Meta Learning for Low-Resource Natural Language Understanding

Sanghyun Seo

Feb 14, 2020

Department of Computer Engineering at Dongguk University

Artificial Intelligence Laboratory

Papers

- MAML, FOMAML (2017)
  - Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.
- Reptile (2018)
  - Nichol, Alex, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms." *arXiv preprint arXiv:1803.02999* (2018).
- Low-Resource NLP(2018)
  - Dou, Zi-Yi, Keyi Yu, and Antonios Anastasopoulos. "Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.

Survey Paper Overview

- Tentative Title
  - 소수샷 학습을 위한 메타러닝 연구동향 (A survey on meta learning algorithms for few-shot learning)
- Introduction
  - Meta learning vs Transfer learning or Multi-task Learning
- Problem Definition
  - Supervised learning → Meta Learning
  - Few-shot learning
  - Dataset
- Meta Learning Algorithms
  - Model based approaches
    - MANN, MetaNet, SNAIL
  - Metric based approaches
    - Matching Networks, Relation Networks, Prototypical Networks
    - Few-Shot Learning with Graph Neural Networks, Transductive Propagation Networks
  - Optimization based approaches
    - MAML, FOMAML, Meta-SGD, Reptile
    - MT-Net, LEO
    - Hierarchical bayesian model, probabilistic MAML, Bayesian MAML
  - Set-input approaches
    - Deep sets, Neural statistician

Meta Learning

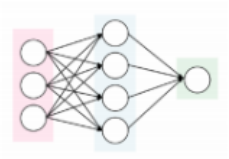
- Two ways to view meta-learning
  - **Mechanistic view**
    - Deep neural network model that can read in an entire dataset and make predictions for new datapoints
    - Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task
    - This view makes it easier to implement meta learning algorithms
  - Probabilistic view
    - Extract prior information from a set of (meta-training) tasks that allows efficient learning of new tasks
    - Learning a new task uses this prior and (small) training set to infer most likely posterior parameters
    - This view makes it easier to understand meta learning algorithms

- ICML 2019 tutorial, https://sites.google.com/view/icml19metalearning

Meta Learning

- Limits of Supervised Learning

supervised learning:



$$\arg\max_{\phi} \log p(\phi|\mathcal{D})$$

model parameters, training data

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

input (e.g., image), label

$$= \arg\max_{\phi} \log p(\mathcal{D}|\phi) + \log p(\phi)$$

data likelihood, regularizer (e.g., weight decay)

$$= \arg\max_{\phi} \sum_i \log p(y_i|x_i, \phi) + \log p(\phi)$$

What is wrong with this?

- ➢ The most powerful models typically require large amounts of labeled data
- ➢ Labeled data for some tasks may be very limited

- ICML 2019 tutorial, https://sites.google.com/view/icml19metalearning

Meta Learning

- Additional data
  - Unsupervised learning
  - Transfer learning
  - Multi-task learning
  - **Meta learning**

supervised learning:

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}) \qquad \mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

can we incorporate *additional* data?

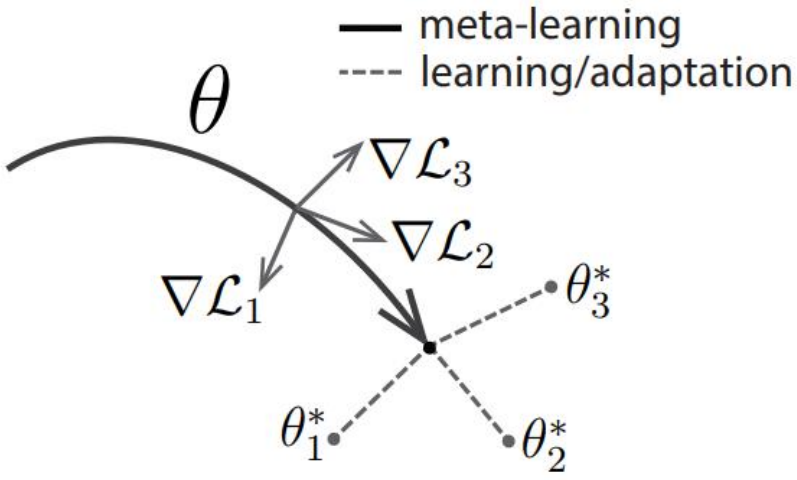
$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) \qquad \mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

- ICML 2019 tutorial, https://sites.google.com/view/icml19metalearning

MAML

- How MAML can learn prior knowledge from meta train dataset?
- Procedure of MAML algorithms



$$\phi \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}(\theta, D_i^{tr})$$

$$\min_\theta \sum_{task\ i} \mathcal{L}(\theta - \beta\nabla_\theta\mathcal{L}(\theta, D_i^{tr}), D_i^{ts})$$

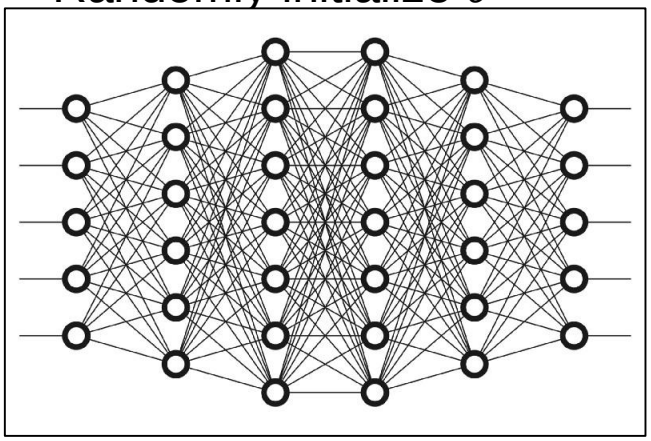
Algorithm 2 MAML for Few-Shot Supervised Learning

