Efficient Meta Learning via Minibatch Proximal Update

Pan Zhou

Joint work with Xiao-Tong Yuan, Huan Xu, Shuicheng Yan, Jiashi Feng

National University of Singapore

pzhou@u.nus.edu

Dec 11, 2019

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** w from observed tasks such that

w **is close to the optimal models of new similar tasks, promoting new task learning**

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ **is close to the optimal models of new similar tasks, promoting new task learning**

- **Training model:** given a task distribution $\mathcal{T}$, we minimize a **bi-level** meta learning model

$$\min_{\boldsymbol{w}} \ \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2,$$

where $T_i \sim \mathcal{T}$ has K training samples $D_{T_i} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$

$\mathcal{L}_{D_{T_i}} = \frac{1}{K}\sum_{(\boldsymbol{x},\boldsymbol{y})\in D_{T_i}} \ell(f(\boldsymbol{w},\boldsymbol{x}),\boldsymbol{y})$ is empirical loss with predictor f and loss ℓ.

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ is close to the optimal models of new similar tasks, promoting new task learning

- **Training model:** given a task distribution $\mathcal{T}$, we minimize a **bi-level** meta learning model

update task-specific solution

$$\min_{\boldsymbol{w}} \ \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \boxed{\mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2},$$

where $T_i \sim \mathcal{T}$ has K training samples $D_{T_i} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$

$\mathcal{L}_{D_{T_i}} = \frac{1}{K}\sum_{(\boldsymbol{x},\boldsymbol{y})\in D_{T_i}} \ell(f(\boldsymbol{w}, \boldsymbol{x}), \boldsymbol{y})$ is empirical loss with predictor f and loss ℓ.

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ is close to the optimal models of new similar tasks, promoting new task learning

- **Training model:** given a task distribution $\mathcal{T}$, we minimize a **bi-level** meta learning model

update the prior model

$$\min_{\boldsymbol{w}} \boxed{\min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2,}$$

where $T_i \sim \mathcal{T}$ has K training samples $D_{T_i} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$

$\mathcal{L}_{D_{T_i}} = \frac{1}{K}\sum_{(\boldsymbol{x},\boldsymbol{y})\in D_{T_i}} \ell(f(\boldsymbol{w}, \boldsymbol{x}), \boldsymbol{y})$ is empirical loss with predictor f and loss ℓ.

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ is close to the optimal models of new similar tasks, promoting new task learning

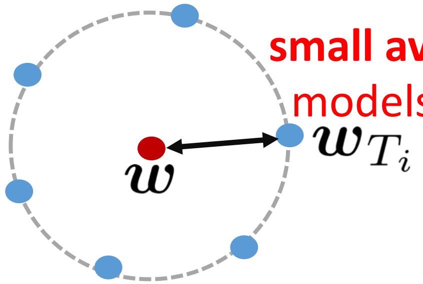
- **Training model:** given a task distribution $\mathcal{T}$, we minimize a **bi-level** meta learning model

update the prior model

$$\min_{\boldsymbol{w}} \boxed{\min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2,}$$

where $T_i \sim \mathcal{T}$ has K training samples $D_{T_i} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$

$\mathcal{L}_{D_{T_i}} = \frac{1}{K}\sum_{(\boldsymbol{x},\boldsymbol{y})\in D_{T_i}} \ell(f(\boldsymbol{w},\boldsymbol{x}),\boldsymbol{y})$ is empirical loss with predictor f and loss ℓ.



small average distance to optimum models of all tasks in expectation

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ **is close to the optimal models of new similar tasks, promoting new task learning**

- **Test model:** given a randomly sample a task $T \sim \mathcal{T}$ consisting of K samples $D_T = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^K$

$$\min_{\boldsymbol{w}_T} \; \mathcal{L}_{D_T}(\boldsymbol{w}_T) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_T\|_2^2,$$

Meta Learning via Minibatch Proximal Update (Meta-MinibatchProx)

Meta-MinibatchProx learns a good **prior model initialization** $\boldsymbol{w}$ from observed tasks such that

$\boldsymbol{w}$ is close to the optimal models of new similar tasks, promoting new task learning

