'}}} \|\mathbf{w} - \mathbf{w}^\star_{\tau'}\|$ *is the distance between* $\mathbf{w}^\star_{\tau'}$ *and* $\mathbb{S}_{k_{\tau'}}$.

From Theorem 4.3.2, $\mathcal{R}(\mathcal{S}) - \mathcal{R}^\star$ is upper-bounded by three terms: (i) The first term measures the distance between the approximate minimizer $\mathbf{v}_{\tau',k_{\tau'}}$ and exact minimizer $\mathbf{v}^\star_{\tau',k_{\tau'}}$; (ii) The second term arises from the approximation error of $\mathbf{w}^\star_{\tau'}$ using the learned subspaces; (iii) The third term depends on the complexity of subspaces (i.e., m and K). For the centroid-based clustering method in [180], the upper bound of its expected excess risk contains a term $\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\|\boldsymbol{\omega}_{k^\star_{\tau'}} - \mathbf{w}^\star_{\tau'}\|^2$, where $\boldsymbol{\omega}_{k^\star_{\tau'}}$ is the centroid of the cluster that τ' is assigned to. The distance $\|\boldsymbol{\omega}_{k^\star_{\tau'}} - \mathbf{w}^\star_{\tau'}\|^2$ plays the same role as the term $\mathrm{dist}(\mathbf{w}^\star_{\tau'}, \mathbb{S}_{k_{\tau'}})$ in Theorem 4.3.2, which measures how far the optimal model $\mathbf{w}^\star_{\tau'}$ is away from the subspaces or clusters.

4.4 Experiments on Few-shot Regression

4.4.1 Synthetic Data

In this experiment, we use a synthetic 1-dimensional data set to examine whether MUSML can discover subspaces that the task model parameters lie in. We use a 5-shot regression setting, with 14,000 meta-training, 2,000 meta-validation, and 6,000 meta-testing tasks. The model for task τ is $f(x;\mathbf{w}_\tau) = \exp(0.1w_{\tau,1}x) + w_{\tau,2}|\sin(x)|$, in which $\mathbf{w}_\tau = [w_{\tau,1}; w_{\tau,2}]$ is randomly sampled from one of the two subspaces: (i) *Line-A*: $\mathbf{w}_\tau = \mathbf{S}_1 a_\tau + 0.1\boldsymbol{\xi}_\tau$, where $\mathbf{S}_1 = [1;1]$, $a_\tau \sim \mathcal{U}(1,5)$, and $\boldsymbol{\xi}_\tau \sim \mathcal{N}(\mathbf{0};\mathbf{I})$; and (ii) *Line-B*: $\mathbf{w}_\tau = \mathbf{S}_2 a_\tau + 0.1\boldsymbol{\xi}_\tau$, where $\mathbf{S}_2 = [-1;1]$, $a_\tau \sim \mathcal{U}(0,2)$, and $\boldsymbol{\xi}_\tau \sim \mathcal{N}(\mathbf{0};\mathbf{I})$. The samples of task τ are generated as $y = f(x;\mathbf{w}_\tau) + 0.05\xi$, where $x \sim \mathcal{U}(-5,5)$ and $\xi \sim \mathcal{N}(0,1)$. The experiment is repeated 10 times with different seeds.

The subspace bases are trained for $T = 30,000$ iterations using the Adam optimizer [73]. For the meta-learner, the initial learning rate is 0.001, which is then reduced by half every 5,000 iterations. The base learner uses a learning rate of $\alpha = 0.05$, $\mathbf{v}^{(0)} = \frac{1}{m}\mathbf{1}$, and T_{in} is 5 (resp. 20) at meta-training (resp. meta-testing). The temperature is $\gamma_t =$

$\max(10^{-5}, 0.5 - t/T)$, a linear annealing schedule as in [19, 181]. To prevent overfitting, we evaluate the meta-validation performance every 2,000 iterations, and stop training when the meta-validation accuracy has no significant improvement for 5 consecutive evaluations.

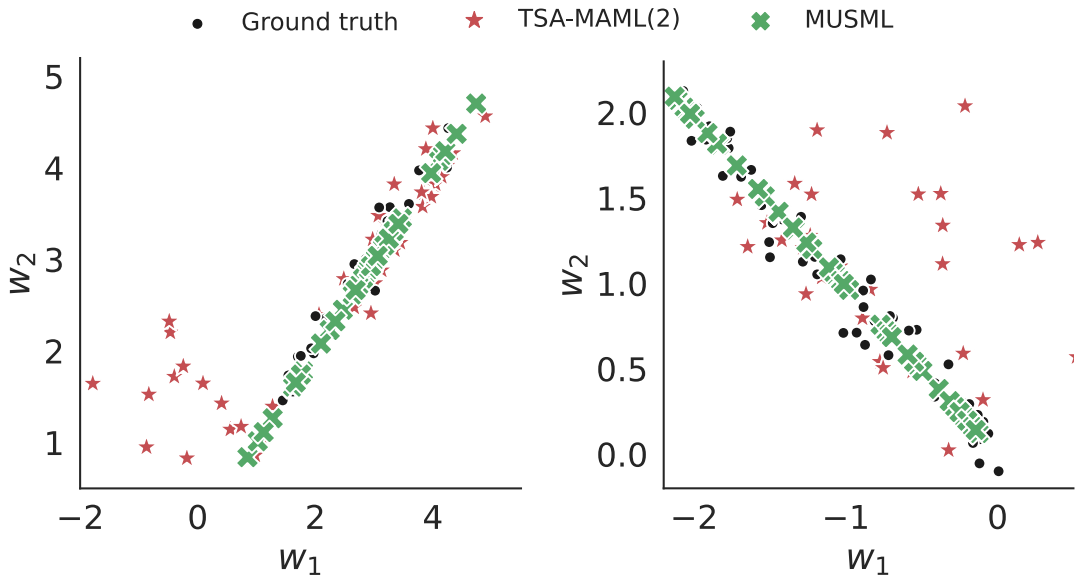
The proposed MUSML (with $K = 2, m = 1$) is compared with the following meta-learning baselines: (i) MAML [35], (ii) BMG [38], which uses target bootstrap, and structured meta-learning algorithms including (iii) Dirichlet process mixture model (DPMM) [65], (iv) hierarchically structured meta-learning (HSML) [167], (v) automated relational meta-learning (ARML) [168] using a graph structure, and (vi) task similarity aware meta-learning (TSA-MAML) [180] with different numbers of clusters. We use these baselines' official implementations (except for DPMM and BMG whose implementations are not publicly available). For performance evaluation, the mean squared error (MSE) on the meta-testing set is used.

Results. Table 4.1 shows the meta-testing MSE. As can be seen, structured meta-learning methods (DPMM, HSML, ARML, TSA-MAML, and MUSML) are significantly better than methods with a globally-shared meta-model (MAML and BMG). In particular, MUSML performs the best. Furthermore, simply increasing the number of clusters in TSA-MAML fails to beat MUSML.

Figure 4.1 visualizes the task model parameters obtained by TSA-MAML(2) and MUSML on 100 randomly sampled meta-testing tasks (50 per subspace). As can be seen, MUSML successfully discovers the underlying subspaces, while the centroid-based clustering method TSA-MAML does not. Table 4.2 shows the average Euclidean distance between

Table 4.1: Meta-testing MSE (with standard deviation) of 5-shot regression on synthetic data. For TSA-MAML, the number in brackets is the number of clusters used.

MAML [35]	0.74 ± 0.03
BMG [38]	0.67 ± 0.03
DPMM [65]	0.56 ± 0.09
HSML [167]	0.49 ± 0.10
ARML [168]	0.60 ± 0.07
TSA-MAML(2) [180]	0.58 ± 0.10
TSA-MAML(10) [180]	0.24 ± 0.09
TSA-MAML(20) [180]	0.12 ± 0.10
TSA-MAML(40) [180]	0.14 ± 0.09
TSA-MAML(80) [180]	0.13 ± 0.08
MUSML	$\mathbf{0.07} \pm 0.01$



(a) Tasks from *Line-A*. (b) Tasks from *Line-B*.

Figure 4.1: Visualization of task model parameters.

Table 4.2: Average Euclidean distance (with standard deviation) between the estimated task model parameters and ground-truth in 5-shot setting on synthetic data. For TSA-MAML, the number in brackets is the number of clusters used.

MAML [35]	1.69 ± 0.02
BMG [38]	1.55 ± 0.03
DPMM [65]	0.85 ± 0.10
HSML [167]	0.80 ± 0.09
ARML [168]	0.91 ± 0.11
TSA-MAML(2) [180]	0.88 ± 0.12
TSA-MAML(10) [180]	0.47 ± 0.19
TSA-MAML(20) [180]	0.33 ± 0.18
TSA-MAML(40) [180]	0.36 ± 0.19
TSA-MAML(80) [180]	0.36 ± 0.18
MUSML	$\mathbf{0.17} \pm 0.01$

the estimated task model parameters and the ground truth. As can be seen, MUSML is more accurate in estimating the task models, confirming the effectiveness of the learned subspaces.

4.4.2 *Pose* Data

While the synthetic data used in the previous experiment is tailored for the proposed subspace model, in this section, we perform experiments on a real-world pose prediction dataset from [171]. This is created based on the Pascal 3D data [162]. Each object contains

100 samples, where input $\mathbf{x}$ is a 128×128 grey-scale image and output y is its orientation relative to a fixed canonical pose. Following [171], we adopt a 15-shot regression setting and randomly select 50 objects for meta-training, 15 for meta-validation, and 15 for meta-testing. The experiment is repeated 15 times with different seeds.

The MR-MAML regularization [171] is used on all the methods except the vanilla MAML. MUSML uses the same encoder-decoder network in [171] as the model $f(\mathbf{x}; \mathbf{w})$. Hyperparameters K and m, as well as the number of clusters in TSA-MAML, are chosen from 1 to 5 using the meta-validation set. For performance evaluation, the MSE on the meta-testing set is used.

Table 4.3 shows the meta-testing MSE. As can be seen, MUSML is again better than the other baselines, confirming the effectiveness of the learned subspaces.

Table 4.3: Meta-testing MSE (with standard deviation) of 15-shot regression on *Pose*. Results on MAML and MR-MAML are from [171].

