MAML: Model-**A**gnostic **M**eta-**L**earning for Fast Adaptation for Deep Networks

Yanjie Ze

July 2021

Contents

1. Meta-Learning Problem Setup (Definition & Goal)
2. Prior Work
3. Model Agnostic Meta Learning
    1. Characteristics
    2. Intuition
4. Approach
    1. Supervised Learning
    2. Gradient of Gradient
    3. Reinforcement Learning
5. Experiment

1. Problem Set-up

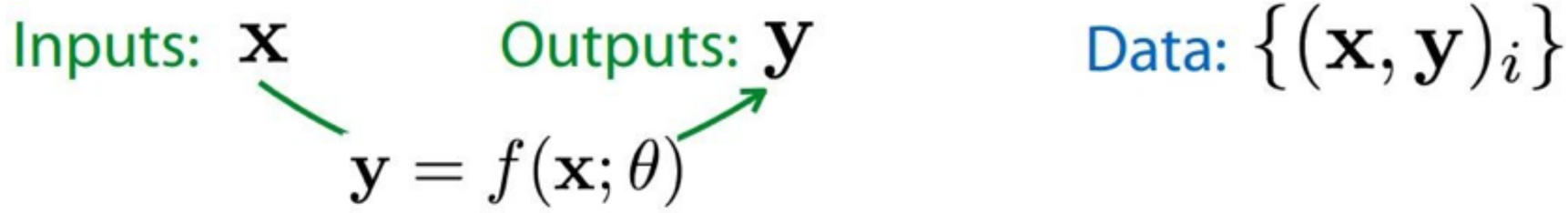
Meta-Learning Problem Set-up

- Definition of meta-learning:
  - Learner is trained by meta-learner to be able to learn on many different tasks.
- Goal:
  - Learner quickly learn new tasks from a small amount of new data.

1. Problem Set-up

Meta Supervised Learning

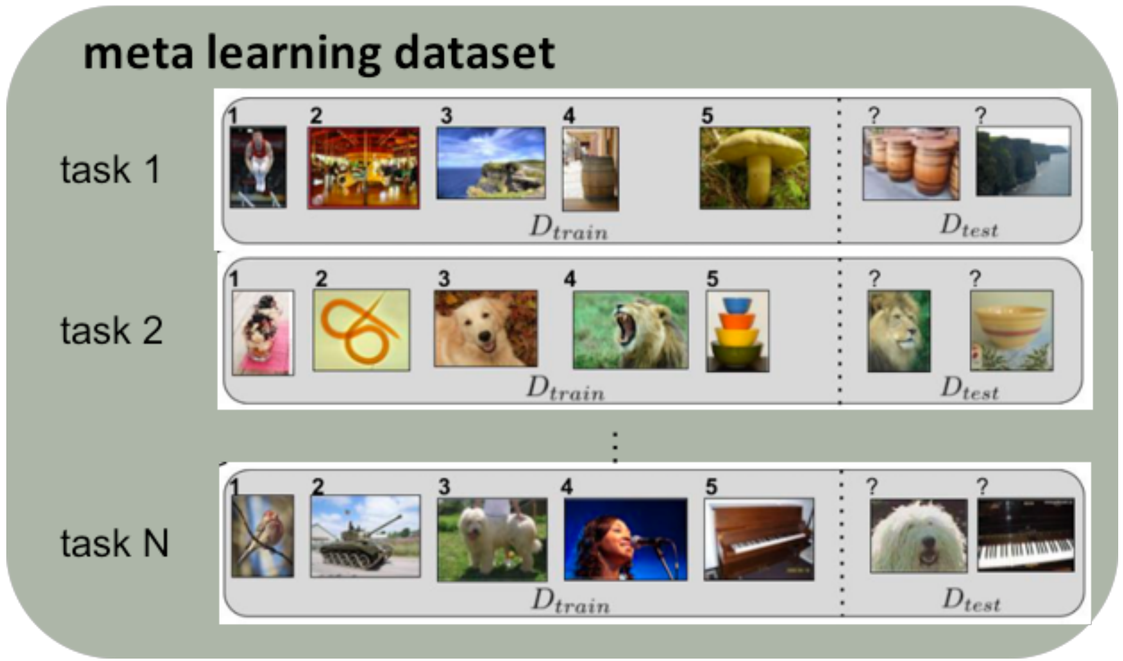
- Supervised learning:



- Meta supervised learning?

