Properties of Good Meta-Learning Algorithms
(And How to Achieve Them)

Chelsea Finn
UC Berkeley & Google Brain -> Stanford
Why Learn to Learn?

- effectively **reuse data** on other tasks
- **replace manual engineering** of architecture, hyperparameters, etc.
- learn to **quickly adapt to unexpected scenarios** (inevitable failures, long tail)
- learn how to learn **with weak supervision**
Problem Domains:
- few-shot classification & generation
- hyperparameter optimization
- architecture search
- faster reinforcement learning
- domain generalization
- learning structure
- …

Approaches:
- recurrent networks
- learning optimizers or update rules
- learning initial parameters & architecture
- acquiring metric spaces
- Bayesian models
- …

What is the meta-learning problem statement?
The Meta-Learning Problem

Supervised Learning:

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

$\mathbf{y} = f(\mathbf{x}; \theta)$

Meta-Supervised Learning:

Inputs: $\mathcal{D}_{\text{train}}$, $\mathbf{x}_{\text{test}}$  
Outputs: $\mathbf{y}_{\text{test}}$  
Data: $\{\mathcal{D}_i\}$

$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$

Why is this view useful?
Reduces the problem to the design & optimization of $f$.

Finn, Levine. Meta-learning and Universality: Deep Representation… ICLR 2018
Example: Few-Shot Classification

Given 1 example of 5 classes:

- training data $D_{\text{train}}$

Classify new examples:

- test set $X_{\text{test}}$

meta-training

$T_1$

$T_2$

:::

:::

Diagram adapted from Ravi & Larochelle ’17
What do we want from our meta-learning algorithms?

- **Expressive power**: the ability for $f$ to represent a range of learning procedures.
- **Consistency**: learned learning procedure will solve task with enough data. Result: reasonable performance on out-of-distribution tasks.
- **No need to differentiate through learning**: better scalability to long learning processes.
- **Uncertainty awareness**: ability to reason about ambiguity during learning.
Design of $f$?

Recurrent network (LSTM, NTM, Conv)

$$y_{test} = f(D_{train}, x_{test}; \theta)$$

Santoro et al. '16, Duan et al. '17, Wang et al. '17, Munkhdalai & Yu '17, Mishra et al. '17, …
Design of $f$?

- **Recurrent network** (LSTM, NTM, Conv)
  \[ y_{\text{test}} = f(D_{\text{train}}, x_{\text{test}}; \theta) \]

- **Learned optimizer** (often uses recurrence)
  \[ y_{\text{test}} = f(x_{\text{test}}; g(D_{\text{train}}; \theta)) \]
Design of $f$?

Recurrent network
(LSTM, NTM, Conv)

Learned optimizer
(often uses recurrence)

$y_{test} = f(D_{train}, x_{test} ; \theta)$


$y_{test} = f(x_{test} ; g(D_{train} ; \theta))$


Expressive power Consistency

Black box approaches ✓ ×

Can we incorporate structure into the learning procedure?
Approaches that incorporate learning structure

**Nearest Neighbors** (in a learned metric space)

Koch ‘15, Vinyals et al. ’16, Snell et al. ’17, Reed et al. ’17, Li et al. ’17, …

---

**Koch ’15**

**Consistency**

Vinyals et al. ’16

**Expressive power**

Can we get both?
Key idea: Train over many tasks, to learn parameter vector $\theta$ that transfers via fine-tuning

Approaches that incorporate learning structure

**Nearest Neighbors**
(in a learned metric space)
Koch ‘15, Vinyals et al. ’16, Snell et al. ’17,
Reed et al. ’17, Li et al. ’17, …

**Gradient Descent**
(from learned initialization)
Finn et al. ’17, Grant et al. ‘17,
Reed et al. ’17, Li et al. ’17, …

Learning procedure: \[ y_{test} = f(x_{test}; \theta - \alpha \nabla_{\theta} \mathcal{L}(D_{train}) ) \]

Consistency \[ \checkmark \] Expressive power \[ ? \]

Does consistency come at a cost?
How can we define a notion of expressive power for meta-learning?

Universal Function Approximation Theorem
A neural network with one hidden layer of finite width can approximate any continuous function.

Hornik et al. ’89, Cybenko ’89, Funahashi ’89

“universal function approximator”

\[ y = f(x; \theta) \]

“universal learning procedure approximator”

\[ y_{\text{test}} = f(D_{\text{train}}, x_{\text{test}}; \theta) \]

Recurrent network
\[ y_{\text{test}} = f(D_{\text{train}}, x_{\text{test}}; \theta) \]

Learned optimizer
\[ y_{\text{test}} = f(x_{\text{test}}; g(D_{\text{train}}; \theta)) \]

Recurrent network
\[ y_{test} = f(D_{train}, x_{test}; \theta) \]

MAML
\[ y_{test} = f(x_{test}; \theta - \alpha \nabla_{\theta} \mathcal{L}(D_{train})) \]

For a sufficiently deep \( f \),
MAML function can approximate any function of \( D_{train}, x_{test} \)

Assumptions:
- nonzero \( \alpha \)
- loss function gradient does not lose information about the label
- datapoints in \( D_{train} \) are unique

Why is this interesting?
MAML has benefit of consistency without losing expressive power.

