Optimization-based Meta-Learning

Outline

- Brief overview of optim-based meta-learning
  - Meta-learning models aim to learn **an global/initial params** and **a update rule** of how to quickly adapt to individual tasks (i.e., adjusting global initial params to individual tasks)
  - Meta-learning objective is to minimize the expected generalization loss of the meta-learner on the task space
- LSTM-based meta-learning (Meta-LSTM) (Ravi and Larochelle, 2016)
  - Meta-LSTM uses a sequential LSTM update rule
- Meta-SGD (Li et al., 2017)
  - Meta-SGD learns the global learning rate and update direction α addition to the initial params
- Meta-Curvature (Park and Oliva, 2020)
  - Meta-Curvature further improves MAML by adding second-order information into the gradient
    - the additional info is is decomposed into 3 component matrices and is multiplied to gradient during inner gradient update steps
- IMO, the current trend is modifying and adding components to the update rules to learn more about individual tasks, but the components are controlled by some global learnable variables

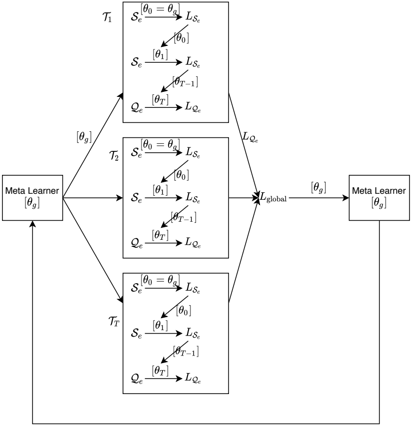
Optim-based Meta-Learning: Intuition

- Meta-learning objective is to minimize the expected generalization loss of the meta-learner on the task space

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \underbrace{\mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\underbrace{\theta_g - \alpha \nabla_{\theta_g} L_{\mathcal{S}_e(\theta_g)}}_{\text{inner}})}]}_{\text{outer}}$$

  - **inner-loop:** given an individual task $\mathcal{T}_e$, the **learner** (params θ_e) is optimized based on the loss of task support set $L_{\mathcal{S}_e}$
  - **outer-loop:** the loss of task query set $L_{\mathcal{Q}_e}$ is computed with the updated learner and is accumulated to the generalization loss $L_{\text{global}} = \sum_e L_{\mathcal{Q}_e}$. the **meta-learner** (params θ_g) is then optimized based the generalization loss.

Model-Agnostic Meta-Learning (MAML): Model

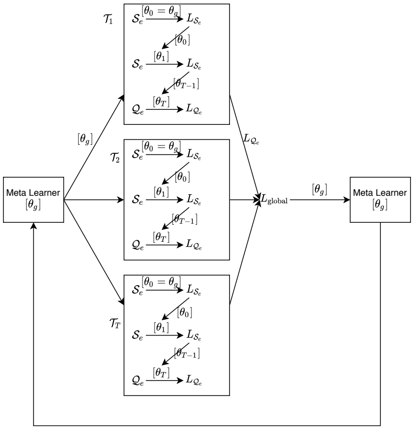


Algorithm : MAML's training procedure

Require: train meta-set $\mathscr{T}^{\text{train}}$
Return: global params θ_g that are adaptable to many tasks

initialize θ_g
for epoch in num epochs **do**
  for episode e in num episodes **do**
    sample a local task $\mathcal{T}_e = (\mathcal{S}_e, \mathcal{Q}_e)$ from $\mathscr{T}^{\text{train}}$
    $\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
    set $\theta_0 = \theta_g$
    for step t in num steps **do**
      compute loss of the local task support set
      $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
      compute gradient and update local params for the local task
      $\theta_t = \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
    end for
    $\mapsto$ **outer loop**: learn global params via all the losses of all task query sets
    compute loss of the local task query set with local params
    $L_{\mathcal{Q}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{Q}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_T), y)$
    accumulate the loss of the local task query set to the global loss
    $L_{\text{global}} \mathrel{+}= L_{\mathcal{Q}_e}$
  end for
  compute global gradient (w.r.t. sum of all local losses) and update global params
  $\theta_g = \theta_g - \beta \nabla_{\theta_g} L_{\text{global}}$
end for

Model-Agnostic Meta-Learning (MAML): Model



Algorithm : MAML's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g
Return: individual task params θ_T after T update steps

1: $\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
2: set $\theta_0 = \theta_g$
3: **for** step t in num steps **do**
4: compute loss of the local task support set
$L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
5: compute gradient and update local params for the local task
$\theta_t = \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
6: **end for**

Algorithm : MAML's training procedure, outer-loop

Require: task query set $\mathcal{Q}_e$, individual task params θ_T
Return: loss of task query set $L_{\mathcal{Q}_e}$

1: $\mapsto$ **outer loop**: learn global params via all the losses of all task query sets
2: compute loss of the local task query set with local params
$L_{\mathcal{Q}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{Q}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_T), y)$
3: accumulate the loss of the local task query set to the global loss
$L_{\text{global}} \mathrel{+}= L_{\mathcal{Q}_e}$

