Meta Learning: Learn to learn

Hung-yi Lee

What does “meta” mean? meta-X = X about X
這門課的作業在做甚麼？
Using 1000 GPUs to try 1000 sets of hyperparameters

“Telepathize” (通靈) a set of good hyperparameters

Can machine automatically determine the hyperparameters?
Machine Learning 101
Machine Learning
= Looking for a function

**Step 1:** Function with unknown

**Step 2:** Define loss function

**Step 3:** Optimization

Dog-Cat Classification

\[ f(\theta) = \text{“cat”} \]

Weights and biases of neurons are unknown parameters *learnable*. Using \( \theta \) to represent the unknown parameters.
Machine Learning

**Step 1:** Function with unknown

**Step 2:** Define loss function

**Step 3:** Optimization

$$L(\theta) = \sum_{k=1}^{K} e_k$$

*Training Examples*

*Ground Truth*
**Step 1:** Function with unknown

**Step 2:** Define loss function

**Step 3:** Optimization

loss: \( L(\theta) = \sum_{k=1}^{K} e_k \) sum over training examples

\( \theta^* = \arg\min_{\theta} L(\theta) \) done by gradient descent

\( f_{\theta^*} \) is the function learned by learning algorithm from data
Introduction of Meta Learning
What is Meta Learning?

Can we learn this function?

Following the same three steps in ML!

Training Examples

function

Learning algorithm

Hand-crafted

Input

Testing

Classifier

Learned from data

Output
Meta Learning – Step 1

• What is **learnable** in a learning algorithm?

In meta, we will try to learn some of them.
Meta Learning – Step 1

• What is **learnable** in a learning algorithm?

![Diagram](image)

- **Training Examples**
  - cat
  - dog

- **$F\phi$**
- **$F$**
- **$f^*$**
- **classifier**
- **Component**
  - Net Architecture,
  - Initial Parameters,
  - Learning Rate,
  - ...
  - **$\phi$: learnable components**

- **Categorize meta learning based on what is learnable**
Meta Learning – Step 2

• Define **loss function** for **learning algorithm** $F_\phi$

$L(\phi)$

$L(\phi) \downarrow \uparrow L(\phi)$

**Task 1**

Apple & Orange

**Training Tasks**

**Task 2**

Car & Bike

<table>
<thead>
<tr>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>apple</td>
</tr>
<tr>
<td>orange</td>
<td>orange</td>
</tr>
<tr>
<td>bike</td>
<td>bike</td>
</tr>
<tr>
<td>car</td>
<td>car</td>
</tr>
</tbody>
</table>
Meta Learning – Step 2

Task 1

Training Examples

apple  orange

\( F_\phi \)

\( f_{\theta^1} \)

\( \theta^1 \): parameters of the classifier learned by \( F_\phi \) using the training examples of task 1

How to define \( L(\phi) \)

\( L(\phi) \)
**Meta Learning – Step 2**

**Task 1**

*Training Examples*

- apple
- orange

How can we know a classifier is good or bad?

Evaluate the classifier on testing set

How to define $L(\phi)$?

$L(\phi) \uparrow$
Meta Learning – Step 2

Task 1

Training Examples

apple
orange

$F_\phi$

prediction

Compute difference

$l^1$

Testing Examples

apple
orange

$f_{\theta^1*}$

Ground Truth

Cross-entropy

apple
orange

Cross-entropy

apple
orange
**Meta Learning – Step 2**

**Task 1**

*Training Examples*

- Apple
- Orange

*Testing Examples*

- Apple
- Orange

\[ F_\phi \]

\[ f_{\theta^*} \]

**prediction**

\[ l^1 \]

**Compute difference**

\[ f_{\theta^*} \]

\[ \text{Cross-entropy} \]

\[ \text{Ground Truth} \]
Task 1

Training Examples

apple  orange

Testing Examples

apple  orange

\[ l^1 \]
Compute difference

\[ F_{\phi} \]

\[ f_{\theta^1*} \]
prediction

\[ f_{\theta^1*} \]

Testing Examples

apple  orange

Ground Truth

Cross-entropy

Cross-entropy
Meta Learning – Step 2

Task 1

Training Examples

apple orange

Testing Examples

apple orange

Total loss: \( L(\phi) = l^1 + l^2 \) (sum over all the training tasks)

Task 2

Testing Examples

bike car

Prediction

\( F_\phi \)