MAML [35]	5.39 ± 1.31
MR-MAML [171]	2.26 ± 0.09
BMG [38]	2.16 ± 0.15
DPMM [65]	1.99 ± 0.08
HSML [167]	2.04 ± 0.13
ARML [168]	2.21 ± 0.15
TSA-MAML [180]	1.96 ± 0.07
MUSML	$\mathbf{1.83} \pm 0.05$

4.5 Experiments on Few-shot Classification

4.5.1 Experimental Setup

Setup. In this experiment, we use three meta-datasets: (i) *Meta-Dataset-BTAF*, proposed in [167], which consists of four image classification datasets: (a) *Bird*; (b) *Texture*; (c) *Aircraft*; and (d) *Fungi*. Sample images are shown in Figure 4.2. (ii) *Meta-Dataset-ABF*, proposed in [180], which consists of *Aircraft*, *Bird*, and *Fungi*. (iii) *Meta-Dataset-CIO*, which consists of three widely-used few-shot datasets: *CIFAR-FS* [8], *mini-ImageNet* [152], and *Omniglot* [76]. We use the meta-training/meta-validation/meta-testing splits in [168, 180, 76]. A summary of the datasets is in Table 4.4.

As for the network architecture, we use the standard *Conv4* backbone [35, 168, 180], and a simple prototype classifier with cosine similarity on top [140, 44] as $f(\mathbf{x}; \mathbf{w})$.

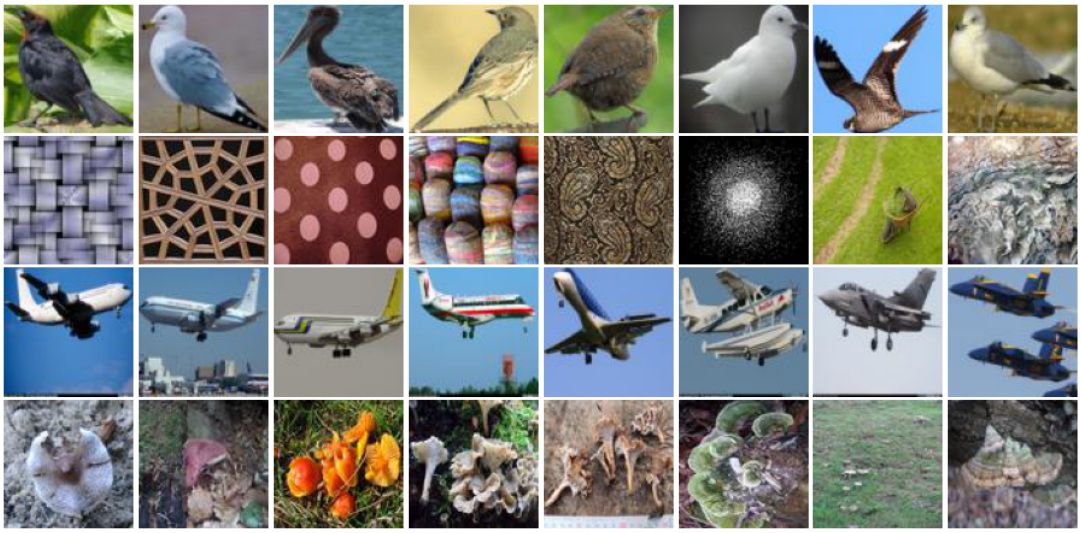


Figure 4.2: Some random images from the meta-testing set of *Meta-Dataset-BTAF* (Top to bottom: *Bird*, *Texture*, *Aircraft*, and *Fungi*).

Table 4.4: Statistics of the datasets.

	#classes (meta-training/meta-validation/meta-testing)
Bird	64/16/20
Texture	30/7/10
Aircraft	64/16/20
Fungi	64/16/20
CIFAR-FS	64/16/20
mini-ImageNet	64/16/20
Omniglot	71/15/16

Hyperparameters K and m are chosen from 1 to 5 on the meta-validation set.

We use the cross-entropy loss for $\ell(\cdot,\cdot)$. The number of parameters in Conv4 is 113,088. For the base learner, $\alpha = 0.01$, $\mathbf{v}^{(0)} = \frac{1}{m}\mathbf{1}$, and T_{in} is set to 5 (resp. 15) at meta-training (resp. meta-validation or meta-testing). We train the subspace bases for $T = 100,000$ iterations using the Adam optimizer [73] with an initial learning rate of 0.001, which is then reduced by half every 5,000 iterations. The temperature is set to $\gamma_t = \max(10^{-5}, 0.8 - t/T)$, which is again a linear annealing schedule [19, 181]. To prevent overfitting, we evaluate the meta-validation performance every 2,000 iterations and stop training when the meta-validation accuracy has no significant improvement for 5 consecutive evaluations. Hyperparameters K and m are chosen from 1 to 5 on the meta-validation set. In practice, as shown in Section 4.5.5, m can simply be fixed at 2, and K can be chosen from 3 to 4. As the search space of K is small, the additional cost of tuning K and m is small.

MUSML is compared with the following state-of-the-arts in the 5-way 5-shot and 5-way 1-shot settings: (i) meta-learning algorithms with a globally-shared meta-model including MAML, ProtoNet, ANIL [119], and BMG; (ii) structured meta-learning algorithms including DPMM, HSML, ARML, TSA-MAML and its variant using ProtoNet as the base learner (denoted TSA-ProtoNet). The number of clusters in TSA-MAML and TSA-ProtoNet are tuned from 1 to 5 on the meta-validation set. For performance evaluation, the classification accuracy on the meta-testing set is used. The experiment is repeated 5 times with different seeds.

4.5.2 *Meta-Dataset-BTAF*

Table 4.5 shows the 5-shot results. As can be seen, MUSML is more accurate than both structured and unstructured meta-learning methods, demonstrating the benefit of structuring task model parameters into subspaces. Figure 4.3 shows the assignment of tasks to the learned subspaces in MUSML. As can be seen, meta-training tasks from the same dataset are always assigned to the same subspace, demonstrating that MUSML can discover the task structure from meta-training tasks. Though the meta-validation and meta-testing classes are not seen during meta-training, most of the corresponding tasks are still assigned to the correct subspaces. The assignment for *Texture* is slightly worse, as the *Texture* and *Fungi* images are more similar to each other (Figure 4.2).

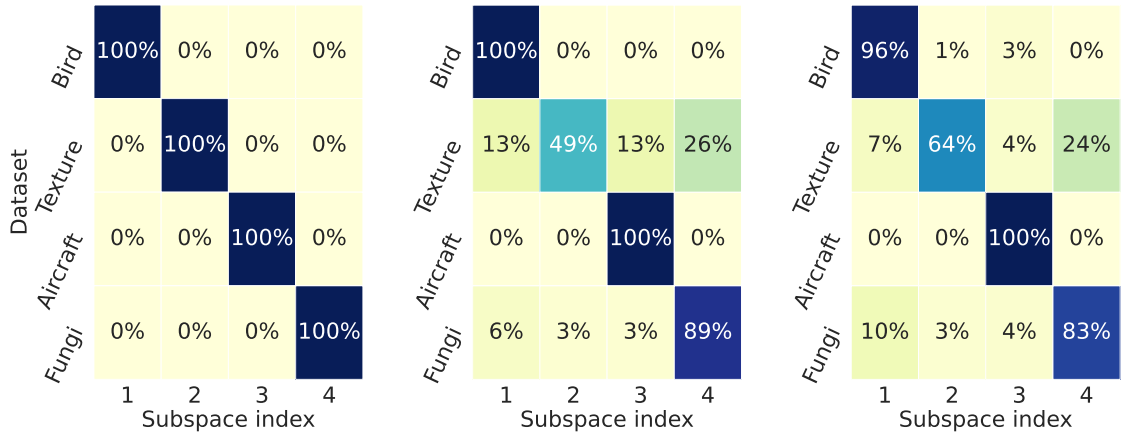
Table 4.6 shows the 1-shot results. MUSML, while still the best overall, has a smaller improvement than in the 5-shot setting. This suggests that having more training samples is beneficial for the base learner to choose a proper subspace. The assignment of tasks to the learned subspaces is shown in Figure 4.4.

4.5.3 *Meta-Dataset-ABF* and *Meta-Dataset-CIO*

Tables 4.7 and 4.8 show the results on *Meta-Dataset-ABF* and *Meta-Dataset-CIO*, respectively. Here, we only consider the 5-shot setting, which is more useful for subspace learning. As can be seen, MUSML consistently outperforms centroid-based clustering methods (DPMM, TSA-MAML, TSA-ProtoNet) and structured meta-learning methods (HSML, ARML). MUSML again outperforms methods with a globally-shared meta-model (MAML, ProtoNet, ANIL, BMG), confirming the effectiveness of using a subspace mixture. The performance of MUSML on *Omniglot* is slightly worse in Table 4.8. This

Table 4.5: 5-way 5-shot accuracy (with 95% confidence interval) on *Meta-Dataset-BTAF*. Results marked with † are from [168].

	Bird	*Texture*	*Aircraft*	*Fungi*	average
MAML† [35]	68.52 ± 0.73	44.56 ± 0.68	66.18 ± 0.71	51.85 ± 0.85	57.78
ProtoNet [140]	71.48 ± 0.72	50.36 ± 0.67	71.67 ± 0.69	55.68 ± 0.82	62.29
ANIL [119]	70.67 ± 0.72	44.67 ± 0.95	66.05 ± 1.07	52.89 ± 0.30	58.57
BMG [38]	71.56 ± 0.76	49.44 ± 0.73	66.83 ± 0.79	52.56 ± 0.89	60.10
DPMM [65]	72.22 ± 0.70	49.32 ± 0.68	73.55 ± 0.69	56.82 ± 0.81	63.00
TSA-MAML [180]	72.31 ± 0.71	49.50 ± 0.68	74.01 ± 0.70	56.95 ± 0.80	63.20
HSML† [167]	71.68 ± 0.73	48.08 ± 0.69	73.49 ± 0.68	56.32 ± 0.80	62.39
ARML† [168]	73.68 ± 0.70	49.67 ± 0.67	74.88 ± 0.64	57.55 ± 0.82	63.95
TSA-ProtoNet [180]	73.70 ± 0.73	50.91 ± 0.74	73.55 ± 0.78	56.11 ± 0.82	63.57
MUSML	**76.79** ± 0.72	**52.41** ± 0.75	**77.76** ± 0.82	**57.74** ± 0.81	**66.18**

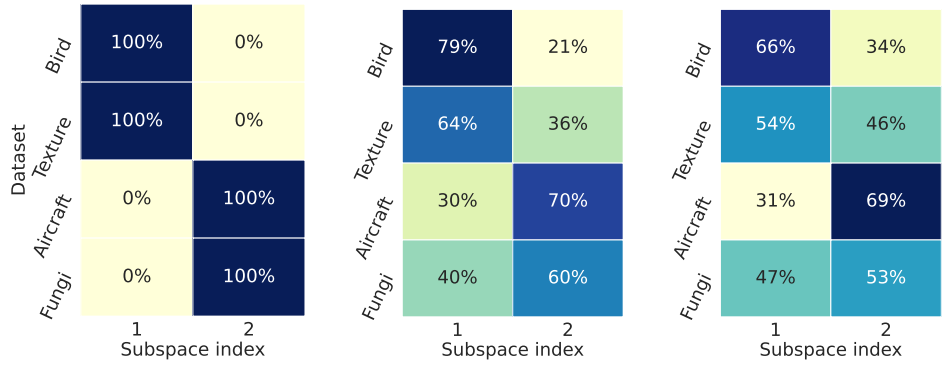


(a) Meta-train. (b) Meta-valid. (c) Meta-test.