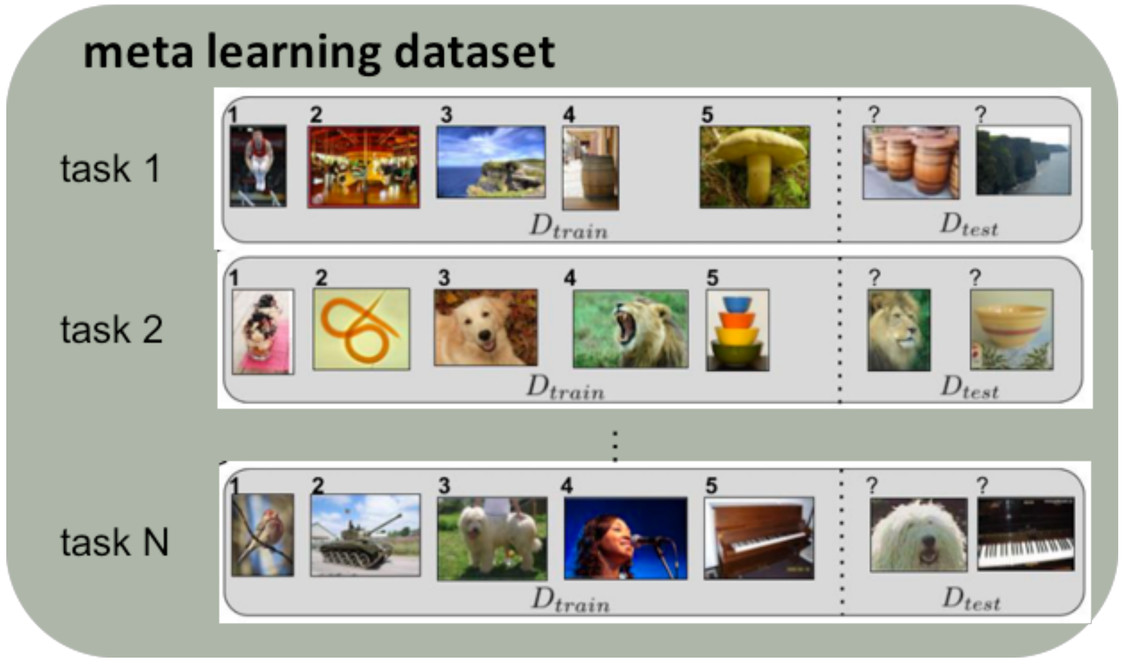
1. Problem Set-up

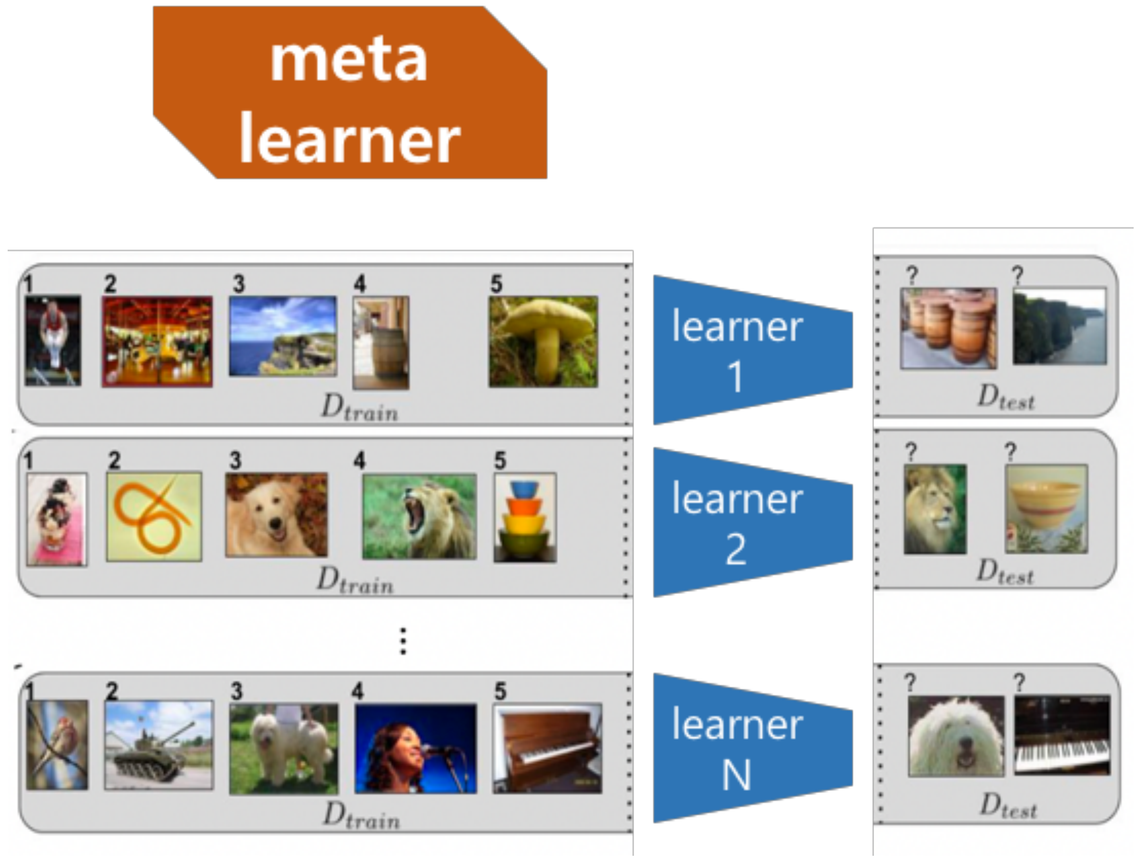
Meta Supervised Learning



1. Problem Set-up

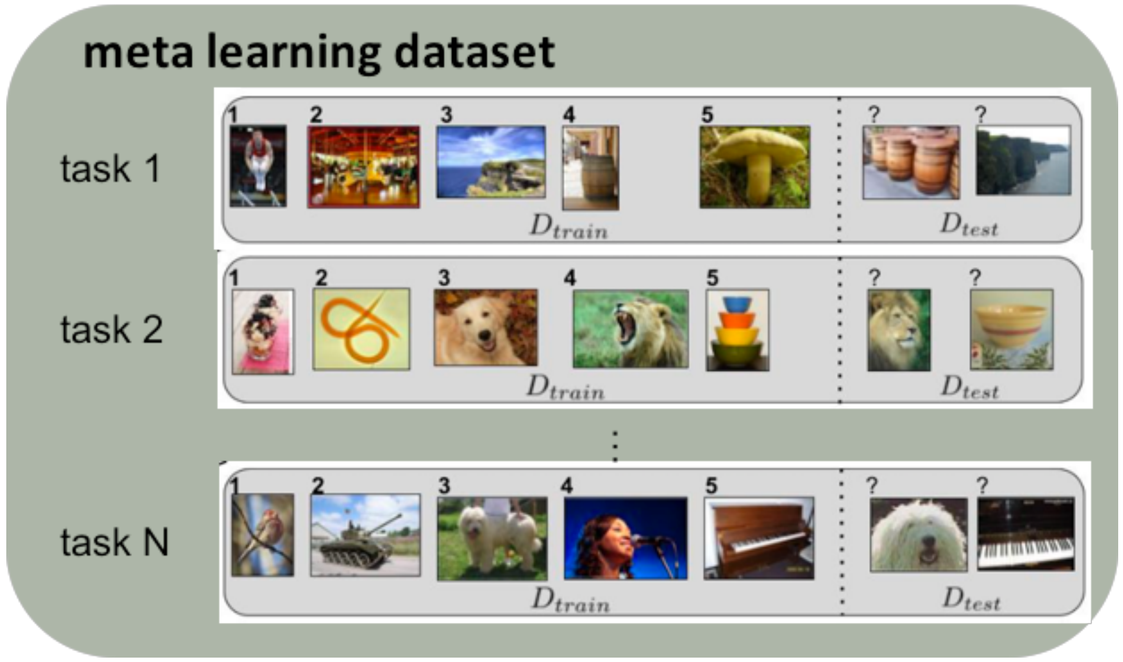
Meta Supervised Learning

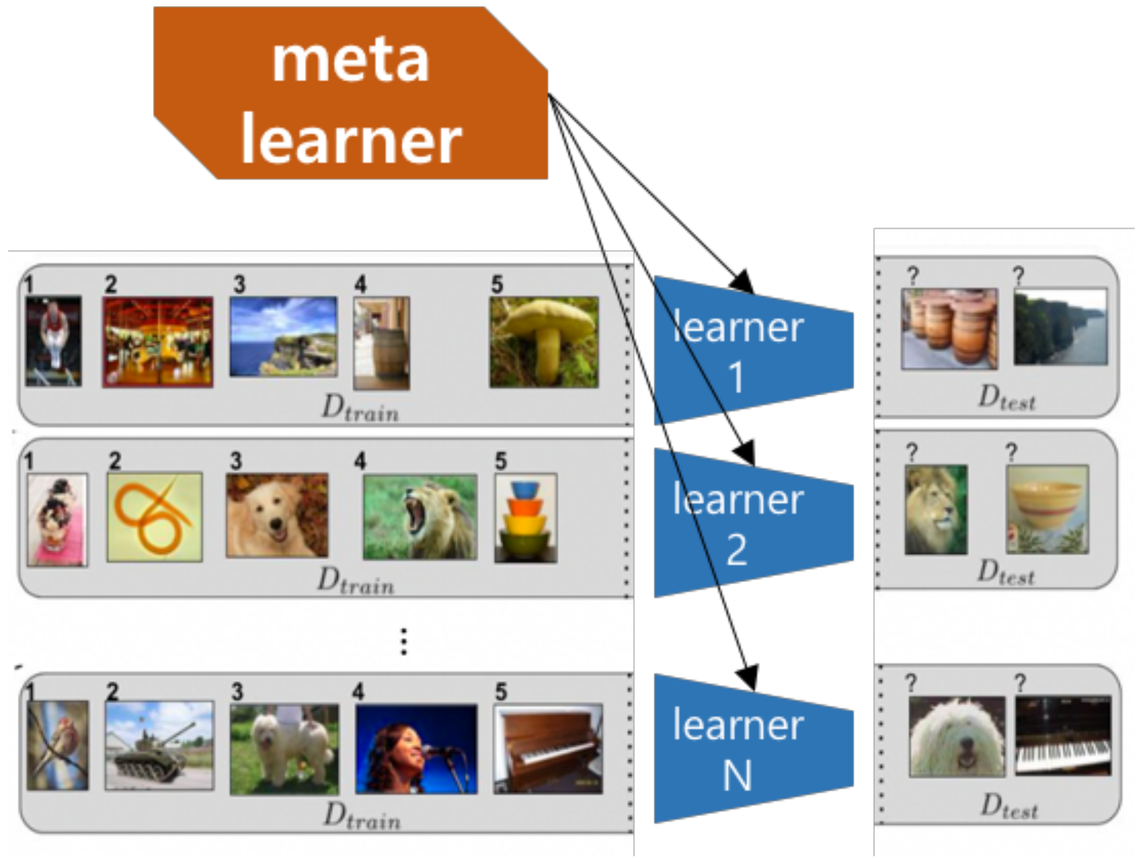




1. Problem Set-up

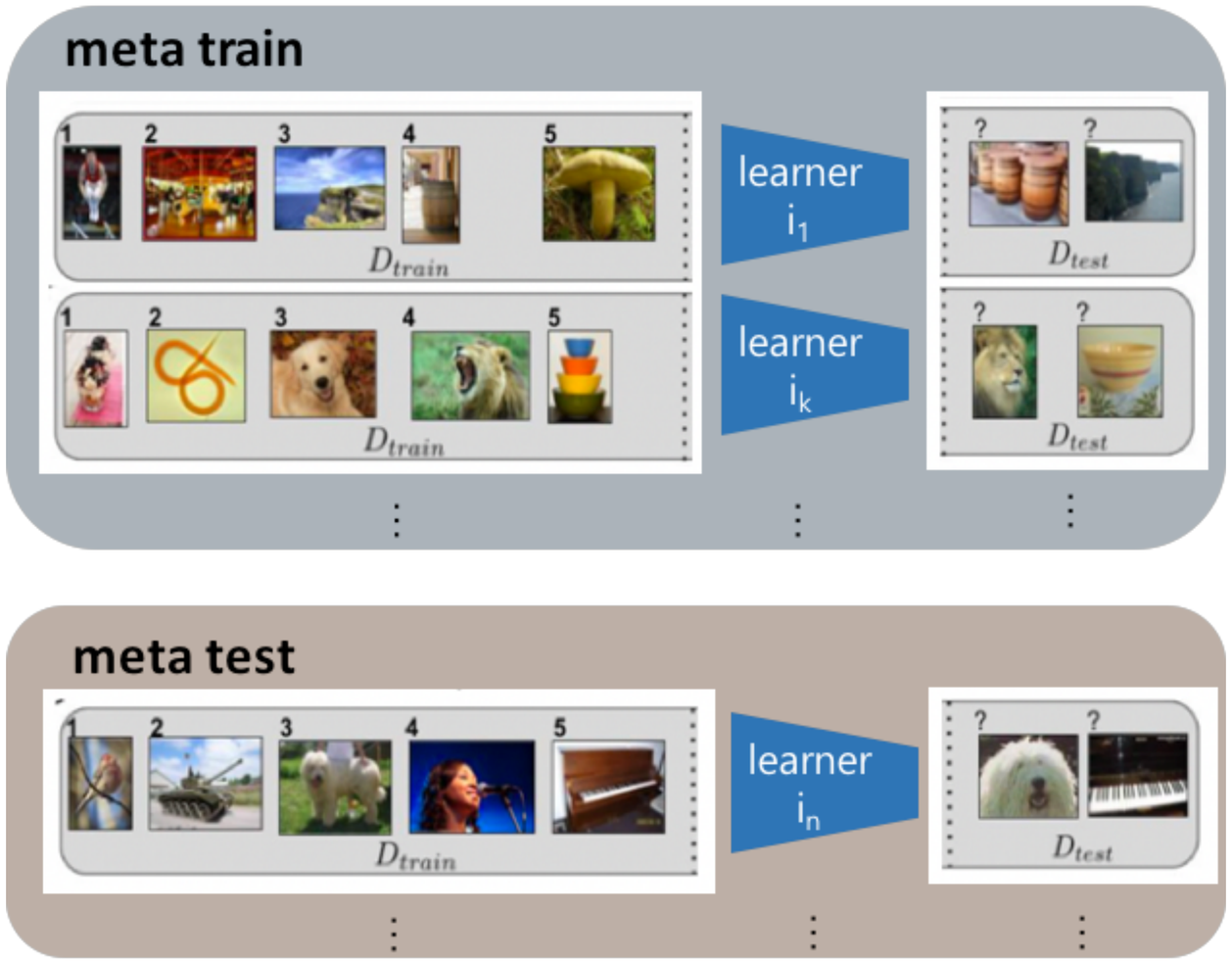
Meta Supervised Learning





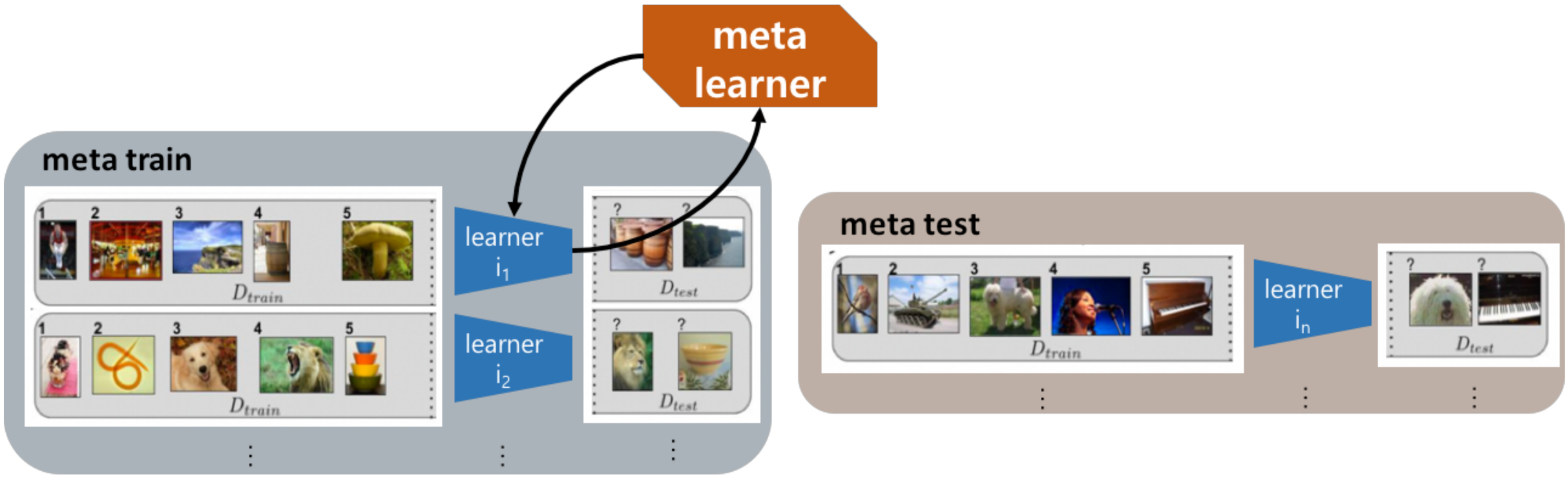
1. Problem Set-up

Meta Supervised Learning



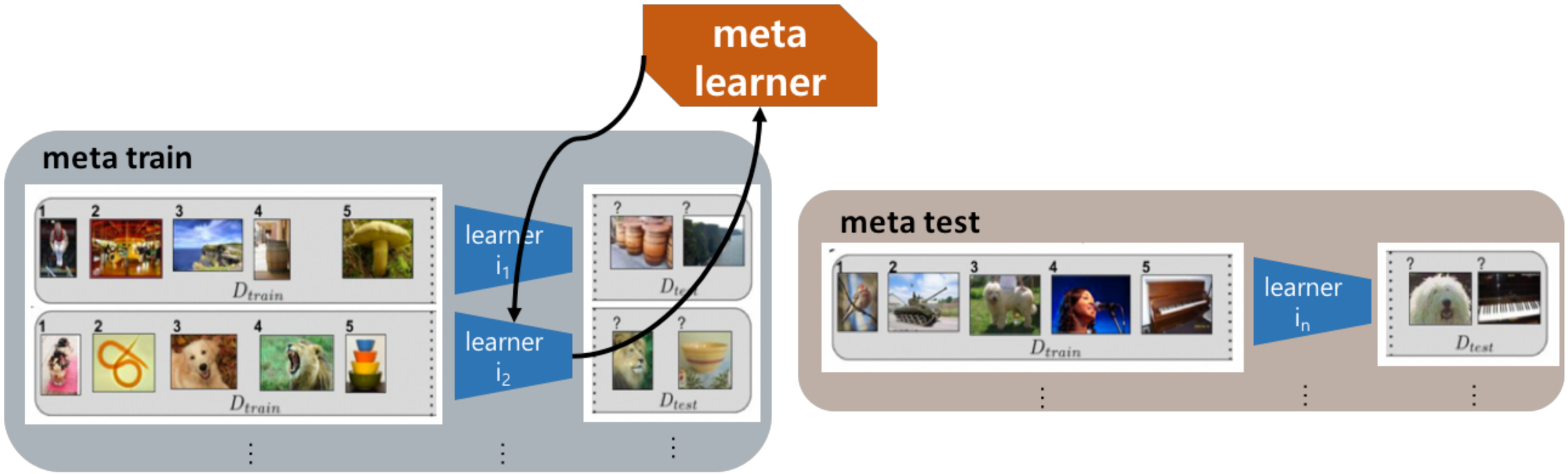
1. Problem Set-up

Meta Supervised Learning



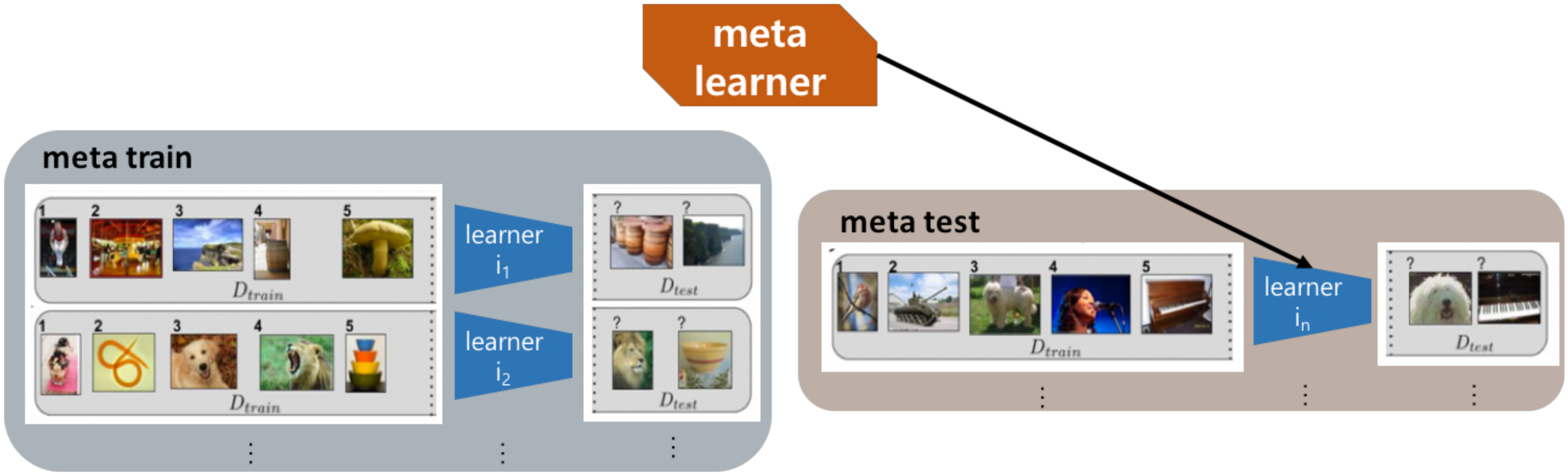
1. Problem Set-up

Meta Supervised Learning



1. Problem Set-up

Meta Supervised Learning



Prior Work

- Memory-augmented meta-learning
  - Memory-augmented models on many tasks
    (Meta-learning with memory-augmented neural networks)

- Parameter initialization of deep networks
  - Exploring sensitivity while maintaining Internal representation
    (Overcoming catastrophic forgetting in NN)
  - Data-dependent Initializer
    (Data-Dependent Initialization of CNN)
  - learned initialization
    (Gradient-based hyperparameter optimization through reversible learning)

Model-Agnostic Meta-Learning (MAML)

- Goal:
  - Quickly adapt to new tasks on distribution with **only small amount of data** and with **only a few gradient steps**, **even one gradient step**.
- Learner:
  - Learn a new task by using a single gradient step.
- Meta-learner:
  - Learn a **generalized parameter initialization** of model.

3. Model-Agnostic Meta-Learning (MAML)

Characteristics of MAML