Prediction

\( F_\phi \)

\( f_{\theta^1*} \)

\( f_{\theta^2*} \)
**Meta Learning – Step 2**

**Task 1**

*Training Examples*

![Images of apple and orange]

*Testing Examples*

![Images of apple and orange]

\[ l^1 \]

\[ F_\phi \]

\[ f_{\theta^1} \]

\[ l^1 \]

**Total loss:**

\[ L(\phi) = \sum_{n=1}^{N} l^n \]

(\( N \) is the number of the training tasks)

**Task 2**

*Testing Examples*

![Images of bike and car]

\[ F_\phi \]

\[ f_{\theta^2} \]

\[ l^2 \]
Meta Learning – Step 2

Task 1

In typical ML, you compute the loss based on training examples.

In meta, you compute the loss based on testing examples.

Hold on! You use testing examples during training???

Testing Examples

\[ f_{\theta^1*} \]

\[ f_{\theta^1*} \]

apple  orange

Ground Truth

apple  orange

apple  orange

prediction

Compute difference

\[ l^1 \]
Task 1

In typical ML, you compute the loss based on **training examples**. In meta, you compute the loss based on **testing examples** of training tasks.

Compute

\[ l^1 \]

**apple**  **orange**

**prediction**

**difference**
Meta Learning – Step 3

• Loss function for learning algorithm
  \[ L(\phi) = \sum_{n=1}^{N} l^n \]

• Find \( \phi \) that can minimize \( L(\phi) \)
  \[ \phi^* = \arg\min_{\phi} L(\phi) \]

• Using the optimization approach you know
  If you know how to compute \( \partial L(\phi) / \partial \phi \)
  Gradient descent is your friend.
  What if \( L(\phi) \) is not differentiable?
  Reinforcement Learning / Evolutionary Algorithm

Now we have a learned “learning algorithm” \( F_{\phi^*} \)
Framework

Not related to the testing task

only need little labeled training data

What we really care about

Training Tasks

Task 1
- apple
- orange

Task 2
- bike
- car

Learning Algorithm

Train
- cat
- dog

Test
- cat

Test Task

F_{\phi^*} \rightarrow f_{\theta^*} \rightarrow \text{cat}
ML v.s. Meta
Goal

**Machine Learning** \( \approx \) find a function \( f \)

Dog-Cat Classification

\[ f(\text{cat}) = \text{“cat”} \]

**Meta Learning**

\( \approx \) find a function \( F \) that finds a function \( f \)

Learning Algorithm

\( F(\text{cat, dog, cat, dog}) = f \)

Training Examples
Training Data

Meta Learning

Training tasks

*Task 1*
Apple & Orange

*Task 2*
Car & Bike

Support set
Query set

(in the literature of “learning to compare”)
Machine Learning

Within-task Training

Train

Task 1
Task 2

Meta Learning

Train

Hand-crafted

Test

Task 1
Task 2

Train

Test

Across-task Training

Training Tasks

Learning Algorithm
Loss

Machine Learning

\[ L(\theta) = \sum_{k=1}^{K} e_k \]

Sum over training examples in one task

Meta Learning

\[ L(\phi) = \sum_{n=1}^{N} l^n \]

Sum over testing examples in one task

Sum over training tasks
$L(\phi) = \sum_{n=1}^{N} l^n$

If your optimization method needs to compute $L(\phi)$

**Outer Loop** in “Learning to initialize”

**Across-task training** includes within-task training and testing

**Inner Loop** in “Learning to initialize”

Within-task Training

Within-task Testing

To compute the loss
Meta Learning v.s ML

• What you know about ML can usually apply to meta learning
  • Overfitting on training tasks
  • Get more training tasks to improve performance
  • Task augmentation
  • There are also hyperparameters when learning a learning algorithm ......
• Development task 😊
What is learnable in a learning algorithm?
Review: Gradient Descent

Gradient Descent (Function $F$)
Learning to initialize

- Model-Agnostic Meta-Learning (MAML)


- Reptile

https://arxiv.org/abs/1803.02999
How to train your Dragon MAML

Antreas Antoniou, Harrison Edwards, Amos Storkey, How to train your MAML, ICLR, 2019
MAML

Task 1

Task 2

Testing Task

find good init

cat
dog
cat
dog

Pre-training (Self-supervised Learning)

Trained by proxy tasks (fill-in the blanks, etc.)

find good init
cat
dog
MAML

Isn’t it domain adaptation / transfer learning?