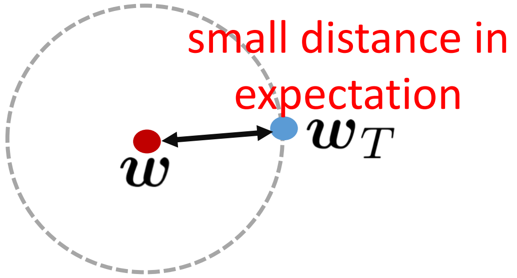
- **Test model:** given a randomly sample a task $T \sim \mathcal{T}$ consisting of K samples $D_T = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{K}$

$$\min_{\boldsymbol{w}_T} \ \mathcal{L}_{D_T}(\boldsymbol{w}_T) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_T\|_2^2,$$

- **Benefit:** a few data is sufficient for adaptation

prior model $\boldsymbol{w}$ is close to optimum $\boldsymbol{w}_T$

when training and test tasks are from a same distribution $\mathcal{T}$.



Optimization Algorithm

We use SGD based algorithm to solve bi-level training model :

$$\min_{\boldsymbol{w}} \; \left\{F(\boldsymbol{w}) := \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\right\}$$

Optimization Algorithm

We use SGD based algorithm to solve bi-level training model :

$$\min_{\boldsymbol{w}} \ \left\{F(\boldsymbol{w}) := \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\right\}$$

- Step1. select a mini-batch of task $\{T_i\}$.

Optimization Algorithm

We use SGD based algorithm to solve bi-level training model :

$$\min_{\boldsymbol{w}} \ \left\{ F(\boldsymbol{w}) := \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2 \right\}$$

- Step1. select a mini-batch of task $\{T_i\}$.
- Step2. for T_i, compute an approximate minimizer:

$$\boldsymbol{w}_{T_i} \approx \operatorname{argmin}_{\boldsymbol{w}_{T_i}} \{g(\boldsymbol{w}_{T_i}) =: \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\} \ \text{s.t.} \|\nabla g(\boldsymbol{w}_{T_i})\|_2^2 \leq \epsilon_s$$

Optimization Algorithm

We use SGD based algorithm to solve bi-level training model :

$$\min_{\boldsymbol{w}} \ \left\{F(\boldsymbol{w}) := \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\right\}$$

- Step1. select a mini-batch of task $\{T_i\}$.
- Step2. for T_i, compute an approximate minimizer:

$$\boldsymbol{w}_{T_i} \approx \mathrm{argmin}_{\boldsymbol{w}_{T_i}} \{g(\boldsymbol{w}_{T_i}) =: \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\} \text{ s.t.} \|\nabla g(\boldsymbol{w}_{T_i})\|_2^2 \leq \epsilon_s$$

- Step3. update the prior model

$$\boldsymbol{w} = \boldsymbol{w} - \eta_s \lambda(\boldsymbol{w} - \frac{1}{b_s}\sum_{i=1}^{b_s} \boldsymbol{w}_{T_i})$$

Optimization Algorithm

We use SGD based algorithm to solve bi-level training model :

$$\min_{\boldsymbol{w}} \ \{F(\boldsymbol{w}) := \min_{\boldsymbol{w}_{T_i}} \sum_{i=1}^{n} \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\}$$

- Step1. select a mini-batch of task $\{T_i\}$.
- Step2. for T_i, compute an approximate minimizer:

$$\boldsymbol{w}_{T_i} \approx \operatorname{argmin}_{\boldsymbol{w}_{T_i}} \{g(\boldsymbol{w}_{T_i}) =: \mathcal{L}_{D_{T_i}}(\boldsymbol{w}_{T_i}) + \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{w}_{T_i}\|_2^2\} \text{ s.t.} \|\nabla g(\boldsymbol{w}_{T_i})\|_2^2 \leq \epsilon_s$$

- Step3. update the prior model

$$\boldsymbol{w} = \boldsymbol{w} - \eta_s \lambda(\boldsymbol{w} - \frac{1}{b_s}\sum_{i=1}^{b_s} \boldsymbol{w}_{T_i})$$

Theorem 1 (convergence guarantees, informal).

