MAML: Model-Agnostic Meta-Learning for Fast Adaptation for Deep Networks

Chelsea Finn  Pieter Abbeel  Sergey Levine
Presented by Zixuan Huang
Motivation

- Today’s machine learning accomplishes numerous challenging tasks
- Specialists
- Learn **one task in one environment from scratch**
  - Take long to master new tasks!
Motivation

• Humans are generalists that learn and adapt quickly
• We’re able to
  • Learn new skills
  • Adapt to new environment
  • Recognize new objects
• In a few shots
Motivation

• With past experiences and prior knowledges, we figure out how to learn more efficiently
• Meta-learning: learning to learn
• How do we equip a ML model with such capability?
How does meta-learning work? An example

Given 1 example of 5 classes:

training data $D_{train}$

Classify new examples

test set $X_{test}$
How does meta-learning work? 
An example

Can replace image classification with regression, skill learning, language generation and etc.
Problem setup | Meta Learning

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably.

**Key assumption**: meta-training tasks and meta-test task drawn i.i.d. from same task distribution

$\mathcal{T}_1, \ldots, \mathcal{T}_n \sim p(\mathcal{T}), \mathcal{T}_j \sim p(\mathcal{T})$

Like before, tasks must share structure.
Comparison to supervised learning

Supervised Learning:
Inputs: $\mathbf{x}$
Outputs: $\mathbf{y}$
Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

$\mathbf{y} = g_{\phi}(\mathbf{x})$

Meta Supervised Learning:
Inputs: $D_{tr}^{\mathbf{x}}, \mathbf{x}_{ts}$
Outputs: $\mathbf{y}_{ts}$
Data: $\{D_i\}$

$\{(\mathbf{x}, \mathbf{y})_{1:K}\}$

$\mathbf{y}_{ts} = f_{\theta}(D_{tr}^{\mathbf{x}}, \mathbf{x}_{ts})$

Why is this view useful?
Reduces the meta-learning problem to the design & optimization of $f$. 
Prior works

• Learning an update function or update rule
  • LSTM optimizer (Learning to learn by gradient descent by gradient descent)
  • Meta LSTM optimizer (Optimization as a model for few-shot learning)

• Few shot (or meta) learning for specific tasks
  • Generative modeling (Neural Statistician)
  • Image classification (Matching Net., Prototypical Net.)
  • Reinforcement learning (Benchmarking deep reinforcement learning for continuous control)

• Memory-augmented model
  • Learning an RNN that ingests experience
MAML | Overview

• Model-agnostic
  • Compatible with any model trained with gradient descent

• General
  • Applicable to a variety of different learning problems, including classification, regression, and reinforcement learning.

• Optimization-based
  • An explicit optimization procedure is embedded
MAML | Intuition

• Some internal representations are more transferrable than others.
• Desired model parameter set is $\theta$ such that:
  • Applying one (or a small # of) gradient step to $\theta$ on a new task will produce optimal behavior
• Find $\theta$ that commonly decreases loss of each task after adaptation.

$\theta$ parameter vector being meta-learned

$\phi_i^*$ optimal parameter vector for task $i$
**MAML | Objective**

**Fine-tuning** [test-time]

\[
\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_{\text{tr}})
\]

**Meta-learning**

\[
\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_{i\text{tr}}), \mathcal{D}_{i\text{ts}})
\]

**Key idea:** Over many tasks, learn parameter vector $\theta$ that transfers via fine-tuning
MAML | Algorithm

1. Sample task $\mathcal{T}_i$ (or mini batch of tasks)
2. Sample disjoint datasets $\mathcal{D}^{\text{tr}}_i, \mathcal{D}^{\text{test}}_i$ from $\mathcal{D}_i$
3. Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}_i)$
4. Update $\theta$ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}^{\text{test}}_i)$
MAML | Second-order gradient

\[ \theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim p(T)} \mathcal{L}_{T_i}(f_{\theta'_i}) \quad \text{(Recall: } \theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)) \]

\[ = \theta - \beta \sum_{T_i \sim p(T)} \nabla_\theta \mathcal{L}_{T_i}(f_{\theta'_i}) \quad \text{( } \mathcal{L} \text{ is differentiable)} \]

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

\[ = \theta - \beta \sum_{T_i \sim p(T)} \left( I - \alpha \nabla^2_\theta \mathcal{L}_{T_i}(f_\theta) \right) \nabla_{\theta'_i} \mathcal{L}_{T_i}(f_{\theta'_i}) \]

Calculation of Hessian matrix is required.
→ MAML suggest 1st order approximation.
MAML | Second-order gradient

\[ \theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} \mathcal{L}_{T_i}(f_{\theta_i'}) \quad \text{(Recall: } \theta_i' = \theta - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta})\text{)} \]

\[ = \theta - \beta \sum_{T_i \sim p(T)} \nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta_i'}) \quad \text{( } \mathcal{L} \text{ is differentiable)} \]

