Effective Meta-Regularization by Kernelized Proximal Regularization

Weisen Jiang◇†, James T. Kwok†, Yu Zhang◇

◇Southern University of Science and Technology †Hong Kong University of Science and Technology

Introduction

- Deep networks are data-hungry, and massive training samples are required to avoid overfitting. To reduce the labor-intensive and time-consuming process of data labeling, **meta-learning** aims to extract meta-knowledge from seen tasks to accelerate learning on unseen tasks with limited samples.
- One representative approach is called **meta-regularization**, in which the base learner learns the task-specific model by minimizing a regularized loss. Recently, Denevi et al. [1] study a linear model with efficient closed-form solution. However, extending to nonlinear models requires computing the meta-gradient using matrix inversion, which can be infeasible for large models like neural networks [2].
- To introduce nonlinearity to the base learner, MetaOptNet [3] uses kernel trick and achieves the state-of-the-art performance. However, its base learner uses a Tikhonov regularizer rather than a learnable proximal regularizer as in meta-regularization methods.
- In this paper, we propose a kernel-based algorithm (called **MetaProx**) to meta-learn a proximal regularizer for a nonlinear base learner.

Our Approach

- Notations:
  - $\mathcal{T}$ is a collection of tasks for meta-training. Each task $\tau \in \mathcal{T}$ contains a support set S_τ and a query set Q_τ ($n_s = |S_\tau|$).
  - An input $\mathbf{x}$ is mapped to $\mathbf{z} = \text{NN}(\mathbf{x}; \boldsymbol{\phi})$ in an embedding space $\mathcal{E}$. $\mathbf{Z}_\tau = [\mathbf{z}_1^\top; \dots; \mathbf{z}_{n_s}^\top]$, where $\mathbf{z}_i = \text{NN}(\mathbf{x}_i; \boldsymbol{\phi})$ for $\mathbf{x}_i \in S_\tau$.
  - $\mathcal{K}$ is a base kernel on $\mathcal{E} \times \mathcal{E}$, $\mathcal{H}$ is the corresponding RKHS.
- The problem in the inner loop is:

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

- By representer theorem, $f_\tau(\cdot; \boldsymbol{\alpha}_\tau) = f_{\boldsymbol{\theta}}(\cdot) + \mathcal{K}(\mathbf{Z}_\tau, \cdot)^\top \boldsymbol{\alpha}_\tau$.
- **Meta-Regularization by Kernelized Proximal Regularization**:

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

$$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\phi}) - \eta \sum_{(\mathbf{x}, y) \in Q_\tau} \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \ell(f_\tau(\mathbf{z}; \boldsymbol{\alpha}_\tau), y). \quad \text{(outer)}$$

- **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 $f_{\boldsymbol{\theta}} = 0$, MetaProx recovers MetaOptNet [3]. Experiment results demonstrate that MetaProx significantly outperforms MetaOptNet, which verifies the effectiveness of a learnable proximal regularizer.
  3. For square loss, $\boldsymbol{\alpha}_\tau = (\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1}(\mathbf{y}_\tau - f_{\boldsymbol{\theta}}(\mathbf{Z}_\tau))$ has an efficient closed-form solution. For general losses, the dual problem is convex and can be solved 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].

MetaProx Algorithm

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**
    $\mathbf{z}_i = \text{NN}(\mathbf{x}_i; \boldsymbol{\phi}_t)$ for each $(\mathbf{x}_i, y_i) \in S_\tau$;
    $f_\tau(\mathbf{z}; \boldsymbol{\alpha}) \equiv f_{\boldsymbol{\theta}_t}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}$ denote the task model w.r.t. dual variables;
    $\boldsymbol{\alpha}_\tau = \operatorname{argmin}_{\boldsymbol{\alpha}} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i; \boldsymbol{\alpha}), y_i) + \boldsymbol{\alpha}^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \boldsymbol{\alpha}$;
    $\mathbf{g}_\tau = \sum_{(\mathbf{x}, y) \in Q_\tau} \nabla_{(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t)} \ell(\hat{y}, y)$, where $\hat{y} = f_\tau(\mathbf{z}; \boldsymbol{\alpha}_\tau)$ and $\mathbf{z} = \text{NN}(\mathbf{x}; \boldsymbol{\phi}_t)$;
  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

In experiments, (i) in regression, $\mathcal{K}$ is the linear kernel and $f_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\theta}^\top \mathbf{z}$; (ii) in classification, $\mathcal{K}$ is the cosine kernel, and $f_{\boldsymbol{\theta}}$ is a weighted prototype classifier on $\mathcal{E}$, where $\boldsymbol{\theta}$ is the weight.

