ICML2019-meta learning MAML改进

Hierarchically Structured Meta-learning

Fast Context Adaptation via Meta-Learning

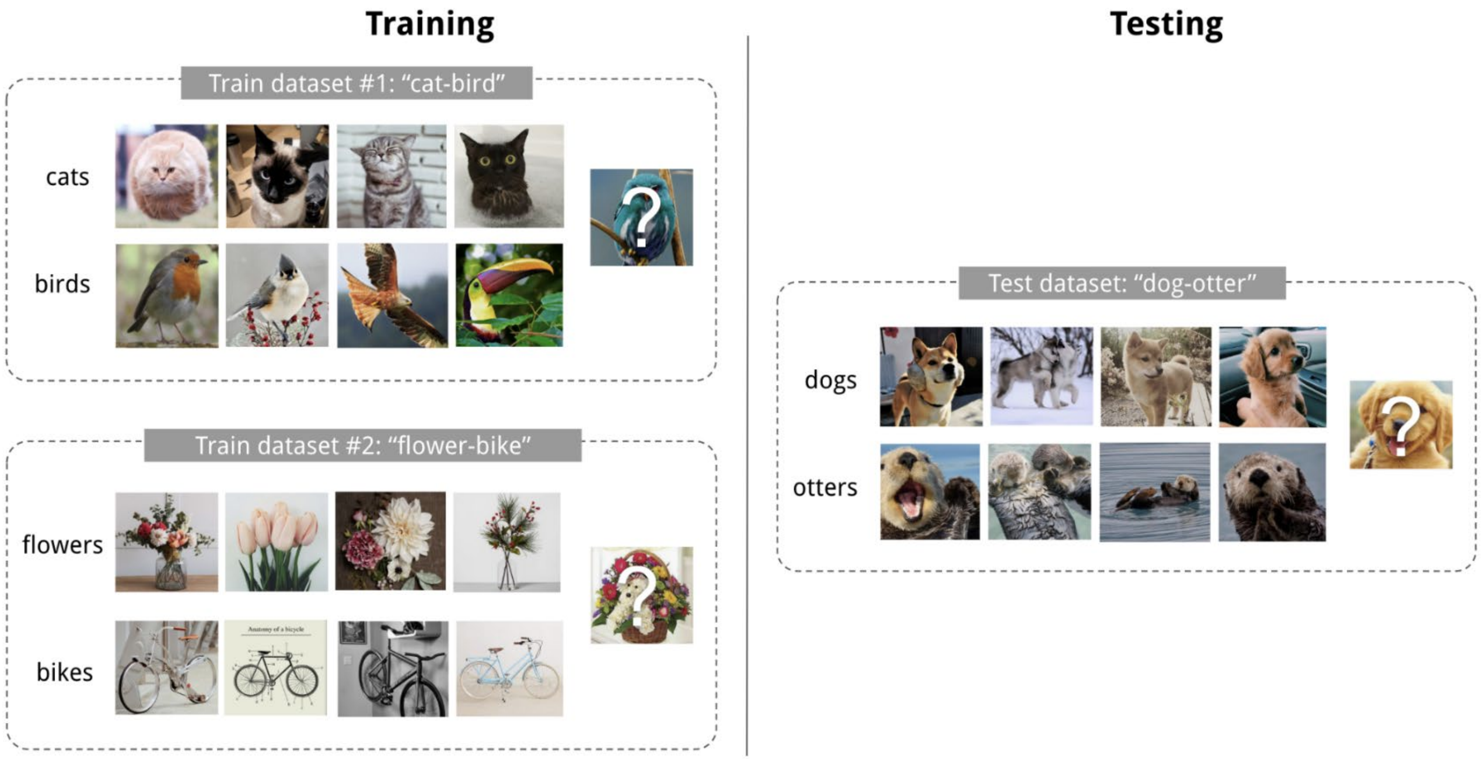
Define the Meta-Learning Problem

■ Problem Setting

$\{\mathcal{T}_1, \ldots, \mathcal{T}_{N_t}\}$ sampled from p

for each $\mathcal{T}_i \in \mathbf{T}$ $\quad \mathcal{D}_i^{\text{train}} = \{(x, y)^{i,m}\}_{m=1}^{M_i^{\text{train}}}, \quad \mathcal{D}_i^{\text{test}} = \{(x, y)^{i,m}\}_{m=1}^{M_i^{\text{test}}}$

$$\min_{\theta_0} \sum_{i=1}^{N_t} \mathcal{L}(f_{\theta_0 - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}^{tr}_{\mathcal{T}_i})}, \mathcal{D}^{te}_{\mathcal{T}_i})$$



4-shot 2-class

4-shot 2-way

Meta Learning

Meta-learning, also known as "learning to learn", intends to design models that can learn new skills or adapt to new environments rapidly with a few training examples.

■ common approaches

1. use a recurrent neural network equipped with either external or internal memory storing and querying meta-knowledge **model based**
2. Learn an effective distance metric between examples (Matching Networks) **metric based**
3. learn a meta-optimizer which can quickly optimize the model parameters
4. learn an appropriate initialization from which the model parameters can be updated within a few gradient steps (**MAML**)

Optimization based

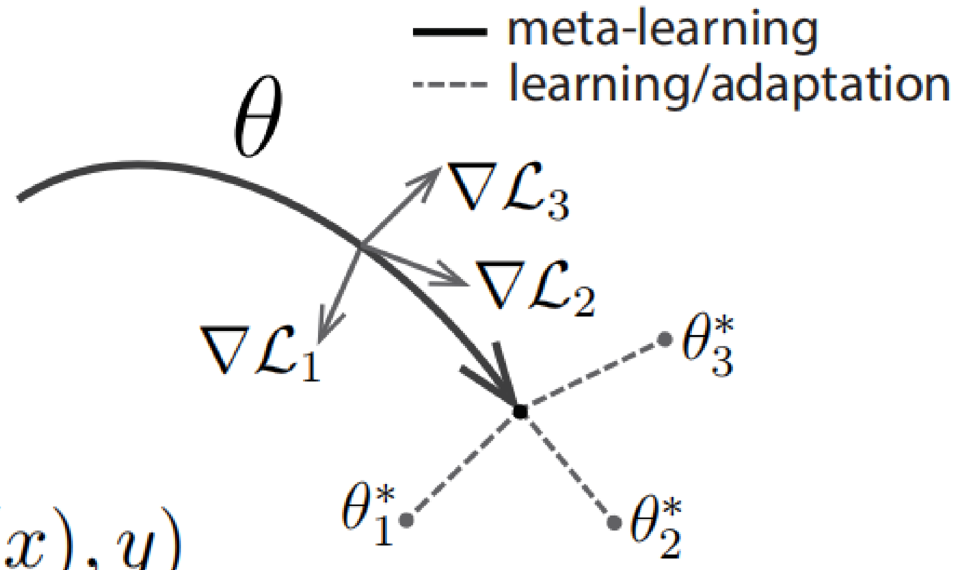
Model-Agnostic Meta-Learning

■ Intuition

MAML learns an **initialisation** for the parameters $\boldsymbol{\theta}$ of a model f_θ such that, given a new task, a good model for that task can be learned with only a small number of gradient steps and data points

inner loop

$$\theta_i = \theta - \alpha \nabla_\theta \frac{1}{M^i_{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_\theta(x), y)$$



outer loop

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i\in\mathbf{T}} \frac{1}{M^i_{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}(x), y)$$