- The MAML learner's **weights are updated using the gradient**, rather than a learned update.
  - Not need additional parameters nor require a particular learner architecture.

- **Fast adaptability** through good parameter initialization
  - Explicitly optimizes to learn Internal representation (i.e. suitable for many tasks).
  - Maximizes sensitivity of new task losses to the model parameters.

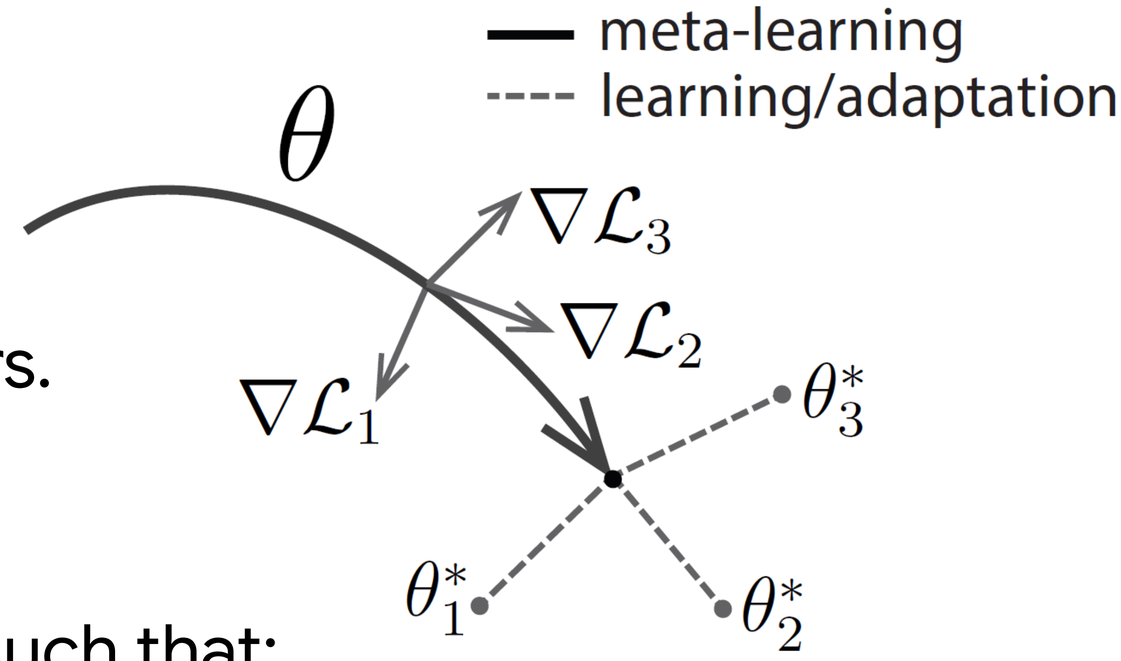
Characteristics of MAML

- **Model-agnostic** (No matter whatever model is)
  - Classification & regression with differentiable losses, Policy gradient RL.
  - The model should be parameterized.
  - No other assumptions on the form of the model.

- **Task-agnostic** (No matter whatever task is)
  - Adopted all knowledge-transferable tasks.
  - No other assumption is required.

Intuition of MAML

- Some internal representations are **more transferrable** than others.



- Desired model parameter set is θ such that:
  Applying one (or a small # of) gradient step to θ
  on a new task will produce maximally effective behavior.

  → Find θ that commonly decreases loss of each task
  after adaptation.

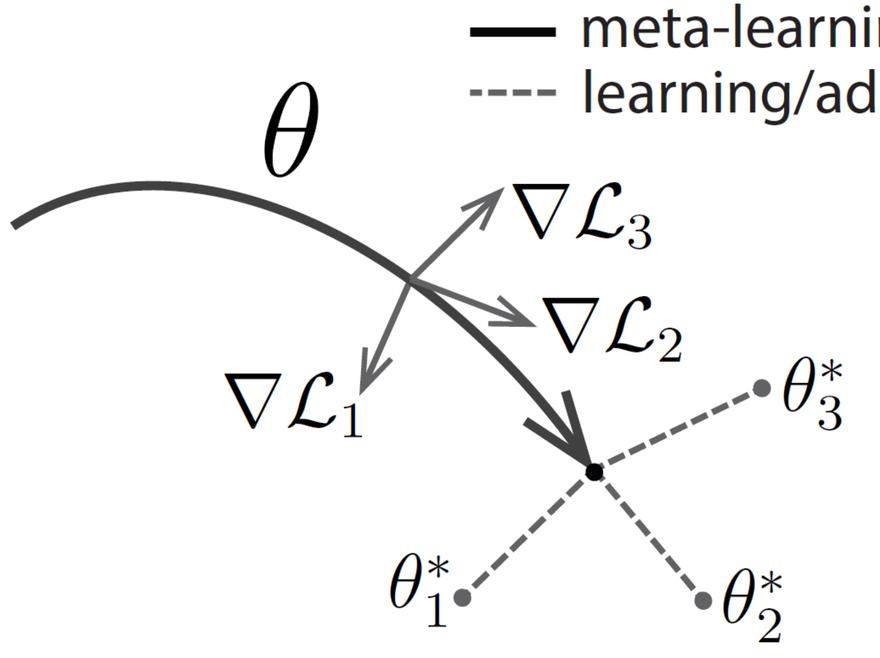
Supervised Learning

- Notations
  - Model: f_θ (Model-agnostic)
  - Task distribution: $p(\mathcal{T})$
- For each Task: $T = \{L(x_1, y_1, \ldots, x_K, y_K), (x, y) \sim q\}$
  - Data distribution: q (K samples are drawn from q)
  - Loss function: $\mathcal{L}$
    - MSE for Regression
    - Cross entropy for classification
    - Any other **differentiable** loss functions can be used.