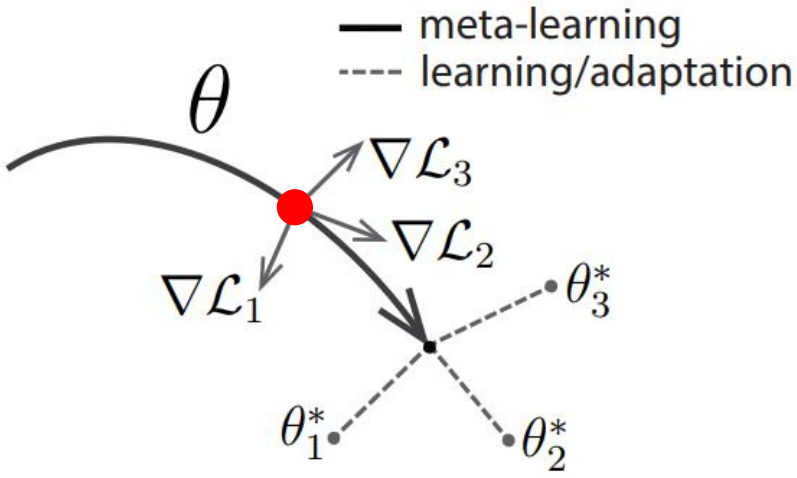
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

- Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

MAML

- Meta Train (Meta Learning)
- Randomly initialize θ



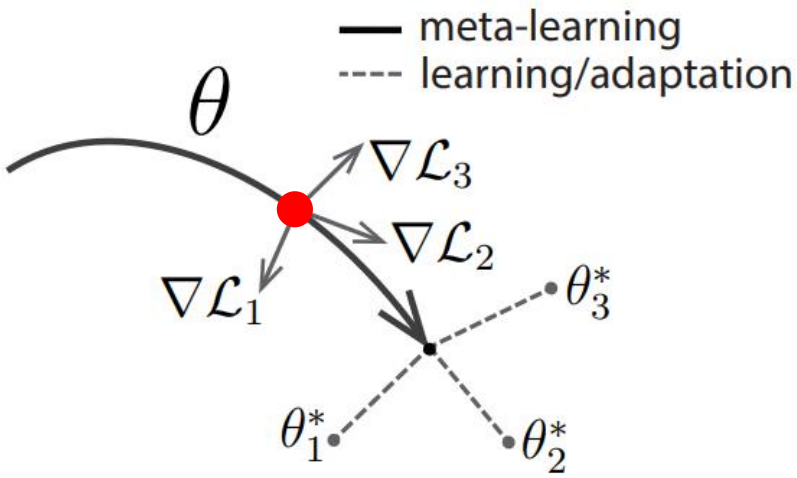


Algorithm 2 MAML for Few-Shot Supervised Learning

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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Sample batch of tasks $T_i \sim p(T)$

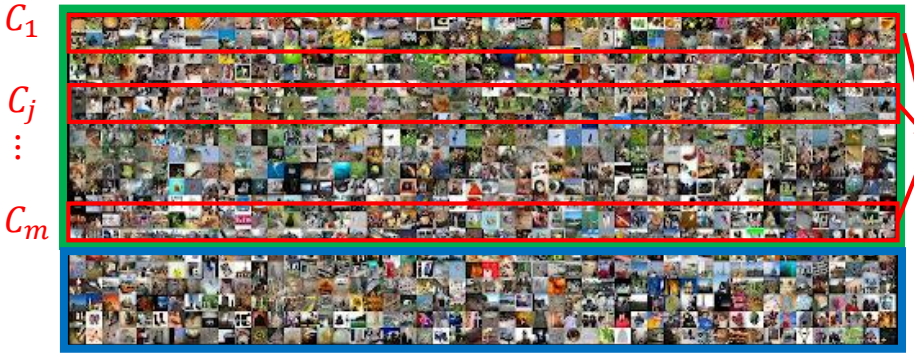


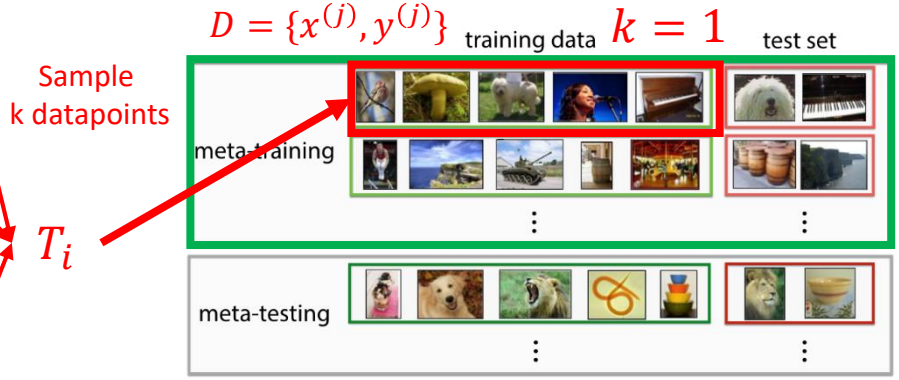
Algorithm 2 MAML for Few-Shot Supervised Learning

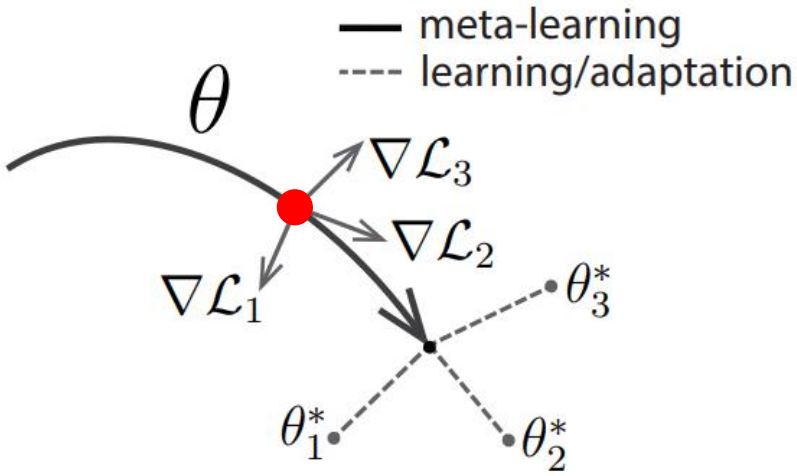
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Sample K datapoints D







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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

