Effective Meta-Regularization by Kernelized Proximal Regularization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of meta-learning, which has proved to be advantageous to accelerate learning new tasks with a few samples. The recent approaches based on deep kernel learning have achieved the recent state-of-the-art in few-shot learning. However, the regularizers in their base learners are not learnable. In this paper, we propose an algorithm to learn a proximal regularizer for the base learner. Experimental results confirm the advantage of the proposed algorithm.

1 Introduction

Humans, by leveraging prior knowledge and experience, can easily learn new tasks from a handful of examples. In contrast, deep networks are data-hungry, and a large number of training samples are required to avoid overfitting. To reduce the labor-intensive and time-consuming process of data labeling, *meta-learning* (or *learning to learn*) [3, 37] 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 [41, 12, 39, 35], hyperparameter optimization [14], neural architecture search [24, 44], and reinforcement learning [29].

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 [12, 25, 11, 38], such as the well-known MAML [12]. 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 there are many gradient steps. During testing, it is common for MAML's base learner to perform many gradient steps to seek a more accurate solution [12]. 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 [28, 43, 7, 8, 9], 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. [7] 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 [28].

To introduce nonlinearity to the base learner, a recent approach is to make use of the kernel trick. For example, R2D2 [4] and MetaOptNet [22] use deep kernel learning [42] 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.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

As learning a meta-regularization has been shown to be effective in linear models for regression [7] and classification [8], in this paper we propose a kernel-based algorithm to meta-learn a proximal regularizer for a nonlinear base learner. After kernel extension, the learnable function in the proximal regularizer is a function in the reproducing kernel Hilbert space (RKHS) induced by the base kernel. We establish convergence and global convergence of the proposed algorithm. Experiments on various benchmark regression and classification datasets demonstrate the superiority of the proposed algorithm over the recent state-of-the-arts.

Notations. Vectors are denoted by lowercase boldface, and matrices by uppercase boldface. For a vector $\mathbf{x}$, $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$ and $\mathrm{diag}[\mathbf{x}]$ constructs a diagonal matrix with $\mathbf{x}$ on the diagonal. $\|\cdot\|_{\mathcal{H}}$ is the norm on the Hilbert space $\mathcal{H}$. $\mathcal{N}(0, \sigma^2)$ is the univariate normal distribution with mean zero and variance σ^2. $\mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$ is the multivariate normal distribution with mean $\mathbf{m}$ and covariance matrix $\mathbf{\Sigma}$.

2 Related Work

In meta-learning, a collection $\mathcal{T}$ of tasks 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 is sampled from a given distribution $p(\tau)$, and has a support set $S_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_s\}$ and a query set $Q_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_q\}$, where $\mathbf{x} \in \mathbb{R}^d$ are the features and y the labels. Let $f(\cdot; \mathbf{w})$ be a model parameterized by $\mathbf{w}$ and $\mathcal{L}(D; \mathbf{w}) \equiv \sum_{(\mathbf{x},y)\in D} \ell(f(\mathbf{x}; \mathbf{w}), y)$ be the supervised loss on data set D, where $\ell(\cdot, \cdot)$ is a loss function that is assumed to be convex w.r.t. the first argument. In each meta-training iteration, a 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 batch are processed by the base learner, the meta-learner minimizes the loss $\sum_{\tau\in\mathcal{B}} \mathcal{L}(Q_\tau; \mathbf{w}_\tau)$ 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 $S_{\tau'}$ and $\boldsymbol{\theta}$. Finally, its performance is evaluated on $Q_{\tau'}$.

Popular meta-learning algorithms usually construct the task-specific model by: (i) meta-initialization [12, 25, 11, 38], (ii) meta-regularization [7, 8, 9, 28, 43], or (iii) metric learning [35, 4, 22, 27]. A representative meta-learning algorithm based on meta-initialization is MAML [12]. Its base learner learns $\mathbf{w}_\tau$ by gradient descent from a learnable initialization. Computing the meta-gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(Q_\tau; \mathbf{w}_\tau)$ needs to back-propagate through the inner optimization path and involves second-order derivatives. This can be expensive for large models and/or when there are many gradient steps.

For meta-learning algorithms based on meta-regularization, the base learner learns $\mathbf{w}_\tau$ by minimizing a proximal regularized loss

$$\mathcal{L}(S_\tau; \mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2, \tag{1}$$

where $\lambda > 0$ is the regularization parameter. The meta-gradient can be computed directly from the learned $\mathbf{w}_\tau$, without back-propagating through the inner optimization trajectory [28]. Though more efficient, this still takes $\mathcal{O}(n_{\boldsymbol{\theta}}^3)$ time [28], where $n_{\boldsymbol{\theta}}$ is the number of parameters in $\boldsymbol{\theta}$.