Supervised Learning

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$
11: **end while**



Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

(Recall: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$)

Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta)) \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Calculation of Hessian matrix is required.

Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Calculation of Hessian matrix is required.
→ MAML suggest 1st order approximation.

1st Order Approximation

- Update rule of MAML:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

1st Order Approximation

- Update rule of MAML:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \boxed{\alpha \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta)}}\right)$$

This part needs calculating Hessian.
Hessian makes MAML be slow.

1st Order Approximation

- Update rule of MAML:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

- Update rule of MAML with 1st order approximation:

$$\delta \leftarrow \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \delta}\right)$$

1st Order Approximation

- Update rule of MAML:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

- Update rule of MAML with 1st order approximation:

$$\delta \leftarrow \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta) \qquad \text{(Regard } \delta \text{ as constant)}$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \delta}\right)$$

Recall: Gradient of Gradient

- From line 10 in Algorithm 2,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\text{)}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \qquad (\mathcal{L} \text{ is differentiable})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

In 1st order approximation,
we regard this as identity matrix *I*.

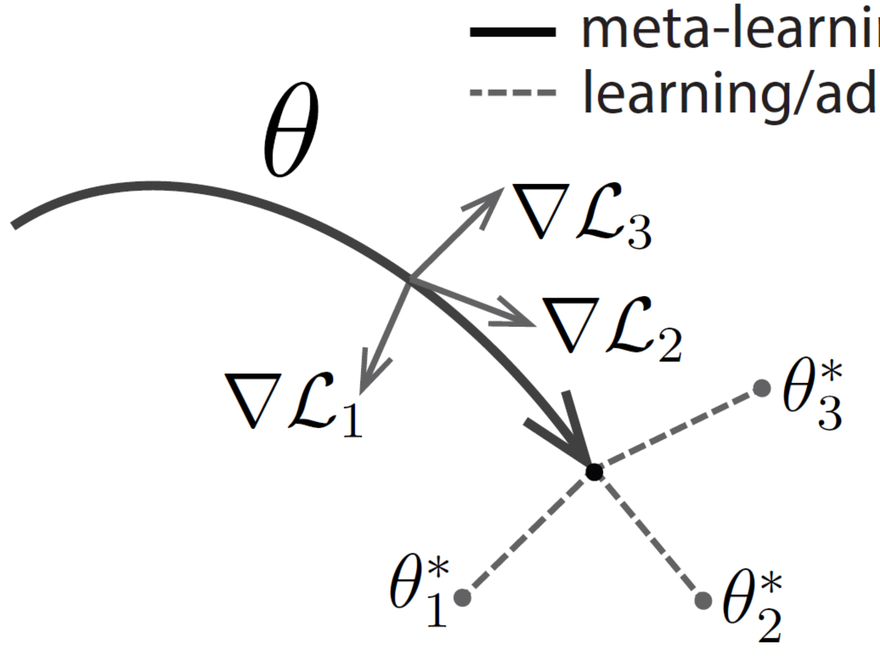
Reinforcement Learning

- Notations
  - Policy of an agent: f_θ such that a_t \sim f_\theta (x_t)
  - Task distribution: $p(\mathcal{T})$
- For each Task: $T = \{L(e_1, e_2, \ldots, e_K), x_1 \sim q(x), x_{t+1} \sim q(x|x_t, a_t), H\}$
  - Episode: $\mathbf{e} = (x_1, a_1, \ldots, x_H, a_H)$
  - Transition distribution: *q* (*K* trajectories are drawn from *q* and *f*)
  - Episode length: *H*
  - Loss function: $\mathcal{L}$ = expectation of sum of rewards

Reinforcement Learning

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$
11: **end while**



Experiment on Few-Shot Classification

- Omniglot (Lake et al., 2012)
  - 50 different alphabets, 1623 characters.
  - 20 instances for each characters were drawn by 20 different people.
  - 1200 for training, 423 for test.
- Mini-Imagenet (Ravi & Larochelle, 2017)
  - Classes for each set: train=64, validation=12, test=24.

Results of Few-Shot Classification

- MAML outperforms state-of-arts algorithms.

Omniglot (Lake et al., 2011)	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

Effect of 1st Order Approximation

- In 1st order approximation, computation is roughly 33% better.
- Performances are similar.

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	5-shot
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (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, first order approx. (ours)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Effect of 1st Order Approximation

- In 1st order approximation, computation is roughly 33% better.
- Performances are similar.

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	5-shot
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (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, first order approx. (ours)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Performances are not that different.

Effect of 1st Order Approximation

- In 1st order approximation, computation is roughly 33% better.
- Performances are similar.

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	5-shot
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (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, first order approx. (ours)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Some performance is even better.

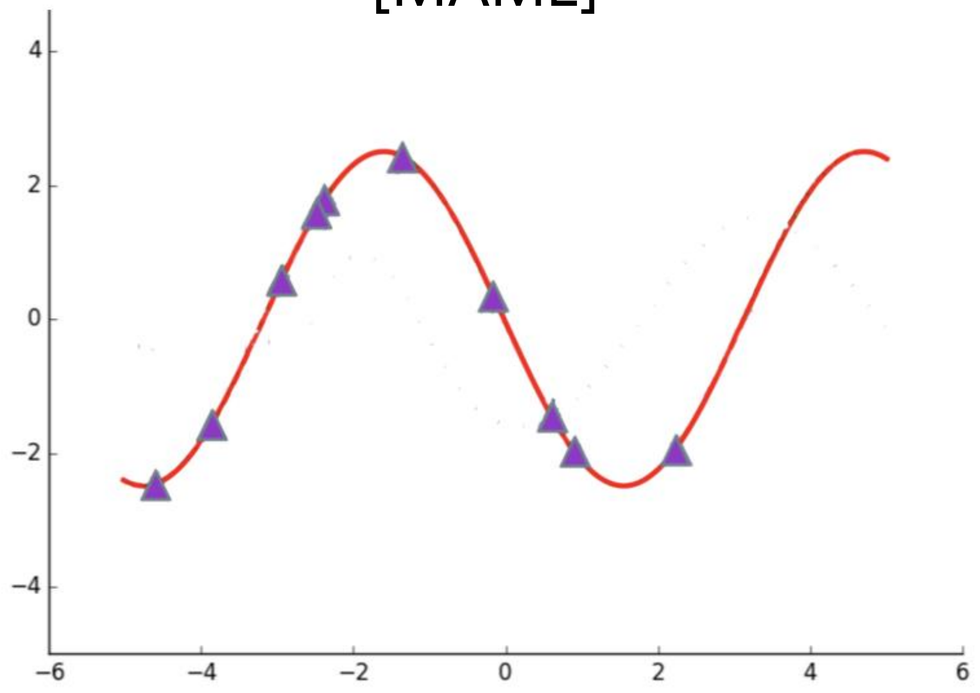
Experiments on Regression

- Sinusoid function:
  - Amplitude (A) and phase (ϕ) are varied between tasks
    - A in [0.1, 0.5]
    - ϕ in [0, π]
  - x in [-5.0, 5.0]
- Loss function: Mean Squared Error (MSE)
- Regressor: 2 hidden layers with 40 units and ReLU
- Training
  - Use **only 1 gradient step** for learner
  - K = 5 or 10 example (5-shot learning or 10-shot learning)
  - Fixed step size (α=0.01) for Adam optimizer.

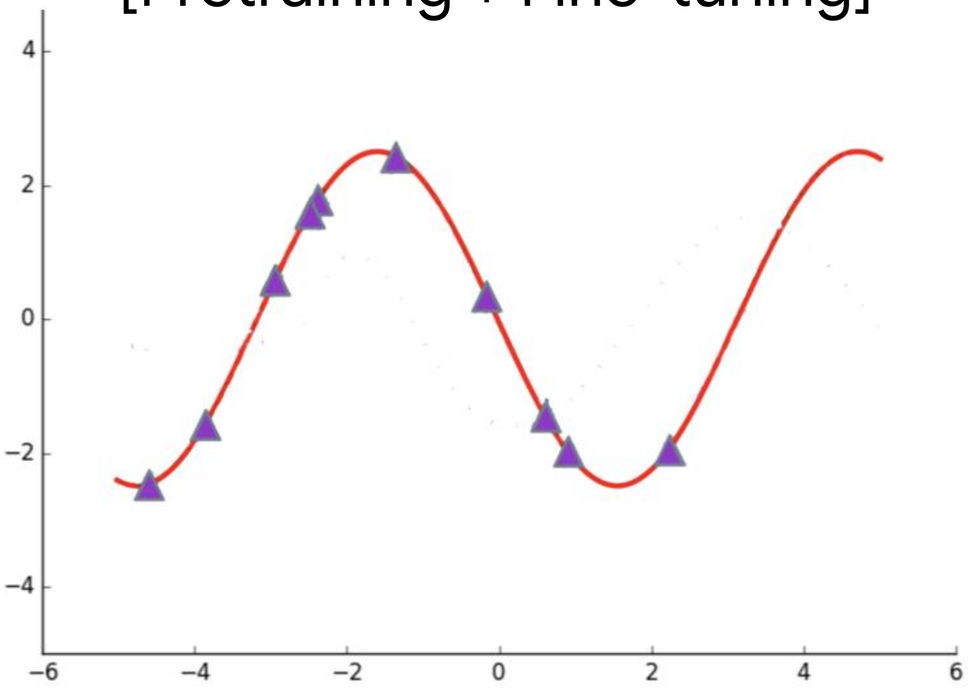
5. Experiment

Results of 10-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]

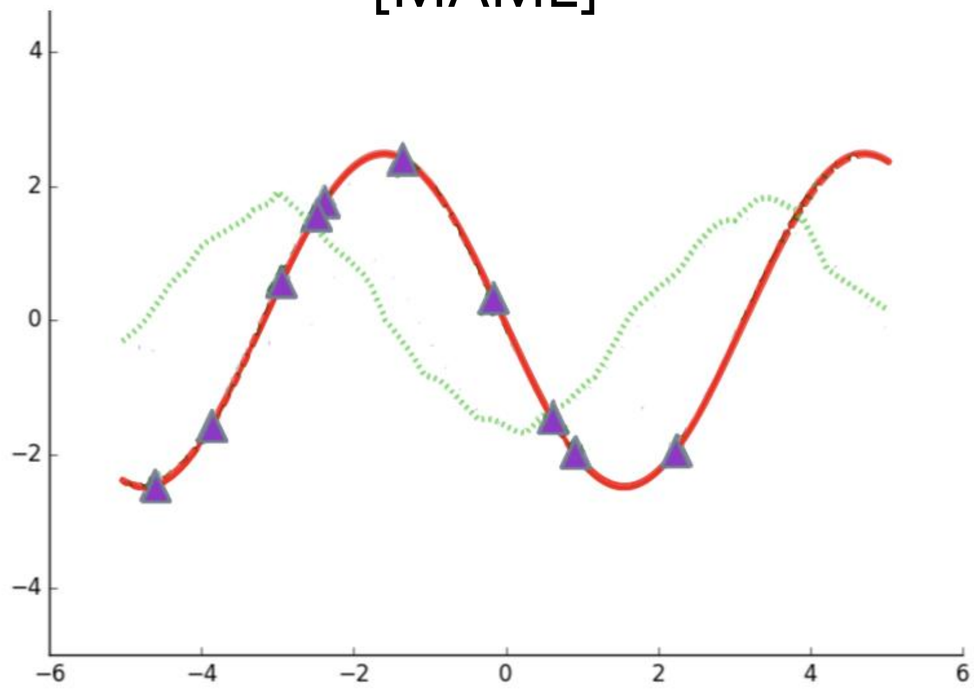


The red line is ground truth.
Fit this sine function with only few (10) samples.