Meta-LSTM vs MAML: Model

Algorithm : MAML's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g
Return: individual task params θ_T after T update steps

$\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
set $\theta_0 = \theta_g$
for step t in num steps **do**
  compute loss of the local task support set
  $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
  compute gradient and update local params for the local task
  $\theta_t = \theta_{t-1} - \alpha\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
end for

Algorithm : Meta-LSTM's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g
Return: individual task params θ_T after T update steps

$\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set sequentially with the LSTM mechanism
set $\theta_0 = \theta_g$
for step t in num steps **do**
  compute loss of the local task support set
  $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
  compute input gate i_t and forget gate f_t
  $i_t = \sigma(\boldsymbol{W}^I \cdot [\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}, L_{\mathcal{S}_e}, \theta_{t-1}, i_{t-1}] + \boldsymbol{b}^I)$
  $f_t = \sigma(\boldsymbol{W}^F \cdot [\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}, L_{\mathcal{S}_e}, \theta_{t-1}, f_{t-1}] + \boldsymbol{b}^F)$
  update local params for the local task according to LSTM rule
  $\theta_t = f_t \odot \theta_{t-1} - i_t \odot \nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
end for

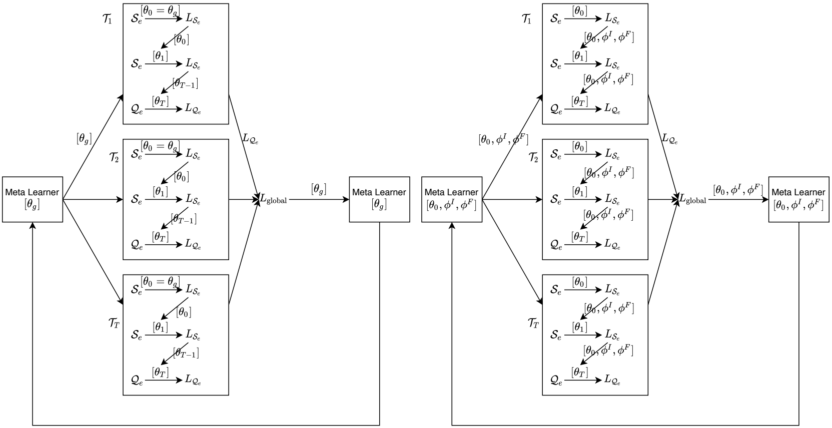
- MAML's objective

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_{t-1} - \alpha\nabla_{\theta_{t-1}} L_{\mathcal{S}_e})}]$$

- Meta-SGD uses a sequential LSTM update rule, its objective becomes

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(f_{t-1}\odot\theta_{t-1} - i_{t-1}\odot\nabla_{\theta_{t-1}} L_{\mathcal{S}_e})}]$$

Meta-LSTM vs MAML: Model



MAML

Meta-LSTM

Meta-SGD vs MAML: Model

Algorithm : MAML's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g
Return: individual task params θ_T after T update steps

$\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
set $\theta_0 = \theta_g$
for step t in num steps **do**
  compute loss of the local task support set $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
  compute gradient and update local params for the local task $\theta_t = \theta_{t-1} - \alpha\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
end for

Algorithm : Meta-SGD's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g, global lr α_g
Return: individual task params θ_T after T update steps

$\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set using global params θ_g and global lr α_g
set $\theta_0 = \theta_g$
for step t in num steps **do**
  compute loss of the local task support set $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
  compute gradient and update local params for the local task $\theta_t = \theta_{t-1} - \alpha_g\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
end for

- MAML's objective

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_{t-1}-\alpha\nabla_{\theta_{t-1}}L_{\mathcal{S}_e})}]$$

- In addition to learning global initial params as in MAML, Meta-SGD also learns the global learning rate (and update direction) alpha

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_{t-1}-\alpha_g\nabla_{\theta_{t-1}}L_{\mathcal{S}_e})}]$$

Meta-Curvature vs MAML: Model

Algorithm : MAML's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g
Return: individual task params θ_T after T update steps
1: $\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
2: set $\theta_0 = \theta_g$
3: **for** step t in num steps **do**
4: compute loss of the local task support set $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
5: compute gradient and update local params for the local task $\theta_t = \theta_{t-1} - \alpha\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}$
6: **end for**

Algorithm : Meta-Curvature's training procedure, inner-loop

Require: task support set $\mathcal{S}_e$, global params θ_g, curvature params $\boldsymbol{M}_f, \boldsymbol{M}_i, \boldsymbol{M}_o$
Return: individual task params θ_T after T update steps
1: $\mapsto$ **inner loop**: learn local params for the local task via the loss of task support set
2: set $\theta_0 = \theta_g$
3: **for** step t in num steps **do**
4: compute loss of the local task support set $L_{\mathcal{S}_e} = \sum_{(\boldsymbol{x},y)\in\mathcal{S}_e} L_{\text{cls}}(f(\boldsymbol{x};\theta_{t-1}), y)$
5: compute gradient and update local params for the local task $\theta_t = \theta_{t-1} - \alpha\,\text{MC}(\nabla_{\theta_{t-1}} L_{\mathcal{S}_e}; \boldsymbol{M}_f, \boldsymbol{M}_i, \boldsymbol{M}_o)$
6: **end for**