Metric learning methods have been widely studied in few-shot learning [39, 35, 4, 22, 27]. The meta-learner maps raw samples to an embedding space via a a deep network, then feeds the embeddings to the base learner to train a simple task model. Typical models include non-parametric prototype classifier (ProtoNet [35]), linear models like ride regression (R2D2 [4]), SVM classifiers (MetaOptNet-SVM [22]), and a softmax classifier (ANIL [27]). In particular, the base learners in R2D2 [4], MetaOptNet [22] and DKT [26] seek solutions in dual space, which achieve the recent state-of-the-art performance. However, their base learners use a Tikhonov regularizer, which is not learnable as in meta-regularization approaches.

3 Meta-Regularization by Kernelized Proximal Regularization

3.1 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 $\mathbf{w}_\tau^* \in \mathbb{R}^d$. We assume that each input $\mathbf{x}$ is sampled

Algorithm 1 MAML [12].

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**
    $\mathbf{w}_\tau^{(\text{gd})} = \boldsymbol{\psi}_t - \gamma \mathbf{X}_\tau^\top (\mathbf{X}_\tau \boldsymbol{\psi}_t - \mathbf{y}_\tau)$;
    $\mathbf{g}_\tau = \frac{1}{2} \sum_{(\mathbf{x},y)\in Q_\tau} \nabla_{\boldsymbol{\psi}} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{gd})} - y)^2$;
  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 [7].

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**
    $\mathbf{w}_\tau^{(\text{prox})} = \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{X}_\tau \mathbf{w} - \mathbf{y}_\tau\|^2 + \frac{\lambda}{2}\|\mathbf{w} - \boldsymbol{\theta}_t\|^2$;
    $\mathbf{g}_\tau = \frac{1}{2} \sum_{(\mathbf{x},y)\in Q_\tau} \nabla_{\boldsymbol{\theta}} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{prox})} - y)^2$;
  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

from $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{x}}^2 \mathbf{I})$, and the output y is obtained as $\mathbf{x}^\top \mathbf{w}_\tau^* + \xi$, where $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the random noise. We compare two representative meta-learning algorithms: (i) MAML [12], 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 by CommonMean) [7], 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 $\boldsymbol{\theta}$:

$$\mathbf{w}_\tau^{(\text{prox})} = \operatorname*{argmin}_{\mathbf{w}} \sum\nolimits_{(\mathbf{x}_i, y_i)\in S_\tau} \frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \frac{\lambda}{2}\|\mathbf{w} - \boldsymbol{\theta}\|^2 = (\lambda \mathbf{I} + \mathbf{X}_\tau^\top \mathbf{X}_\tau)^{-1}(\lambda \boldsymbol{\theta} + \mathbf{X}_\tau^\top \mathbf{y}_\tau), \quad (2)$$

where $\mathbf{X}_\tau = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_s}]$ is the sample matrix from S_τ, and $\mathbf{y}_\tau = [y_1, \ldots, y_{n_s}]$ is the corresponding label vector. Algorithms 1 and 2 show MAML and CommonMean, respectively, for this problem.

Recently, Balcan et al. [2] 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. All proofs are in the appendix.

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

The following Proposition shows that $\bar{\mathbf{w}}$ in Proposition 1 is also the best $\boldsymbol{\psi}$ (for MAML) or $\boldsymbol{\theta}$ (for CommonMean) that can lead to the smallest population risk for this meta-learning problem.

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

During meta-testing, we sample a task $\tau' \sim p(\tau)$ with parameter $\mathbf{w}_{\tau'}^*$. 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 will drop the subscript τ' in the following. Let the singular value decomposition of $\mathbf{X}$ be $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ (where $\boldsymbol{\Sigma} = \operatorname{diag}[[\nu_1, \ldots, \nu_{n_s}]]$), 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 [12]. With the convex loss and linear model here, the base learner can obtain a globally optimal solution $\mathbf{w}^{(\text{gd}\infty)}$ 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, $\mathbf{w}^{(\text{gd}\infty)}$ is not unique but depends on the learned initialization. Let $\mathbf{w}^{(\text{gd}\infty)} = \mathbf{V}\mathbf{a}^{(\text{gd}\infty)} + \mathbf{V}^\perp \mathbf{b}^{(\text{gd}\infty)}$ For gradient descent, its update direction $\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top(\mathbf{X}\mathbf{w} - \mathbf{y})$ is always in the span of $\mathbf{V}$ and so $\mathbf{b}^{(\text{gd}\infty)}$ remains unchanged.