Figure 4.3: Task assignment to the learned subspaces in 5-way 5-shot setting on *Meta-Dataset-BTAF* (the number of subspaces K selected by the meta-validation set is 4). Darker color indicates higher percentage.

Table 4.6: 5-way 1-shot accuracy (with 95% confidence interval) on *Meta-Dataset-BTAF*. Results marked with † are from [168].

	Bird	*Texture*	*Aircraft*	*Fungi*	average
MAML† [35]	53.94 ± 1.45	31.66 ± 1.31	51.37 ± 1.38	42.12 ± 1.36	44.77
ProtoNet [140]	60.37 ± 1.31	40.57 ± 0.78	52.83 ± 0.93	44.10 ± 1.36	49.50
ANIL [119]	53.36 ± 1.42	31.91 ± 1.25	52.87 ± 1.34	42.30 ± 1.28	45.11
BMG [38]	54.12 ± 1.46	32.19 ± 1.21	52.09 ± 1.35	43.00 ± 1.37	45.35
DPMM [65]	61.30 ± 1.47	35.21 ± 1.35	57.88 ± 1.37	43.81 ± 1.45	49.55
TSA-MAML [180]	61.37 ± 1.42	35.41 ± 1.39	**58.78** ± 1.37	44.17 ± 1.25	49.93
HSML† [167]	60.98 ± 1.50	35.01 ± 1.36	57.38 ± 1.40	44.02 ± 1.39	49.35
ARML† [168]	**62.33** ± 1.47	35.65 ± 1.40	58.56 ± 1.41	44.82 ± 1.38	50.34
TSA-ProtoNet [180]	60.41 ± 1.02	40.98 ± 1.20	53.29 ± 0.89	43.91 ± 1.31	49.64
MUSML	60.52 ± 0.33	**41.33** ± 1.30	54.69 ± 0.69	**45.60** ± 0.43	**50.53**



(a) Meta-train. (b) Meta-valid. (c) Meta-test.

Figure 4.4: Task assignment to the learned subspaces in 5-way 1-shot on *Meta-Dataset-BTAF* (K selected by meta-validation set is 2).

Table 4.7: Accuracy (with 95% confidence interval) of 5-way 5-shot classification on *Meta-Dataset-ABF*. Results marked with † are from [180].

	Aircraft	*Bird*	*Fungi*	average
MAML† [35]	67.82 ± 0.65	70.55 ± 0.77	53.20 ± 0.82	63.86
ProtoNet[140]	69.74 ± 0.64	71.46 ± 0.69	55.66 ± 0.68	65.62
ANIL [119]	69.24 ± 0.87	70.34 ± 1.20	53.71 ± 0.67	64.43
BMG [38]	69.75 ± 0.72	73.04 ± 0.77	54.61 ± 0.84	65.80
DPMM [65]	70.22 ± 0.69	73.28 ± 1.33	54.28 ± 1.01	66.26
TSA-MAML† [180]	72.84 ± 0.63	74.80 ± 0.76	56.86 ± 0.67	68.17
HSML† [167]	69.89 ± 0.90	68.99 ± 1.01	53.63 ± 1.03	64.17
ARML [168]	70.20 ± 0.91	69.12 ± 1.01	54.23 ± 1.07	64.52
TSA-ProtoNet [180]	74.42 ± 0.62	75.11 ± 0.72	56.77 ± 0.69	68.77
MUSML	**79.88** ± 0.61	**75.63** ± 0.73	**57.80** ± 0.80	**71.10**

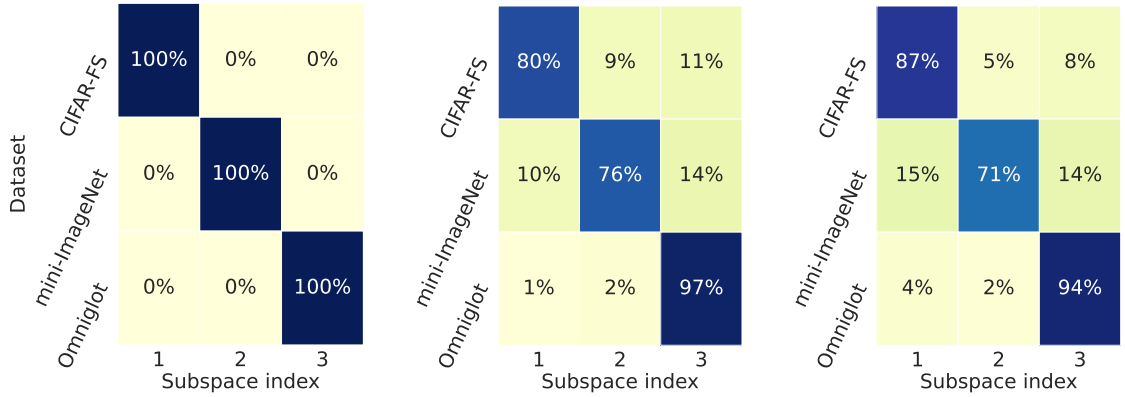
Table 4.8: Accuracy (with 95% confidence interval) of 5-way 5-shot classification on *Meta-Dataset-CIO*.

	CIFAR-FS	*mini-ImageNet*	*Omniglot*	average
MAML [35]	66.28 ± 1.61	60.20 ± 1.20	96.91 ± 0.39	74.46
ProtoNet [140]	71.32 ± 1.54	62.90 ± 1.07	95.32 ± 0.25	76.51
ANIL [119]	66.08 ± 0.90	60.62 ± 0.94	97.13 ± 0.13	74.61
BMG [38]	70.49 ± 1.22	63.97 ± 1.19	**97.92** ± 0.42	77.46
DPMM [65]	69.84 ± 1.42	62.92 ± 1.28	97.14 ± 0.28	76.63
TSA-MAML [180]	71.11 ± 1.55	62.57 ± 1.31	96.99 ± 0.31	76.89
HSML [167]	69.24 ± 1.57	62.28 ± 1.23	95.10 ± 0.32	75.54
ARML [168]	68.88 ± 1.91	63.26 ± 1.33	96.23 ± 0.31	76.12
TSA-ProtoNet [180]	72.37 ± 1.46	63.23 ± 1.52	96.21 ± 0.33	77.27
MUSML	**73.25** ± 1.42	**65.12** ± 1.48	95.13 ± 0.28	**77.83**

may be due to that *Omniglot* is a simple dataset and a single meta-model is good enough. As shown in Figure 4.5, its meta-validation and meta-testing tasks are often assigned to

Table 4.9: Accuracy of cross-domain 5-way 5-shot classification (*Meta-Dataset-BTAF* → *Meta-Dataset-CIO*).

MAML	ProtoNet	ANIL	BMG	DPMM	TSA-MAML	HSML	ARML	TSA-ProtoNet	MUSML
64.25	66.13	65.19	66.98	66.73	66.85	65.18	65.37	66.92	**67.41**



(a) Meta-train. (b) Meta-valid. (c) Meta-test.

Figure 4.5: Task assignment to the learned subspaces in 5-way 5-shot setting on *Meta-Dataset-CIO* (K selected by the meta-validation set is 3). Darker color indicates higher percentage.

the same subspace.

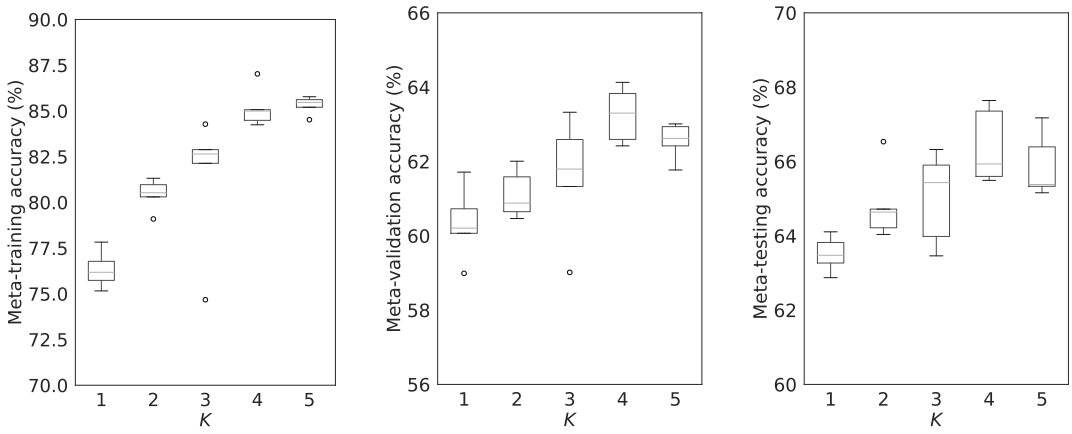
4.5.4 Cross-Domain Few-Shot Classification

We examine the effectiveness of MUSML on cross-domain few-shot classification, which is more challenging as the testing domain is unseen at meta-training. We perform 5-way 5-shot classification, where *Meta-Dataset-BTAF* is used for meta-training, and *Meta-Dataset-CIO* for meta-testing. Table 4.9 shows the meta-testing accuracy. As can be seen, MUSML is also effective on unseen domains.

4.5.5 Effects of K and m

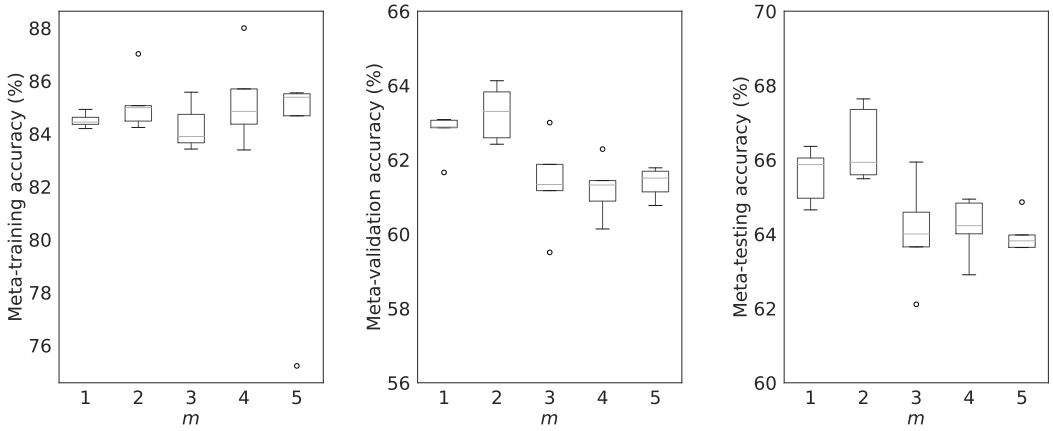
In this experiment, we study the effects of K and m on the 5-shot performance of MUSML on *Meta-Dataset-BTAF*. Figure 4.6(a) shows that the meta-training accuracy increases with K. However, a large $K = 5$ is not advantageous at meta-validation (Figure 4.6(b)) and meta-testing (Figure 4.6(c)).