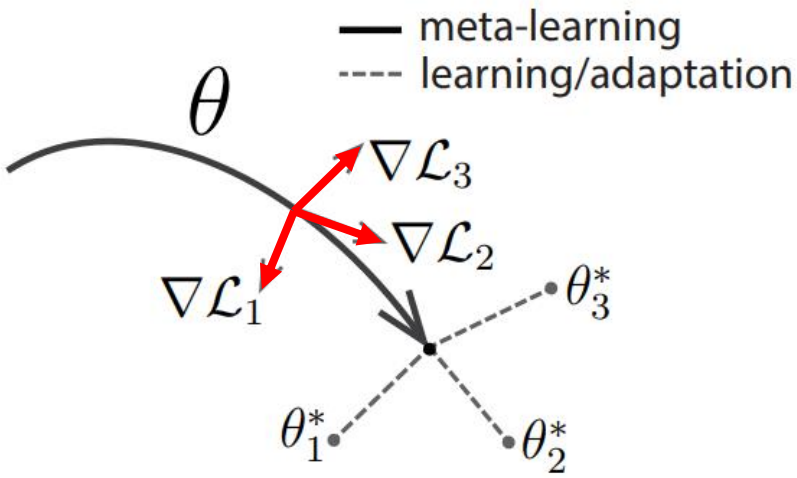
MAML

- Regression tasks: MSE

$$(2)\ \mathcal{L}_{T_i}(f_\phi) = \sum_{x^{(j)}, y^{(j)} \sim T_i} \left\| f_\phi(x^{(j)}) - y^{(j)} \right\|_2^2$$

- Classification tasks: CE

$$(3)\ \mathcal{L}_{T_i}(f_\phi) = \sum_{x^{(j)}, y^{(j)} \sim T_i} y^{(j)} \log f_\phi(x^{(j)}) + (1 - y^{(j)}) \log(1 - f_\phi(x^{(j)}))$$



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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

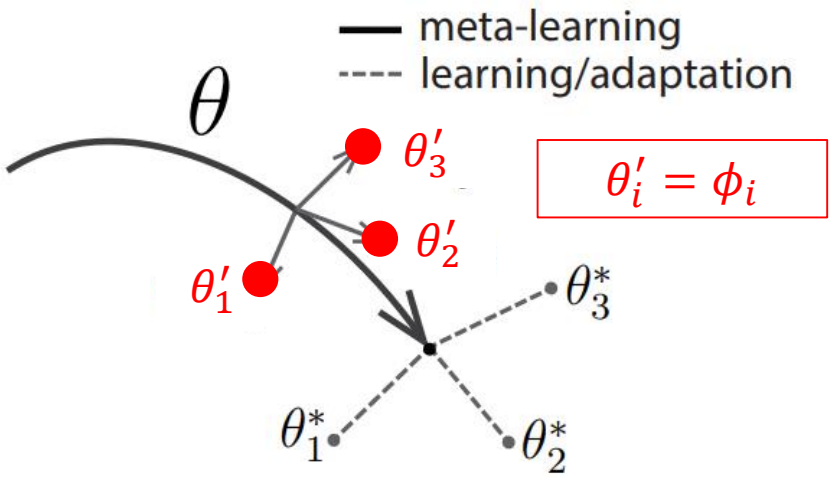
MAML

- Regression tasks: MSE

$$(2)\ \mathcal{L}_{T_i}(f_\phi) = \sum_{x^{(j)},y^{(j)}\sim T_i} \left\| f_\phi(x^{(j)}) - y^{(j)} \right\|_2^2$$

- Classification tasks: CE

$$(3)\ \mathcal{L}_{T_i}(f_\phi) = \sum_{x^{(j)},y^{(j)}\sim T_i} y^{(j)} \log f_\phi(x^{(j)}) + (1-y^{(j)}) \log(1 - f_\phi(x^{(j)}))$$

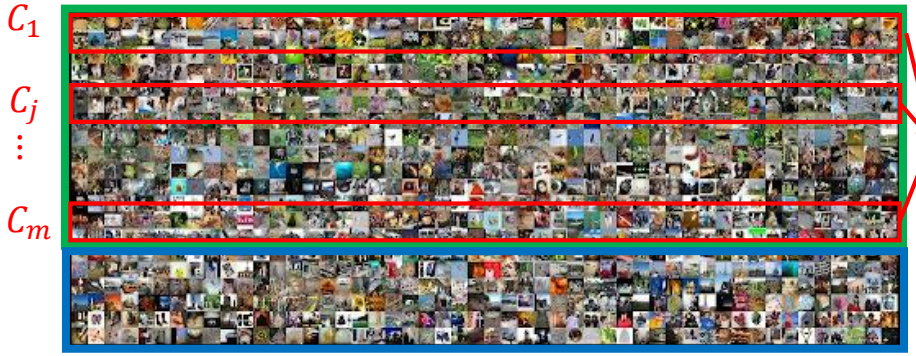


Algorithm 2 MAML for Few-Shot Supervised Learning

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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

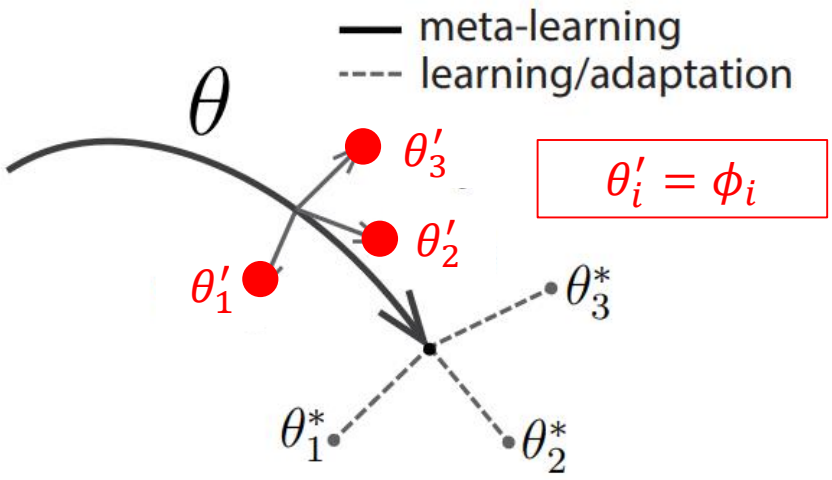
- Sample datapoints D_i'



