A Survey on Meta-Learning

Xiang Li

Nanyang Technological University

September 10, 2019
What’s Not Included in This Survey?

- Meta-Learning for Reinforcement Learning
- Bayesian-Based Meta-Learning
Overview

Problem Definition
  Few-shot Learning

Approaches
  Non-Parametric Methods (Metric Learning)
  Model-Based Methods (Black-Box Adaptation)
  Optimization-Based Methods
  Other Methods

Summary
Problem Definition

- Over a task distribution $p(T)$:

$$
\theta^* \leftarrow \arg \min_{\theta} \sum_{t \sim p(T)} L_t(\theta)
$$

- Common tasks:
  - Few-shot Learning
  - Regression
  - Reinforcement Learning
Few-shot Learning: Definition

- Training set: $D_{\text{meta-train}} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$
- Test set: $D_{\text{meta-test}} = \{D_1, \ldots, D_n\}$, $D_i = \{(x^i_1, y^i_1), \ldots, (x^i_m, y^i_m)\}$
- N-way, K-shot problem (usually 5-way, 5-shot or 5-way, 1-shot)

Ravi et al. '17
Few-shot Learning: Dataset

- Omniglot dataset (Lake et al. ’15)
  - 1623 characters, 20 instances of each character
  - "transpose" of MNIST

- Mini-ImageNet
  - subset of ImageNet

- Other datasets: CIFAR, CUB, CelebA and others
Problem Definition

Few-shot Learning

Approaches

Non-Parametric Methods (Metric Learning)
Model-Based Methods (Black-Box Adaptation)
Optimization-Based Methods
Other Methods

Summary
Non-Parametric Methods (Metric Learning)

▶ Idea:
Simply compare query images to images in support set in feature space

▶ Challenge:
  ▶ Feature space (compare in what space?)
  ▶ Distance metric
Siamese Network (Koch et al. ’15)
Can we make the training condition match the test condition?
Can the feature space be conditioned on specific task?
Matching Networks for One Shot Learning (Vinyals et al. ’16)

- Sampling Strategy: test and training conditions must match
  - Sample "episodes" in a training batch
- Attention Kernel
  - \( \hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \)
  - \( a(\hat{x}, x_i) = \frac{e^c(f(\hat{x}), g(x_i))}{\sum_{j=1}^{k} e^c(f(\hat{x}), g(x_j))} \)
- Full Context Embeddings

\( g_\theta \) is a bidirectional LSTM to encode \( x_i \) in context of \( S \)
\( f_\theta \) is a LSTM to encode \( x \) in context of \( S \)
Other Metric-Based Works

▶ Prototypical Networks for Few-shot Learning (Snell et al. ’17)

\[
\mathbf{c}_k = \frac{1}{|S_k|} \sum (\mathbf{x}_i, y_i) \in S_k \ f_\phi (\mathbf{x}_i)
\]

\[
p_\phi (y = k | x) = \frac{\exp(-d(f_\phi (x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi (x), \mathbf{c}_{k'}))}
\]

▶ Learning to Compare: Relation Network for Few-Shot Learning (Sung et al. ’18)
## Performance

### Omniglot

<table>
<thead>
<tr>
<th>Model</th>
<th>Fine Tune</th>
<th>5-way Acc.</th>
<th>20-way Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>MANN [32]</td>
<td>N</td>
<td>82.8%</td>
<td>94.9%</td>
</tr>
<tr>
<td>CONVOLUTIONAL SIAMESE NETS [20]</td>
<td>N</td>
<td>96.7%</td>
<td>98.4%</td>
</tr>
<tr>
<td>CONVOLUTIONAL SIAMESE NETS [20]</td>
<td>Y</td>
<td>97.3%</td>
<td>98.4%</td>
</tr>
<tr>
<td>MATCHING NETS [39]</td>
<td>N</td>
<td>98.1%</td>
<td>98.9%</td>
</tr>
<tr>
<td>MATCHING NETS [39]</td>
<td>Y</td>
<td>97.9%</td>
<td>98.7%</td>
</tr>
<tr>
<td>SIAMESE NETS WITH MEMORY [18]</td>
<td>N</td>
<td>98.4%</td>
<td>99.6%</td>
</tr>
<tr>
<td>NEURAL STATISTICIAN [8]</td>
<td>N</td>
<td>98.1%</td>
<td>99.5%</td>
</tr>
<tr>
<td>META NETS [27]</td>
<td>N</td>
<td>99.0%</td>
<td>-</td>
</tr>
<tr>
<td>PROTOTYPICAL NETS [36]</td>
<td>N</td>
<td>98.8%</td>
<td>99.7%</td>
</tr>
<tr>
<td>MAML [10]</td>
<td>Y</td>
<td>98.7 ± 0.4%</td>
<td>99.9 ± 0.1%</td>
</tr>
<tr>
<td>RELATION NET</td>
<td>N</td>
<td><strong>99.6 ± 0.2%</strong></td>
<td><strong>99.8 ± 0.1%</strong></td>
</tr>
</tbody>
</table>