Proposition 3 ([17])**.** *Let $\mathbf{w}^* = \mathbf{V}\mathbf{a}^* + \mathbf{V}^\perp \mathbf{b}^*$, $\boldsymbol{\theta} = \boldsymbol{\psi} = \mathbf{V}\mathbf{a}_0 + \mathbf{V}^\perp \mathbf{b}_0$, $\tilde{\mathbf{a}} = \mathbf{a}_0 - \mathbf{a}^*$, and $\tilde{\mathbf{b}} = \mathbf{b}_0 - \mathbf{b}^*$. Assume that $\gamma < \min_{1\le j\le n_s} 1/\nu_j^2$. We have $\mathbb{E}_{\boldsymbol{\xi}} \|\mathbf{w}^{(gd\infty)} - \mathbf{w}^*\|^2 = \|\tilde{\mathbf{b}}\|^2 + \sum_{j=1}^{n_s} (\sigma_\xi/\nu_j)^2$, where the expectation is over the label noise vector $\boldsymbol{\xi}$.*

For CommonMean, using the Woodbury matrix identity, $\mathbf{w}^{(\text{prox})} = \boldsymbol{\theta} + \mathbf{X}^\top(\lambda \mathbf{I} + \mathbf{X}\mathbf{X}^\top)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$, and $\mathbf{b}^{(\text{prox})}$ is also unchanged. Thus, $\mathbf{w}^{(\text{gd}\infty)}$ and $\mathbf{w}^{(\text{prox})}$ only differ in the estimation of $\mathbf{w}^*$ on $\mathbf{V}$.

Proposition 4. $\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(prox)} - \mathbf{w}^*\|^2 = \|\tilde{\mathbf{b}}\|^2 + \sum_{j=1}^{n_s}\left(\frac{\lambda\tilde{\mathrm{a}}_j}{\lambda+\nu_j^2}\right)^2 + \sum_{j=1}^{n_s}\left(\frac{\sigma_\xi}{\lambda/\nu_j+\nu_j}\right)^2$, *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 and introducing an estimation bias of $\mathbf{a}^*$.

3.2 Proposed Algorithm

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

$$\boldsymbol{\alpha}_\tau = (\mathbf{I} + \lambda^{-1}\mathbf{X}_\tau\mathbf{X}_\tau^\top)^{-1}(\mathbf{y}_\tau - \mathbf{X}_\tau\boldsymbol{\theta}). \tag{3}$$

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. Deep kernel learning [42] combines the expressive power of deep architectures with kernels. An input $\mathbf{x}$ is mapped to $\mathbf{z} = \mathrm{NN}(\mathbf{x}; \boldsymbol{\phi})$ in an embedding space $\mathcal{E}$ via a deep network parameterized by $\boldsymbol{\phi}$. Moreover, the presence of $\boldsymbol{\theta}$ in (2) allows extra flexibility over [42, 22, 4, 26]. Specifically, $\boldsymbol{\theta}$ becomes a function $f_{\boldsymbol{\theta}} \in \mathcal{H}$, the reproducing kernel Hilbert spaces (RKHS) corresponding to a given kernel function $\mathcal{K}$. The base learner obtains a model $f_\tau \in \mathcal{H}$ by minimizing

$$\min_{f\in\mathcal{H}} \sum\nolimits_{(\mathbf{x}_i, y_i)\in S_\tau} \ell(f(\mathbf{z}_i), y_i) + \frac{\lambda}{2}\|f - f_{\boldsymbol{\theta}}\|_{\mathcal{H}}^2 , \tag{4}$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm on $\mathcal{H}$. By setting $f_{\boldsymbol{\theta}} = 0$, this formulation can recover the state-of-the-arts of MetaOptNet [22], R2D2 [4], and DKT [26], which also uses the deep kernel in meta-learning. However, Proposition 4 suggests that a good $f_{\boldsymbol{\theta}}$ reduces the noise disturbance.

By the representer theorem [34], the solution of (4) is $f_\tau = f_{\boldsymbol{\theta}} + \sum_{(\mathbf{x}_i, y_i)\in S_\tau} \alpha_i\mathcal{K}_{\mathbf{z}_i}$, where $\mathcal{K}_{\mathbf{z}_i} = \mathcal{K}(\mathbf{z}_i, \cdot) \in \mathcal{H}$, and $\boldsymbol{\alpha}_\tau = [\alpha_{\tau,1}, \ldots, \alpha_{\tau,n_s}]$ is obtained from the convex program

$$\min_{\boldsymbol{\alpha}} \sum\nolimits_{(\mathbf{x}_i, y_i)\in S_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \boldsymbol{\alpha}^\top\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\boldsymbol{\alpha}, \tag{5}$$

where $\mathbf{Z}_\tau = [\mathbf{z}_1^\top, \ldots, \mathbf{z}_{n_s}^\top] \in \mathbb{R}^{n_s\times d}$, and $\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$ is the kernel matrix. Note that we have dropped the tricky hyperparameter λ in (4) as the kernel is learnable. With the square loss, the dual solution of (5) 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 the $f_{\boldsymbol{\theta}}$ and network parameter $\boldsymbol{\phi}$ by performing gradient descent on the meta-loss

$$\mathcal{L}_{\mathrm{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv \sum\nolimits_{\tau\in\mathcal{T}} \sum\nolimits_{(\mathbf{x}, y)\in Q_\tau} \ell(f_\tau(\mathbf{z}), y), \tag{6}$$

where $\hat{y} \equiv f_\tau(\mathbf{z}) = f_{\boldsymbol{\theta}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top\boldsymbol{\alpha}_\tau$. Using the chain rule,

$$\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\ell(\hat{y}, y) = \nabla_1\ell(\hat{y}, y)\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\hat{y}, \tag{7}$$

$$\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\hat{y} = \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}f_{\boldsymbol{\theta}}(\mathbf{z}) + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}))^\top\boldsymbol{\alpha}_\tau + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\boldsymbol{\alpha}_\tau)^\top\mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}). \tag{8}$$

