Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

ICML 2017

SangHyeon Lee

Dept. of Artificial intelligence

2023.3.30.

Contents

(ICML 2017) Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Chelsea Finn, Pieter Abbeel, Sergey Levine

Introduction

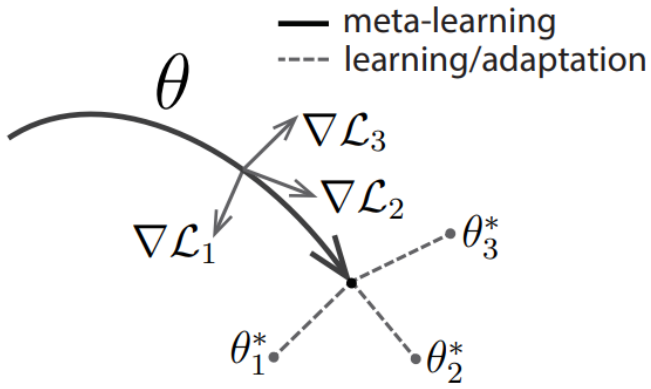
Challenge

- Fast and Flexible learning is challenging
  - The agent must integrate its prior experience with a small amount of new information, while avoiding overfitting to the new data.
  - The form of prior experience and new data will depend on the task.

Contribution

1. We proposed a general and agnostic meta-learning approach.
   - It can be applied to many types of models that use gradient descent.
2. We applied MAML to several algorithms, including fully connected, CNN, few-shot regression, image classification, and RL, and demonstrated that it works.
3. In particular, compared to the SOTA one-shot meta-learning algorithm in supervised learning, MAML performed better with fewer parameters.

Introduction



Key-Idea

- Train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task.
- Feature learning standpoint
  - Building an internal representation that is broadly suitable for many tasks.
- Dynamical systems standpoint
  - The learning process can be viewed as maximizing the sensitivity of the loss functions of new tasks with respect to the parameters.

Background

Terminology

- x: observations(inputs)
- a: outputs
- f_θ: A model parametrized by θ
  - $f_\theta(\mathrm{x}) = \mathrm{a}$
- $\mathcal{L}(\mathrm{x}_1, \mathrm{a}_1, \ldots, \mathrm{x}_H, \mathrm{a}_H) \to \mathbb{R}$: loss function
- $q(\mathrm{x}_1)$: distribution over initial observations
- $q(\mathrm{x}_{t+1}|\mathrm{x}_t, \mathrm{a}_t)$: transition distribution
- H: horizon, sample length
- $\mathcal{T} = \{\mathcal{L}(\mathrm{x}_1, \mathrm{a}_1, \ldots, \mathrm{x}_H, \mathrm{a}_H),\ q(\mathrm{x}_1),\ q(\mathrm{x}_{t+1}|\mathrm{x}_t, \mathrm{a}_t),\ H\}$: task
- $p(\mathcal{T})$: distribution of tasks

Method

MAML(Model-Agnostic Meta-Learning)

Algorithm 1 Model-Agnostic Meta-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: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

1. Set the learning rate α, meta-learning rate β, task distribution $p(\mathcal{T})$ and randomly initialize the model's parameter θ.
2. Select i(batch-size) tasks according to $p(\mathcal{T})$ and start an inner learning loop for each task.
3. Select K(K-shot) train samples from task $\mathcal{T}_i$, then calculate the loss and gradient of model f_θ.

4. Instead of updating the existing parameter θ immediately, save the parameter separately as θ_i' for each task $\mathcal{T}_i$.

5. After the inner loop terminates, calculate the loss and gradient of model $f_{\theta_i'}$ using test data from task $\mathcal{T}_i$, then update θ by calculating the sum of the gradients.

$\rightarrow$ The MAML fine-tuning each task based on the meta-parameter θ and then updates θ by considering all the gradient directions of the tuned parameter θ_i'.

Method

MAML for 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 $\mathcal{L}_{\mathcal{T}_i}$ 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**

- Loss function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathrm{x}^{(j)}, \mathrm{y}^{(j)} \sim \mathcal{T}_i} ||f_\phi(\mathrm{x}^{(j)}) - \mathrm{y}^{(j)}||_2^2 \qquad \text{(MSE)}$$

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathrm{x}^{(j)}, \mathrm{y}^{(j)} \sim \mathcal{T}_i} \mathrm{y}^{(j)} \log f_\phi(\mathrm{x}^{(j)}) + (1 - \mathrm{y}^{(j)}) \log(1 - f_\phi(\mathrm{x}^{(j)})) \qquad \text{(Cross-entropy)}$$

- line 8: sampling query set $\mathcal{D}_i'$

Method

MAML for 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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
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 $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