(1) Convex setting, i.e. convex $\phi_{D_{T_i}}(\boldsymbol{w})$. We prove $\mathbb{E}[\|\boldsymbol{w}^S - \boldsymbol{w}^*\|_2^2] \leq \mathcal{O}(\frac{1}{S})$.

(2) Nonconvex setting, i.e. smooth $\phi_{D_{T_i}}(\boldsymbol{w})$. We prove $\mathbb{E}_s[\|\nabla F(\boldsymbol{w}^s)\|_2^2] \leq \mathcal{O}(\frac{1}{\sqrt{S}})$.

Generalization Performance Guarantee

- Ideally, for a given task T, one should train the model on the population risk

$$\text{Population solution: } \boldsymbol{w}_P^* = \text{argmin}_{\boldsymbol{w}_T} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim T} \ell(f(\boldsymbol{w}_T, \boldsymbol{x}), \boldsymbol{y}).$$

- In practice, we only has K samples and adapt the prior model $\boldsymbol{w}^*$ to the new task:

$$\text{Empirical solution: } \boldsymbol{w}_T^* = \text{argmin}_{\boldsymbol{w}_T} \mathcal{L}_{D_T}(\boldsymbol{w}_T) + \frac{\lambda}{2}\|\boldsymbol{w}^* - \boldsymbol{w}_T\|_2^2.$$

Generalization Performance Guarantee

- Ideally, for a given task T, one should train the model on the population risk

$$\text{Population solution: } \boldsymbol{w}_P^* = \text{argmin}_{\boldsymbol{w}_T} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim T} \ell(f(\boldsymbol{w}_T, \boldsymbol{x}), \boldsymbol{y}).$$

- In practice, we only has K samples and adapt the prior model $\boldsymbol{w}^*$ to the new task:

$$\text{Empirical solution: } \boldsymbol{w}_T^* = \text{argmin}_{\boldsymbol{w}_T} \mathcal{L}_{D_T}(\boldsymbol{w}_T) + \frac{\lambda}{2}\|\boldsymbol{w}^* - \boldsymbol{w}_T\|_2^2.$$

- **Since $\boldsymbol{w}_P^* \neq \boldsymbol{w}_T^*$, why $\boldsymbol{w}_T^*$ is good for generalization in few-shot learning problem?**

Theorem 2 (generalization performance guarantee, informal).

Suppose each loss $\phi_{D_{T_i}}(\boldsymbol{w})$ is convex and is smooth. Let $D_T = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^K \sim T$. Then we have

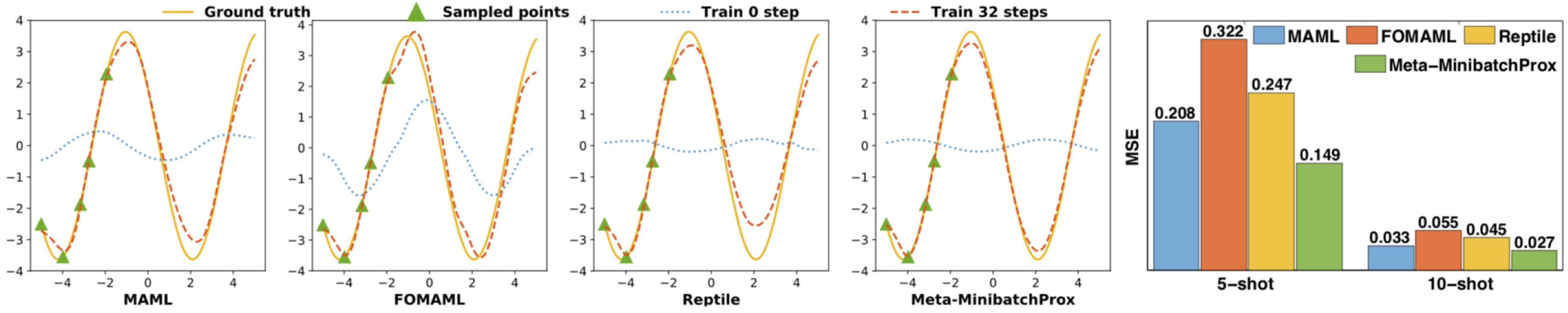
$$\mathbb{E}_{T\sim\mathcal{T}} \mathbb{E}_{D_T\sim T}(\mathcal{L}(\boldsymbol{w}_T^*) - \mathcal{L}(\boldsymbol{w}_P^*)) \leq \frac{c}{\sqrt{K}} \mathbb{E}[\|\boldsymbol{w}^* - \boldsymbol{w}_P^*\|_2^2].$$