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta learning for fast adaptation of deep networks. ICML-2017 1162引用

Model-Agnostic Meta-Learning

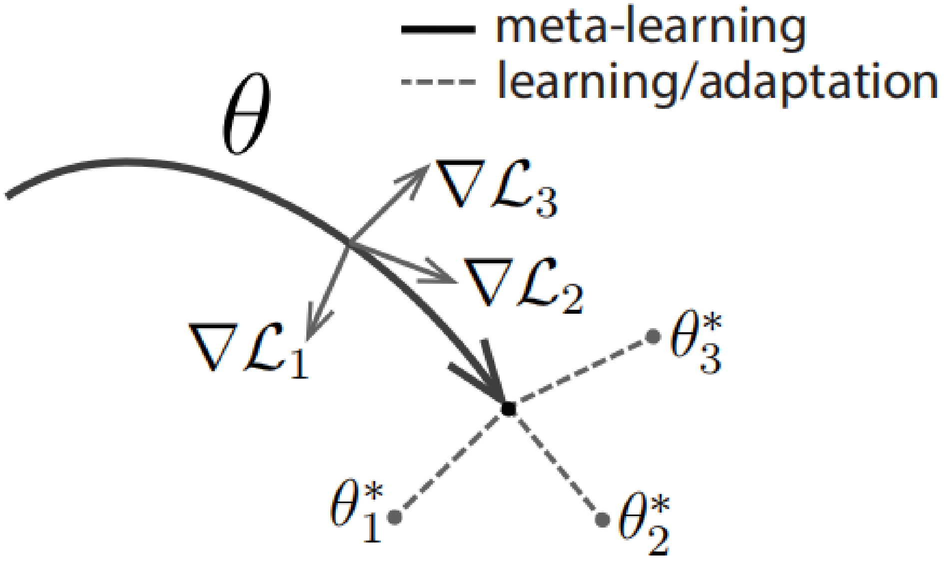
■ Intuition

MAML learns an **initialisation** for the parameters $\boldsymbol{\theta}$ of a model f_θ such that, given a new task, a good model for that task can be learned with only a small number of gradient steps and data points

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while



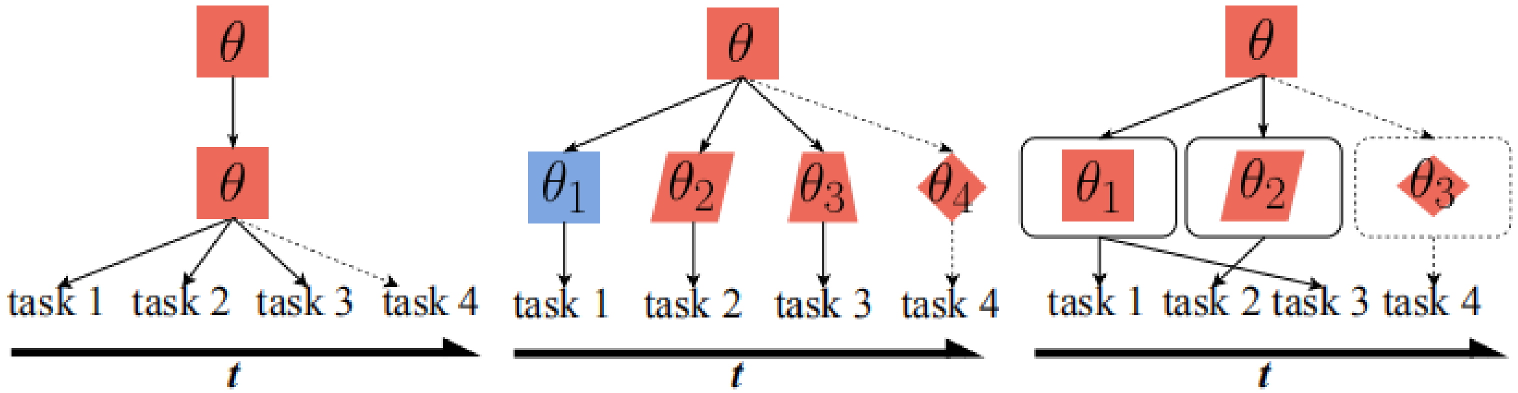
Hierarchically Structured Meta-learning

■ Intuition

A critical challenge in meta-learning is task uncertainty and heterogeneity, which can not be handled via globally sharing knowledge among tasks

■ Method

Based on MAML, propose a **h**ierarchically **s**tructured **m**eta-**l**earning (HSML) algorithm that explicitly tailors the transferable knowledge to different clusters of tasks

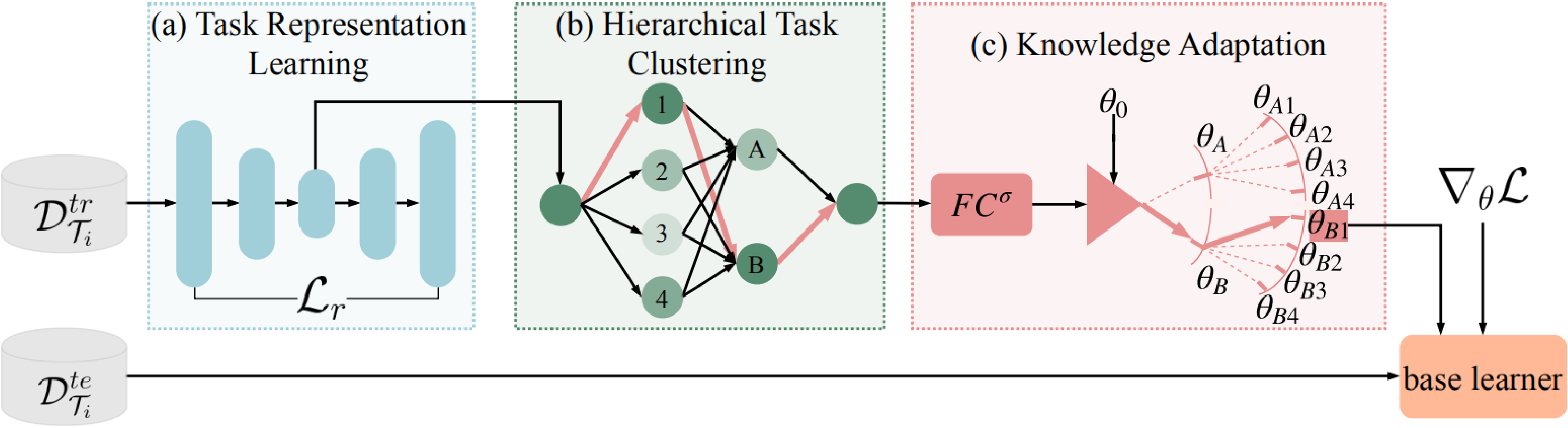


(a) : globally shared (b) : task specific (c) : HSML

balance
generalization and customization

HSML-Method

■ Framework



HSML-Method

(a)Task Representation Learning

1. high representational capacity
2. permutational invariance to its inputs

Pooling Autoencoder Aggregator(PAA)

$$\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr}) = \sum_{j=1}^{n^{tr}} \|\mathrm{FC}_{dec}(\mathbf{g}_{i,j}) - \mathcal{F}(\mathbf{x}_{i,j}^{tr}, \mathbf{y}_{i,j}^{tr})\|_2^2,$$