Sample datapoints
From new task T_i

T_i

$D_i' = \{x^{(j)}, y^{(j)}\}$



$\theta_i' = \phi_i$

Algorithm 2 MAML for Few-Shot Supervised Learning

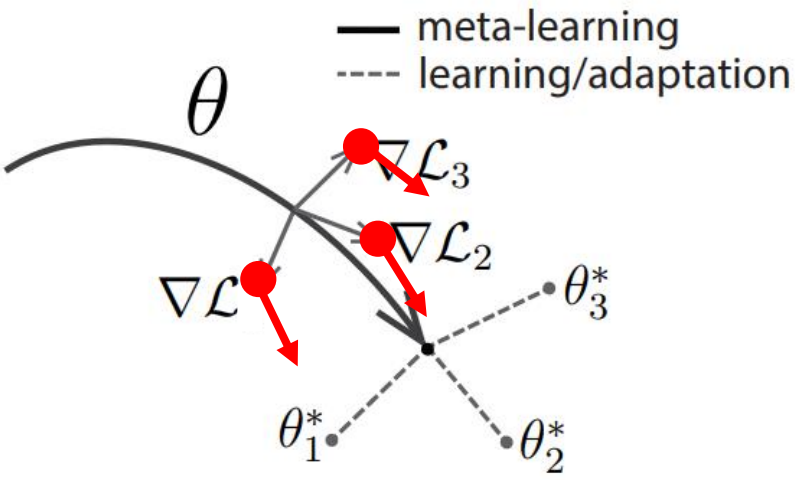
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Why MAML uses D_i' and θ_i'?
  - D_i and D_i' are disjoint

Note that the meta-optimization is performed over the model parameters θ, whereas the objective is computed using the updated model parameters θ'. In effect, our proposed method aims to optimize the model parameters such that one or a small number of gradient steps on a new task will produce maximally effective behavior on that task.

$$\beta \nabla_\theta \sum_{T_i \sim p(T)} \mathcal{L}_{T_i}\left(f_{\theta_i'}\right)$$



$D_i' = \{x^{(j)}, y^{(j)}\}$

Algorithm 2 MAML for Few-Shot Supervised Learning

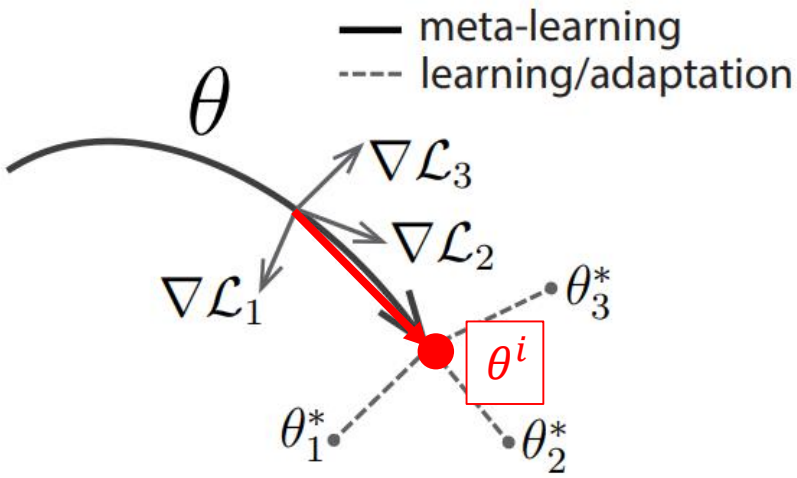
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Meta update

$$D_i' = \{x^{(j)}, y^{(j)}\}$$

$$\min_{\theta} \sum_{T_i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, D_i), D_i')$$

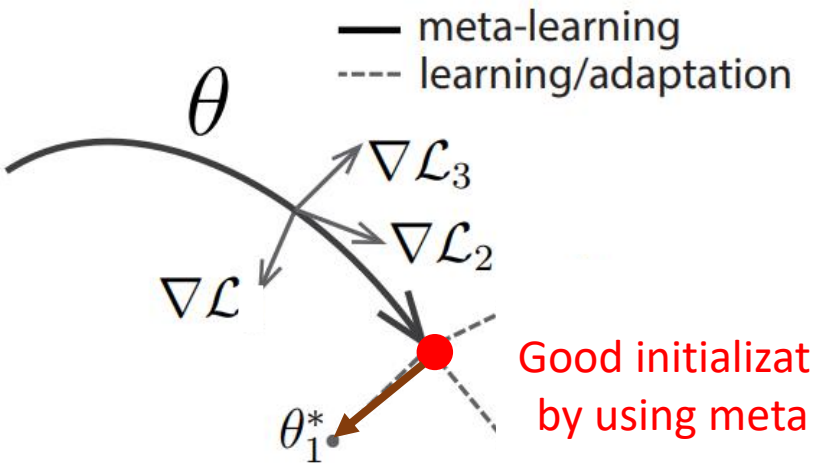


Algorithm 2 MAML for Few-Shot Supervised Learning

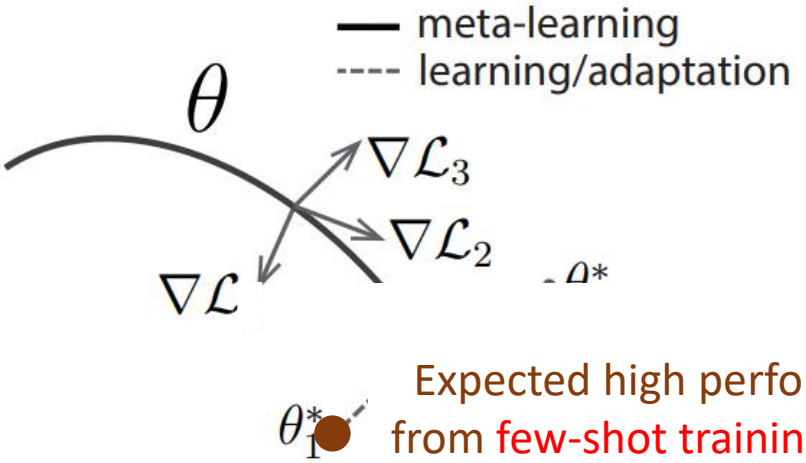
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Meta Test (Learning→Evaluation)