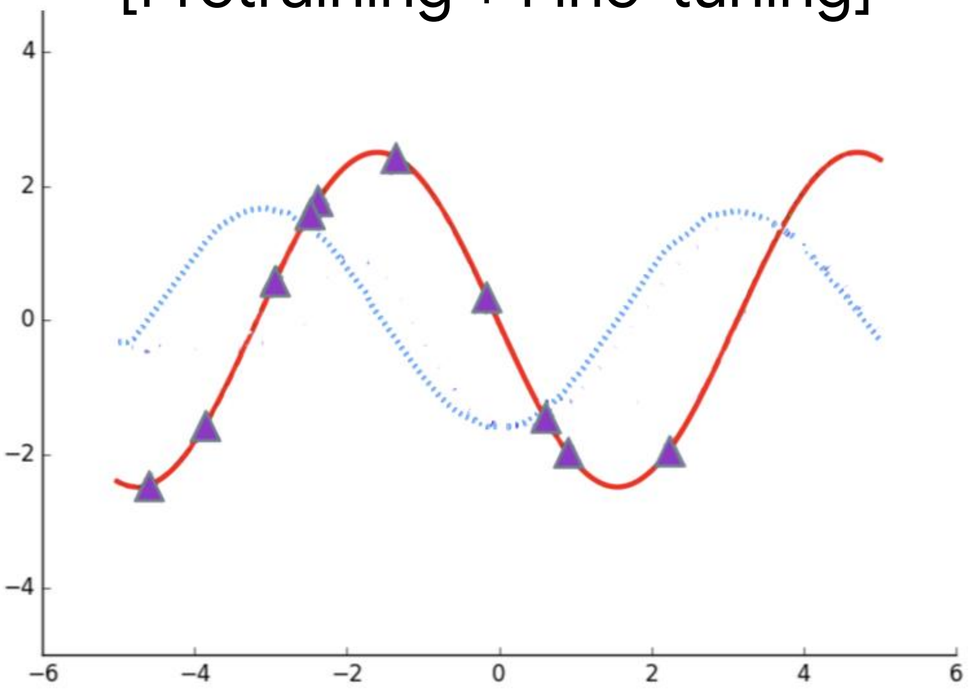
5. Experiment

Results of 10-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]

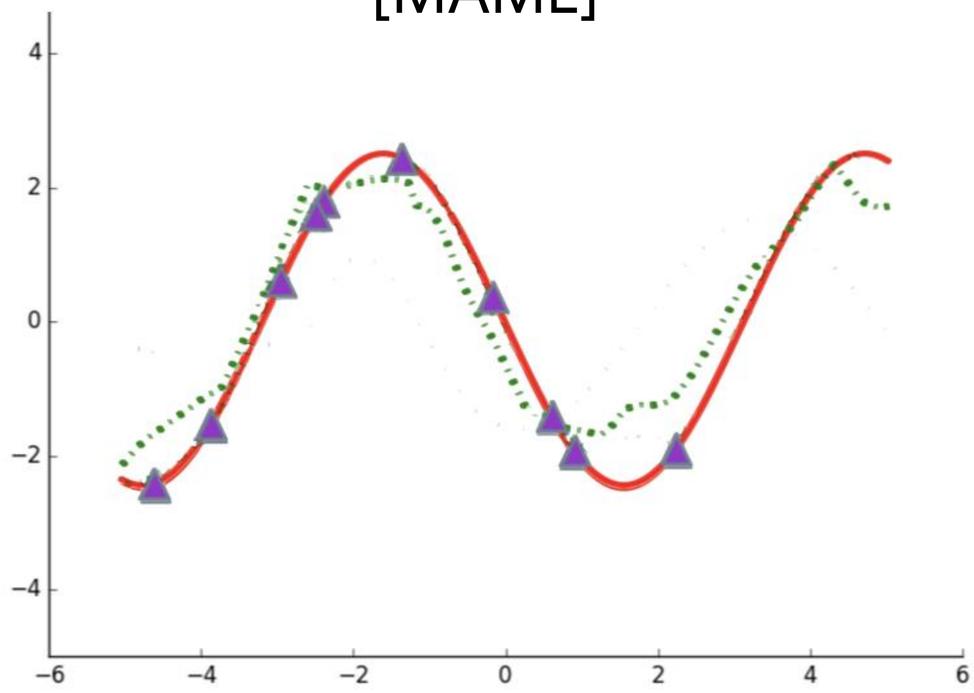


Above plots are the pre-trained function of two models.
(The prediction of meta-parameter of MAML,
The prediction of co-learned parameter of vanilla multi-task learning)

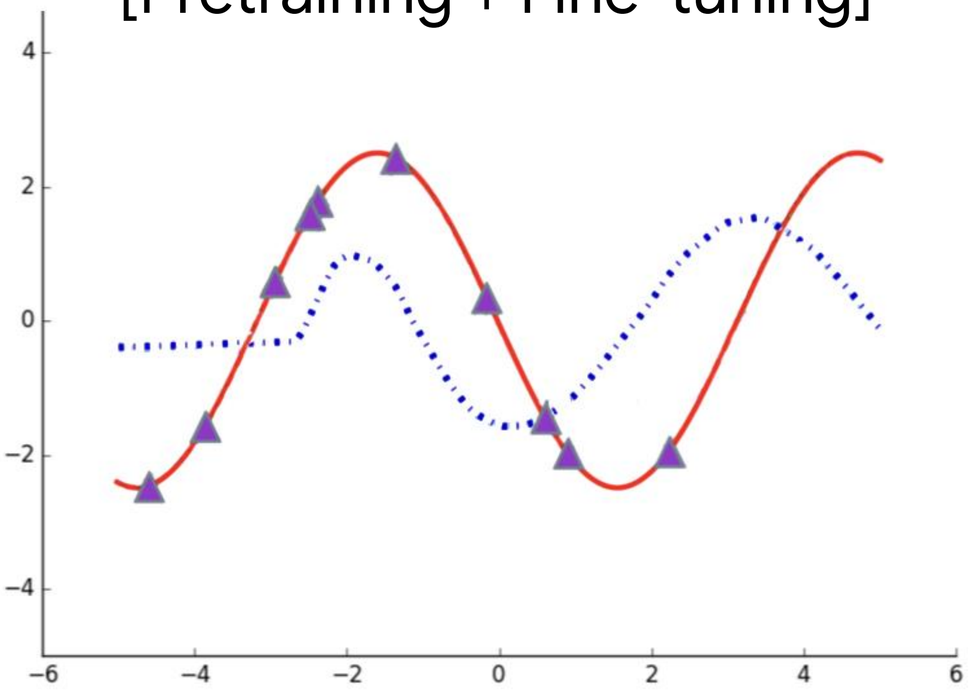
5. Experiment

Results of 10-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]

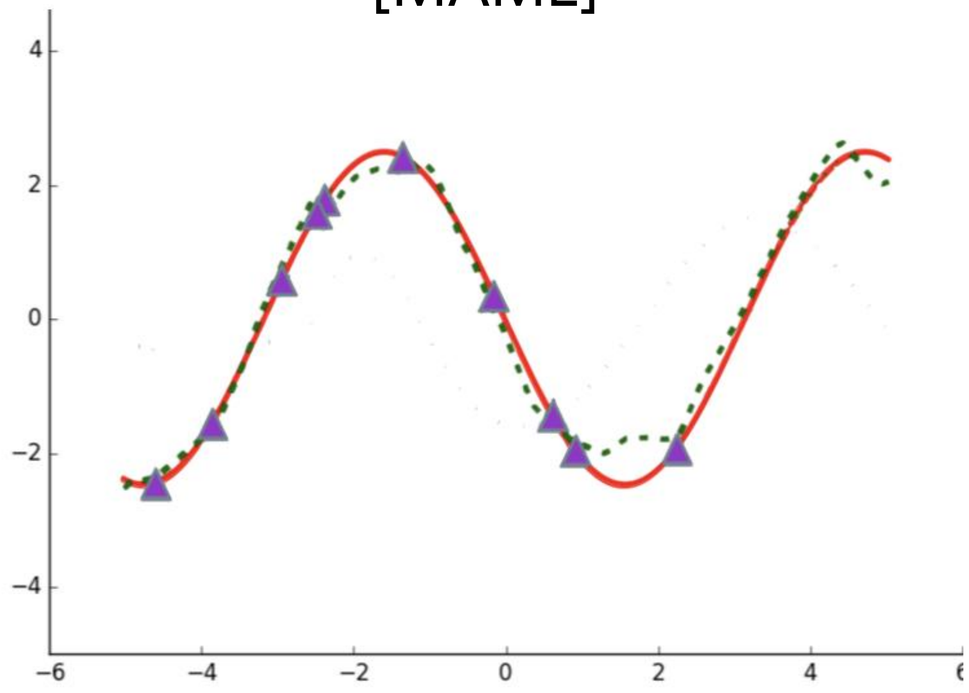


After 1 gradient step update.