$\mathbf{g}_{i,j} = \mathrm{FC}_{enc}(\mathcal{F}(\mathbf{x}_{i,j}^{tr}, \mathbf{y}_{i,j}^{tr}))$ is the representation for the j-th example

$\mathcal{F}(\cdot,\cdot)$ preliminarily embed both features and predictions, varies from dataset to dataset

$$\mathbf{g}_i = \mathrm{Pool}_{j=1}^{n^{tr}}(\mathbf{g}_{i,j}),$$

$\mathbf{g}_i \in \mathbb{R}^d$ is the desired representation of task $\mathcal{T}_i$

Pool denotes a max or mean pooling operator over examples

HSML-Method

(a)Task Representation Learning

1. high representational capacity
2. permutational invariance to its inputs

Recurrent Autoencoder Aggregator(RAA)

Different from the pooling autoencoder, examples are sequentially fed into the recurrent autoencoder

$$\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr}) = \sum_{j=1}^{n^{tr}} \| \overset{\mathbf{d}_{i,j}}{\cancel{\mathrm{FC}_{dec}(\mathbf{g}_{i,j})}} - \mathcal{F}(\mathbf{x}_{i,j}^{tr}, \mathbf{y}_{i,j}^{tr}) \|_2^2, \quad \mathbf{g}_i = \frac{1}{n^{tr}} \sum_{j}^{n^{tr}} (\mathbf{g}_{i,j})$$

$$\mathcal{F}(\mathbf{x}_{i,1}^{tr}, \mathbf{y}_{i,1}^{tr}) \rightarrow \mathbf{g}_{i,1} \rightarrow \cdots \rightarrow \mathbf{g}_{i,n^{tr}} \rightarrow \mathbf{d}_{i,n^{tr}} \rightarrow \cdots \rightarrow \mathbf{d}_{i,1}, \quad (4)$$

where $\forall j, \mathbf{g}_{i,j} = \mathrm{RNN}_{enc}(\mathcal{F}(\mathbf{x}_{i,j}^{tr}, \mathbf{y}_{i,j}^{tr}), \mathbf{g}_{i,j-1})$ and $\mathbf{d}_{i,j} = \mathrm{RNN}_{dec}(\mathbf{d}_{i,j+1})$ represent the learned representation and the reconstruction of the j-th example, respectively. Here

HSML-Method

(b)Hierarchical Task Clustering

1. **Assignment step (soft assignment)**

Assign a task represented in the k^l-th cluster of the l-th level $\mathbf{h}_i^{k^l} \in \mathbb{R}^d$

soft
$$p_i^{k^l \to k^{l+1}} = \frac{\exp\left(-\|(\mathbf{h}_i^{k^l} - \mathbf{c}_{k^{l+1}})/\sigma^l\|_2^2/2\right)}{\sum_{k^{l+1}=1}^{K^{l+1}} \exp\left(-\|(\mathbf{h}_i^{k^l} - \mathbf{c}_{k^{l+1}})/\sigma^l\|_2^2/2\right)},$$

computed by applying softmax over Euclidean distances between $\mathbf{h}_i^{k^l}$ and the learnable cluster centers $\{\mathbf{c}_{k^{l+1}}\}_{k^{l+1}=1}^{K^{l+1}}$,

2. **Update step**

$$\mathbf{h}_i^{k^{l+1}} = \sum_{k^l=1}^{K^l} p_i^{k^l \to k^{l+1}} \tanh\left(\mathbf{W}^{k^{l+1}} \mathbf{h}_i^{k^l} + \mathbf{b}^{k^{l+1}}\right),$$

where $\mathbf{W}^{k^{l+1}} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^{k^{l+1}} \in \mathbb{R}^d$ are learned to transform from representations of the l-th to those of the $(l+1)$-th level

HSML-Method

(c)Knowledge Adaptation

Inspired by that similar tasks activate similar meta-parameters, design a cluster-specific parameter gate.

$$\mathbf{o}_i = \mathrm{FC}^{\sigma}_{\mathbf{W}_g}(\underline{\mathbf{g}_i \oplus \mathbf{h}_i^L})$$

Not only preserves but also reinforces the cluster-specific property of the parameter gate.

The globally transferable knowledge is adapted to the cluster-specific initial parameters via the parameter gate

$$\theta_{0i} = \theta_0 \circ \mathbf{o}_i$$

■ Continual Adaptation

a new task does not fit any of the learned task clusters, which implies that additional clusters should be introduced to the hierarchical clustering structure

$$\min_{\Theta} \sum_{i=1}^{N_t} \mathcal{L}(f_{\theta_{0i} - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}_{\mathcal{T}_i})}, \mathcal{D}^{te}_{\mathcal{T}_i}) + \xi \mathcal{L}_r(\mathcal{D}^{tr}_{\mathcal{T}_i})$$

reconstruction error

Algorithm

Algorithm 1 Meta-training of HSML

Require: $\mathcal{E}$: distribution over tasks; $\{K^1, \cdots, K^L\}$: # of clusters in each layer; α, β: stepsizes; μ: threshold

Randomly initialize Θ

while not done **do**

  if $\bar{\mathcal{L}}_{new} > \mu \bar{\mathcal{L}}_{old}$ **then**

    Increase the number of clusters

  end if

  Sample a batch of tasks $\mathcal{T}_i \sim \mathcal{E}$

  for all $\mathcal{T}_i$ **do**

    Sample $\mathcal{D}_{\mathcal{T}_i}^{tr}$, $\mathcal{D}_{\mathcal{T}_i}^{te}$ from $\mathcal{T}_i$

    Compute $\mathbf{g}_i$ in Eqn. (3) or Eqn. (5), $\mathbf{h}_i^L$ in Eqn. (7), and reconstruction error $\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})$

    Compute $\mathbf{o}_i$ in Eqn. (8) and evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}_{\mathcal{T}_i}^{tr})$

    Update parameters with gradient descent (taking one step as an example): $\theta_{\mathcal{T}_i} = \theta_{0i} - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_{\mathcal{T}_i}^{tr})$

  end for

  Update $\Theta \leftarrow \Theta - \beta \nabla_\Theta \sum_{i=1}^{N_t} \mathcal{L}(f_{\theta_{\mathcal{T}_i}}, \mathcal{D}_{\mathcal{T}_i}^{te}) + \xi \mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})$