Few-shot Regression on *Sine* and *Sale*

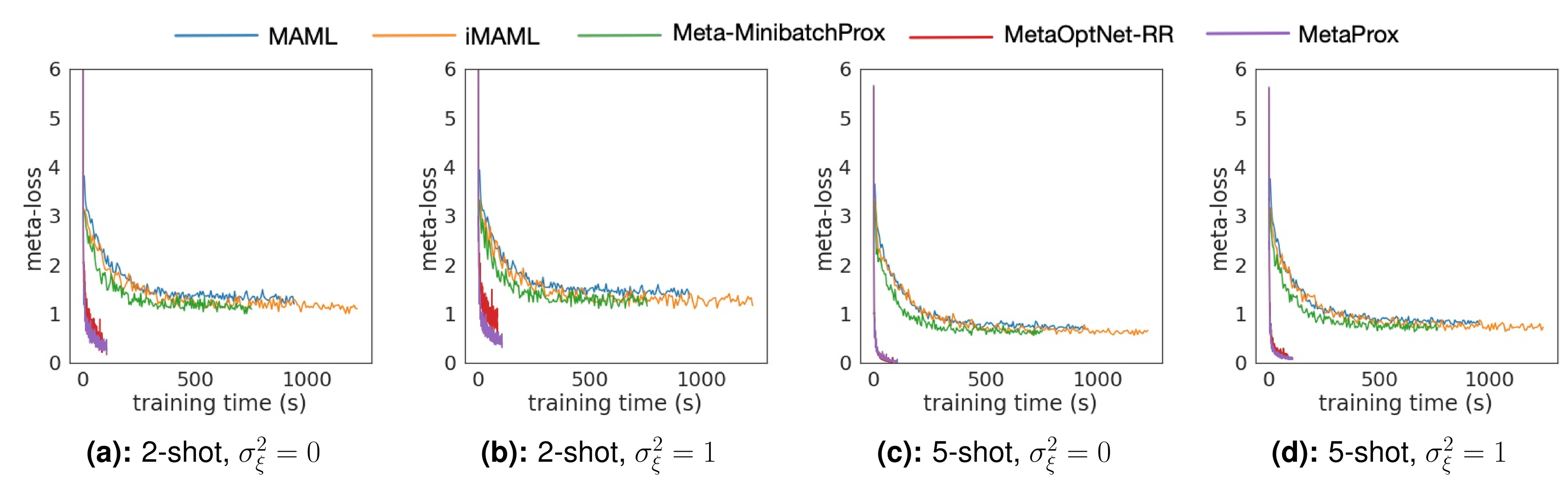

Figure 1: Convergence curves for few-shot sinusoid regression. MetaProx converges much faster and better than the non-kernel-based methods (MAML, iMAML and Meta-MinibatchProx). In the 2-shot settings, MetaProx converges to a loss smaller than that of MetaOptNet-RR.