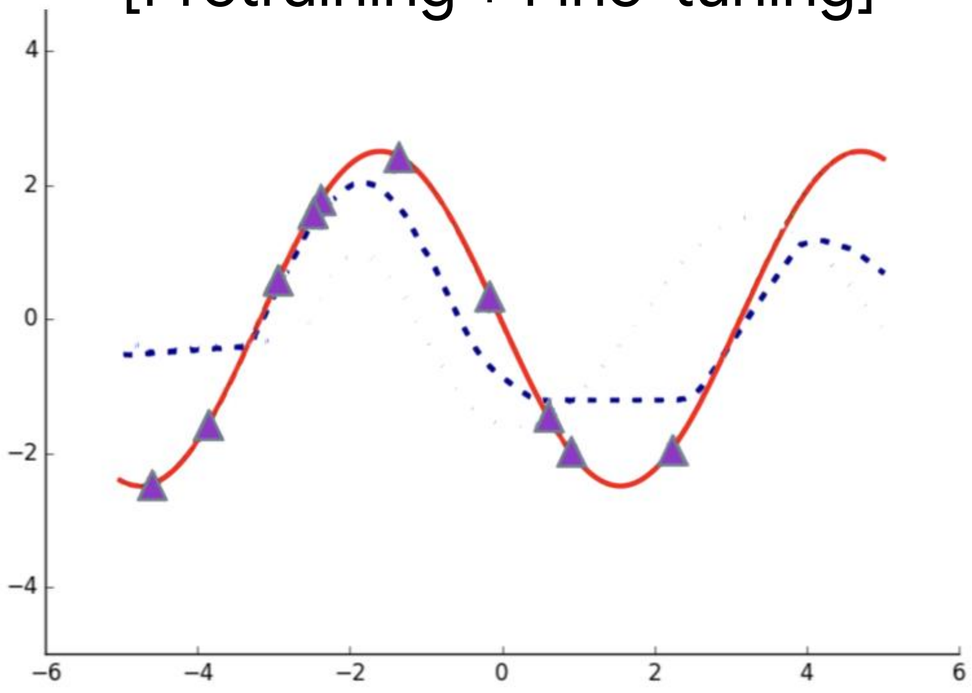
5. Experiment

Results of 10-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]

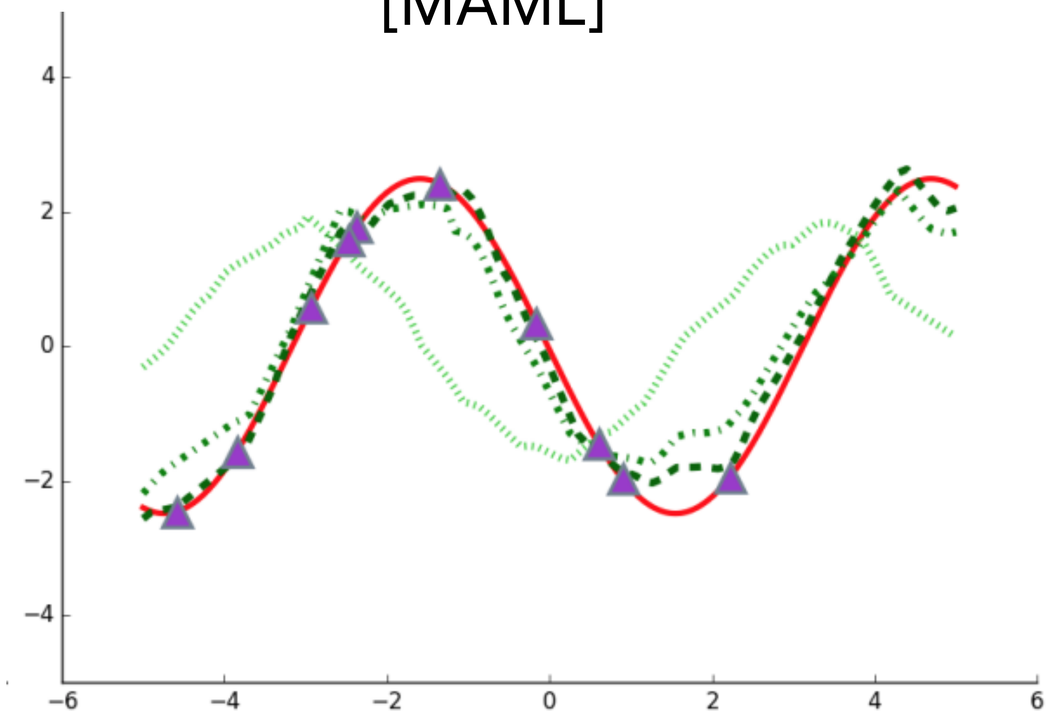


After 10 gradient step update.

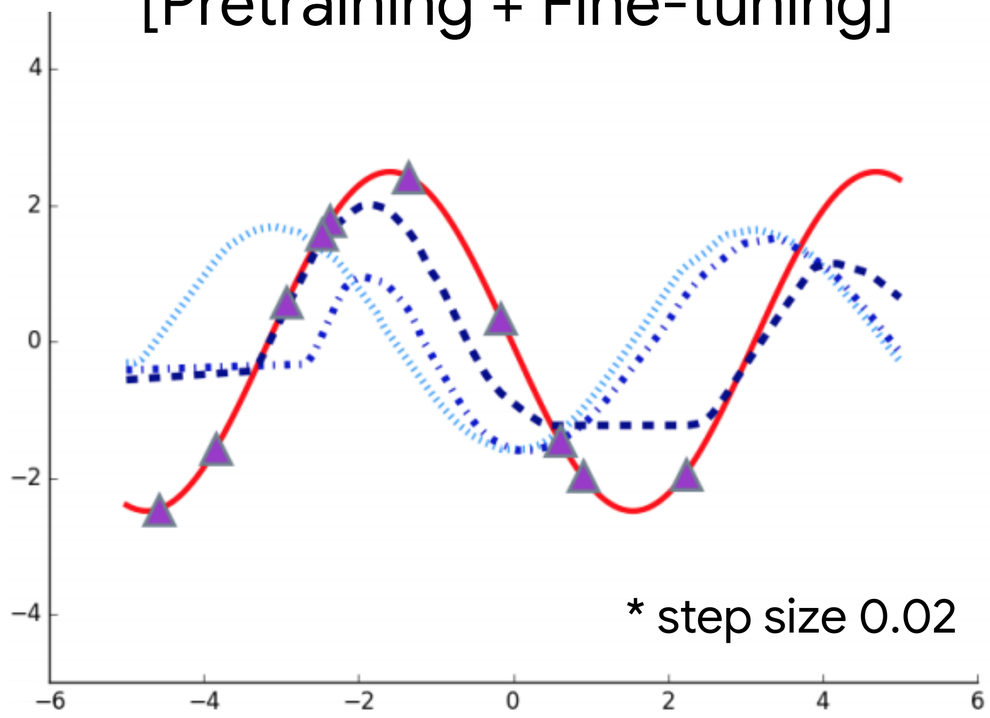
5. Experiment

Results of 10-Shot Learning Regression

[MAML]



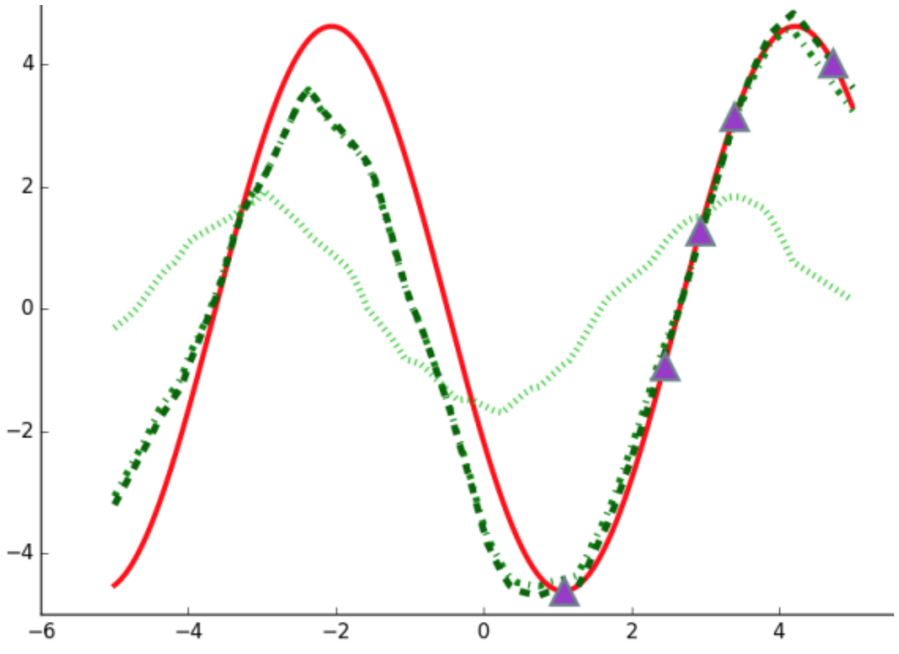
[Pretraining + Fine-tuning]



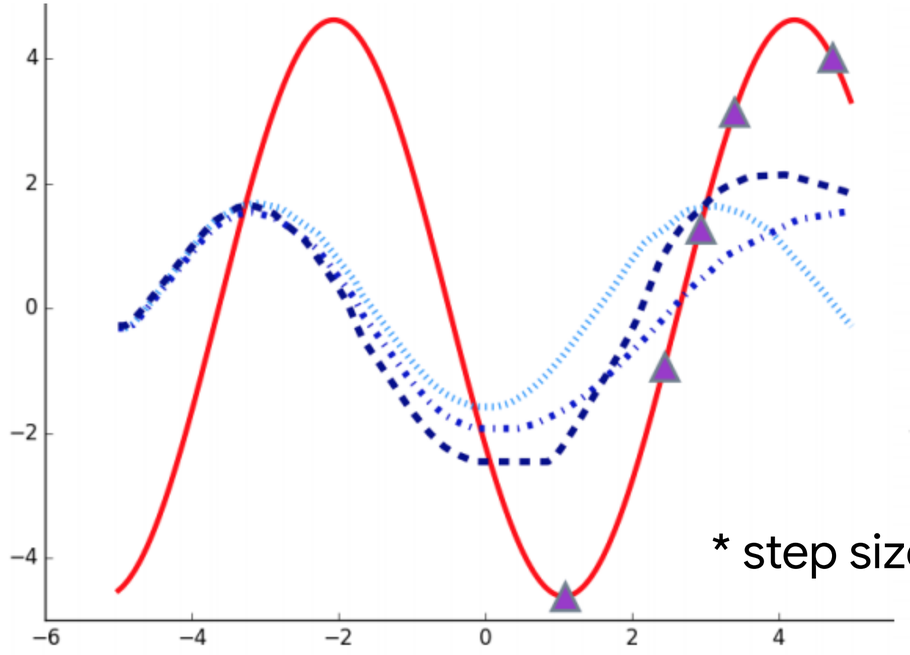
In the case of MAML, they quickly adapted to new samples.
But, adaptation failed in the case of pretraining model.