  Compute $\bar{\mathcal{L}}_{new}$ and save $\bar{\mathcal{L}}_{old}$ for every Q rounds

end while

Continual Adaptation

$$\mathbf{g}_i = \mathrm{Pool}_{j=1}^{n^{tr}}(\mathbf{g}_{i,j}), \quad (3)$$

$$\mathbf{h}_i^{k^{l+1}} = \sum_{k^l=1}^{K^l} p_i^{k^l \to k^{l+1}} \tanh\left(\mathbf{W}^{k^{l+1}} \mathbf{h}_i^{k^l} + \mathbf{b}^{k^{l+1}}\right) \quad (7)$$

$$\mathbf{o}_i = \mathrm{FC}_{\mathbf{W}_g}^{\sigma}(\mathbf{g}_i \oplus \mathbf{h}_i^L) \quad (8)$$

$$\min_\Theta \sum_{i=1}^{N_t} \mathcal{L}(f_{\theta_{0i} - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_{\mathcal{T}_i}^{tr})}, \mathcal{D}_{\mathcal{T}_i}^{te}) + \xi \mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})$$

Experiment

Toy Regression

number of layers to be three with 4, 2, 1 clusters in each layer

Table 1. Performance of MSE ± 95% confidence intervals on toy regression tasks, averaged over 4,000 tasks. Both 5-shot and 10-shot results are reported.

Model	5-shot	10-shot
MAML	2.205 ± 0.121	0.761 ± 0.068
Meta-SGD	2.053 ± 0.117	0.836 ± 0.065
MT-Net	2.435 ± 0.130	0.967 ± 0.056
BMAML	2.016 ± 0.109	0.698 ± 0.054
MUMOMAML	1.096 ± 0.085	0.256 ± 0.028
HSML (ours)	$\mathbf{0.856 \pm 0.073}$	$\mathbf{0.161 \pm 0.021}$

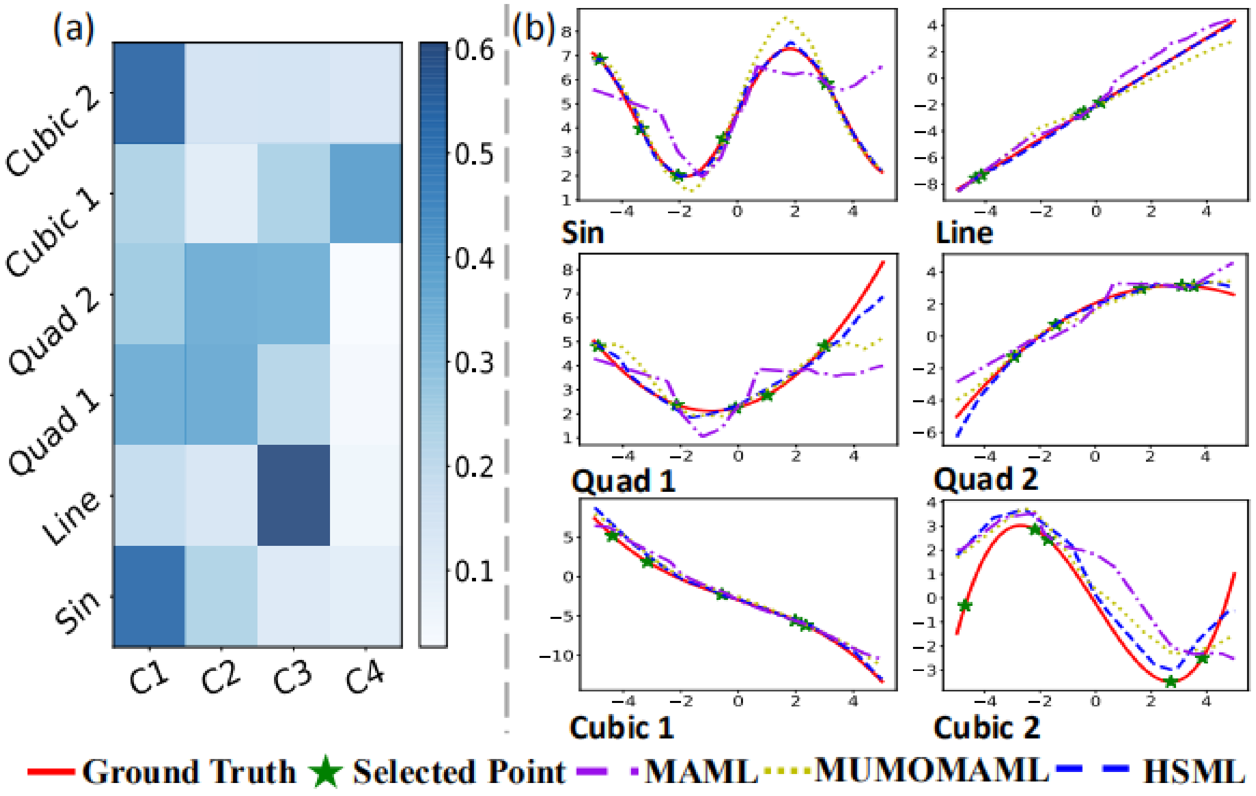
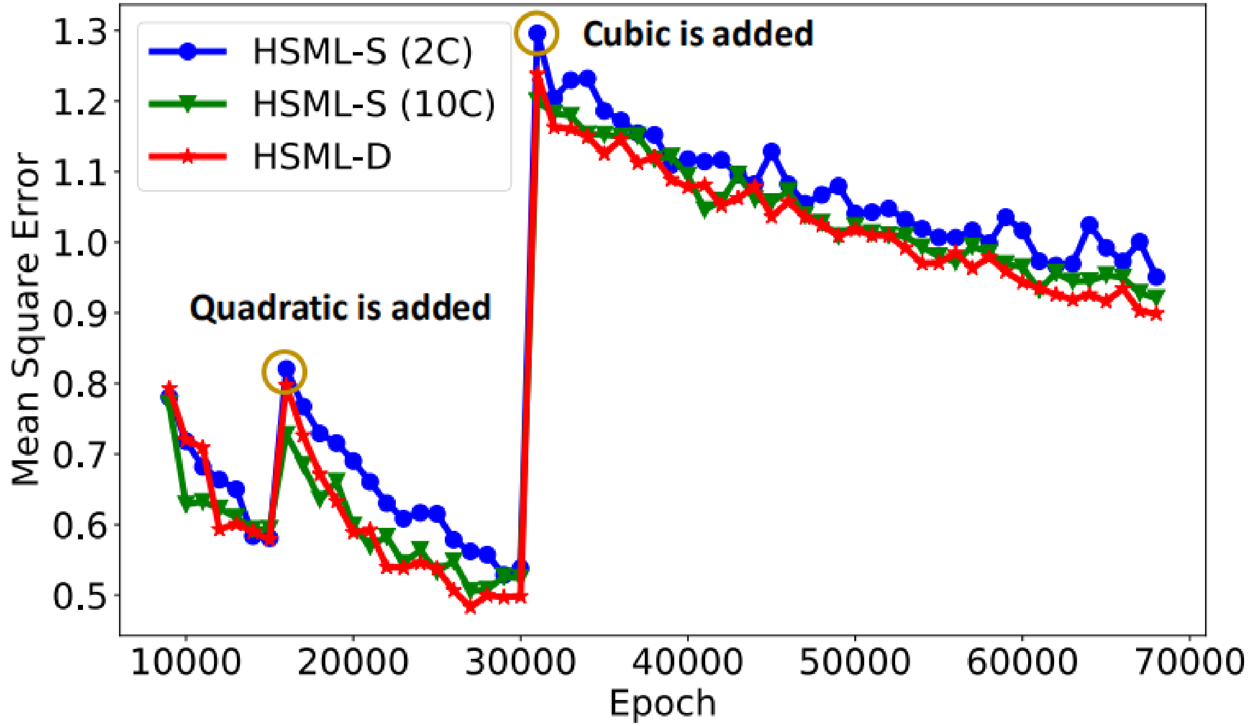


Figure 3. (a) The visualization of soft-assignment in Eqn. (6) of six selected tasks. Darker color represents higher probability. (b) The corresponding fitting curves. The ground truth underlying a function is shown in red line with data samples marked as green stars. C1-4 mean cluster 1-4, respectively.