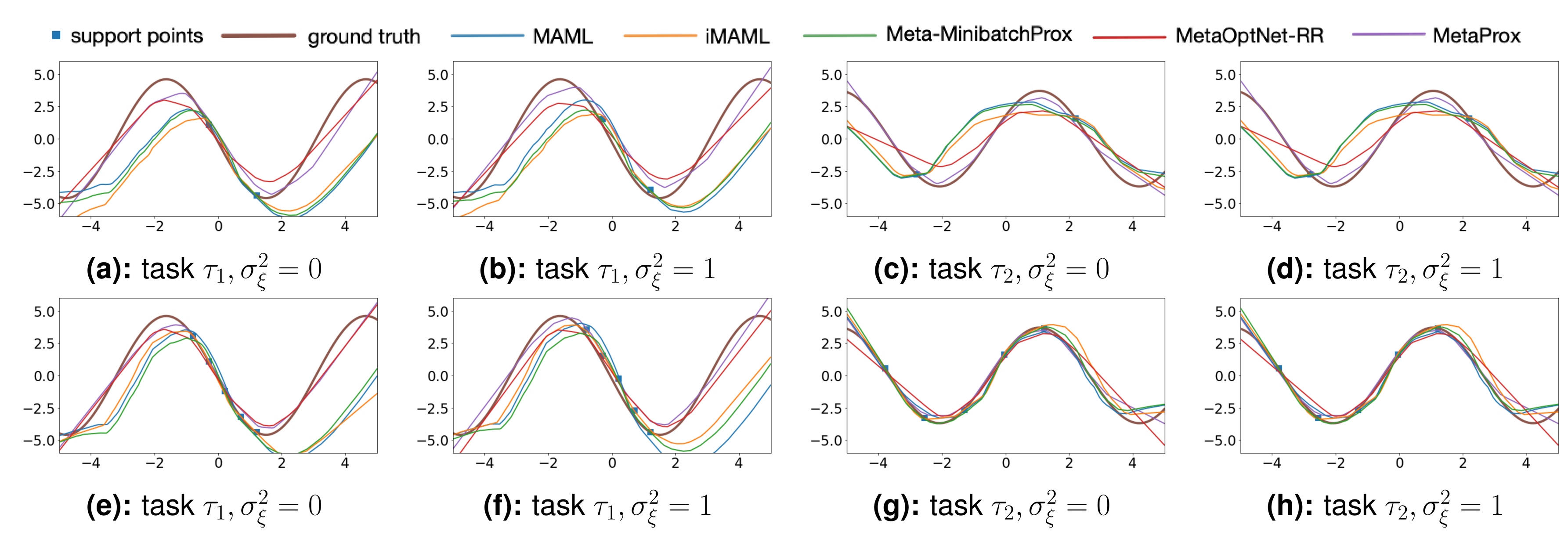

Figure 2: Sinusoid regression: Two meta-testing tasks τ_1 and τ_2 with different σ_ξ's in 2-shot (a–d) and 5-shot (e–h) settings. MetaProx always fits the target curve well.

Table 1: Average MSE (with 95% confidence intervals) of few-shot regression on the *Sine* and *Sale* datasets. MetaProx (with the learned $f_{\boldsymbol{\theta}}$) performs better than MetaOptNet-RR.

	Sine (2-shot)		*Sine* (5-shot)		*Sale*	
	noise-free	noisy	noise-free	noisy	1-shot	5-shot
CommonMean	4.58 ± 0.07	4.59 ± 0.07	4.29 ± 0.06	4.31 ± 0.06	0.090	0.074
MAML	1.24 ± 0.12	1.91 ± 0.13	0.41 ± 0.03	1.15 ± 0.05	0.069	0.063
iMAML	1.12 ± 0.11	1.84 ± 0.10	0.38 ± 0.02	1.02 ± 0.05	0.068	0.063
Meta-MinibatchProx	1.15 ± 0.08	1.87 ± 0.09	0.37 ± 0.02	1.01 ± 0.03	0.081	0.064
MetaOptNet-RR	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**

Few-shot Regression on *QMUL*

Table 2: Average MSE (with 95% confidence intervals) of few-shot regression on *QMUL* (10-shot). MetaProx with the learnable $f_{\boldsymbol{\theta}}$ reduces the errors of MetaOptNet-RR by half.

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

Few-shot Classification on *mini-ImageNet*

Table 3: Accuracies (with 95% confidence intervals) of 5-way few-shot classification using *Conv4*.

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

Table 4: Accuracies (with 95% confidence intervals) of 5-way few-shot classification using *ResNet-12*.

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

As can be seen from Table 3 and Table 4, compared with MetaOptNet-SVM, MetaProx performs better due to the learnable regularizer.

Summary

1. We proposed an effective meta-regularization algorithm (MetaProx) by kernelized proximal regularization.
2. MetaProx combines deep kernel and meta-regularization. By reformulating the 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 by the meta-learner.
3. Extensive experiments on standard datasets for regression and classification verify the effectiveness of the proposed meta-regularization algorithm.

Reference

[1] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. Learning to learn around a common mean. In *NeurIPS* 2018.

[2] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *NeurIPS* 2019.

[3] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *CVPR* 2019.