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 $\mathbf{p} = [f_{\boldsymbol{\theta}}(\mathbf{z}_1), \ldots, f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s}), \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_1), \ldots, \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{n_s})] \in$

$\mathbb{R}^{n_s+n_s^2}$ is the input to the dual problem. $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathbf{p}$ can be directly computed. When the square loss is used, $\nabla_{\mathbf{p}}\boldsymbol{\alpha}_\tau = -(\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau))^{-1}\left[\mathbf{I} \mid \mathbf{I}\otimes\boldsymbol{\alpha}_\tau^\top\right]$. For a general loss, $\boldsymbol{\alpha}_\tau$ is obtained by solving the convex program, and $\nabla_{\mathbf{p}}\boldsymbol{\alpha}_\tau$ can be obtained by implicit differentiation. Denote the dual objective in (5) by $g(\mathbf{p},\boldsymbol{\alpha})$. By the implicit function theorem [32], $\nabla_{\mathbf{p}}\boldsymbol{\alpha}_\tau = -\left(\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p},\boldsymbol{\alpha}_\tau)\right)^{-1}\frac{\partial^2}{\partial\mathbf{p}\partial\boldsymbol{\alpha}}g(\mathbf{p},\boldsymbol{\alpha}_\tau)$, where $\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p},\boldsymbol{\alpha}_\tau) = \sum_{(\mathbf{x}_i,y_i)\in S_\tau}\nabla_1^2\ell(f_\tau(\mathbf{z}_i),y_i)\mathcal{K}(\mathbf{Z}_\tau,\mathbf{z}_i)\mathcal{K}(\mathbf{Z}_\tau,\mathbf{z}_i)^\top + \mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau)$, $\frac{\partial^2}{\partial\mathbf{p}\partial\boldsymbol{\alpha}}g(\mathbf{p},\boldsymbol{\alpha}_\tau) = \left[\mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau)\mathbf{D} \mid (\mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau)\mathbf{D})\otimes\boldsymbol{\alpha}_\tau^\top + \mathbf{v}^\top\otimes\mathbf{I} + \mathbf{I}\otimes\boldsymbol{\alpha}_\tau^\top\right]$, $\mathbf{D} = \mathrm{diag}[\nabla_1^2\ell(f_\tau(\mathbf{z}_1),y_1),\ldots,\nabla_1^2\ell(f_\tau(\mathbf{z}_{n_s}),y_{n_s})]$, $\mathbf{v} = [\nabla_1\ell(f_\tau(\mathbf{z}_1),y_1),\ldots,\nabla_1\ell(f_\tau(\mathbf{z}_{n_s}),y_{n_s})]$, 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 much faster that the other meta-learning algorithms (e.g., MAML [12] with single step takes $\mathcal{O}(n_\phi^2)$ time, iMAML [28]: $\mathcal{O}(n_\phi^3)$, CommonMean [7]: $\mathcal{O}(d^3)$).

Algorithm 3 MetaProx.

Require: step size η_t, batch size b;
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**
    $\mathbf{z}_i = \mathrm{NN}(\mathbf{x}_i;\boldsymbol{\phi}_t)$ for each $(\mathbf{x}_i,y_i)\in S_\tau$;
    $f_\tau(\mathbf{z}) \equiv f_{\boldsymbol{\theta}_t}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau,\mathbf{z})^\top\boldsymbol{\alpha}$ denote the task model w.r.t. dual variables;
    $\boldsymbol{\alpha}_\tau = \mathrm{argmin}_{\boldsymbol{\alpha}}\sum_{(\mathbf{x}_i,y_i)\in S_\tau}\ell(f_\tau(\mathbf{z}_i;\boldsymbol{\alpha}),y_i) + \boldsymbol{\alpha}^\top\mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau)\boldsymbol{\alpha}$;
    $\mathbf{g}_\tau = \sum_{(\mathbf{x},y)\in Q_\tau}\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\ell(\hat{y},y)$, where $\hat{y} = f_\tau(\mathbf{z};\boldsymbol{\alpha}_\tau)$ and $\mathbf{z} = \mathrm{NN}(\mathbf{x};\boldsymbol{\phi}_t)$;
  end for
  meta-learner: $(\boldsymbol{\theta}_{t+1},\boldsymbol{\phi}_{t+1}) = (\boldsymbol{\theta}_t,\boldsymbol{\phi}_t) - \frac{\eta_t}{b}\sum_{\tau\in\mathcal{B}_t}\mathbf{g}_\tau$;
end for