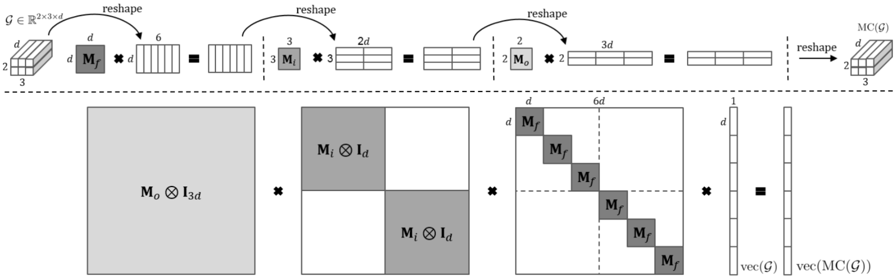
- MAML's objective

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_{t-1}-\alpha\nabla_{\theta_{t-1}} L_{\mathcal{S}_e})}]$$

- Meta-Curvature further improves MAML by adding second-order information into the gradient

$$\min_{\theta_g} \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_e)}] = \mathbb{E}_{\mathcal{T}_e}[L_{\mathcal{Q}_e(\theta_{t-1}-\alpha\,\text{MC}(\nabla_{\theta_{t-1}} L_{\mathcal{S}_e};\boldsymbol{M}_f,\boldsymbol{M}_i,\boldsymbol{M}_o))}]$$

Meta-Curvature vs MAML: Model



- The second-order information is characterized/decomposed by into 3 component matrices, the info is multiplied to the gradient during updates

$$\mathrm{MC}(\nabla_{\theta_g} L_{\mathcal{S}_e(\theta_g)}; \boldsymbol{M}_f, \boldsymbol{M}_i, \boldsymbol{M}_o) = \nabla_{\theta_g} L_{\mathcal{S}_e(\theta_g)} \times_3 \boldsymbol{M}_f \times_2 \boldsymbol{M}_i \times_1 \boldsymbol{M}_o$$

  - all component matrices are updated globally similar to global initial params

FSL Benchmark

Model	*mini*ImageNet test accuracy 1-shot	5-shot
Matching networks (Vinyals et al., 2016)	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
Meta-learner LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML (Finn et al., 2017)	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
LLAMA (Grant et al., 2018)	$49.40 \pm 1.83\%$	-
REPTILE (Nichol & Schulman, 2018)	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
PLATIPUS (Finn et al., 2018)	$50.13 \pm 1.86\%$	-
Meta-SGD (our features)	$54.24 \pm 0.03\%$	$70.86 \pm 0.04\%$
SNAIL (Mishra et al., 2018)	$55.71 \pm 0.99\%$	$68.88 \pm 0.92\%$
(Gidaris & Komodakis, 2018)	$56.20 \pm 0.86\%$	$73.00 \pm 0.64\%$
(Bauer et al., 2017)	$56.30 \pm 0.40\%$	$73.90 \pm 0.30\%$
(Munkhdalai et al., 2017)	$57.10 \pm 0.70\%$	$70.04 \pm 0.63\%$
DEML+Meta-SGD (Zhou et al., 2018)[4]	$58.49 \pm 0.91\%$	$71.28 \pm 0.69\%$
TADAM (Oreshkin et al., 2018)	$58.50 \pm 0.30\%$	$76.70 \pm 0.30\%$
(Qiao et al., 2017)	$59.60 \pm 0.41\%$	$73.74 \pm 0.19\%$
LEO (ours)	$\mathbf{61.76 \pm 0.08\%}$	$\mathbf{77.59 \pm 0.12\%}$

Model	*tiered*ImageNet test accuracy 1-shot	5-shot
MAML (deeper net, evaluated in Liu et al. (2018))	$51.67 \pm 1.81\%$	$70.30 \pm 0.08\%$
Prototypical Nets (Ren et al., 2018)	$53.31 \pm 0.89\%$	$72.69 \pm 0.74\%$
Relation Net (evaluated in Liu et al. (2018))	$54.48 \pm 0.93\%$	$71.32 \pm 0.78\%$
Transductive Prop. Nets (Liu et al., 2018)	$57.41 \pm 0.94\%$	$71.55 \pm 0.74\%$
Meta-SGD (our features)	$62.95 \pm 0.03\%$	$79.34 \pm 0.06\%$
LEO (ours)	$\mathbf{66.33 \pm 0.05\%}$	$\mathbf{81.44 \pm 0.09\%}$

Table 1: Test accuracies on *mini*ImageNet and *tiered*ImageNet. For each dataset, the first set of results use convolutional networks, while the second use much deeper residual networks, predominantly in conjuction with pre-training.

Thank you !

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to Learn Quickly for Few-Shot Learning. *arXiv:1707.09835 [cs]*, 2017.

Eunbyung Park and Junier B. Oliva. Meta-Curvature. In *Proceedings of the NeurIPS 2019*, 2020.

Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. In *Proceedings of the ICLR 2016*, 2016.