Experiment

Toy Regression

Results of Continual Adaptation



Model	HSML-S (2C)	HSML-S (10C)	HSML-D
MSE ± 95% CI	0.933 ± 0.074	0.889 ± 0.071	**0.869 ± 0.072**

Figure 4. The performance comparison for the 5-shot toy regression problem in the continual adaptation scenario. The curve of MSE in meta-training process is shown in the top figure and the performance of meta-testing is reported in the bottom table.

Experiment

Few-shot Classification

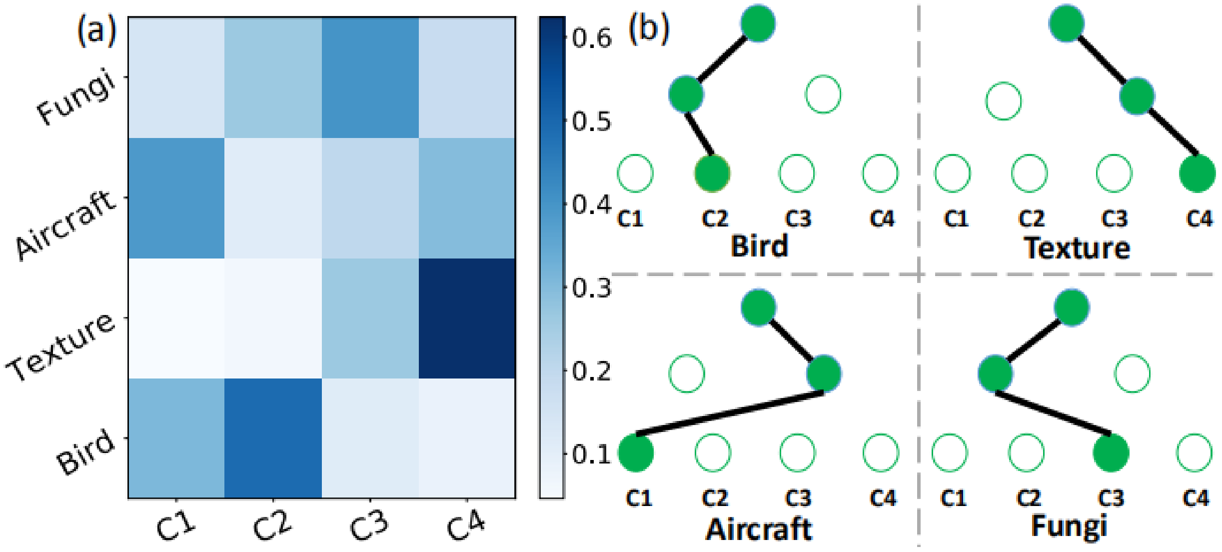


Figure 5. (a) The visualization of soft-assignment in Eqn. (6) of four selected tasks. (b) Learned hierarchical structure of each task. In each layer, top activated cluster is shown in dark color.

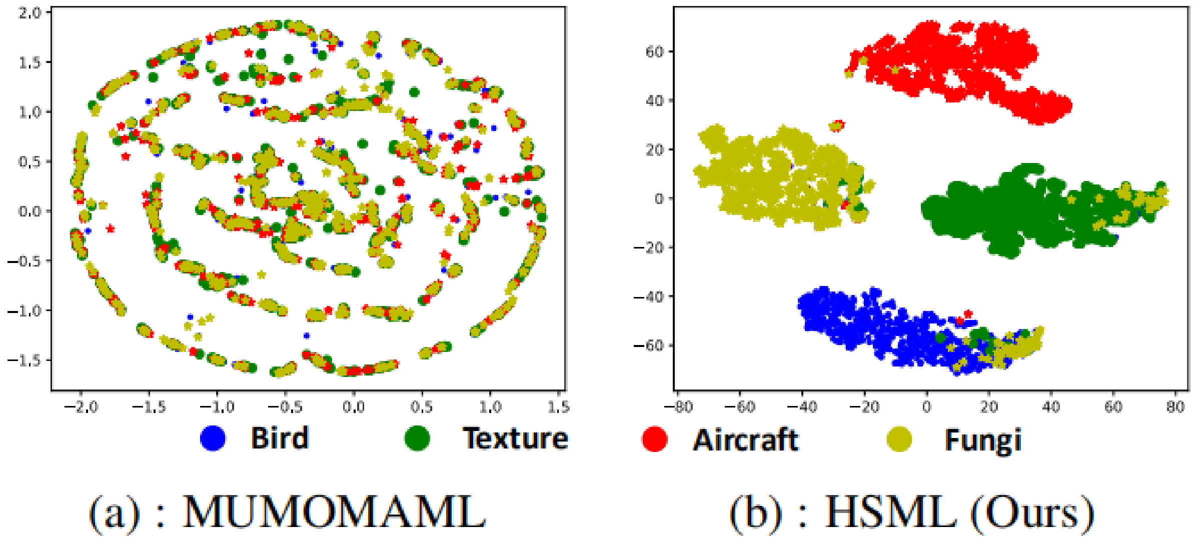


Figure 6. t-SNE visualization of gated weight, i.e., θ_{0i}, in Eqn. (9)

$$\min_{\Theta} \sum_{i=1}^{N_t} \mathcal{L}(f_{\theta_{0i}-\alpha\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}_{\mathcal{T}_i})}, \mathcal{D}^{te}_{\mathcal{T}_i}) + \xi \mathcal{L}_r(\mathcal{D}^{tr}_{\mathcal{T}_i}), \quad (9)$$