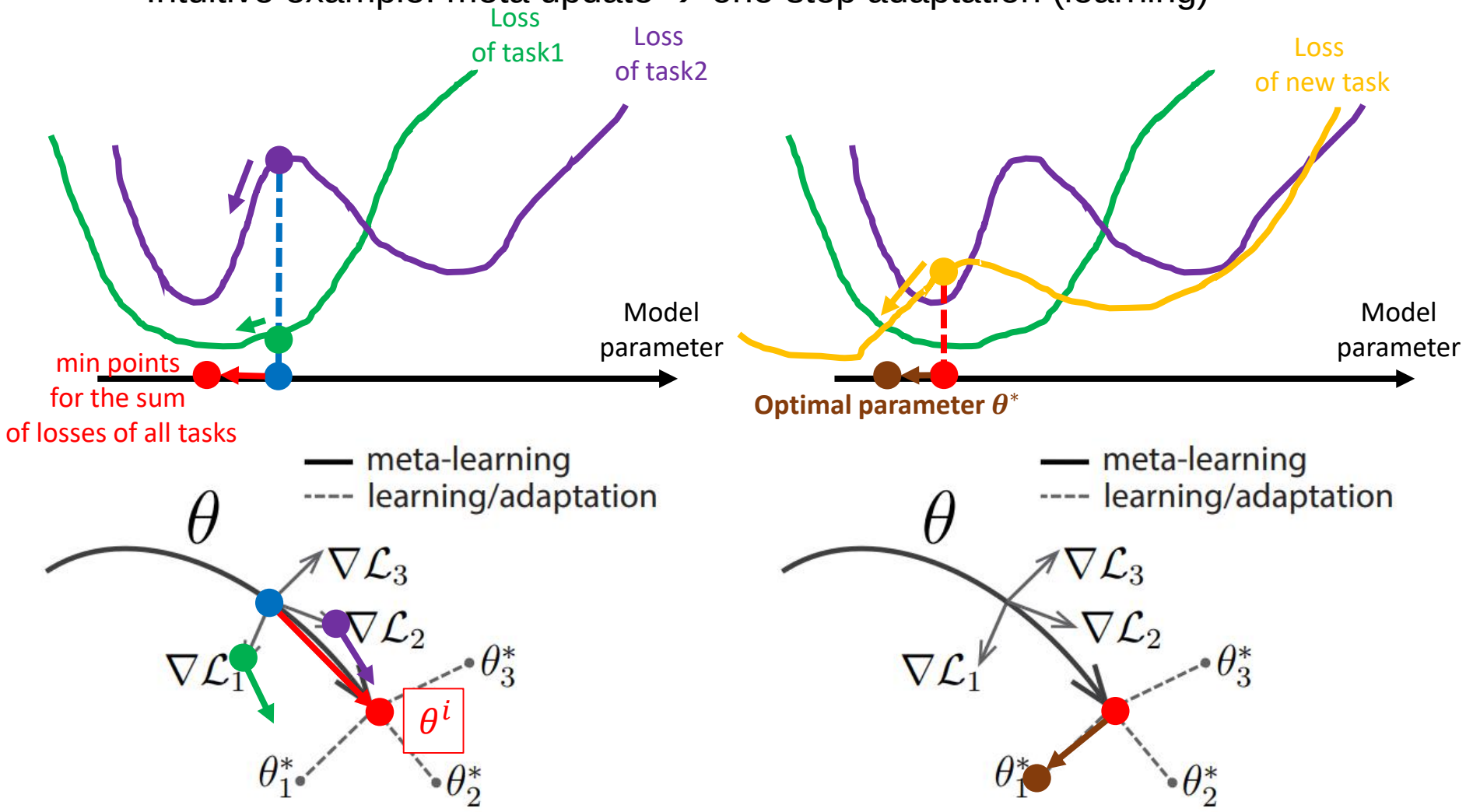
Good initialization point
by using meta learning



Expected high performance
from few-shot training dataset

MAML

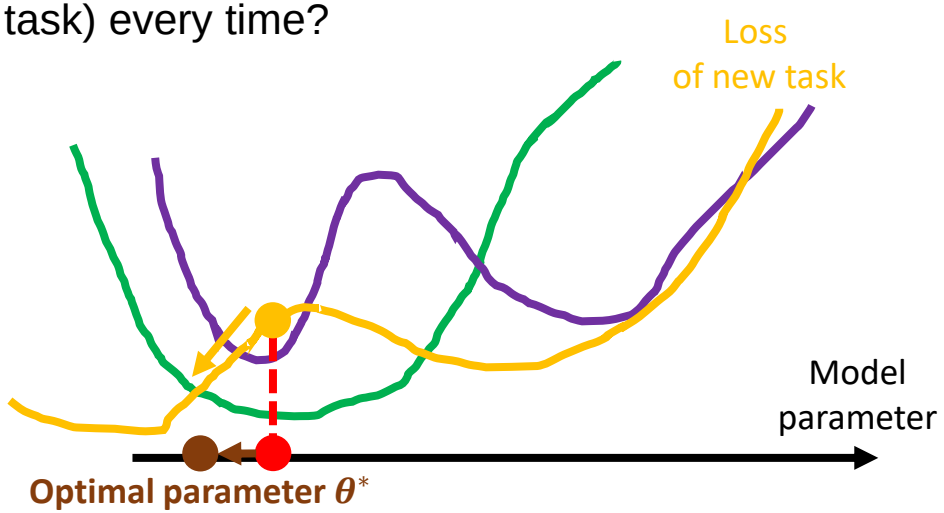
- Intuitive example: meta update → one-step adaptation (learning)

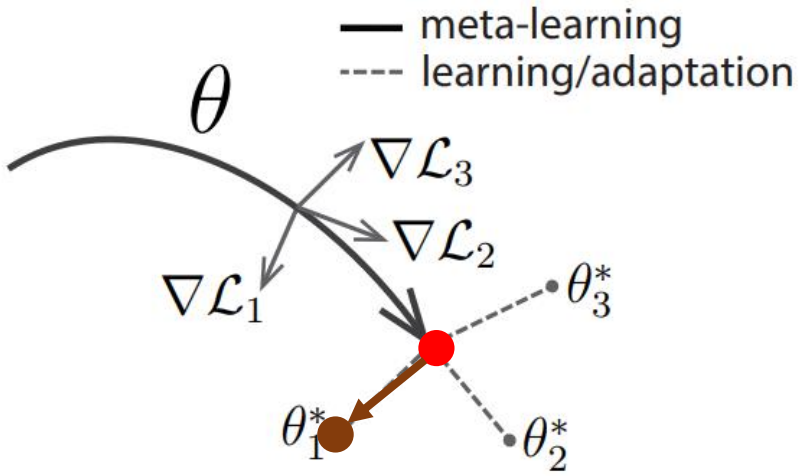


MAML

- Intuitive example
  - Can we meet sweet examples(new task) every time?