Results of 5-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]



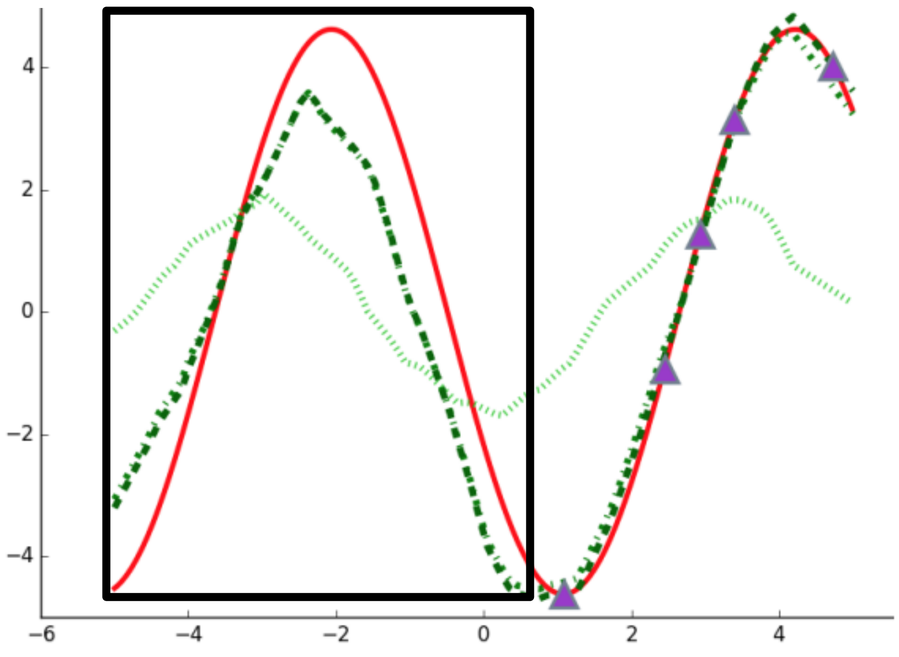
* step size 0.01

In the 5-shot learning, the difference is pervasive.

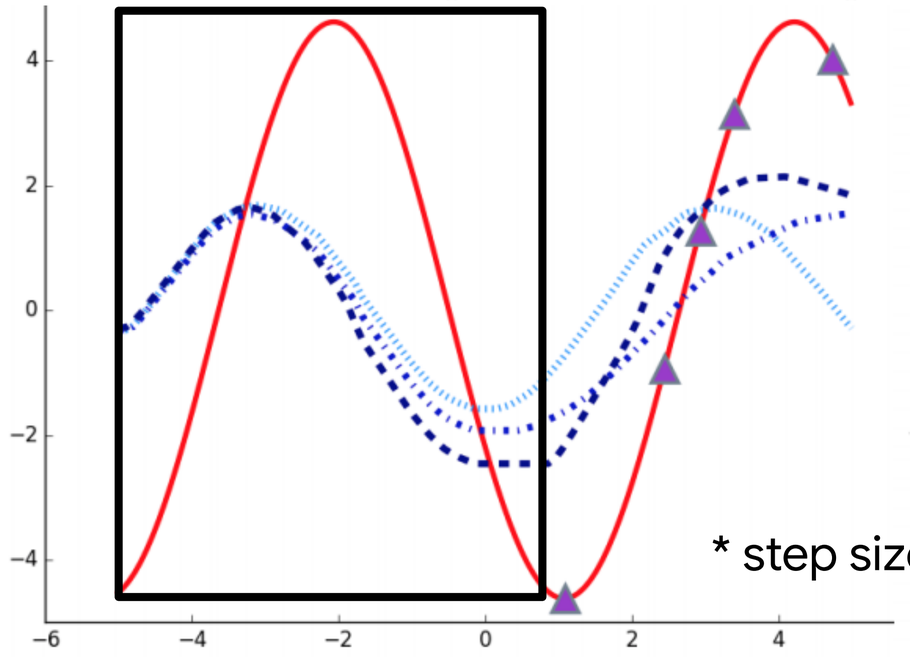
pre-update · 1 grad step · 10 grad steps · ground truth · ▲ used for grad · pre-update · 1 grad step · 10 grad steps

Results of 5-Shot Learning Regression

[MAML]



[Pretraining + Fine-tuning]



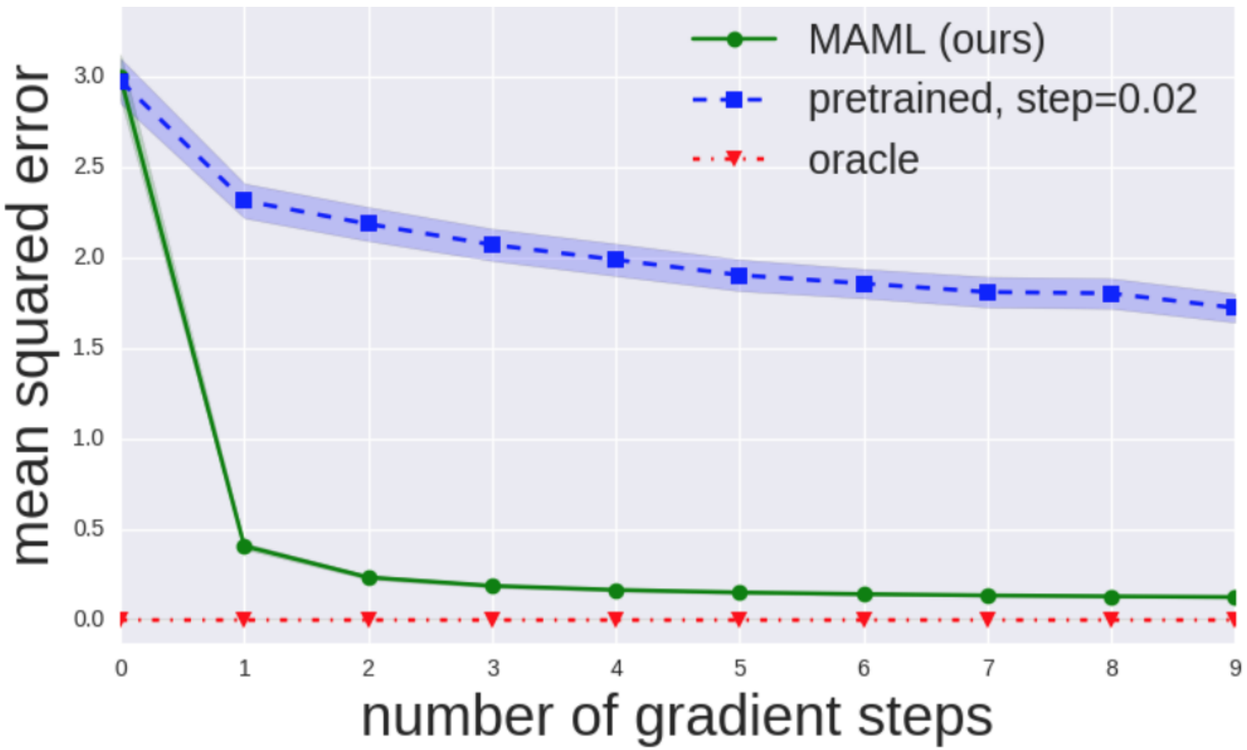
* step size 0.01

In the 5-shot learning, the difference is pervasive.
Good predictions are also made for ranges not particularly seen.

pre-update　1 grad step　10 grad steps　ground truth　▲ ▲ used for grad　pre-update　1 grad step　10 grad steps

5. Experiment

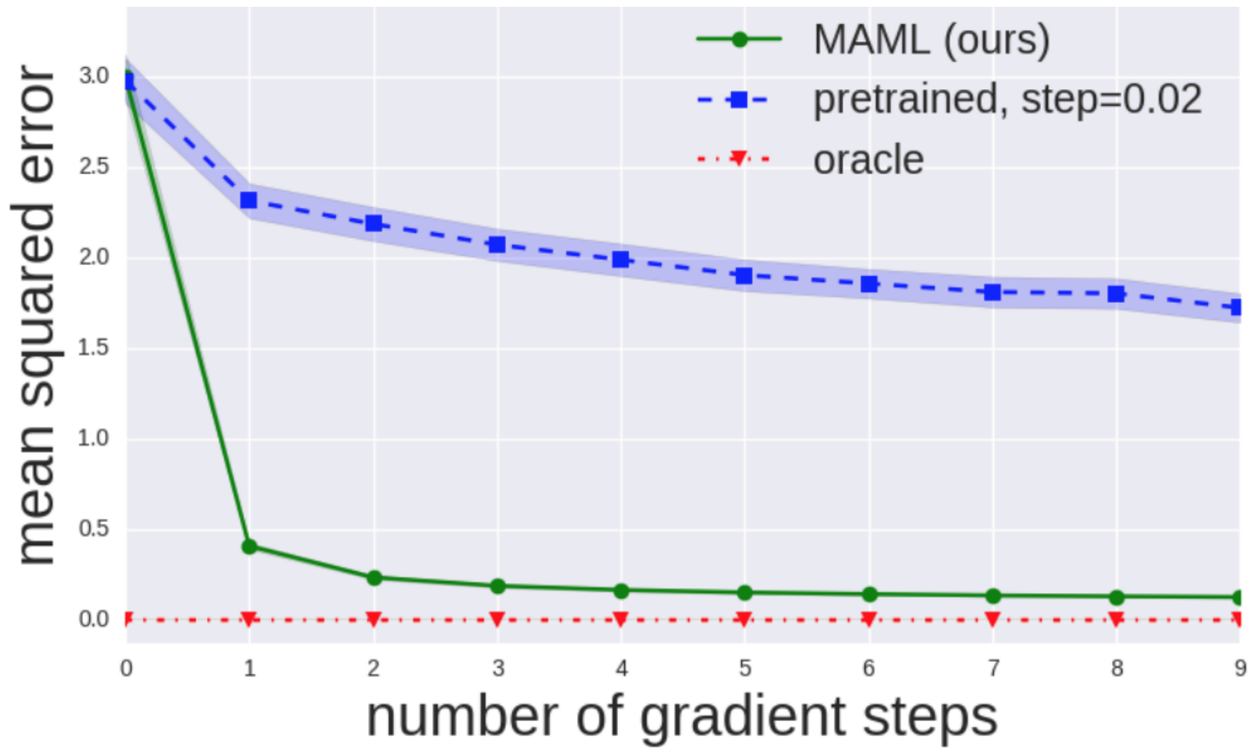
MAML Needs Only One Gradient Step



Vanilla pretrained model adapted slowly,
but, the MAML method quickly adapted **even in one gradient step**.