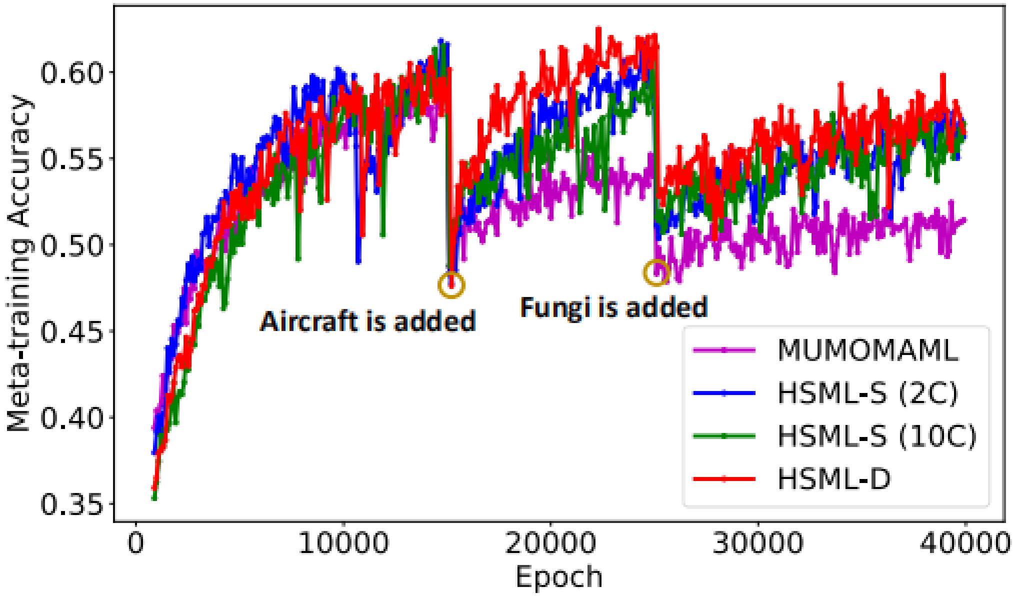
Experiment

Few-shot Classification

Table 2. Comparison between HSML and other gradient-based meta-learning methods on the 5-way, 1-shot/5-shot image classification problem, averaged over 1000 tasks for each dataset. Accuracy $\pm$ 95% confidence intervals are reported.

	Model	Bird	Texture	Aircraft	Fungi	Average
5-way 1-shot	MAML	$53.94 \pm 1.45\%$	$31.66 \pm 1.31\%$	$51.37 \pm 1.38\%$	$42.12 \pm 1.36\%$	44.77%
	Meta-SGD	$55.58 \pm 1.43\%$	$32.38 \pm 1.32\%$	$52.99 \pm 1.36\%$	$41.74 \pm 1.34\%$	45.67%
	MT-Net	$58.72 \pm 1.43\%$	$32.80 \pm 1.35\%$	$47.72 \pm 1.46\%$	$43.11 \pm 1.42\%$	45.59%
	BMAML	$54.89 \pm 1.48\%$	$32.53 \pm 1.33\%$	$53.63 \pm 1.37\%$	$42.50 \pm 1.33\%$	45.89%
	MUMOMAML	$56.82 \pm 1.49\%$	$33.81 \pm 1.36\%$	$53.14 \pm 1.39\%$	$42.22 \pm 1.40\%$	46.50%
	HSML (ours)	$\mathbf{60.98 \pm 1.50\%}$	$\mathbf{35.01 \pm 1.36\%}$	$\mathbf{57.38 \pm 1.40\%}$	$\mathbf{44.02 \pm 1.39\%}$	$\mathbf{49.35\%}$
5-way 5-shot	MAML	$68.52 \pm 0.79\%$	$44.56 \pm 0.68\%$	$66.18 \pm 0.71\%$	$51.85 \pm 0.85\%$	57.78%
	Meta-SGD	$67.87 \pm 0.74\%$	$45.49 \pm 0.68\%$	$66.84 \pm 0.70\%$	$52.51 \pm 0.81\%$	58.18%
	MT-Net	$69.22 \pm 0.75\%$	$46.57 \pm 0.70\%$	$63.03 \pm 0.69\%$	$53.49 \pm 0.83\%$	58.08%
	BMAML	$69.01 \pm 0.74\%$	$46.06 \pm 0.69\%$	$65.74 \pm 0.67\%$	$52.43 \pm 0.84\%$	58.31%
	MUMOMAML	$70.49 \pm 0.76\%$	$45.89 \pm 0.69\%$	$67.31 \pm 0.68\%$	$53.96 \pm 0.82\%$	59.41%
	HSML (ours)	$\mathbf{71.68 \pm 0.73\%}$	$\mathbf{48.08 \pm 0.69\%}$	$\mathbf{73.49 \pm 0.68\%}$	$\mathbf{56.32 \pm 0.80\%}$	$\mathbf{62.39\%}$

Results of Continual Adaptation



Model	Bird	Texture	Aircraft	Fungi
MUMOMAML	56.66%	33.68%	45.73%	40.38%
HSML-S (2C)	60.77%	33.41%	51.28%	40.78%
HSML-S (10C)	59.16%	34.48%	52.30%	40.56%
HSML-D	**61.16%**	**34.53%**	**54.50%**	**41.66%**

Figure 7. The performance comparison for the 5-way 1-shot few-shot classification problem in the continual adaptation scenario. The top figure and bottom table show the meta-training accuracy curves and the meta-testing accuracy, respectively.

Table 3. Comparison of different cluster numbers. The numbers in first column represents the number of clusters from bottom layer to top layer. Accuracy for 5-way 1-shot classification are reported.

Num. of Clu.	Bird	Texture	Aircraft	Fungi
(2, 2, 1)	58.37%	33.18%	56.15%	42.90%
(4, 2, 1)	**60.98%**	**35.01%**	57.38%	44.02%
(6, 3, 1)	60.55%	34.02%	55.79%	43.43%
(8, 4, 2, 1)	59.55%	34.74%	**57.84%**	**44.18%**

Effect of Cluster Numbers

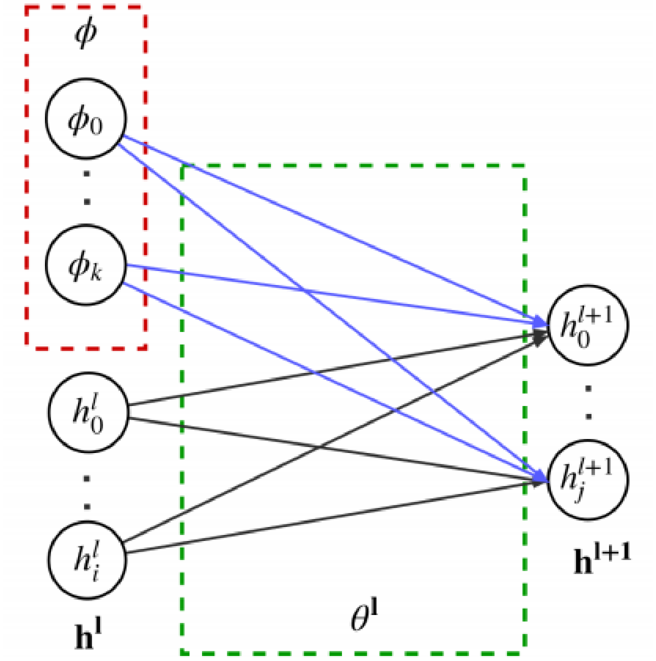
Fast Context Adaptation via Meta-Learning

■ Intuition

A simple extension to MAML that is **less prone to meta-overfitting**, easier to parallelise, and more interpretable.