Finn, Levine. *Meta-learning and Universality: Deep Representation*… ICLR 2018
<table>
<thead>
<tr>
<th></th>
<th>Expressive power</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Black box approaches</strong></td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td><strong>MAML</strong></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Empirically, what does consistency get you?
How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

MAML, TCML, MetaNetworks

Finn, Levine. *Meta-learning and Universality: Deep Representation…* ICLR 2018
How well can methods generalize to similar, but extrapolated tasks?

The world is non-stationary.

**Sinusoid curve regression**

*Takeaway:* Strategies learned with MAML consistently generalize better to out-of-distribution tasks

Finn, Levine. *Meta-learning and Universality: Deep Representation…* ICLR 2018
What do we want from our meta-learning algorithms?

- **Expressive power**: the ability for $f$ to represent a range of learning procedures
  - ✔️

- **Consistency**: learned learning procedure will solve task with enough data
  - result: reasonable performance on out-of-distribution tasks
  - ✔️

- **No need to differentiate through learning**: better scalability to long learning processes

- **Uncertainty awareness**: ability to reason about ambiguity during learning
What do we want from our meta-learning algorithms?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressive power</td>
<td>the ability for $f$ to represent a range of learning procedures ✔️</td>
</tr>
</tbody>
</table>
| Consistency                         | learned learning procedure will solve task with enough data
  result: reasonable performance on **out-of-distribution tasks** ✔️ |
| No need to differentiate through learning | better scalability to long learning processes                             |
| Uncertainty awareness               | ability to reason about ambiguity during learning                          |
Model-Agnostic Meta-Learning: (MAML)

\[
\min_{\theta} \sum_{\text{task } i} \mathcal{L}^i_{\text{test}}(\theta - \alpha \nabla_{\theta} \mathcal{L}^i_{\text{train}}(\theta))
\]

What if we stop the gradient through this term?

<table>
<thead>
<tr>
<th>MiniImagenet (Ravi &amp; Larochelle, 2017)</th>
<th>5-way Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>fine-tuning baseline</td>
<td>1-shot</td>
</tr>
<tr>
<td></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></td>
</tr>
<tr>
<td></td>
<td>43.56 ± 0.84%</td>
</tr>
<tr>
<td>meta-learner LSTM (Ravi &amp; Larochelle, 2017)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>43.44 ± 0.77%</td>
</tr>
<tr>
<td>MAML, first order approx. (ours)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48.07 ± 1.75%</td>
</tr>
<tr>
<td>MAML (ours)</td>
<td>48.70 ± 1.84%</td>
</tr>
</tbody>
</table>

Note: In some tasks, we have observed a more substantial drop.

What do we want from our meta-learning algorithms?

**Expressive power**
the ability for \( f \) to represent a range of learning procedures

**Consistency**
learned learning procedure will solve task with enough data
result: reasonable performance on out-of-distribution tasks

**No need to differentiate through learning**
better scalability to long learning processes

**Uncertainty awareness**
ability to reason about ambiguity during learning
<table>
<thead>
<tr>
<th>What do we want from our meta-learning algorithms?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expressive power</strong></td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>No need to differentiate through learning</strong></td>
</tr>
<tr>
<td><strong>Uncertainty awareness</strong></td>
</tr>
</tbody>
</table>

**No uncertainty awareness in original MAML**
that’s nice but is it… **Bayesian?**
kind of… but it can be more Bayesian

A useful property:

start from $\phi = \theta$ and follow gradient of $\log p(Y|X, \phi)$ for $K$ steps

this is equivalent to MAP on $p(\phi|X, Y)$

for a prior $p(\phi|\theta) = \mathcal{N}(\theta, \Sigma)$

and for a **linear** model $E[Y] = X^T\phi$

(Santos, 1996)
A Probabilistic Interpretation of MAML

start from $\phi = \theta$ and follow gradient of $\log p(Y|X, \phi)$ for $K$ steps
this is equivalent to MAP on $p(\phi|X, Y)$

for a prior $p(\phi|\theta) = \mathcal{N}(\theta, \Sigma)$
and for a linear model $E[Y] = X^T\phi$

MAML adaptation:
$\phi_i = \theta - \alpha\nabla_{\theta} \mathcal{L}(\theta, D_{\text{train}})$

MAP inference in this model
$p(\phi_i|\theta) = \mathcal{N}(\theta, \Sigma)$

estimate Hessian for neural nets with KFAC

can we do better than MAP?
can use Laplace estimate:

$$-\log p(X|\theta) \approx \sum_i \left[ -\log p(X_j|\hat{\phi}_j) - \log p(\hat{\phi}_j|\theta) + \frac{1}{2} \log \det(H_j) \right]$$

Grant, Finn, Levine, Darrell, Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. ICLR ‘18
Modeling ambiguity

Can we *sample* classifiers?