5. Experiment

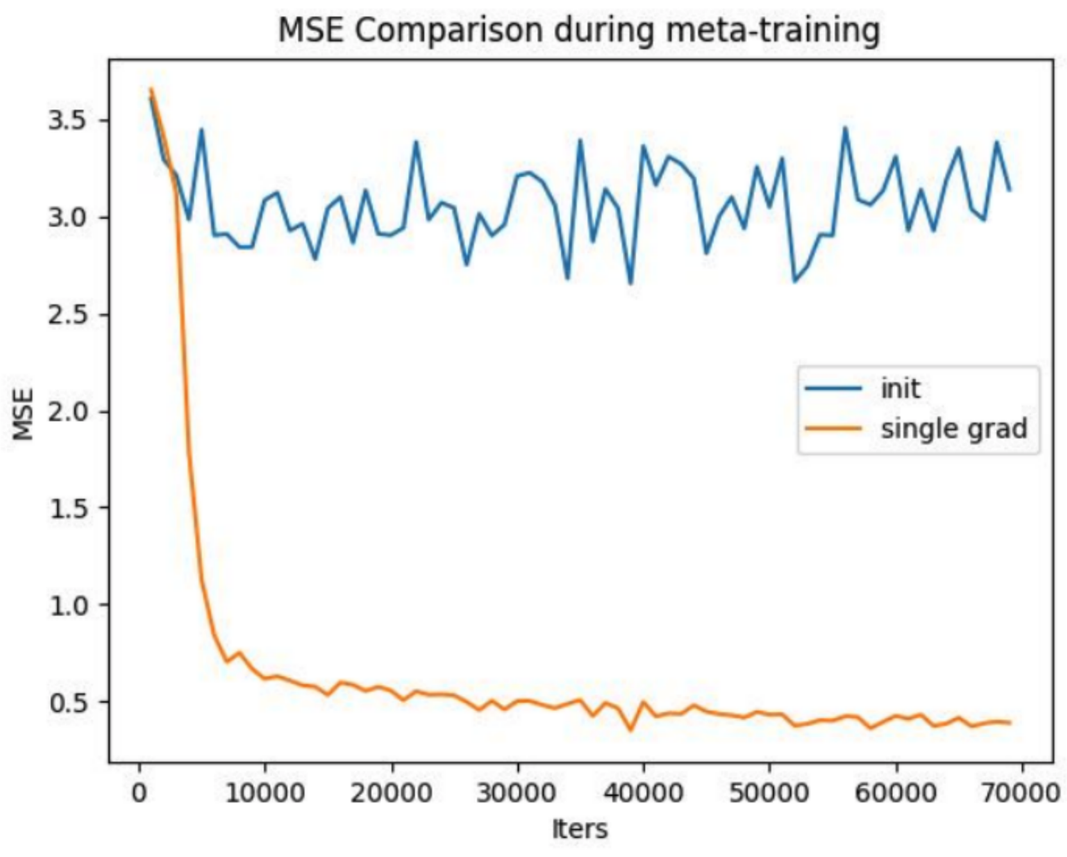
MAML Needs Only One Gradient Step



Vanilla pretrained model adapted slowly,
but, the MAML method quickly adapted **even in one gradient step**.
Meta parameter theta of MAML is **more sensitive**
than pre-updated theta of vanilla pretrained model.

5. Experiment

MSE of the Meta-Parameters



- The performance of the-meta parameters was not improved much in training.
- However, the performance of the single gradient updated parameters
  started on meta-parameters improved as training progressed.

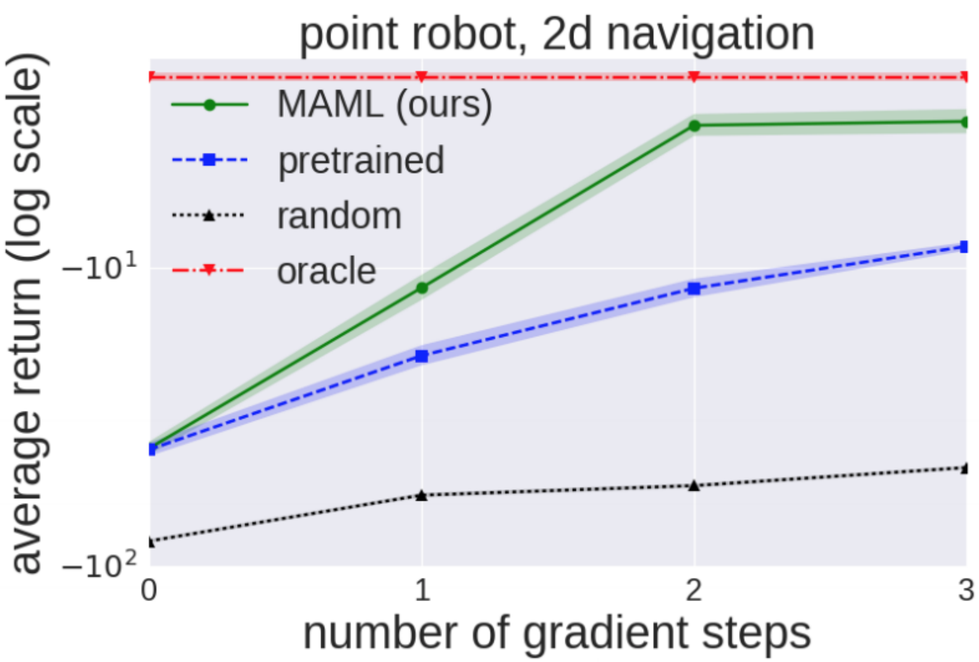
Experiments on Reinforcement Learning

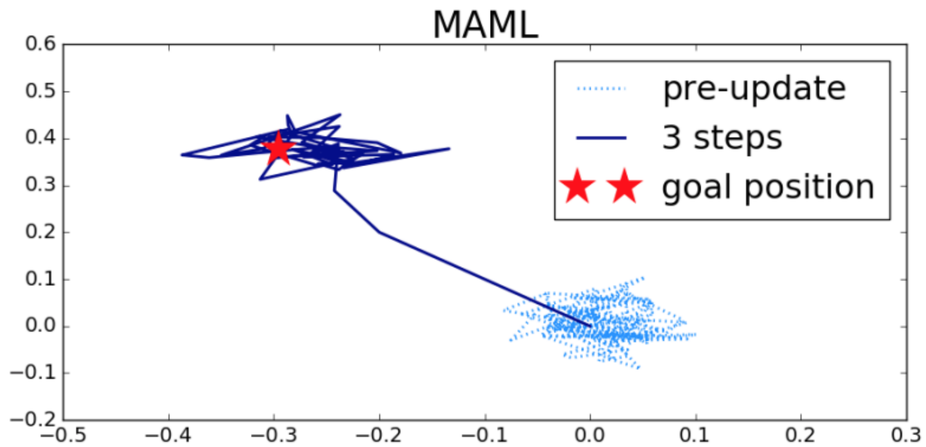
- rllab benchmark suite
- Neural network policy with two hidden layers of size 100 with ReLU
- Gradients updates are computed using vanilla policy gradient (REINFORCE)
  and trust-region policy (TRPO) optimization as meta-optimizer.

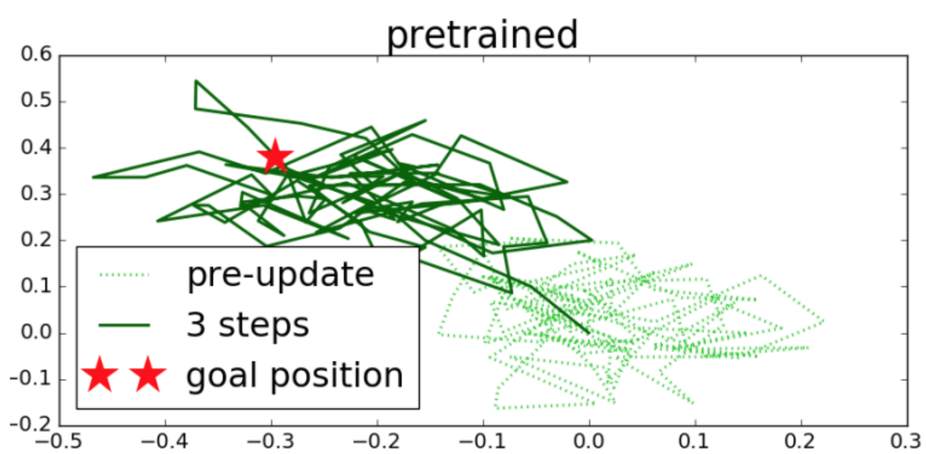
- Comparison
  - Pretraining one policy on all of the tasks and fine-tuning
  - Training a policy from randomly initialized weights
  - Oracle policy

Results on Reinforcement Learning

- 2d navigation



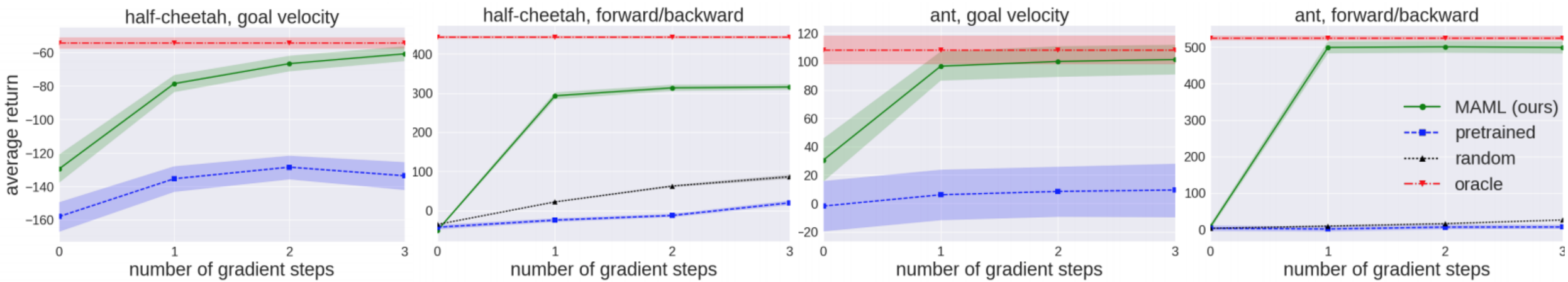




num. grad steps	0	1	2	3
context vector	−42.42	−13.90	−5.17	**−3.18**
MAML (ours)	−40.41	**−11.68**	**−3.33**	−3.23

Results on Reinforcement Learning

- Locomotion
  - High-dimensional locomotion tasks with the MuJoCo simulator



num. grad steps	0	1	2	3
context vector	-40.49	-44.08	-38.27	-42.50
MAML (ours)	-50.69	$\mathbf{293.19}$	$\mathbf{313.48}$	$\mathbf{315.65}$

Thank you!

Reference

- Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." arXiv preprint arXiv:1703.03400 (2017).
- Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." Advances in neural information processing systems. 2014.
- Baldi, Pierre. "Autoencoders, unsupervised learning, and deep architectures." Proceedings of ICML workshop on unsupervised and transfer learning. 2012.
- Li, Zhenguo, et al. "Meta-sgd: Learning to learn quickly for few shot learning." arXiv preprint arXiv:1707.09835 (2017)

Reference

- https://www.slideshare.net/TaesuKim3/pr12094-modelagnostic-metalearning-for-fast-adaptation-of-deep-networks
- https://people.eecs.berkeley.edu/~cbfinn/_files/nips2017_metaworkshop.pdf
- https://github.com/cbfinn/maml
- Lab seminar slide by Young-ki Hong.