Classification. In this section, we consider extension from regression to N-way few-shot classification. For task τ, the base learner learns the model $\mathbf{f}_\tau = [f_\tau^{(1)},\ldots,f_\tau^{(N)}]$ for the N classes by minimizing

$$\min_{f^{(1)},\ldots,f^{(N)}\in\mathcal{H}}\sum\nolimits_{(\mathbf{x}_i,y_i)\in S_\tau}\ell(\hat{\mathbf{y}}_i,y_i) + \frac{\lambda}{2}\sum\nolimits_{c=1}^{N}\|f^{(c)} - f_{\boldsymbol{\theta}^{(c)}}\|_{\mathcal{H}}^2, \tag{9}$$

where $\hat{\mathbf{y}}_i = [f^{(1)}(\mathbf{z}_i),\ldots,f^{(N)}(\mathbf{z}_i)]$, and $f_{\boldsymbol{\theta}^{(1)}}\ldots,f_{\boldsymbol{\theta}^{(N)}}\in\mathcal{H}$ are functions learned by the meta-learner. By the representer theorem [34], $\mathbf{f}_\tau = [f_{\boldsymbol{\theta}^{(1)}} + \mathcal{K}(\mathbf{Z}_\tau,\cdot)^\top\boldsymbol{\alpha}_\tau^{(1)},\ldots,f_{\boldsymbol{\theta}^{(N)}} + \mathcal{K}(\mathbf{Z}_\tau,\cdot)^\top\boldsymbol{\alpha}_\tau^{(N)}]$, where $[\boldsymbol{\alpha}_\tau^{(1)},\cdots,\boldsymbol{\alpha}_\tau^{(N)}]$ is obtained from the convex program $\min_{\boldsymbol{\alpha}^{(1)},\ldots,\boldsymbol{\alpha}^{(N)}}\sum_{(\mathbf{x}_i,y_i)\in S_\tau}\ell(\hat{\mathbf{y}}_i,y_i) + \sum_{c=1}^{N}\boldsymbol{\alpha}^{(c)\top}\mathcal{K}(\mathbf{Z}_\tau,\mathbf{Z}_\tau)\boldsymbol{\alpha}^{(c)}$. As $[\boldsymbol{\alpha}_\tau^{(1)},\cdots,\boldsymbol{\alpha}_\tau^{(N)}]\in\mathbb{R}^{Nn_s}$ and both N, n_s are typically very small (e.g., $N = 5, n_s = 1$ in 5-way 1-shot classification), this convex program can be solved efficiently. The meta-learner then updates the network parameter $\boldsymbol{\phi}$ and $\{f_{\boldsymbol{\theta}^{(1)}},\ldots,f_{\boldsymbol{\theta}^{(N)}}\}$ by gradient descent on the meta-loss $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(N)},\boldsymbol{\phi}) \equiv \sum_{\tau\in\mathcal{T}}\sum_{(\mathbf{x}_i,y_i)\in Q_\tau}\ell(\hat{\mathbf{y}}_i,y_i)$. Computing $\nabla_{(\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(N)},\boldsymbol{\phi})}\ell(\hat{\mathbf{y}},y)$ takes $\mathcal{O}((Nn_s)^3 + (Nn_s)^2(n_{\boldsymbol{\phi}} + \sum_{c=1}^{N}n_{\boldsymbol{\theta}^{(c)}}))$ time, where $n_{\boldsymbol{\theta}^{(c)}}$ is the size of $\boldsymbol{\theta}^{(c)}$. This is again linear in the number of meta-parameters and thus very efficient.

3.3 Theoretical Analysis

With the linear kernel and square loss, the dual solution (3) 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 $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta},\boldsymbol{\phi})$ in (6) is convex and convergence follows from convex optimization [5, 7]. 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 [15, 31] and meta-learning in non-convex settings [11, 43].

Assumption 1 (Smoothness)**.** *(i) The deep network* $\mathrm{NN}(\mathbf{x};\boldsymbol{\phi})$ *is Lipschitz-smooth, i.e.,* $\|\nabla_{\boldsymbol{\phi}}\mathrm{NN}(\mathbf{x};\boldsymbol{\phi})-\nabla_{\boldsymbol{\phi}}\mathrm{NN}(\mathbf{x};\boldsymbol{\phi}')\|\leq\beta\|\boldsymbol{\phi}-\boldsymbol{\phi}'\|$ *with a Lipschitz constant* $\beta>0$*; (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) the loss* $\ell(\hat{y},y)$ *is twice-differentiable and Lipschitz-smooth w.r.t.* $\hat{y}$*.*

The following lemma guarantees smoothness of the meta-loss.