**Intuition:** we want to learn a prior where a random kick can put us in different modes

\[ \mathcal{L}(\phi, D_{\text{train}}) \]

\[ \phi \leftarrow \theta + \epsilon \]

\[ \phi \leftarrow \phi + \alpha \nabla_{\phi} \mathcal{L}(\phi, D_{\text{train}}) \]
Meta-learning with ambiguity

\[ \theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta) \]
\[ \phi_i \sim p(\phi_i | \theta) \]

Goal: sample \( \phi_i \sim p(\phi_i | x^{\text{train}}_i, y^{\text{train}}_i, x^{\text{test}}_i) \)

Finn*, Xu*, Levine. *Probabilistic Model-Agnostic Meta-Learning*
Sampling parameter vectors

\[ \theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta) \]
\[ \phi_i \sim p(\phi_i | \theta) \]

\[ \log p(y^\text{train}_i | x^\text{train}_i, \phi_i) \]
\[ \log p(y^\text{test}_i | x^\text{test}_i, \phi_i) \]

Goal: sample \( \phi_i \sim p(\phi_i | x^\text{train}_i, y^\text{train}_i) \)

\[ p(\phi_i | x^\text{train}_i, y^\text{train}_i) \propto \int p(\theta) p(\phi_i | \theta) p(y^\text{train}_i | x^\text{train}_i, \phi_i) d\theta \]

\[ \Rightarrow \text{this is completely intractable!} \]

what if we knew \( p(\phi_i | \theta, x^\text{train}_i, y^\text{train}_i) \)?

\[ \Rightarrow \text{now sampling is easy! just use ancestral sampling!} \]

**key idea:** \( p(\phi_i | \theta, x^\text{train}_i, y^\text{train}_i) \approx \delta(\hat{\phi}_i) \)

this is extremely crude

but extremely convenient!

\[ \hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y^\text{train}_i | x^\text{train}_i, \theta) \]

Finn*, Xu*, Levine. *Probabilistic Model-Agnostic Meta-Learning*
Sampling parameter vectors

\[ \theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta) \]

**key idea:**
\[ p(\phi_i | \theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \delta(\hat{\phi}_i) \quad \hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}} | x_i^{\text{train}}, \theta) \]

**What does ancestral sampling look like?**

1. \[ \theta \sim \mathcal{N}(\mu_\theta, \Sigma_\theta) \]
2. \[ \phi_i \sim p(\phi_i | \theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \hat{\phi}_i = \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}} | x_i^{\text{train}}, \theta) \]

Finn*, Xu*, Levine. *Probabilistic Model-Agnostic Meta-Learning*
PLATIPUS
Probabilistic LATent model for Incorporating Priors and Uncertainty in few-Shot learning

Ambiguous regression:

Ambiguous classification:

PLATIPUS
Probabilistic LAstent model for Incorporating Priors and Uncertainty in few-Shot learning

<table>
<thead>
<tr>
<th>Ambiguous celebA (5-shot)</th>
<th>Accuracy</th>
<th>Coverage (max=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAML</td>
<td>69.26 ± 2.18%</td>
<td>1.00 ± 0.0</td>
</tr>
<tr>
<td>MAML + noise</td>
<td>54.73 ± 0.8%</td>
<td>2.60 ± 0.12</td>
</tr>
<tr>
<td>PLATIPUS (ours)</td>
<td>69.97 ± 1.32%</td>
<td>2.62 ± 0.11</td>
</tr>
</tbody>
</table>

Finn*, Xu*, Levine. *Probabilistic Model-Agnostic Meta-Learning*
Related concurrent work

Kim et al. “Bayesian Model-Agnostic Meta-Learning”: uses Stein variational gradient descent for sampling parameters

Gordon et al. “Decision-Theoretic Meta-Learning”: unifies a number of meta-learning algorithms under a variational inference framework
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressive power</td>
<td>the ability for $f$ to represent a range of learning procedures</td>
</tr>
</tbody>
</table>
| Consistency                   | learned learning procedure will solve task with enough data  

  *result*: reasonable performance on out-of-distribution tasks |
| No need to differentiate through learning | better scalability to long learning processes |
| Uncertainty awareness         | ability to reason about ambiguity during learning |


We can build meta-learning algorithms with many nice properties (expressive power, consistency, first-order, uncertainty awareness)
Collaborators

Sergey Levine  Tianhe Yu  Erin Grant  Trevor Darrell  Sudeep Dasari  Josh Abbott

Pieter Abbeel  Kelvin Xu  Tom Griffiths  Annie Xie  Tianhao Zhang  Josh Peterson

Blog post, code, and papers: eecs.berkeley.edu/~cbfinn

Questions?