Key idea
"our training procedure is based on a simple machine learning principle: test and train conditions must match"



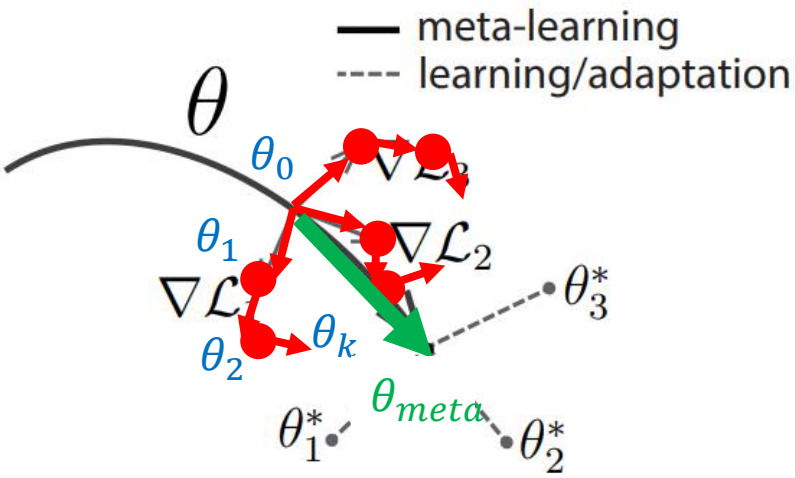


MAML

- Computational Problem
  - Consider the case of k>1 inner gradient step

$$\theta_0 = \theta_{meta}$$
$$\theta_1 = \theta_0 - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_0)$$
$$\theta_2 = \theta_1 - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_1)$$
$$\dots$$
$$\theta_k = \theta_{k-1} - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_{k-1})$$

$$\theta_0 = \theta_{meta} = \theta_{meta} - \beta \nabla_\theta \mathcal{L}^{(1)}(\theta_k)$$



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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ ← What if K iteration?
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

MAML

- Computational Problem
  - Consider the case of k>1 inner gradient step

$$\nabla_\theta \mathcal{L}^{(1)}(\theta_k)$$
$$= \frac{\partial \mathcal{L}^{(1)}(\theta_k)}{\partial \theta} = \frac{\partial \mathcal{L}^{(1)}(\theta_k)}{\partial \theta_k} \cdot \frac{\partial \theta_k}{\partial \theta}^T$$
$$= \frac{\partial \mathcal{L}^{(1)}(\theta_k)}{\partial \theta_k} \cdot \prod_{i=1}^{k} \frac{\partial \theta_i}{\partial \theta_{i-1}}^T$$
$$= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \nabla_{\theta_{k-1}} \mathcal{L}(\theta_k) \cdots \nabla_{\theta_0} \mathcal{L}(\theta_1) \cdot \nabla_\theta \theta_0$$
$$= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} \nabla_{\theta_{i-1}} \mathcal{L}(\theta_i)$$
$$= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} \nabla_{\theta_{i-1}} \left(\theta_{i-1} - \alpha \nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})\right)$$
$$= \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \prod_{i=1}^{k} (I - \alpha \nabla_{\theta_{i-1}} \left(\nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})\right))$$

$$\theta_{meta} = \theta_{meta} - \beta \nabla_\theta \mathcal{L}^{(1)}(\theta_k)$$

FOMAML

$$\nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k)$$
$$\cdot \prod_{i=1}^{k} (I - \alpha \nabla_{\theta_{i-1}} \left(\nabla_\theta \mathcal{L}^{(0)}(\theta_{i-1})\right))$$

→ second order derivative problem

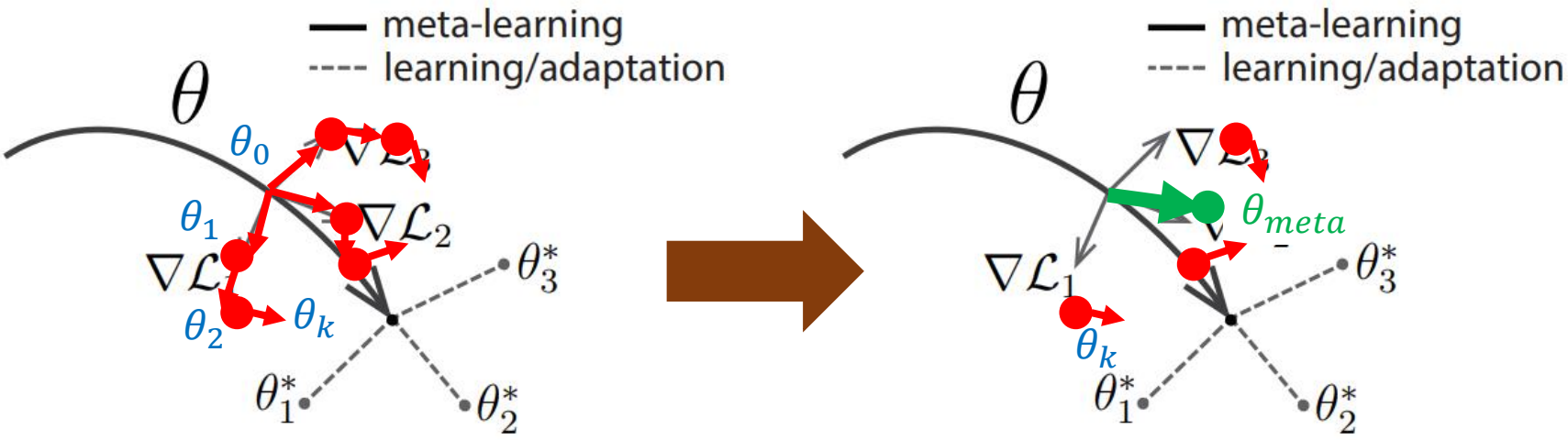
FOMAML

- First-Order MAML ignores the second derivative part

$$\nabla_{\theta_k}\mathcal{L}^{(1)}(\theta_k)\cdot\prod_{i=1}^{k}\left(I-\alpha\nabla_{\theta_{i-1}}\left(\nabla_{\theta}\mathcal{L}^{(\theta)}(\theta_{i-1})\right)\right)$$

$$\theta_{meta}=\theta_{meta}-\beta\nabla_{\theta}\mathcal{L}^{(1)}(\theta_k)$$

$$\theta_{meta}=\theta_{meta}-\beta\nabla_{\boldsymbol{\theta_k}}\mathcal{L}^{(1)}(\theta_k)$$



Reptile

- The Reptile works by repeatedly:
  - 1) sampling a task,
  - 2) training on it by multiple gradient descent steps,
  - 3) and then moving the model weights towards the new parameters.