Lemma 1. $\mathcal{L}_{meta}(\boldsymbol{\theta},\boldsymbol{\phi})$ *is Lipschitz-smooth w.r.t.* $(\boldsymbol{\theta},\boldsymbol{\phi})$ *with a Lipschitz constant* β_{meta}*.*

Theorem 1. *Let the step size be* $\eta_t=\min(1/\sqrt{T},1/\beta_{meta})$*. Algorithm 3 satisfies* $\min_{1\leq t\leq T}\mathbb{E}\|\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{L}_{meta}(\boldsymbol{\theta}_t,\boldsymbol{\phi}_t)\|^2=\mathcal{O}\left(1/\sqrt{T}\right)$*, where the expectation is taken over the random training samples.*

This rate is the same as MAML [11, 19] and Meta-MinibatchProx [43]. For MAML with $J>1$ gradient steps, Ji et al., [19] 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 [13, 43] 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 [40] studies the global convergence of MAML in over-parameterized neural networks. Over-parameterization is closely related to the assumption of uniform conditioning [18, 21, 23].

Assumption 2 (Uniform conditioning [23])**.** *A multivariable function* $\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi})$ *is* μ*-uniformly conditioning if its tangent kernel [18] satisfies* $\min_{(\boldsymbol{\theta},\boldsymbol{\phi})}\lambda_{\min}(\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi})\nabla^{\top}_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{M}(\boldsymbol{\theta},\boldsymbol{\phi}))\geq\mu>0$*, 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 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 $\mathcal{O}(1/\sqrt{T})$. The rate is improved to exponential if the meta-learner adopts full gradient descent.

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

4 Experiments

4.1 Few-shot Regression

Data sets. Experiments are performed on three data sets.

(i) *Sine*. This is the sinusoid regression problem in [12]. 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 [12], 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 [36], which contains weekly purchased quantities of 811 products over 52 weeks. For each product (task), we use its sales quantities in the previous 5 weeks to predict the sales quantity for the current week. Thus, each task 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 [16] 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 [26] and evaluate the model on 10-shot regression. Each person has 133 facial images covering a viewsphere of $\pm 90^\circ$ in yaw and $\pm 30^\circ$ 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 $[-90^\circ, 0^\circ]$. 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 [26], we do not use a meta-validation set since the dataset is small.

Network Architecture. For *Sine* and *Sale*, we use the network in [12], 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 [26]. 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 [20] with a learning rate of 0.001. Each mini-batch has 8 tasks. For *Sine* and *Sale*, the model (kernel 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 [26] 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 [7], MAML [12], MetaOptNet-RR [22], Meta-MinibatchProx [43], and iMAML [28]. CommonMean is a linear model, and MetaOptNet-RR is equivalent to MetaProx when $f_{\boldsymbol{\theta}} = 0$. Following [12], 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 [43] and iMAML [28] are meta-regularization approaches. For QMUL, we compare MetaProx with the baselines reported in [26] (namely, DKT [26], Feature Transfer [10], and MAML). As further baselines, we also compare with Meta-MinibatchProx and MetaOptNet-RR to evaluate the improvement of MetaProx due to the learnable $f_{\boldsymbol{\theta}}$.

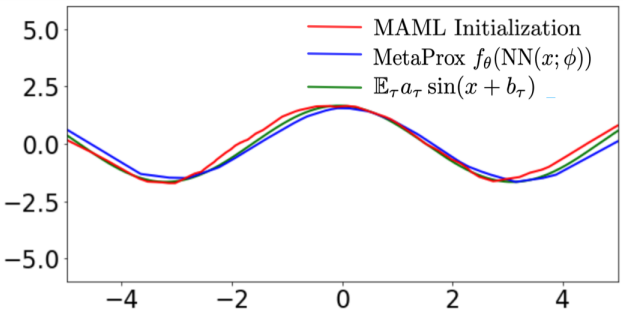


Figure 1: The learned $f_{\boldsymbol{\theta}}(\mathrm{NN}(\mathbf{x};\boldsymbol{\phi}))$ by MetaProx and MAML initialization, and the expected output, in the noise-free 5-shot experiment.

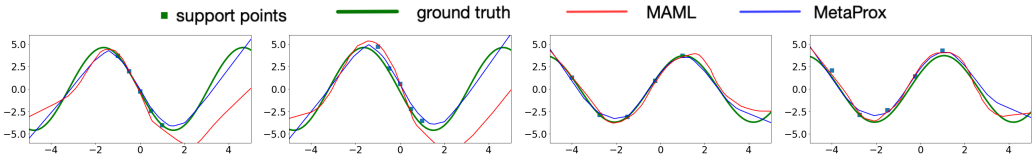
Results on Sine. Table 2 shows the MSE results. Obviously, CommonMean (a linear model) fails in this non-linear regression task. MetaProx and MetaOptNet-RR significantly outperform the other baselines. MetaProx (with the learned $f_{\boldsymbol{\theta}}$) performs better than MetaOptNet-RR, particularly when the data is noisy.