Task 1

Task 2

find good init

find good init

Pre-training (more typical ways)

Use data from different tasks to train a model

Also known as multi-task learning (baseline of meta)
MAML v.s. Pre-training

• https://youtu.be/vUwOA3SNb_E

影片中有防不勝防的業配

這就是“meta 業配”
MAML is good because …

- ANIL (Almost No Inner Loop)

Aniruddh Raghu, Maithra Raghu, Samy Bengio, Oriol Vinyals, Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, ICLR, 2020
More about MAML

• More mathematical details behind MAML
  • https://youtu.be/mxqzGwP_Qys

• First order MAML (FOMAML)
  • https://youtu.be/3z997JhL9Oo

• Reptile
  • https://youtu.be/9jJe2AD35P8
Basis form: $\theta^{t+1} \leftarrow \theta^t - \lambda g^t$

Adagrad, RMSprop, NAG, Adam ....

Is the optimizer learnable?

Can be learned by MAML
Optimizer

Marcin Andrychowicz, et al., Learning to learn by gradient descent by gradient descent, NIPS, 2016
Network Architecture Search (NAS)

$$\phi$$

1. **Network Structure**
   - Init: $\theta^0$
2. **Update**
   - Compute Gradient
   - $\theta' = \theta' + \text{gradient}$
3. **Update**
   - Compute Gradient
   - $\theta'' = \theta'' + \text{gradient}$

- **Training Data**
Network Architecture Search (NAS)

\[ \hat{\phi} = \arg \min_{\phi} L(\phi) \quad \nabla_{\phi} L(\phi) = ? \]

- Reinforcement Learning
  - Barret Zoph, et al., Neural Architecture Search with Reinforcement Learning, ICLR 2017
  - Barret Zoph, et al., Learning Transferable Architectures for Scalable Image Recognition, CVPR, 2018
  - Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML, 2018

An agent uses a set of actions to determine the network architecture.

\( \phi \): the agent’s parameters

\( -L(\phi) \): Reward to be maximized
Network Architecture Search (NAS)

Across-task Training

Update $\phi$ to maximize reward $-L(\phi)$

Form a network

Accuracy of the network

Train the network

Within-task Training
Network Architecture Search (NAS)

\[ \hat{\phi} = \arg \min_{\phi} L(\phi) \quad \nabla_{\phi} L(\phi) =? \]

- **Reinforcement Learning**
  - Barret Zoph, et al., Neural Architecture Search with Reinforcement Learning, ICLR 2017
  - Barret Zoph, et al., Learning Transferable Architectures for Scalable Image Recognition, CVPR, 2018
  - Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML, 2018

- **Evolution Algorithm**
  - Esteban Real, et al., Large-Scale Evolution of Image Classifiers, ICML 2017
  - Esteban Real, et al., Regularized Evolution for Image Classifier Architecture Search, AAAI, 2019
  - Hanxiao Liu, et al., Hierarchical Representations for Efficient Architecture Search, ICLR, 2018
**Network Architecture Search (NAS)**

\[
\hat{\phi} = \arg \min_{\phi} L(\phi) \quad \nabla_{\phi} L(\phi) = ?
\]

- **DARTS**  
  Hanxiao Liu, et al., DARTS: Differentiable Architecture Search, ICLR, 2019
Data Processing?

Network Structure

Gradient Descent (Function $F$)

Gradient Descent

Compute Gradient

Update

Compute Gradient

Update

Training Data

Training Data

$\theta^*$
Data Augmentation

Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M. Robertson, Yongxin Yang, DADA: Differentiable Automatic Data Augmentation, ECCV, 2020

Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, Xi Chen, Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules, ICML, 2019

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le, AutoAugment: Learning Augmentation Policies from Data, CVPR, 2019
Sample Reweighting

• Give different samples different weights

Larger weights (focus on tough examples)?
Smaller weights (the labels are noisy)?

Sample Weighting Strategies → Learnable $\phi$

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, Deyu Meng,
Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting, NeurIPS, 2019
Mengye Ren, Wenyuan Zeng, Bin Yang, Raquel Urtasun, Learning to Reweight Examples for Robust Deep Learning, ICML, 2018
Beyond Gradient Descent

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, Raia Hadsell,
Meta-Learning with Latent Embedding Optimization, ICLR, 2019

This is a Network.
Its parameter is $\phi$

(Invent new learning algorithm! Not gradient descent anymore)
Until now ......