fast **c**ontext **a**daptation **via** meta-learning (**CAVIA**)

context parameters ϕ



Network parameters θ

- context parameters ϕ are adapted in the **inner** loop for each task
- parameters θ are meta-learned in the **outer** loop and shared across tasks

Model-Agnostic Meta-Learning

■ Intuition

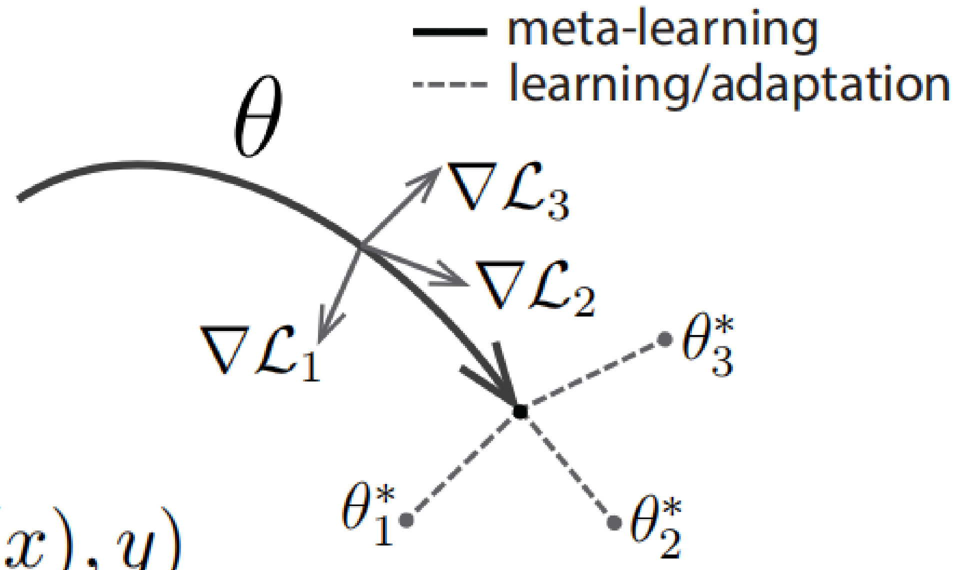
MAML learns an initialisation for the parameters θ of a model f_θ such that, given a new task, a good model for that task can be learned with only a small number of gradient steps and data points

inner loop

$$\theta_i = \theta - \alpha \nabla_\theta \frac{1}{M^i_{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_\theta(x), y)$$

outer loop

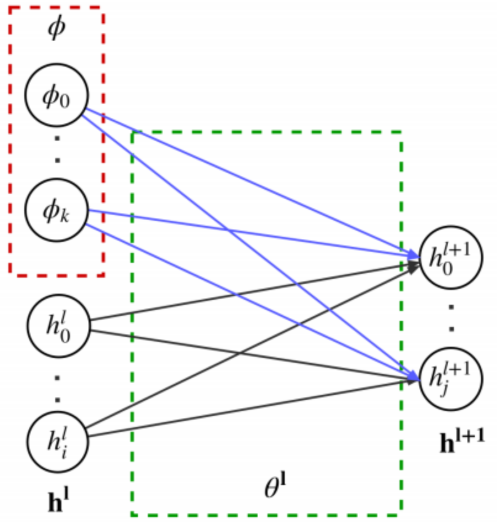
$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \frac{1}{M^i_{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}(x), y)$$



Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta learning for fast adaptation of deep networks. ICML-2017

CAVIA-Method

■ Conditioning on Context Parameters



simply concatenate $\boldsymbol{\phi}$ to the inputs of that layer

$$h_i^{(l)} = g\left(\sum_{j=1}^{J} \theta_{j,i}^{(l,h)}\, h_j^{(l-1)} + \sum_{k=1}^{K} \theta_{k,i}^{(l,\phi)}\, \phi_{0,k} + b\right)$$

$\theta_{j,i}^{(l,h)}$ are the weights associated with layer input $h_j^{(l-1)}$

$\theta_{k,i}^{(l,\phi)}$ are the weights associated with the context parameter $\phi_{0,k}$

use feature-wise linear modulation **FiLM** $FiLM(h_i) = \gamma_i h_i + \beta$, where $\gamma, \beta \in \mathbb{R}^M$

■ Context Parameter Initialisation

When learning a new task, $\boldsymbol{\phi}$ have to be initialised to a vector filled with zeros, $\phi_0 = \mathbf{0} = [0, \ldots, 0]^\top$

CAVIA-Method

■ Supervised Learning

Starting from a fixed value ϕ_0 we learn task-specific parameters ϕ_i via one gradient update

$$\phi_i = \phi_0 - \alpha \nabla_\phi \frac{1}{M_i^{\text{train}}} \sum_{(x,y) \in \mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_0,\theta}(x), y)$$

Given updated parameters ϕ_i for all sampled tasks, we proceed to the meta-learning step, in which θ is updated

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \frac{1}{M_i^{\text{test}}} \sum_{(x,y) \in \mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i,\theta}(x), y)$$

CAVIA-Method

Algorithm 1 CAVIA for Supervised Learning

Require: Distribution over tasks $p(\mathcal{T})$
Require: Step sizes α and β
Require: Initial model $f_{\phi_0,\theta}$ with θ initialised randomly and $\phi_0 = 0$

while not done **do**
  Sample batch of tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^{N}$ where $\mathcal{T}_i \sim p$
  for all $\mathcal{T}_i \in \mathbf{T}$ **do**
    $\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}} \sim q_{\mathcal{T}_i}$
    $\phi_0 = 0$
    $\phi_i = \phi_0 - \alpha \nabla_\phi \frac{1}{M_i^{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_0,\theta}(x), y)$
  end for
  $\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \frac{1}{M_i^{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i,\theta}(x, y))$
end while

Algorithm 2 CAVIA for RL

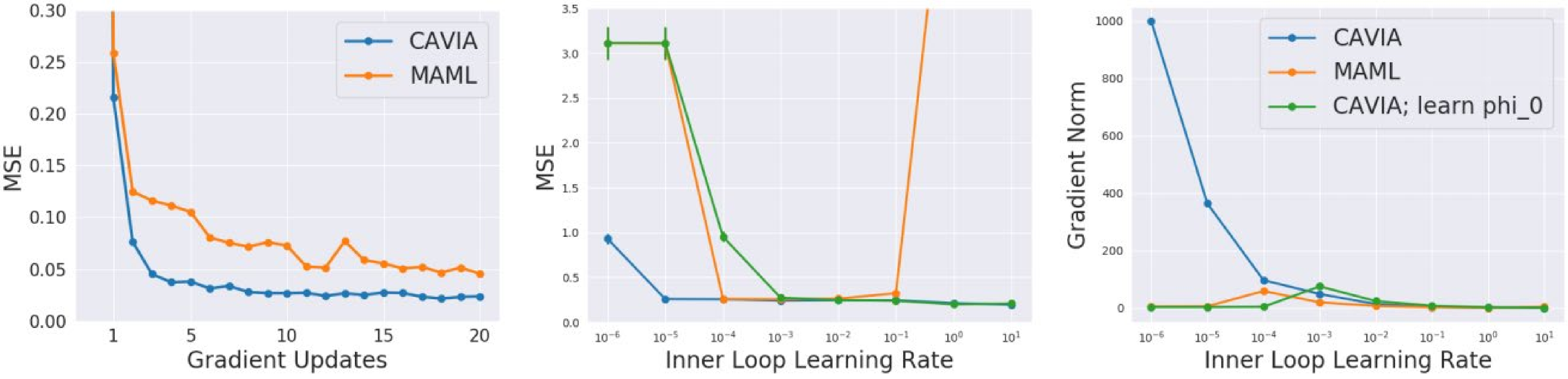
Require: Distribution over tasks $p(\mathcal{T})$
Require: Step sizes α and β
Require: Initial policy $\pi_{\phi_0,\theta}$ with θ initialised randomly and $\phi_0 = 0$