Table 1: Average MSE (with 95% confidence intervals) of few-shot regression on *QMUL* (10-shot). Results of the first four methods are from [26] and the standard deviations are dropped.

method	in-range	out-of-range
Feature Transfer [10]	0.22	0.18
MAML [12]	0.21	0.18
DKT + RBF [26]	0.12	0.14
DKT + Spectral [26]	0.10	0.11
Meta-MinibatchProx [43]	0.171 ± 0.022	0.193 ± 0.025
MetaOptNet-RR [22]	0.021 ± 0.007	0.039 ± 0.009
MetaProx (proposed)	$\mathbf{0.012 \pm 0.003}$	$\mathbf{0.020 \pm 0.005}$

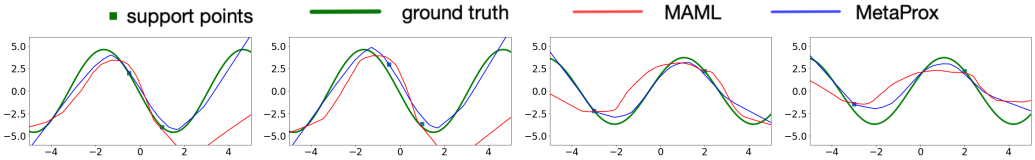
Figure 1 shows the learned $f_{\boldsymbol{\theta}}(\mathrm{NN}(\mathbf{x};\boldsymbol{\phi}))$ by MetaProx and MAML's learned initialization in the 5-shot experiment with $\sigma_\xi^2 = 0$. As can be seen, both are close to the expected output.

Figures 2 and 3 show the learned functions on 2 meta-testing tasks (with ($a = 4.6$, $b = 3.2$) and ($a = 3.7$, $b = 0.5$)) in the 5-shot setting and more challenging 2-shot setting. As can be seen, MetaProx always fits the target curve well. Though MAML can fit the support samples, it performs worse in regions far from the support samples. This is especially noticeable in the 2-shot setting.



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

Figure 2: Sinusoid regression: Two 5-shot meta-testing tasks τ_1 and τ_2 with different σ_ξ's.



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

Figure 3: Sinusoid regression: Two 2-shot meta-testing tasks τ_1 and τ_2 with different σ_ξ's.

Table 2: 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 [7]	4.58 ± 0.07	4.59 ± 0.07	4.29 ± 0.06	4.31 ± 0.06	0.090	0.074
MAML [12]	1.24 ± 0.12	1.91 ± 0.13	0.41 ± 0.03	1.15 ± 0.05	0.069	0.063
iMAML [28]	1.12 ± 0.11	1.84 ± 0.10	0.38 ± 0.02	1.02 ± 0.05	0.068	0.063
Meta-MinibatchProx [43]	1.15 ± 0.08	1.87 ± 0.09	0.37 ± 0.02	1.01 ± 0.03	0.081	0.064
MetaOptNet-RR [22]	0.18 ± 0.01	0.79 ± 0.01	0.01 ± 0.00	0.19 ± 0.01	0.088	0.068
MetaProx (proposed)	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.43 \pm 0.01}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.13 \pm 0.01}$	**0.061**	**0.060**

Results on Sale. As can be seen from Table 2, 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 1 shows that MetaProx achieves the lowest MSE and the kernel methods (DKT+RBF, DKT+Spectral, MetaOptNet-RR, and MetaProx) perform better than non-kernel methods (Feature Transfer, MAML, and Meta-MinibatchProx). MetaProx with the learnable $f_{\boldsymbol{\theta}}$ reduces the errors of MetaOptNet-RR by half.

4.2 Few-shot Classification

Datasets. We use the standard 5-way K-shot setting ($K = 1$ or 5) on the *mini-ImageNet* [39] dataset, which consists of 100 randomly chosen classes from *ILSVRC*-2012 [33]. Each class contains 600 84×84 images. We use the commonly-used split in [30]: the 100 classes are randomly split into 64 for meta-training, 16 for meta-validation, and 20 for meta-testing.

Network Architecture. We use the Conv4 network in [12, 39], which has 4 modules. Each module is a 3×3 convolutional layer with 64 filters, followed by batch normalization, ReLU activation, and 2×2 max-pooling. The output feature dimensionalities are 1600 and 256 for *mini-ImageNet* and *CIFAR-FS*, respectively. As the cosine similarity is more effective than ℓ_2 distance in few-shot classification [6], we adopt the cosine kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}') = \cos(\mathbf{z}, \mathbf{z}')$ as base kernel, where $\mathbf{z}$ is the

Table 3: Accuracies (with 95% confidence interval) of 5-way few-shot classification on *mini-ImageNet*. † 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 [28]).