Algorithm 1 Reptile (serial version)

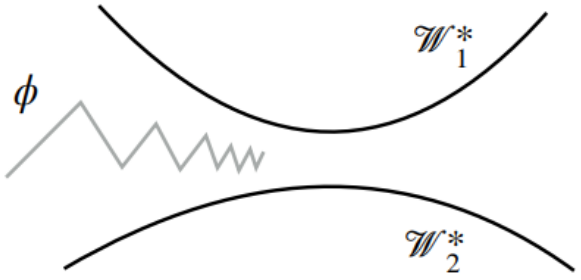
Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

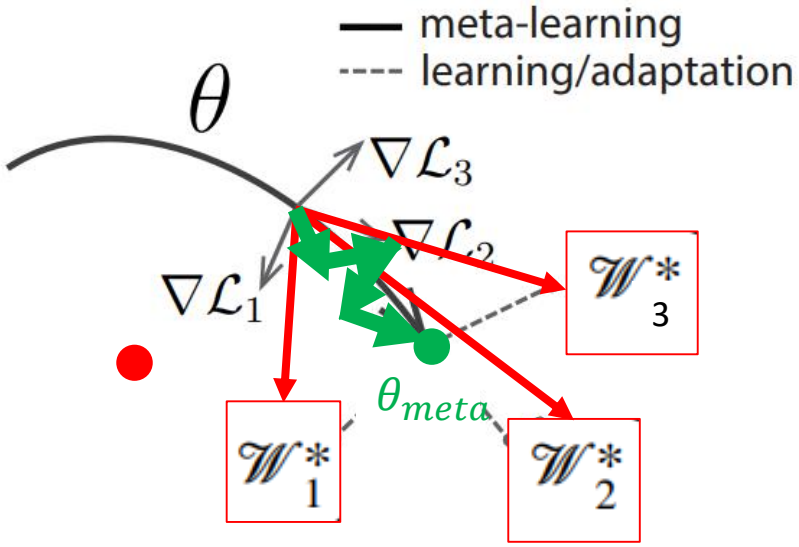
U corresponds to performing gradient descent

- Nichol, Alex, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms." arXiv preprint arXiv:1803.02999 (2018).

Reptile

- Intuitive example
  - Assuming that a task $\tau \sim p(\tau)$ has a manifold of optimal network configuration, W_τ^*. The model f_θ achieves the best performance for task τ when θ lays on the surface of W_τ^*
  - To find a solution that is good across tasks, we would like to find a parameter close to all the optimal manifolds of all tasks

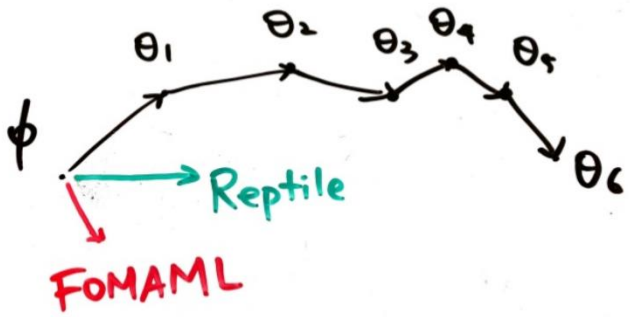


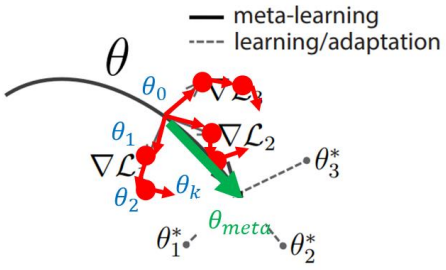


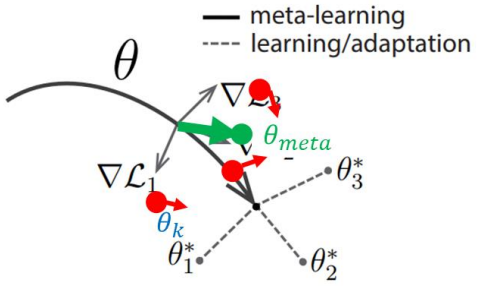
- Nichol, Alex, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms." arXiv preprint arXiv:1803.02999 (2018).

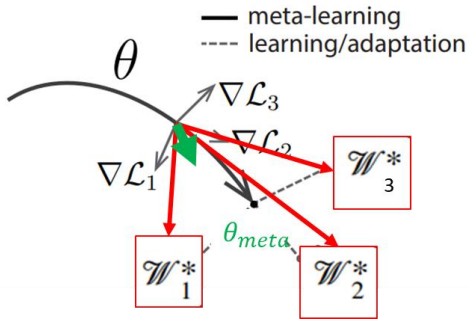
MAML vs FOMAML vs Reptile

- Can you know the difference between each methodology in the following figure?









$$\mathbb{E}\left[g_{\text{MAML}}\right] = (1)\text{AvgGrad} - (2\alpha)\text{AvgGradInner} + O(\alpha^2) \quad (33)$$

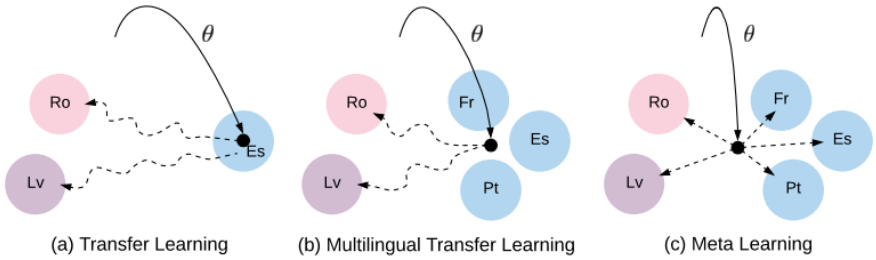
$$\mathbb{E}\left[g_{\text{FOMAML}}\right] = (1)\text{AvgGrad} - (\alpha)\text{AvgGradInner} + O(\alpha^2) \quad (34)$$

$$\mathbb{E}\left[g_{\text{Reptile}}\right] = (2)\text{AvgGrad} - (\alpha)\text{AvgGradInner} + O(\alpha^2) \quad (35)$$

- https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html, slides on reptile by Yoonho Lee
- Nichol, Alex, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms." arXiv preprint arXiv:1803.02999 (2018).