- Loss function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{\mathrm{x}_t, \mathrm{a}_t \sim f_\phi, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} R_i(\mathrm{x}_t, \mathrm{a}_t) \right] \qquad \text{(negative reward function)}$$

- line 8: sampling query set $\mathcal{D}_i'$

Method

MAML using first-order approximation(in experiments)

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{1}$$

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

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_\theta \theta_i') \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \ , \ \text{(by chain rule)} \tag{3}$$

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \{\nabla_\theta (\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta))\} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{4}$$

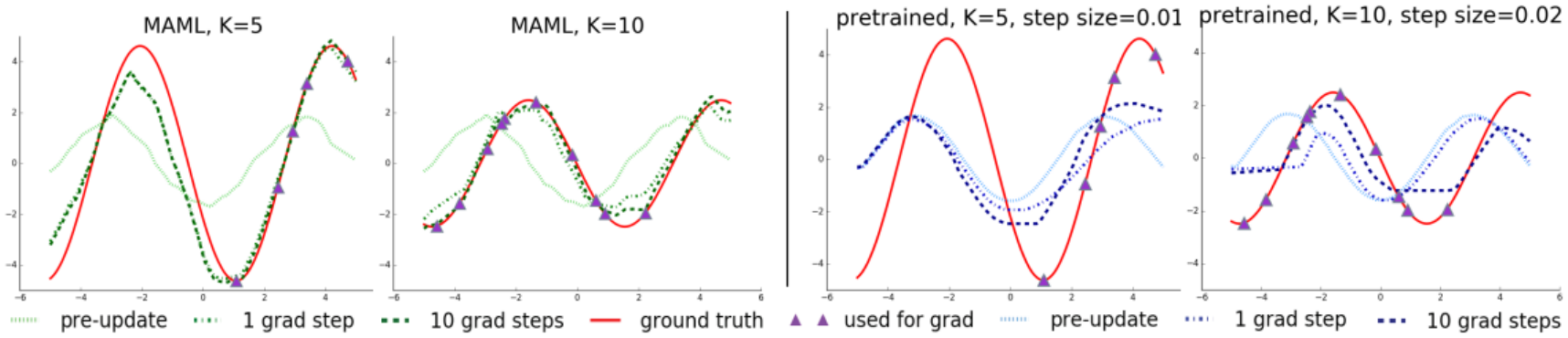
$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta)) \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{5}$$

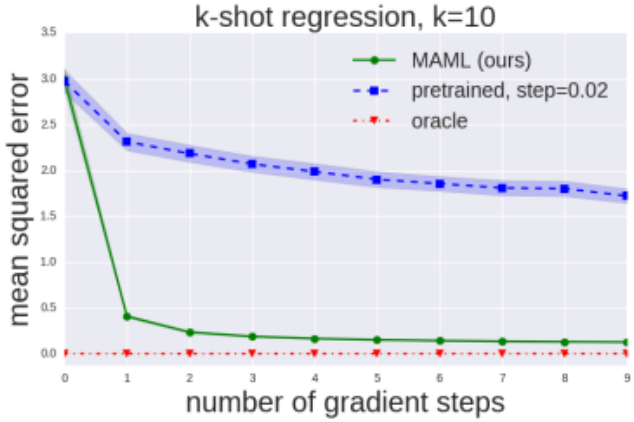
$$\approx \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) \tag{6}$$

- The second derivative $\nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ is computationally expensive, so in this paper, also experimented with a method that reduces the amount of computation by using a first-order approximation.

Experiments

Regression





- Figure 1 shows that The MAML learns well even though K=5 samples are not enough to represent the sin wave. On the other hand, the pretrained model does not learn well.

  → MAML properly learns the structural features of sin waves during meta-training

- Figure 2 shows that the MAML can quickly adapt to new tasks even one-gradient step.

Experiments

Classification

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%**

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	5-shot
fine-tuning baseline	28.86 ± 0.54%	49.79 ± 0.79%
nearest neighbor baseline	41.08 ± 0.70%	51.04 ± 0.65%
matching nets (Vinyals et al., 2016)	43.56 ± 0.84%	55.31 ± 0.73%
meta-learner LSTM (Ravi & Larochelle, 2017)	43.44 ± 0.77%	60.60 ± 0.71%
MAML, first order approx. (ours)	**48.07 ± 1.75%**	**63.15 ± 0.91%**
MAML (ours)	**48.70 ± 1.84%**	**63.11 ± 0.92%**

- The table below compares the accuracy of the algorithms on the MiniImagenet dataset and shows that MAML outperforms the accuracy of other SOTA algorithms, and in particular, the performance of first-order approximation MAML is nearly the same as that obtained with full second derivative MAML.

Experiments

Reinforcement learning