method	1-shot	5-shot
MAML [12]	48.7 ± 1.8	63.1 ± 0.9
FOMAML [12]	48.1 ± 1.8	63.2 ± 0.9
REPTILE [25]	50.0 ± 0.3	66.0 ± 0.6
Proto-MAML† [38]	51.2 ± 0.8	65.8 ± 0.6
iMAML [28]	49.0 ± 1.8	$-$
Meta-MinibatchProx [43]	50.8 ± 0.9	67.4 ± 0.9
R2D2 [4]	49.5 ± 0.2	65.4 ± 0.3
MetaOptNet-SVM(lin)† [22]	49.8 ± 0.9	66.9 ± 0.7
MetaOptNet-SVM(cos)† [22]	50.1 ± 0.9	67.2 ± 0.6
ProtoNet [35]	49.4 ± 0.8	68.2 ± 0.7
ANIL [27]	46.7 ± 0.4	61.5 ± 0.5
MetaProx (proposed)	$\mathbf{52.4 \pm 1.0}$	$\mathbf{68.8 \pm 0.8}$

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, where $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(5)}$ are the class centroids of samples in S_τ.

Baselines. We compare MetaProx with the state-of-the-arts: (i) meta-initialization: MAML [12] and its variants include FOMAML [12], REPTILE [25] and Proto-MAML [38]; (ii) meta-regularization: iMAML [28] and Meta-MinibatchProx [43]; and (iii) metric learning: R2D2 [4], MetaOptNet [22] with SVM using the linear kernel and cosine kernel, ProtoNet [35], and ANIL [27].

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 $25,000$ iterations, and each mini-batch has 4 tasks. We use the Adam optimizer [20] 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 loss 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. Table 3 shows the results. As can be seen, MetaProx is always the best. Compared with R2D2 and MetaOptNet-SVM, MetaProx is better due to the learnable regularizer.

5 Conclusion

In this paper, we proposed the MetaProx, an efficient meta-regularization algorithm for meta-learning. Through reformulating the problem in dual space, a learnable proximal regularizer is introduced to the base learner. The algorithm combines the deep kernel and meta-regularization. Moreover, the time complexities of computing the meta-gradient is only linear in the number of meta-parameters. The effectiveness of the proposed algorithm is verified on standard datasets for regression and classification.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[2] M.-F. Balcan, M. Khodak, and A. Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433. PMLR, 2019.

[3] Y. Bengio, S. Bengio, and J. Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[4] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *Proceedings of International Conference on Learning Representations*, 2018.

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

[6] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *Proceedings of International Conference on Learning Representations*, 2018.

[7] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. Learning to learn around a common mean. In *Proceedings of Advances in Neural Information Processing Systems*, pages 10169–10179, 2018.

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

[9] G. Denevi, M. Pontil, and C. Ciliberto. The advantage of conditional meta-learning for biased regularization and fine tuning. *Proceedings of Advances in Neural Information Processing Systems*, 33, 2020.

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

[11] A. Fallah, A. Mokhtari, and A. Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR, 2020.

[12] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.

[13] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. In *Proceedings of International Conference on Machine Learning*, pages 1920–1930, 2019.

[14] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2018.

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

[16] S. Gong, S. McKenna, and J. J. Collins. An investigation into face pose distributions. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 265–270. IEEE, 1996.

[17] S. Gunasekar, J. Lee, D. Soudry, and N. Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

[18] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[19] K. Ji, J. Yang, and Y. Liang. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. *arXiv preprint arXiv:2002.07836*, 2020.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

[21] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[22] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[23] C. Liu, L. Zhu, and M. Belkin. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *arXiv preprint arXiv:2003.00307*, 2020.

[24] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[25] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[26] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. J. Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2019.

[28] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Proceedings of Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[29] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[30] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *Proceedings of International Conference on Learning Representations*, 2017.

[31] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[32] W. Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[34] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

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

[36] S. C. Tan and J. P. San Lau. Time series clustering: A superior alternative for market basket analysis. In *Proceedings of the First International Conference on Advanced Data and Information Engineering*, pages 241–248. Springer, Singapore, 2014.

[37] S. Thrun and L. Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[38] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

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

[40] H. Wang, R. Sun, and B. Li. Global convergence and induced kernels of gradient-based meta-learning with neural nets. *arXiv preprint arXiv:2006.14606*, 2020.

[41] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[42] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, 2016.

[43] P. Zhou, X. Yuan, H. Xu, S. Yan, and J. Feng. Efficient meta learning via minibatch proximal update. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1534–1544, 2019.

[44] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and last 2 paragraphs in Introduction
   (b) Did you describe the limitations of your work? [No] We are still conducting research to find the limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Because no potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] I have read it.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See theorems in Section 3.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix in the supplemental material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4, e.g.. Table 2, Table 3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.
   (b) Did you mention the license of the assets? [Yes] See the cited reference.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] No new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] All datasets are public.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] No identifiable information or offensive content in the datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] Not human subjects research.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No] Not human subjects research.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] Not human subjects research.