Learning Algorithm (Function $F$)

$\theta^*$

Training Data

Testing Data

Learning + Classification (Function $F$)

Training Data

Testing Data

Learning to compare
(metric-based approach)

https://youtu.be/yyKaACh_j3M
https://youtu.be/scK2EIT7klw
https://youtu.be/semSxPP2Yzg
https://youtu.be/ePimv_k-H24
Applications
Few-shot Image Classification

• Each class only has a few images.

• N-ways K-shot classification: In each task, there are N classes, each has K examples.

• In meta learning, you need to prepare many N-ways K-shot tasks as training and testing tasks.
Omniglot

https://github.com/brendenlake/omniglot

• 1623 characters
• Each has 20 examples
Omniglot

- **20 ways**
- **1 shot**

Each character represents a class

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
</tr>
</tbody>
</table>

Training set (Support set)

<table>
<thead>
<tr>
<th>Testing set (Query set)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
</tr>
</tbody>
</table>

- Split your characters into training and testing characters
  - Sample N training characters, sample K examples from each sampled characters → one training task
  - Sample N testing characters, sample K examples from each sampled characters → one testing task

Demo: https://openai.com/blog/reptile/
<table>
<thead>
<tr>
<th></th>
<th>(A) Learning to initialize</th>
<th>(B) Learning to compare</th>
<th>(C) Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound Event Detection</td>
<td>(Shi et al., 2020)</td>
<td>(Chou et al., 2019)</td>
<td>Network architecture search: (Li et al., 2020)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Wang et al., 2020)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Shimada et al., 2020b)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Shi et al., 2020)</td>
<td></td>
</tr>
<tr>
<td>Keyword Spotting</td>
<td>(Chen et al., 2020a)</td>
<td>(Huh et al., 2020)</td>
<td>Net2Net: (Veniat et al., 2019)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Network architecture search: (Mazzaoui et al., 2019)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Network architecture search: (Mo et al., 2020)</td>
</tr>
<tr>
<td>Text Classification</td>
<td>(Dou et al., 2019)</td>
<td>(Yu et al., 2018)</td>
<td>Learning the learning algorithm: (Wu et al., 2019)</td>
</tr>
<tr>
<td></td>
<td>(Bansal et al., 2019)</td>
<td>(Tan et al., 2019)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Geng et al., 2019)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Sun et al., 2019)</td>
<td></td>
</tr>
<tr>
<td>Voice Cloning</td>
<td></td>
<td></td>
<td>Learning the learning algorithm: (Chen et al., 2019b)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Serrà et al., 2019)</td>
</tr>
<tr>
<td>Sequence Labeling</td>
<td>(Wu et al., 2020)</td>
<td>(Hou et al., 2020)</td>
<td></td>
</tr>
<tr>
<td>Machine Translation</td>
<td>(Gu et al., 2018)</td>
<td></td>
<td>Learning to optimize: (Klejch et al., 2018)</td>
</tr>
<tr>
<td></td>
<td>(Indurthi et al., 2020)</td>
<td></td>
<td>Network architecture search: (Chen et al., 2020b)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Bamuwa et al., 2019)</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td>(Hsu et al., 2020)</td>
<td>(Ye and Ling, 2019)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Klejch et al., 2019)</td>
<td>(Chen et al., 2019a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Winata et al., 2020a)</td>
<td>(Xiong et al., 2018)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Winata et al., 2020b)</td>
<td>(Gao et al., 2019)</td>
<td></td>
</tr>
<tr>
<td>Knowledge Graph</td>
<td>(Obamuyide and Vlachos, 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Bose et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Lv et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dialogue / Chatbot</td>
<td>(Qian and Yu, 2019)</td>
<td></td>
<td>Learning to optimize: (Chien and Lieow, 2019)</td>
</tr>
<tr>
<td></td>
<td>(Madotto et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Mi et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parsing</td>
<td>(Guo et al., 2019)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Huang et al., 2018)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word Embedding</td>
<td>(Hu et al., 2019)</td>
<td>(Sun et al., 2018)</td>
<td></td>
</tr>
<tr>
<td>Multi-model</td>
<td>(Eloff et al., 2019)</td>
<td></td>
<td>Learning the learning algorithm: (Surfs et al., 2019)</td>
</tr>
</tbody>
</table>