Figures 4.7(b) and 4.7(c) show that the meta-validation and meta-testing accuracies of MUSML increase when m increases from 1 to 2, but larger m's ($m = 3, 4, 5$) lead to worse performance. This is because the obtained task model parameters ($\mathbf{W}$) lie



(a) Meta-train. (b) Meta-valid. (c) Meta-test.

Figure 4.6: 5-way 5-shot classification accuracy on *Meta-Dataset-BTAF* with varying K (m is fixed at 2).



(a) Meta-train. (b) Meta-valid. (c) Meta-test.

Figure 4.7: 5-way 5-shot classification accuracy on *Meta-Dataset-BTAF* with varying m (K is fixed at 4).

close to the union of 2-dimensional subspaces[2], and so a larger m does not improve performance. Figure 4.8 also shows that for the 4 subspaces, the first 2 singular values of $\mathbf{W}$ are dominant.

To demonstrate the theoretical results in Section 4.3, we further study the effects of K and m on the meta-testing loss. The average training (resp. testing) loss of meta-testing tasks is an empirical estimate of $\hat{\mathcal{R}}(\mathcal{S})$ (resp. $\mathcal{R}(\mathcal{S})$), while their gap measures the generalization performance.

Figure 4.9(a) shows that, for $m \geq 2$, increasing K leads to a reduction in the training loss. However, the testing loss does not always decrease when K increases (Figure

[2]For example, for the $\mathbf{W}$ solution obtained with $m = 5$ on *Meta-Dataset-BTAF* (under 5-way 5-shot setting), approximation by a rank-2 matrix $\hat{\mathbf{W}}$ leads to a relative error ($\|\mathbf{W} - \hat{\mathbf{W}}\|_{\mathrm{F}} / \|\mathbf{W}\|_{\mathrm{F}}$) of only 4.1%.

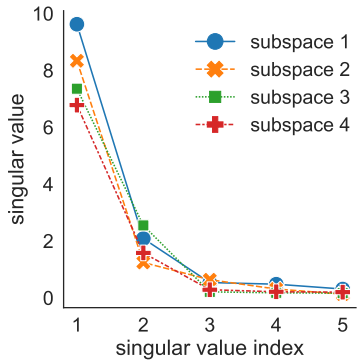
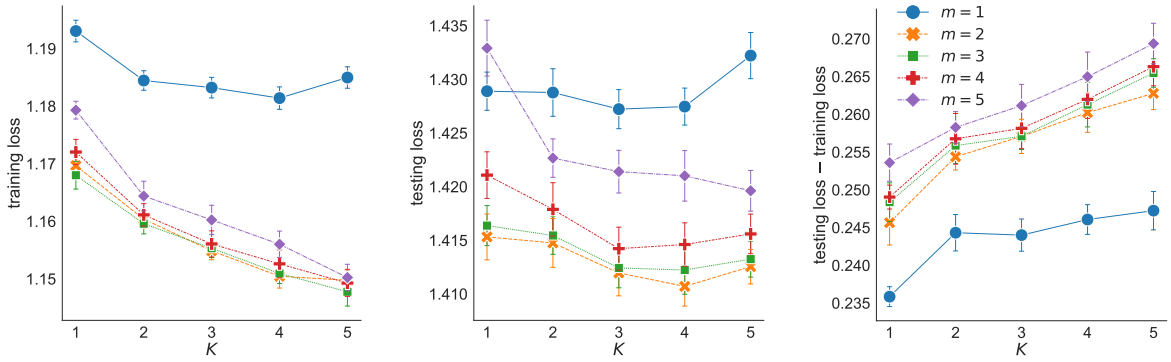


Figure 4.8: Singular values of model parameters of meta-testing tasks under the 5-way 5-shot setting on *Meta-Dataset-BTAF* ($K = 4$ and $m = 5$).



(a) Training loss. (b) Testing loss. (c) Generalization gap.

Figure 4.9: Effects of K and m on the training loss, testing loss, and generalization gap (with 95% confidence interval) of meta-testing tasks under the 5-way 5-shot setting on *Meta-Dataset-BTAF*.

4.9(b)). Figure 4.9(c) shows that a large K or m may enlarge the generalization gap, which justifies Theorem 4.3.1. As shown in Figure 4.9(c), the generalization gap is approximately linear with K, which agrees with the relationship between the upper bound of $\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S})$ and K in Theorem 4.3.1.

4.5.6 Effects of Temperature Scaling Schedule

The temperature schedule used is linear annealing as in DynamicConvolution [19] and ProbMask [181]. We conduct a 5-way 5-shot experiment on *Meta-Dataset-BTAF* to evaluate MUSML with a constant temperature. Table 4.10 reports the meta-testing

Table 4.10: Accuracy of 5-way 5-shot classification on *Meta-Dataset-BTAF*.

γ	0.0001	0.001	0.01	0.1	1.0	2.0	MUSML
accuracy	51.22	60.12	61.15	63.16	62.11	62.02	**66.18**

accuracy. We can see that using a constant γ is inferior.

4.5.7 Improving Existing Meta-Learning Approaches

As the proposed MUSML is general, a subspace mixture is also useful for other meta-learning approaches. In this experiment, we combine MUSML with Meta-Curvature [109] and Meta-SGD [86]. Table 4.11 reports 5-way 5-shot accuracies on meta-datasets. As can be seen, MUSML is beneficial for both Meta-Curvature and Meta-SGD.

Table 4.11: Accuracy of 5-way 5-shot classification on meta-datasets.

	Meta-Dataset-BTAF	*Meta-Dataset-ABF*	*Meta-Dataset-CIO*
Meta-SGD [86]	58.93	64.19	75.95
MUSML-SGD	65.72	69.15	77.48
Meta-Curvature [109]	60.02	64.51	76.13
MUSML-Curvature	**66.10**	**69.23**	**77.96**

4.6 Conclusion

In this chapter, we formulate task model parameters into a subspace mixture and propose a model-agnostic meta-learning algorithm with subspace learning called MUSML. For each task, the base learner builds a task model from each subspace, while the meta-learner updates the meta-parameters by minimizing a weighted validation loss. The generalization performance is theoretically studied. Experimental results on benchmark datasets for classification and regression validate the effectiveness of the proposed MUSML.

CHAPTER 5

Structured Prompting by Meta-Learning and Representative Verbalizer

Meta-learning is a general machine learning tool and has been successfully used in various applications, such as computer vision [35, 140, 8, 113, 72, 164, 157] and natural language processing [106, 169, 14, 52, 163, 170, 85]. This section introduces an application of meta-learning in prompt tuning.

5.1 Introduction

In recent years, large pre-trained language models have achieved great success in solving a variety of downstream tasks [58, 26, 166, 21, 141, 49, 117, 12, 81, 22]. Though fine-tuning the whole model [58, 26] is effective and widely-used, optimizing and storing all the task-specific parameters can be compute- and memory-expensive when the model is large (e.g., GPT-3 [12] contains 100+ billion parameters). To alleviate this issue, many approaches have been proposed. Examples include adapter tuning [57, 88, 59] and prompt learning [116, 137, 12, 81, 94, 84, 95, 114, 92]. However, prompt learning is more preferable due to its effectiveness and also that it can be easily plugged into a pre-trained MLM without invasive modification [84, 51, 55, 142].

Prompt learning formulates the downstream task as a cloze-style MLM problem. It is useful for few-shot tasks due to its effectiveness, parameter-efficiency, and plug-and-play nature [116, 12, 92]. Specifically, prompt learning wraps an input text with a discrete *prompt* (e.g., "`Topic is [MASK]`") and feeds it to the MLM to predict a token at the `[MASK]` position. A *verbalizer* [81, 27, 60] then maps the predicted token to the label. However, designing an effective prompt requires a good understanding of the downstream tasks.

Recently, *prompt tuning* [81, 94, 175] proposes to wrap the input embedding with a *continuous* prompt. To reduce the number of parameters to be learned, the MLM is kept frozen. The continuous prompt can be further combined with discrete tokens to form a *template* [94, 133, 27].

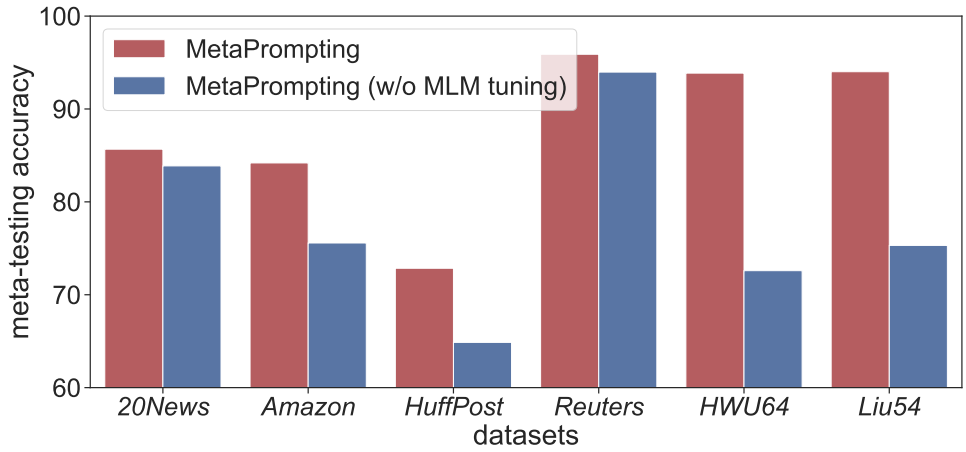


Figure 5.1: 5-way 5-shot classification meta-testing accuracy of MetaPrompting with or without MLM tuning on six data sets.

Prompt tuning can be sensitive to initialization [81]. Recently, a number of approaches have been proposed to alleviate this problem [81, 83, 153]. In particular, MetaPrompting [56] is the state-of-the-art that uses *meta-learning* [145, 35] to learn a meta-initialization for all task-specific prompts. However, MetaPrompting suffers from three problems. (i) When the tasks are complex, it is challenging to obtain good prompts for all tasks and samples from a single meta-initialized prompt. (ii) MetaPrompting uses a hand-crafted verbalizer. However, selecting good label tokens for the hand-crafted verbalizer is labor-intensive and not scalable for a large label set. (iii) MetaPrompting requires expensive tuning the whole MLM. Figure 5.1 shows a large gap in meta-testing accuracies with and without MLM tuning (experimental details are in Section 5.3).