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

\[ = \theta - \beta \sum_{T_i \sim p(T)} \left( I - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta}) \right) \nabla_{\theta_i'} \mathcal{L}_{T_i}(f_{\theta_i'}) \]

In 1st order approximation, we regard this as identity matrix \( I \).
Experiments

• Supervised regression
• Supervised classification
• Reinforcement Learning
Few-shot 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.
Few-shot regression

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

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)
Few-shot regression

[MAML]

[Pretraining + Fine-tuning]

After 1 gradient step update.
Few-shot regression

[MAML] [Pretraining + Fine-tuning]

After 10 gradient step update.
MAML only requires 1 gradient step

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

- 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.
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.
Few-shot classification

MAML outperforms methods that are specially designed for this task

<table>
<thead>
<tr>
<th></th>
<th>5-way Accuracy</th>
<th>20-way Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>Omniglot (Lake et al., 2011)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MANN, no conv (Santoro et al., 2016)</td>
<td>82.8%</td>
<td>94.9%</td>
</tr>
<tr>
<td>MAML, no conv (ours)</td>
<td><strong>89.7 ± 1.1%</strong></td>
<td><strong>97.5 ± 0.6%</strong></td>
</tr>
<tr>
<td>Siamese nets (Koch, 2015)</td>
<td>97.3%</td>
<td>98.4%</td>
</tr>
<tr>
<td>matching nets (Vinyals et al., 2016)</td>
<td>98.1%</td>
<td>98.9%</td>
</tr>
<tr>
<td>neural statistician (Edwards &amp; Storkey, 2017)</td>
<td>98.1%</td>
<td>99.5%</td>
</tr>
<tr>
<td>memory mod. (Kaiser et al., 2017)</td>
<td>98.4%</td>
<td>99.6%</td>
</tr>
<tr>
<td><strong>MAML (ours)</strong></td>
<td><strong>98.7 ± 0.4%</strong></td>
<td><strong>99.9 ± 0.1%</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5-way Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td>MiniImagenet (Ravi &amp; Larochelle, 2017)</td>
<td></td>
</tr>
<tr>
<td>fine-tuning baseline</td>
<td>28.86 ± 0.54%</td>
</tr>
<tr>
<td>nearest neighbor baseline</td>
<td>41.08 ± 0.70%</td>
</tr>
<tr>
<td>matching nets (Vinyals et al., 2016)</td>
<td>43.56 ± 0.84%</td>
</tr>
<tr>
<td>meta-learner LSTM (Ravi &amp; Larochelle, 2017)</td>
<td>43.44 ± 0.77%</td>
</tr>
<tr>
<td><strong>MAML, first order approx. (ours)</strong></td>
<td><strong>48.07 ± 1.75%</strong></td>
</tr>
<tr>
<td><strong>MAML (ours)</strong></td>
<td><strong>48.70 ± 1.84%</strong></td>
</tr>
</tbody>
</table>
Reinforcement learning

- rl lab 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
Reinforcement learning

- 2d navigation

<table>
<thead>
<tr>
<th>num. grad steps</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>context vector</td>
<td>-42.42</td>
<td>-13.90</td>
<td>-5.17</td>
<td>-3.18</td>
</tr>
<tr>
<td>MAML (ours)</td>
<td>-40.41</td>
<td>-11.68</td>
<td>-3.33</td>
<td>-3.23</td>
</tr>
</tbody>
</table>
Reinforcement learning

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

<table>
<thead>
<tr>
<th>num. grad steps</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>context vector</td>
<td>−40.49</td>
<td>−44.08</td>
<td>−38.27</td>
<td>−42.50</td>
</tr>
<tr>
<td>MAML (ours)</td>
<td>−50.69</td>
<td>293.19</td>
<td>313.48</td>
<td>315.65</td>
</tr>
</tbody>
</table>
Conclusion

1. MAML is a meta learning technique that reuses past experiences to achieve fast adaptation on new tasks.

2. It’s simple, model-agnostic, and generally applicable to many tasks such as classification, regression and RL.

3. It can be viewed from:
   1. **Feature learning standpoint**: building an internal representation that is broadly suitable for many tasks
   2. **Dynamical system standpoint**: Maximizing the sensitivity of loss function with respect to the parameters.
Discussion

• **Multi-task Learning vs Meta Learning.**
  • Why don’t we learn a single set of weights that are applicable to many tasks?

• **Assumption of Meta Learning.**
  • Meta Learning assumes the tasks during training and test are drawn from the same distribution. But in reality, it’s inevitable that we encounter tasks that are out-of-distribution. In this case, is MAML still going to work?

• **Continuous setting**
  • MAML assumes access to an offline training dataset
  • What if the training data come in sequentially?
  • How to fight against catastrophic forgetting?