Meta learning for NLP

- Meta learning for low-resource NLU
  - Meta learning can make low-resource NLP model
  - Meta learning for Machine translation
- GLUE
  - The General Language Understanding Evaluation benchmark (https://gluebenchmark.com/) is a collection of resources for training, evaluating, and analyzing natural language understanding systems.




glue

glue/cola
Statistics
Features
Homepage

GLUE, the General Language Understanding Evaluation benchmark (https://gluebenchmark.com/) is a collection of resources for training, evaluating, and analyzing natural language understanding systems.

The Corpus of Linguistic Acceptability consists of English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence.

- URL: https://nyu-mll.github.io/CoLA/
- DatasetBuilder: tfds.text.glue.Glue

glue is configured with tfds.text.glue.GlueConfig and has the following configurations predefined (defaults to the first one):

- cola (v1.0.0) (Size: 368.14 KiB): The Corpus of Linguistic Acceptability consists of English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence.
- sst2 (v1.0.0) (Size: 7.09 MiB): The Stanford Sentiment Treebank consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence. We use the two-way (positive/negative) class split, and use only sentence-level labels.
- mrpc (v1.0.0) (Size: 1.43 MiB): The Microsoft Research Paraphrase Corpus (Dolan & Brockett, 2005) is a corpus of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent.
- qqp (v1.0.0) (Size: 57.73 MiB): The Quora Question Pairs2 dataset is a collection of question pairs from the community question-answering website Quora. The task is to determine whether a pair of questions are semantically equivalent.

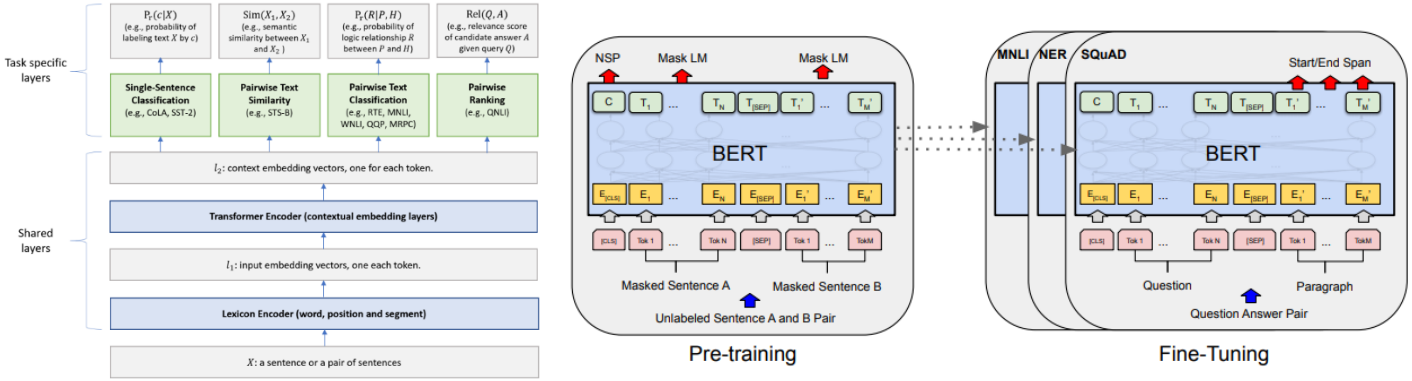
- Gu, Jiatao, et al. "Meta-Learning for Low-Resource Neural Machine Translation." Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.
- Dou, Zi-Yi, Keyi Yu, and Antonios Anastasopoulos. "Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.
- https://www.tensorflow.org/datasets/catalog/glue

Meta learning for NLP

- Meta learning for low-resource NLU

High-resource tasks	Target tasks 1
• SST-2: The Stanford Sentiment Treebank consists of sentences from movie reviews and human annotations of their sentiment • QQP: The Quora Question Pairs2 dataset • MNLI: The Multi-Genre Natural Language Inference Corpusn • QNLI: The Stanford Question Answering Dataset	• CoLA: The Corpus of Linguistic Acceptability consists of English acceptability judgments • MRPC: The Microsoft Research Paraphrase Corpus • STS-B: The Semantic Textual Similarity Benchmark • RTE: The Recognizing Textual Entailment Target tasks 2 • SciTail: entailment dataset created from multiple-choice science exams and web sentences.

- Comparison model
  - MT-DNN
  - BERT



- Liu, Xiaodong, et al. "Multi-Task Deep Neural Networks for Natural Language Understanding." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019.
- Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019.

Meta learning for NLP

- Experiment Results
  - Uniform, Probability Proportional to Size(PPS), Mixed

Model	Test Dataset			
	CoLA	MRPC	STS-B	RTE
BERT	52.1	88.9/84.8	87.1/85.8	66.4
MT-DNN	51.7	89.9/86.3	87.6/86.8	75.4
MAML	**53.4**	89.5/85.8	88.0/87.3	76.4
FOMAML	51.6	89.9/86.4	88.6/88.0	74.1
Reptile	53.2	**90.2/86.7**	**88.7/88.1**	**77.0**

Table 1: Results on GLUE test sets. Metrics differ per task (explained in Appendix A) but the best result is **highlighted**.

Model	#Upt	α	CoLA	MRPC	STS-B	RTE
Reptile	3	1e-3	60.7	89.7	90.2	77.9
	5	1e-4	**62.0**	88.0	90.1	81.2
		1e-3	61.6	**90.0**	**90.3**	**83.0**
		1e-2	60.1	87.8	89.5	73.9
	7	1e-3	57.8	88.7	90.0	81.4

Table 3: Effect of the number of update steps and the inner learning rate α.

Model	CoLA	MRPC	STS-B	RTE
Reptile-PPS	**61.6**	**90.0**	**90.3**	**83.0**
Reptile-Uniform	61.5	84.0	**90.3**	75.7
Reptile-Mixed 2:1	60.3	87.8	**90.3**	71.0
Reptile-Mixed 5:1	**61.6**	85.8	90.1	74.7

Table 2: Effect of task distributions. We report the accuracy or Matthews correlation on development sets.

Figure 2: Results on transfer learning. The target task is SciTail which the model does not come across during the meta-learning stage.

- Dou, Zi-Yi, Keyi Yu, and Antonios Anastasopoulos. "Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.

Q&A
Thank you!