In this chapter, we use a pool of multiple prompts [83, 158, 159] to extract task knowledge from meta-training tasks, and then construct instance-dependent prompts as weighted combinations of all the prompts in the pool via attention [151]. The attention's query vector is the instance's feature embedding. The prompt pool is the shared meta-knowledge and learned by the MAML algorithm [35]. Specifically, given a task with a support set and a query set, the base learner takes the meta-parameter and the support set to build a task-specific prompt pool, then the meta-learner optimizes the meta-parameter on the query set. Meta-learning a prompt pool is more flexible than meta-learning only a single prompt initialization (as in MetaPrompting), and allows better adaptation of complex tasks. Moreover, as only the prompt pool is tuned, it is much more parameter-efficient than MetaPrompting (with 1000$\times$ fewer parameters).

We also propose a novel soft verbalizer called *representative verbalizer* (RepVerb), which constructs label embeddings by averaging feature embeddings of the corresponding

training samples. Unlike manually-designed verbalizers, RepVerb does not incur human effort for label token selection. Moreover, as RepVerb does not require learning any additional parameters, empirical results in Section 5.3.2 demonstrate that RepVerb is more effective than the soft verbalizers in WARP [51], DART [175], ProtoVerb [22]. Besides, the feature embedding learned by RepVerb is more discriminative.

The whole procedure, which combines meta-learning the structured prompts and RepVerb, is called **MetaPrompter** in the sequel. Experiments are performed on six widely used classification data sets. Results demonstrate that RepVerb outperforms existing soft verbalizers, and is also beneficial to other prompt-based methods such as MetaPrompting. Moreover, MetaPrompter achieves better performance than the recent state-of-the-arts.

Our contributions are summarized as follows: (i) We propose a parameter-efficient algorithm MetaPrompter for effective structured prompting. (ii) We propose a simple and effective soft verbalizer (RepVerb). (iii) Experimental results demonstrate the effectiveness and parameter-efficiency of MetaPrompter.

5.2 The Proposed MetaPrompter

In this section, we propose a simple and effective soft verbalizer (representative verbalizer) without inducing additional parameters (Section 5.2.1). Moreover, while MetaPrompting uses a single meta-initialized prompt to build task-specific prompts, we propose in section 5.2.2 the extraction of task knowledge into a pool of multiple prompts, and constructs instance-dependent prompts by attention [151].

5.2.1 Representative Verbalizer (RepVerb)

Instead of explicitly learning an embedding $\mathbf{v}_y$ for each label y [51, 22, 175], we propose the *Representative Verbalizer* (RepVerb), which constructs $\mathbf{v}_y$ from feature embeddings of the corresponding training samples (Algorithm 5). It does not require learning additional parameters, and is thus more effective on limited data as in few-shot learning.

Specifically, let $\mathcal{S}_{\tau,y}$ be the subset of samples in $\mathcal{S}_\tau$ with label y. For an input $\mathbf{x}$, we wrap it with the template and feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta})$ to the pre-trained MLM, and then obtain `[MASK]`'s embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ as its feature embedding. Similar to ProtoNet [140], we

propose to construct $\mathbf{v}_y$ for each y by averaging the corresponding samples' feature embeddings, as:

$$\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}). \tag{5.1}$$

To predict the label of a given $\mathbf{x}$, we measure the cosine similarity[1] between $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ and each $\mathbf{v}_y$ ($y \in \mathcal{Y}_\tau$):

$$\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \frac{\exp(\rho\cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y'\in\mathcal{Y}_\tau}\exp(\rho\cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}, \tag{5.2}$$

where $\rho > 0$ is the temperature. When $\rho \to \infty$, $\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta})$ becomes one-hot; whereas when $\rho \to 0$, $\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta})$ becomes uniform. In the experiments, we set $\rho = 10$ as in Oreshkin et al. [107].

Algorithm 5 Representative Verbalizer (RepVerb).

procedure COMPUTE_LABEL_EMB($\mathcal{S}_\tau$):
    compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $(\mathbf{x},\cdot) \in \mathcal{S}_\tau$;
    $\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|}\sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $y \in \mathcal{Y}_\tau$;
end procedure
procedure PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$)
    compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $\mathbf{x}$;
    $\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \frac{\exp(\rho\cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y'\in\mathcal{Y}_\tau}\exp(\rho\cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}$;
end procedure

5.2.2 Meta Structured-Prompting

In the following, we propose the use of MAML and attention mechanism [151] to meta-learn a prompt pool. While MetaPrompting uses task-specific prompts [56], we propose the construction of instance-specific prompts, which allows more flexibility.

Meta-Learn a Prompt Pool

While MetaPrompting uses only a single initialization for the prompt, we propose to leverage a pool of prompts to extract more task knowledge, which is particularly effective when the tasks are complex. A prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1,\ldots,K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p\times d_i}$ [83, 158, 159]. Note that the size of the prompt pool is negligible compared with that of the MLM. For example, in our

[1] Dissimilarity measures, such as the Euclidean distance, can also be used.

experiments, the MLM has 109.52×10^6 parameters, while the prompt pool has only $55,296$.

The prompt pool can be considered as shared meta-knowledge. Given an input $\mathbf{x}$, the attention weights between $\mathbf{x}$ and the K prompts are computed as $\mathbf{a} = \text{softmax}(\frac{\mathbf{K}\mathbf{q}_\mathbf{x}}{\sqrt{d_o}})$, where $\mathbf{K} = [\mathbf{k}_1^\top; \ldots; \mathbf{k}_K^\top]$, and $\mathbf{q}_\mathbf{x} \in \mathbb{R}^{d_o}$ is the embedding of the `[MASK]` output by a pre-trained and frozen MLM with the wrapped input (e.g., ($\mathbf{x}$. `Topic is [MASK]`)) [158, 159]. Such mapping from $\mathbf{x}$ to $\mathbf{q}_\mathbf{x}$ is called a query function $q(\cdot)$. An instance-dependent prompt is then generated by weighted averaging over all the values ($\boldsymbol{\theta}_i$'s):

$$\boldsymbol{\theta}_\mathbf{x}(\mathbf{K}, \boldsymbol{\Theta}) = \sum_{i=1}^{K} a_i \boldsymbol{\theta}_i, \tag{5.3}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1; \ldots; \boldsymbol{\theta}_K]$. While [158, 159] only selects the top-N most similar prompts from the pool, in (5.3) all the prompts are used and updated simultaneously.

The proposed procedure, which will be called MetaPrompter, is shown in Algorithm 6. At iteration t, the base learner takes $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ and a task τ to optimize for a task-specific prompt pool by gradient descent (steps 4-15). $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ is used as the initialization (step 4). For each inner iteration j, $(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ constructs the instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ in (5.3) (steps 7 and 8). Next, $\boldsymbol{\theta}_{\mathbf{x},j}$ is used to predict the label probability with a combination of the hand-crafted verbalizer (step 9) and soft verbalizer (steps 11 and 12):

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) = (1-\lambda)\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) + \lambda\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}), \tag{5.4}$$

where $\lambda \in [0, 1]$. Let $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}\right)$ be the loss on $\mathcal{S}_\tau$ (step 13). The base learner builds a task-specific prompt pool $(\mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J})$ by taking J gradient updates ($j = 1, \ldots, J$) at step 14:

$$(\mathbf{K}_{t,j}, \boldsymbol{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1}) - \alpha\nabla_{(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})}\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1}).$$

The meta-learner takes $(\mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J})$ and $\mathcal{Q}_\tau$ to update the meta-parameters (steps 17-24). For $(\mathbf{x}, y) \in \mathcal{Q}_\tau$, we use $(\mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J})$ to generate its prompt $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J})$ (steps 18 and 19), which is used for make prediction $\mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J}\right)$ (steps 20 and 21). Let $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J}\right)$ be the negative log-likelihood loss on $\mathcal{Q}_\tau$ (step 23). The meta-learner updates meta-parameters by performing one gradient update on $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J})$ at step 24:

$$(\mathbf{K}_t, \boldsymbol{\Theta}_t) = (\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1}) - \eta\nabla_{(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \boldsymbol{\Theta}_{t,J}).$$

Algorithm 6 MetaPrompter.

Require: prompt length L_p; size of prompt pool K; $\lambda = 0.5$; step size α, η; meta-parameters $(\mathbf{K}, \mathbf{\Theta})$; query function $q(\cdot)$;

1: **for** $t = 1, \ldots, T$ **do**
2: sample a task $\tau = (\mathcal{S}_\tau, \mathcal{Q}_\tau) \in \mathcal{T}$;
3: *base learner:*
4: $(\mathbf{K}_{t,0}, \mathbf{\Theta}_{t,0}) \equiv (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})$;
5: **for** $j = 1, \ldots, J$ **do**
6: **for** $(\mathbf{x}, y) \in \mathcal{S}_\tau$ **do**
7: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
8: $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = \mathsf{softmax}(\mathbf{K}_{t,j-1}\mathbf{q}_\mathbf{x}/\sqrt{d_0})^\top \mathbf{\Theta}_{t,j-1}$;
9: feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ into $\mathcal{M}$, obtain $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$, and $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (2.16);
10: **end for**
11: call COMPUTE_LABEL_EMB($\mathcal{S}_\tau$) of Algorithm 5 to obtain $\{\mathbf{v}_y : y \in \mathcal{Y}_\tau\}$;
12: for $(\mathbf{x}, y) \in \mathcal{S}_\tau$, call PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 5 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$, and compute $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (5.4);
13: $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$;
14: $(\mathbf{K}_{t,j}, \mathbf{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) - \alpha \nabla_{(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})} \mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})$;
15: **end for**
16: *meta-learner:*
17: **for** $(\mathbf{x}, y) \in \mathcal{Q}_\tau$ **do**
18: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
19: $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \mathsf{softmax}(\mathbf{K}_{t,J}\mathbf{q}_\mathbf{x}/\sqrt{d_0})^\top \mathbf{\Theta}_{t,J}$;
20: call PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 5 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
21: compute $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ by (2.16) and (5.4), respectively;
22: **end for**
23: $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
24: $(\mathbf{K}_t, \mathbf{\Theta}_t) = (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1}) - \eta \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$;
25: **end for**
26: **return** $(\mathbf{K}_T, \mathbf{\Theta}_T)$.

The meta-gradient

$$\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) \nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} (\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$$

requires back-propagating through the entire inner optimization path, which is computationally infeasible for large models and J is large. To reduce computation, we discard the second-order derivative and use the first-order approximation $\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) \approx \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ (step 24) as in [35, 56].

Meta-Testing. Given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, the base learner takes $\mathcal{S}_{\tau'}$ and $(\mathbf{K}_T, \mathbf{\Theta}_T)$ to build a task-specific prompt pool $(\mathbf{K}_{T,J}, \mathbf{\Theta}_{T,J})$ as in steps 4-15. This pool is then used to construct instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},J}$ for each $(\mathbf{x}, \cdot) \in \mathcal{Q}_{\tau'}$. The MLM receives the wrapped input $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and predicts the label probability by (5.4).