while not done **do**
  Sample batch of tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^{N}$ where $\mathcal{T}_i \sim p$
  for all $\mathcal{T}_i \in \mathbf{T}$ **do**
    Collect rollout τ_i^{train} using $\pi_{\phi_0,\theta}$
    $\phi_i = \phi_0 + \alpha \nabla_\phi \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{train}}, \pi_{\phi_0,\theta})$
    Collect rollout τ_i^{test} using $\pi_{\phi_i,\theta}$
  end for
  $\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{test}}, \pi_{\phi_i,\theta})$
end while

Experiment

■ Regression

Method	Number of Additional Input Parameters 0	1	2	3	4	5	50
CAVIA	-	0.84(±0.06)	0.21(±0.02)	0.20(±0.02)	**0.19**(±0.02)	**0.19**(±0.02)	**0.19**(±0.02)
MAML	0.33(±0.02)	0.29(±0.02)	0.24(±0.02)	0.24(±0.02)	**0.23**(±0.02)	**0.23**(±0.02)	**0.23**(±0.02)



(a) Test Performance (b) Learning Rates (c) Gradient Norms

Experiment

■ Classification

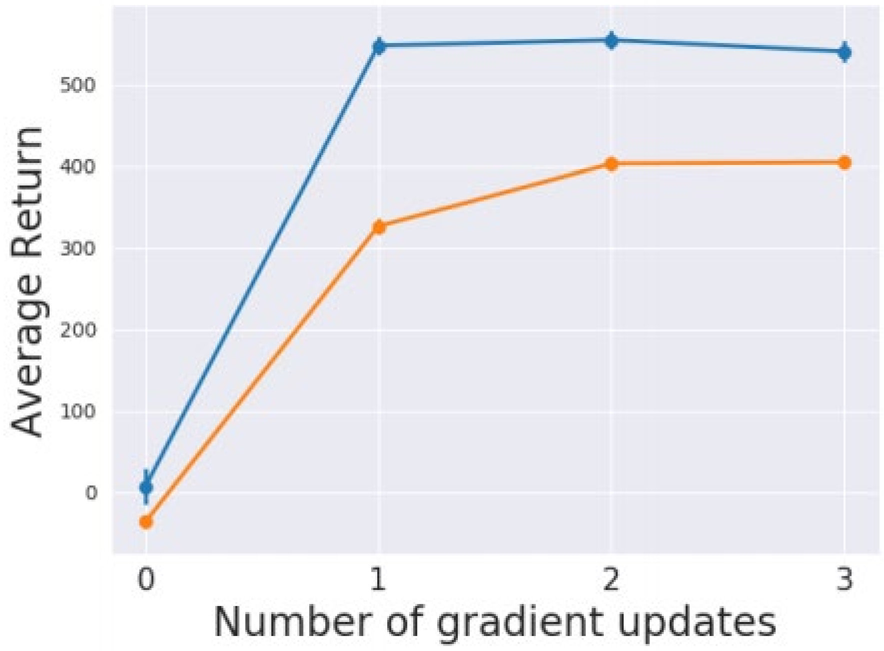
Mini-Imagenet dataset

Method	5-way accuracy 1-shot	5-way accuracy 5-shot
Matching Nets (Vinyals et al., 2016)	46.6%	60.0%
Meta LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
Prototypical Networks (Snell et al., 2017)	$46.61 \pm 0.78\%$	$65.77 \pm 0.70\%$
Meta-SGD (Li et al., 2017)	$50.47 \pm 1.87\%$	$64.03 \pm 0.94\%$
REPTILE (Nichol & Schulman, 2018)	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
MT-NET (Lee & Choi, 2018)	$\mathbf{51.70} \pm 1.84\%$	-
VERSA (Gordon et al., 2018)	$\mathbf{53.40} \pm 1.82\%$	$\mathbf{67.37} \pm 0.86$
MAML (32) (Finn et al., 2017a)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (64)	$44.70 \pm 1.69\%$	$61.87 \pm 0.93\%$
CAVIA (32)	$47.24 \pm 0.65\%$	$59.05 \pm 0.54\%$
CAVIA (128)	$49.84 \pm 0.68\%$	$64.63 \pm 0.54\%$
CAVIA (512)	$\mathbf{51.82} \pm 0.65\%$	$65.85 \pm 0.55\%$
CAVIA (512, first order)	$49.92 \pm 0.68\%$	$63.59 \pm 0.57\%$

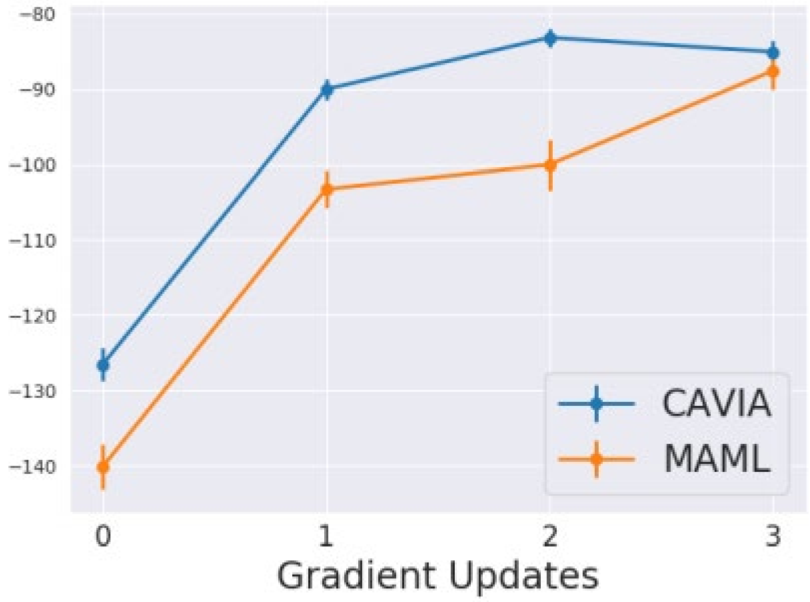
Experiment

■ Reinforcement Learning

MuJoCo
a Cheetah robot



(a) Direction



(b) Velocity

Performance of CAVIA and MAML on the RL Cheetah experiments. Both agents were trained to perform one gradient update, but are evaluated for several update steps. Results are averaged over 40 randomly selected tasks.