Remark: strong generalization performance, as our training model guarantee

prior $\boldsymbol{w}^*$ is close to the optimum model $\boldsymbol{w}_P^*$.

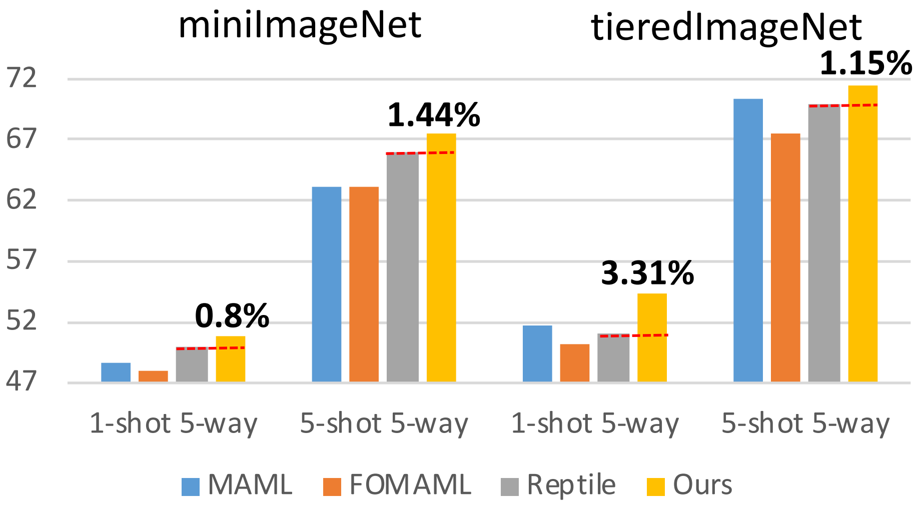
Experimental results

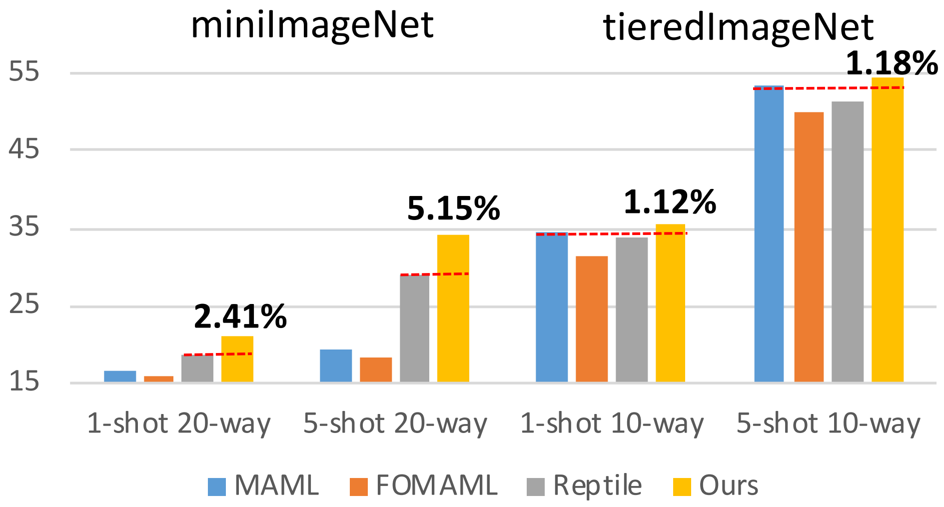
Few-shot regression : smaller mean square error (MSE) between prediction and ground truth



(a) Visual illustration (b) MSE

Few-shot classification: higher classification accuracy





POSTER # 26

05:00 -- 07:00 PM @ East Exhibition Hall B + C

Thanks!