MetaPrompter is Parameter-Efficient. As MetaPrompter only tunes $(\mathbf{K}, \mathbf{\Theta})$, the total number of meta-parameters is $K(d_o + L_p d_i)$ (where d_i and d_o are the dimensions of the input and feature embeddings, respectively). This is much smaller[2] than that of MetaPrompting (which is equal to $d_{\boldsymbol{\phi}} + L_p d_i$, where $d_{\boldsymbol{\phi}}$ is the size of $\boldsymbol{\phi}$), which requires tuning the whole MLM.

5.3 Experiments

5.3.1 Setup

Data sets. Following [14], we perform few-shot classification on six popularly used data sets: (i) *20News* [77], which contains informal discourses from news discussion forums of 20 topics; (ii) *Amazon* [54] consists of customer reviews from 24 products. The task is to classify reviews into product categories; (iii) *HuffPost* [100], which contains news headlines of 41 topics published in the HuffPost between 2012 and 2018. These headlines are shorter and less grammatical than formal sentences, thus are more challenging for classification; (iv) *Reuters* [82] is a collection of Reuters newswire articles of 31 topics from 1996 to 1997; (v) *HWU64* [96] is an intent classification data set, containing user utterances of 64 intents; (vi) *Liu54* [96] is an imbalanced intent classification data set of 54 classes collected on Amazon Mechanical Turk. We use the meta-training/meta-validation/meta-testing splits provided in [14]. A summary of data sets is in Table 5.1.

Following [6, 52, 14, 56], we perform experiments in the 5-way 1-shot and 5-way 5-shot settings with 15 query samples per class. The pre-trained BERT (*bert-base-uncased*) from HuggingFaces [161] is used as the pre-trained MLM as [14, 56]. Experiments are run on a DGX station with 8 V100 32GB GPUs. The experiment is repeated three times with different random seeds.

5.3.2 Evaluation on RepVerb

First, we compare the performance of the proposed RepVerb with the state-of-the-art soft verbalizers of: (i) WARP [51][3], and (ii) ProtoVerb [22]. As the focus is on evaluating

[2]For example, $d_o = d_i = 768$, $d_{\boldsymbol{\phi}} = 109 \times 10^6$ in BERT. Moreover, Both K and L_p are 8 in the experiments.

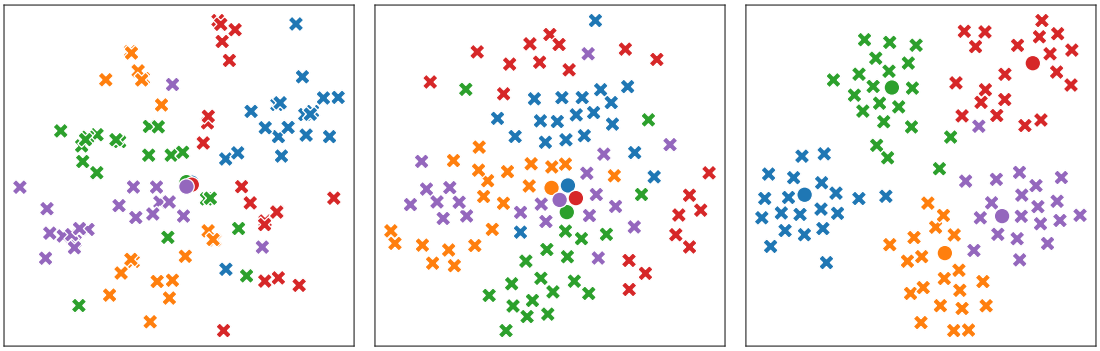
[3]Note that the vebalizer of WARP is the same as that of DART [175].

Table 5.1: Statistics of the data sets.

	#classes (meta-train/valid/test)	#samples	#tokens per sample (mean ± std)
20News	8/5/7	18,820	340 ± 151
Amazon	10/5/9	24,000	140 ± 32
HuffPost	20/5/16	36,900	11 ± 4
Reuters	15/5/11	620	168 ± 136
HWU64	23/16/25	11,036	7 ± 3
Liu54	18/18/18	25,478	8 ± 4

Table 5.2: Meta-testing accuracy of 5-way few-shot classification.

		20News	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
5-shot	WARP [51]	61.43 ± 0.15	59.53 ± 0.20	46.31 ± 0.31	68.67 ± 0.71	68.60 ± 0.40	73.11 ± 0.26
	ProtoVerb [22]	71.33 ± 0.11	71.74 ± 0.21	57.93 ± 0.17	80.93 ± 0.54	73.43 ± 0.51	76.19 ± 0.33
	RepVerb	**78.81** ± 0.08	**77.56** ± 0.16	**61.90** ± 0.08	**88.33** ± 0.40	**78.37** ± 0.49	**82.14** ± 0.23
1-shot	WARP [51]	49.87 ± 0.63	48.94 ± 0.34	38.21 ± 0.35	52.88 ± 0.67	53.20 ± 0.76	58.68 ± 0.64
	ProtoVerb [22]	54.13 ± 0.46	55.07 ± 0.27	41.40 ± 0.21	57.27 ± 0.73	55.17 ± 0.81	60.16 ± 0.37
	RepVerb	**59.86** ± 0.38	**59.18** ± 0.31	**44.65** ± 0.20	**63.63** ± 0.41	**59.83** ± 0.71	**66.17** ± 0.40



(a) WARP. (b) ProtoVerb. (c) RepVerb.

Figure 5.2: t-SNE visualization of `[MASK]`'s embeddings (crosses) and label embeddings (circles) for a 5-way 5-shot task randomly sampled from *Reuters*.

verbalizers, all methods use the same discrete prompt "`Topic is [MASK]`", and fine-tune all parameters for 5 steps with a learning rate of 0.00005 as in [22].

Results. Table 5.2 reports the meta-testing accuracies. As can be seen, RepVerb outperforms WARP and ProtoVerb on both the 1-shot and 5-shot settings.

For a 5-way 5-shot task randomly from *Reuters*, Figure 5.2 shows the t-SNE visualization of the embeddings ($\mathbf{h}_{\texttt{[MASK]}}(\mathbf{x})$'s) of 100 samples ($\mathbf{x}$'s)[4] and learned label embeddings ($\mathbf{v}_y$'s). As can be seen, the RepVerb embedding is more discriminative and compact than WARP and ProtoVerb. Moreover, by design, RepVerb's label embedding is consistent with the samples' feature embeddings, while those of WARP and ProtoVerb are not.

[4] 5-way × (5 support samples + 15 query samples) = 100.

5.3.3 Evaluation on MetaPrompter

For MetaPrompter, hyperparameters K and L_p are chosen from $\{1, 2, 4, 8, 16, 32, 64\}$ using the meta-validation set. For the base learner, $\alpha = 0.1$, and $J = 5$ (resp. 15) at meta-training (resp. meta-validation or meta-testing). We train the prompt pool for $T = 3,000$ iterations using the Adam optimizer [73] with a learning rate of 0.001. To prevent overfitting, we evaluate the meta-validation performance every 100 iterations and choose the checkpoint with the best meta-validation performance for meta-testing. For the hard verbalizer, label tokens are obtained by tokenizing the class name and its synonyms as in [56, 60]. Following [81], prompts are initialized from input embeddings of randomly sampled label tokens for both MetaPrompting and MetaPrompter.

We compare with a variety of baselines. These include state-of-the-art prompt-based methods of (i) MetaPrompting [56], and its variants (ii) MetaPrompting+WARP / MetaPrompting+ProtoVerb / MetaPrompting+RepVerb, which combine meta-prompting with the the soft verbalizer of WARP / ProtoVerb / RepVerb, respectively. Moreover, we also compare with the non-prompt-based methods of: (iii) HATT [41], which meta-learns a prototypical network [140] with a hybrid attention mechanism; (iv) DS [6], which learns attention scores based on word frequency; (v) MLADA [52], which uses an adversarial domain adaptation network to extract domain-invariant features during meta-training; and (vi) ConstrastNet [14], which performs feature extraction by contrastive learning.

Results. Table 5.3 shows the number of parameters and meta-testing accuracy in the 5-shot setting. As can be seen, MetaPrompter is more accurate than both prompt-based and non-prompt-based baselines. Moreover, since MetaPrompter only tunes the prompt pool and keeps the language model frozen, it has much fewer meta-parameters than MetaPrompting and ConstrastNet.

Table 5.3: 5-way 5-shot classification meta-testing accuracy. Results marked with † are from [14]. "–" indicates that the corresponding result is not reported in [14].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [41]	0.07	55.00	66.00	56.30	56.20	-	-
DS† [6]	1.73	68.30	81.10	63.50	96.00	-	-
MLADA† [52]	0.73	77.80	86.00	64.90	96.70	-	-
ConstrastNet† [14]	109.52	71.74	85.17	65.32	95.33	92.57	93.72
MetaPrompting [56]	109.52	85.67 ± 0.44	84.19 ± 0.30	72.85 ± 1.01	95.89 ± 0.23	93.86 ± 0.97	94.01 ± 0.26
MetaPrompting+WARP	109.52	85.81 ± 0.48	85.54 ± 0.20	71.71 ± 0.72	97.28 ± 0.30	93.99 ± 0.76	94.33 ± 0.27
MetaPrompting+ProtoVerb	109.52	86.18 ± 0.51	84.91 ± 0.38	73.11 ± 0.80	97.24 ± 0.25	93.81 ± 0.81	94.38 ± 0.18
MetaPrompting+RepVerb	109.52	86.89 ± 0.39	85.98 ± 0.28	74.62 ± 0.88	97.32 ± 0.31	94.23 ± 0.67	94.45 ± 0.33
MetaPrompter	0.06	**88.57** ± 0.38	**86.36** ± 0.24	**74.89** ± 0.75	**97.63** ± 0.22	**95.30** ± 0.51	**95.47** ± 0.21

Table 5.4: 5-way 1-shot classification meta-testing accuracy. Results marked with † are from [14]. "–" indicates that the corresponding result is not reported in [14].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [41]	0.07	44.20	49.10	41.10	43.20	-	-
DS† [6]	1.73	52.10	62.60	43.00	81.80	-	-
MLADA† [52]	0.73	59.60	68.40	64.90	82.30	-	-
ConstrastNet† [14]	109.52	71.74	76.13	53.06	86.42	86.56	85.89
MetaPrompting [56]	109.52	82.46 ± 0.50	76.92 ± 0.77	68.62 ± 0.56	92.56 ± 0.77	91.06 ± 0.41	87.79 ± 0.29
MetaPrompting +WARP	109.52	82.93 ± 0.39	78.27 ± 0.72	67.78 ± 0.41	94.74 ± 0.56	91.30 ± 0.35	88.69 ± 0.26
MetaPrompting+ProtoVerb	109.52	83.15 ± 0.41	78.19 ± 0.65	68.96 ± 0.52	95.26 ± 0.40	91.27 ± 0.63	90.05 ± 0.15
MetaPrompting+RepVerb	109.52	84.13 ± 0.30	78.59 ± 0.43	$\mathbf{69.02} \pm 0.51$	95.78 ± 0.33	91.32 ± 0.44	90.13 ± 0.20
MetaPrompter	0.06	$\mathbf{84.62} \pm 0.29$	$\mathbf{79.05} \pm 0.21$	67.12 ± 0.23	$\mathbf{96.34} \pm 0.20$	$\mathbf{92.11} \pm 0.30$	$\mathbf{93.72} \pm 0.18$

Furthermore, MetaPrompting+RepVerb performs better than MetaPrompting+WARP and MetaPrompting+ProtoVerb, demonstrating that the proposed RepVerb is beneficial to MetaPrompting.

Table 5.4 shows the number of parameters and meta-testing accuracy in 5-way 1-shot setting. As can be seen, the state-of-the-art prompt-based methods always achieve higher accuracy than the non-prompt-based. Furthermore, MetaPrompter performs the best on 5 of the 6 data sets. Besides, RepVerb is again useful to MetaPrompting on all six data sets.

5.3.4 Visualization

In this section, we visualize the meta-knowledge in the prompt pool learned from the 5-way 5-shot classification task on *Reuters*. Table 5.5 shows the nearest tokens to each of the K ($= 8$) learned prompts. Figure 5.3 shows the average attention weights between the K prompts and meta-training samples belonging to class (topic) y:

$$\frac{1}{|\mathcal{T}_y|} \sum_{\tau \in \mathcal{T}_y} \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y) \in \mathcal{S}_{\tau,y}} \text{softmax}\left(\frac{\mathbf{K}_{T,J}\mathbf{q}_\mathbf{x}}{\sqrt{d_o}}\right),$$

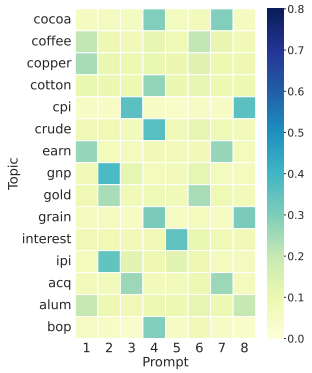


Figure 5.3: Distribution of attention weights on 5-way 5-shot classification of *Reuters* (15 topics).

Table 5.5: Nearest tokens to the learned prompts for *Reuters*.

prompt id	nearest tokens
1	copper, steel, trading, gas, fx, aluminum, earn, coffee
2	gross, ship, index, money, gold, tin, iron, retail
3	product, cpi, industrial, acquisitions, jobs, supplying, orange, sugar
4	cocoa, production, grain, livestock, wholesale, cotton, bop, crude
5	oil, national, rubber, nat, interest, price, reserves, regional
6	nat, wholesale, sugar, golden, reserves, drinks, production, product
7	chocolate, sugar, cheat, orange, trade, fx, cash, acquiring
8	aluminum, livestock, cpc, tin, shops, wheat, petrol, supply

where $\mathcal{T}_y$ is the subset of tasks in $\mathcal{T}$ having class y. As can be seen, samples from each target class prefer prompts whose tokens are related to that class. For example, samples from the topic *cocoa* tend to use the 4th and 7th prompts (whose tokens are close to words like *cocoa*, *chocolate* as can be seen from Table 5.5), while samples from the topic *coffee* tend to use the 1st and 6th prompts (whose tokens are close to words like *coffee* and *sugar*.

Recall that the prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \ldots, K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p \times d_i}$. Let $\boldsymbol{\theta}_i^{(j)}$ be the jth row of $\boldsymbol{\theta}_i$. Moreover, let $\frac{1}{|\mathcal{V}_y|} \sum_{\mathtt{w} \in \mathcal{V}_y} \mathcal{E}(\mathtt{w})$ be the embedding of topic (class) y, where $\mathcal{V}_y$ is a set of tokens relevant to label y (obtained from Hou et al. [56]), and $\mathcal{E}(\cdot)$ is the input embedding. Figure 5.4 shows the cosine similarities between the learned prompt tokens $\{\boldsymbol{\theta}_i^{(j)} : i = 1, \ldots, K, j = 1 \ldots, L_p\}$ and topic embeddings. As can be seen, embedding of *cocoa* is close to $\boldsymbol{\theta}_4^{(1)}$ and $\boldsymbol{\theta}_7^{(1)}$. Thus, samples from *cocoa* prefer the 4th and 7th prompts (Figure 5.3). Similarly, embedding of *coffee* is close to $\boldsymbol{\theta}_1^{(8)}$ and $\boldsymbol{\theta}_6^{(6)}$. Thus, samples from *coffee* prefer the 1st and 6th prompts (Figure 5.3).

5.4 Conclusion

In this chapter, we propose MetaPrompter, an effective and parameter-efficient algorithm for prompt tuning. It combines structured prompting and a novel verbalizer called RepVerb. A prompt pool structure is used to construct instance-dependent prompts by attention, while RepVerb builds label embedding by averaging feature embeddings of the corresponding training samples. The pool of prompts is meta-learned from the meta-training tasks. Experimental results demonstrate the effectiveness of the proposed

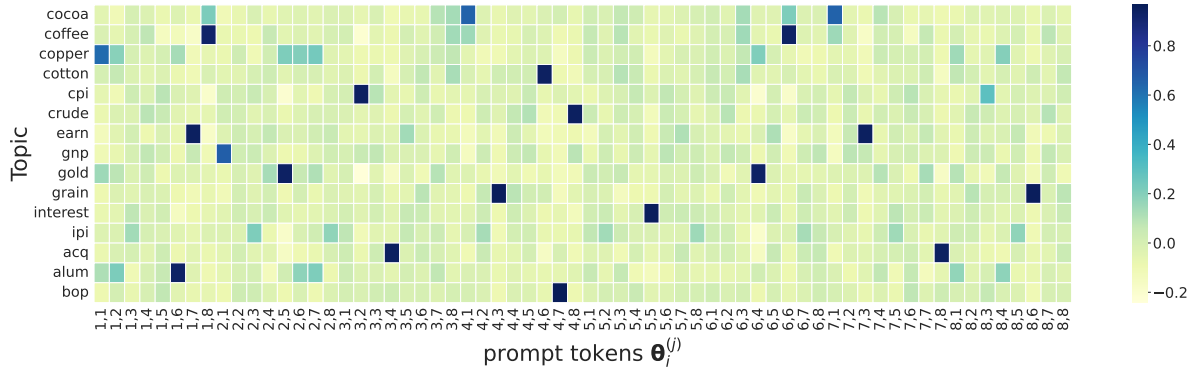


Figure 5.4: Cosine similarities between learned prompt tokens and topic embeddings on 5-way 5-shot classification of *Reuters*. In the x-axis, (i, j) stands for the jth row of $\boldsymbol{\theta}_i$ (i.e., $\boldsymbol{\theta}_i^{(j)}$)

MetaPrompter and RepVerb.

CHAPTER 6

Conclusion and Future Direction

6.1 Conclusion

Meta-learning is used to accelerate learning new tasks from meta-knowledge extracted from historical tasks. In this thesis, we studied the meta-learning problems when tasks are complex. We first extend learning a meta-regularization for linear models to nonlinear models by kernelized extension (Chapter 3). To deal with complex tasks, we formulate task-specific knowledge into a subspace mixture, where each subspace represents one type of knowledge (Chapter 4). As language models are usually large, learning multiple meta-models causes a heavy burden on computation and memory. We further propose to learn a pool of multiple meta-prompts for prompt learning in language models (Chapter 5). Experiments are conducted in each chapter to verify the usefulness of the proposed algorithms. Detailed summaries for each chapter are listed as follows.

1. In Chapter 3, we proposed an efficient algorithm to learn a meta-regularization for *nonlinear* models by kernelized proximal regularization. Nonlinearity allows more powerful models like deep networks to deal with complex tasks. We formulated the inner problem as a dual problem and introduced a learnable proximal regularizer to the base learner. We theoretically established the local and global convergence of the proposed algorithm. Experiments on benchmark regression and classification datasets demonstrate that learning a nonlinear meta-regularizer is effective. Moreover, for regression tasks, the proposed algorithm has a closed-form solution in the base learner and, thus, is very efficient.

2. In Chapter 4, we formulated task model parameters into multiple subspaces and proposed a novel meta-learning algorithm MUSML to learn the subspace bases. MUSML is very general, thus, can be used on linear and nonlinear models. Generalization of the proposed MUSML is analyzed theoretically. Experiments on synthetic demonstrate that MUSML can discover the underlying subspaces of task

model parameters. Empirical results on real-word datasets show that MUSML achieves state-of-the-art performance on complex tasks.

3. In Chapter 5, we proposed using a pool of meta-prompts to extract knowledge from meta-training tasks. We construct instance-dependent prompts by combining the meta-prompts via attention. We propose a novel algorithm MetaPrompter to combine learning a prompt pool with a novel soft verbalizer. Language models are frozen and only the pool is learnable, thus, the proposed MetaPrompter is parameter-efficient. Meta-learning a prompt pool is more flexible than meta-learning only a single prompt initialization (as in MetaPrompting), and allows better adaptation of complex tasks. Experimental results on standard benchmarks demonstrate the effectiveness and parameter-efficiency of the proposed MetaPrompter.

6.2 Future Directions

In the future, we plan to work on using meta-knowledge of LLMs for solving mathematical tasks, which are complex and challenging. Some possible topics are as follows.

1. **Answer verification using LLMs' meta-knowledge.** One can feed a mathematical question to the LLM multiple times and obtain different answers using meta-knowledge of forward reasoning. Existing work proposes choosing the answer received the most votes as the final answer. As Large language models contain diverse meta-knowledge, one research direction is using the meta-knowledge (like backward reasoning) to improve verification. Specifically, we can mask a number in the question and ask the LLM to predict the masked number when a candidate answer is provided. Intuitively, a correct candidate answer is more likely to help predict the masked number than the wrong answers.

2. **Injecting meta-knowledge by data augmentation.** To enhance meta-knowledge of open-source models in solving mathematical tasks, one needs to finetune the models on massive amounts of data. However, creating mathematical questions and answers is labor-intensive and requires expert knowledge. A possible solution is to generate diverse questions based on existing questions and then ask a powerful (closed-source) LLM model to answer the generated questions. For example, a backward question is created by masking a number in the original question and

asking the value of the masked number when an answer is given. Different from the above research topic, which uses backward reasoning for verifying answers, we create backward question-answer pairs here to finetune open-source LLMs for enhancing meta-knowledge of mathematical reasoning.

Bibliography

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Neural Information Processing Systems*, 2019.

[2] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*, 2018.

[3] Sudarshan Babu, Pedro Savarese, and Michael Maire. Online meta-learning via learning with layer-distributed memory. In *Neural Information Processing Systems*, 2021.

[4] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, 2019.

[5] Fan Bao, Guoqiang Wu, Chongxuan Li, Jun Zhu, and Bo Zhang. Stability and generalization of bilevel programming in hyperparameter optimization. In *Neural Information Processing Systems*, 2021.

[6] Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*, 2020.

[7] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. In *International Joint Conference on Neural Networks*, 1991.

[8] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

[9] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, 2006.

[10] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[11] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

[13] Paul H Calamai and Jorge J Moré. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 1987.

[14] Junfan Chen, Richong Zhang, Yongyi Mao, and Jie Xu. ContrastNet: A contrastive learning framework for few-shot text classification. In *AAAI Conference on Artificial Intelligence*, 2022.

[15] Qi Chen, Changjian Shui, and Mario Marchand. Generalization bounds for meta-learning: An information-theoretic analysis. In *Neural Information Processing Systems*, 2021.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[17] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2018.

[18] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[19] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[20] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018.

[21] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Neural Information Processing Systems*, 2019.

[22] Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. Prototypical verbalizer for prompt-based few-shot tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[23] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *Neural Information Processing Systems*, 2018.

[24] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, 2019.

[25] Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. The advantage of conditional meta-learning for biased regularization and fine tuning. In *Neural Information Processing Systems*, 2020.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[27] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. OpenPrompt: An open-source framework for prompt-learning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[28] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[29] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Neural Information Processing Systems*, 2019.

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[31] Benjamin Ehret, Christian Henning, Maria Cervera, Alexander Meulemans, Johannes Von Oswald, and Benjamin F Grewe. Continual learning in recurrent neural networks. In *International Conference on Learning Representations*, 2021.

[32] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[33] Alec Farid and Anirudha Majumdar. Generalization bounds for meta-learning via PAC-Bayes and uniform stability. In *Neural Information Processing Systems*, 2021.

[34] Hongliang Fei and Ping Li. Cross-lingual unsupervised sentiment classification with multi-view transfer learning. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

[35] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[36] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, 2019.

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

[38] Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022.

[39] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018.

[40] Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning*, 2019.

[41] Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *AAAI Conference on Artificial Intelligence*, 2019.

[42] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

[43] Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4): 2341–2368, 2013.

[44] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[45] Shaogang Gong, Stephen McKenna, and John J Collins. An investigation into face pose distributions. In *International Conference on Automatic Face and Gesture Recognition*, 1996.

[46] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, 2020.

[47] Liang-Yan Gui, Yu-Xiong Wang, Deva Ramanan, and José MF Moura. Few-shot human motion prediction via meta-learning. In *European Conference on Computer Vision (ECCV)*, 2018.

[48] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, 2018.

[49] Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

[50] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. SpotTune: transfer learning through adaptive fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[51] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level adversarial reprogramming. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[52] Chengcheng Han, Zeqiu Fan, Dongxiang Zhang, Minghui Qiu, Ming Gao, and Aoying Zhou. Meta-learning adversarial domain adaptation network for few-shot text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[54] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web*, 2016.

[55] Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, 2022.

[56] Yutai Hou, Hongyuan Dong, Xinghao Wang, Bohan Li, and Wanxiang Che. MetaPrompting: Learning to learn better prompts. In *International Conference on Computational Linguistics*, 2022.

[57] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

[58] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2018.

[59] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[60] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[61] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Neural Information Processing Systems*, 2018.

[62] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

[63] Adrián Javaloy and Isabel Valera. RotoGrad: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2021.

[64] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Neural Information Processing Systems*, 2019.

[65] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Neural Information Processing Systems*, 2019.

[66] Kaiyi Ji, Jason D Lee, Yingbin Liang, and H Vincent Poor. Convergence of meta-learning with task-specific adaptation over partial parameters. In *Neural Information Processing Systems*, 2020.

[67] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. Preprint arXiv:2002.07836, 2020.

[68] Weisen Jiang, James Kwok, and Yu Zhang. Effective meta-regularization by kernelized proximal regularization. In *Neural Information Processing Systems*, 2021.

[69] Weisen Jiang, Yu Zhang, and James Kwok. SEEN: Few-shot classification with self-ensemble. In *International Joint Conference on Neural Networks*, 2021.

[70] Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In *International Conference on Machine Learning*, 2022.

[71] Weisen Jiang, Yu Zhang, and James Kwok. Effective structured-prompting by meta-learning and representitive verbalizer. In *International Conference on Machine Learning*, 2023.

[72] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *IEEE International Conference on Computer Vision*, 2019.

[73] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[74] Weihao Kong, Raghav Somani, Zhao Song, Sham Kakade, and Sewoong Oh. Meta-learning for mixed linear regression. In *International Conference on Machine Learning*, 2020.

[75] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[76] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015.

[77] Ken Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of International Machine Learning Conference*, 1995.

[78] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Neural Information Processing Systems*, 2019.

[79] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[80] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2018.

[81] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing*, 2021.

[82] D. Lewis. Reuters-21578 text categorization test collection. *Distribution 1.0, AT&T Labs-Research*, 1997.

[83] Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. Learning to transfer prompts for text generation. In *North American Chapter of the Association for Computational Linguistics*, 2022.

[84] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[85] Yuyang Li, Zhenzhenand Zhang, Jian-Yun Nie, and Dongsheng Li. Improving few-shot relation classification by prototypical representation learning with definition text. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2022.

[86] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. Preprint arXiv:1707.09835, 2017.

[87] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 2007.

[88] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Empirical Methods in Natural Language Processing*, 2020.

[89] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Neural Information Processing Systems*, 2021.

[90] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[91] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[92] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out*, 2022.

[93] Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *European Conference on Computer Vision*, 2020.

[94] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. Preprint arXiv:2103.10385, 2021.

[95] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[96] Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. In *International Workshop on Spoken Dialogue Systems Technology*, 2019.

[97] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[98] Andreas Maurer and Tommi Jaakkola. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 2005.

[99] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *North American Chapter of the Association for Computational Linguistics*, 2022.

[100] Rishabh Misra. News category dataset. Preprint arXiv:2209.11429, 2022.

[101] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, 2017.

[102] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International conference on machine learning*, 2018.

[103] Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021.

[104] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. Preprint arXiv:1803.02999, 2018.

[105] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. BOIL: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021.

[106] Sora Ohashi, Junya Takayama, Tomoyuki Kajiwara, and Yuki Arase. Distinct label representations for few-shot text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[107] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems*, 2018.

[108] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009.

[109] Eunbyung Park and Junier B Oliva. Meta-Curvature. In *Neural Information Processing Systems*, 2019.

[110] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In *Neural Information Processing Systems*, 2020.

[111] Anastasia Pentina and Christoph Lampert. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, 2014.

[112] Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *Neural Information Processing Systems*, 2015.

[113] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[114] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. Preprint arXiv:2203.07281, 2022.

[115] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[116] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners, 2019.

[117] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[118] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

[119] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

[120] Janarthanan Rajendran, Alexander Irpan, and Eric Jang. Meta-learning requires meta-augmentation. In *Neural Information Processing Systems*, 2020.

[121] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019.

[122] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, 2019.

[123] Tiago Ramalho and Marta Garnelo. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*, 2019.

[124] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[125] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, 2016.

[126] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Ali Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *International Conference on Learning Representations*, 2018.

[127] Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. PACOH: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, 2021.

[128] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976.

[129] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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

[131] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[132] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

[133] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *European Chapter of the Association for Computational Linguistics*, 2021.

[134] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, 2001.

[135] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. Preprint arXiv:1709.03410, 2017.

[136] Zhiqiang Shen, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng. Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *AAAI Conference on Artificial Intelligence*, 2021.

[137] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing*, 2020.

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

[139] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. In *Neural Information Processing Systems*, 2020.

[140] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017.

[141] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Neural Information Processing Systems*, 2020.

[142] Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. BBTv2: Towards a gradient-free future with large language models. In *Empirical Methods in Natural Language Processing*, 2022.

[143] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[144] Swee Chuan Tan and Jess Pei San Lau. Time series clustering: A superior alternative for market basket analysis. In *International Conference on Advanced Data and Information Engineering*, 2014.

[145] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*. 1998.

[146] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Neural Information Processing Systems*, 2020.

[147] Michalis K Titsias, Francisco JR Ruiz, Sotirios Nikoloutsopoulos, and Alexandre Galashov. Information theoretic meta learning with gaussian processes. In *Uncertainty in Artificial Intelligence*, 2021.

[148] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

[149] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

[150] Nilesh Tripuraneni, Chi Jin, and Michael Jordan. Provable meta-learning of linear representations. In *International Conference on Machine Learning*, 2021.

[151] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[152] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016.

[153] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[154] Haoxiang Wang, Ruoyu Sun, and Bo Li. Global convergence and induced kernels of gradient-based meta-learning with neural nets. Preprint arXiv:2006.14606, 2020.

[155] Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*, 2020.

[156] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53 (3):1–34, 2020.

[157] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *IEEE International Conference on Computer Vision*, 2019.

[158] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 2022.

[159] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[160] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[161] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. Preprint arXiv:1910.03771, 2019.

[162] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3D object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[163] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In *Empirical Methods in Natural Language Processing*, 2018.

[164] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN: Towards general solver for instance-level low-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

[165] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

[166] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

[167] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, 2019.

[168] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. In *International Conference on Learning Representations*, 2020.

[169] Huaxiu Yao, Ying-xin Wu, Maruan Al-Shedivat, and Eric Xing. Knowledge-aware meta-learning for low-resource text classification. In *Empirical Methods in Natural Language Processing*, 2021.

[170] Zhi-Xiu Ye and Zhen-Hua Ling. Multi-level matching and aggregation network for few-shot relation classification. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[171] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020.

[172] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[173] Matthew D Zeiler. AdaDelta: an adaptive learning rate method. Preprint arXiv:1212.5701, 2012.

[174] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[175] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*, 2022.

[176] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[177] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Uncertainty in Artificial Intelligence*, 2010.

[178] Wenliang Zhong and James Tin Yau Kwok. Convex multitask learning with flexible task clusters. In *International Conference on Machine Learning*, 2012.

[179] Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *Neural Information Processing Systems*, 2019.

[180] Pan Zhou, Yingtian Zou, X Yuan, Jiashi Feng, Caiming Xiong, and SC Hoi. Task similarity aware meta learning: Theory-inspired improvement on MAML. In *Conference on Uncertainty in Artificial Intelligence*, 2021.

[181] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[182] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[183] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.