### Mini-ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>FT</th>
<th>5-way Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td>MATCHING NETS [39]</td>
<td>N</td>
<td>43.56 ± 0.84%</td>
</tr>
<tr>
<td>META NETS [27]</td>
<td>N</td>
<td>49.21 ± 0.96%</td>
</tr>
<tr>
<td>META-LEARN LSTM [29]</td>
<td>N</td>
<td>43.44 ± 0.77%</td>
</tr>
<tr>
<td>MAML [10]</td>
<td>Y</td>
<td>48.70 ± 1.84%</td>
</tr>
<tr>
<td>PROTOTYPICAL NETS [36]</td>
<td>N</td>
<td>49.42 ± 0.78%</td>
</tr>
<tr>
<td>RELATION NET</td>
<td>N</td>
<td><strong>50.44 ± 0.82%</strong></td>
</tr>
</tbody>
</table>
Metric-Based Methods: Summary

- **Meta Knowledge:** feature space & distance metric
- **Task Knowledge:** None

- **Advantage:**
  - Simple and computational fast
  - Entirely feed-forward

- **Disadvantage:**
  - Hard to scale very large K
  - Limited to classification problem
Problem on Metric-Based Methods

Adaptation for each specific task?
- Model-Based Methods
- Optimization-Based Methods
Problem Definition
Few-shot Learning

Approaches
Non-Parametric Methods (Metric Learning)
**Model-Based Methods (Black-Box Adaptation)**
Optimization-Based Methods
Other Methods

Summary
Model-Based Methods (Black-Box Adaptation)

► Idea:
  ▶ Adaptation $\phi_i$ for task $i$
  ▶ Train a network $(\theta)$ to represent $p(\phi_i|D_i^{\text{support}}, \theta)$

\[
\begin{align*}
  f_\theta & \quad (x_1, y_1) & (x_2, y_2) & (x_3, y_3) & \quad \phi_i \\
  \quad D_i^{\text{tr}} & \quad G_{\phi_i} \\
  \quad D_i^{\text{test}} \\
  y^{\text{ts}} & \quad x^{\text{ts}}
\end{align*}
\]
How to aggregate samples in support set (form of $f_{\theta}$)?

- Average (Prototypical Network)
- LSTM (Matching Network)
- Memory-Augmented Neural Networks
Neural Turing Machines (NTMs) (Graves et al. ’14)

- Attention-based
- Memory matrix $M$
- Read: \[
\begin{align*}
\mathbf{r} & \leftarrow \sum_i w(i) \mathbf{M}(i) \\
\sum_i w(i) & = 1
\end{align*}
\]
- Write: \[
\begin{align*}
\tilde{\mathbf{M}}_t(i) & \leftarrow \mathbf{M}_{t-1}(i) \left[ \mathbf{1} - w(i) \mathbf{e} \right] \\
\mathbf{M}_t(i) & \leftarrow \tilde{\mathbf{M}}_t(i) + w(i) \mathbf{a}
\end{align*}
\]
Neural Turing Machines (NTMs) (Graves et al. ’14)

Advantages:
- Large memory
- Addressable
Target: \( p \left( \phi_i \mid D_i^{support}, \theta \right) \)

What formations can \( \phi_i \) be?
- Contain task-specific information
- Contain only useful information

Solution 1: feature representations of samples in \( D^{support} \)
Meta-Learning with Memory Augmented Neural Networks (Santoro et al. ’16)

- Read Memory:

\[ r_i = \sum_{i=1}^{N} w^r_t(i) M_t(i), \]

where

\[ w^r_t(i) = \text{softmax} \left( \frac{k_t \cdot M_t(i)}{\|k_t\| \cdot \|M_t(i)\|} \right) \]

- Least Recently Used Access (LRUA)

- Strategy: 1 time off-set
**Dynamic Few-Shot Visual Learning without Forgetting (Gidaris et al. ’18)**

Use cosine similarity in the classification layer instead of dot-product

Generate prototypical vector of each class using:

- Average of features in support set (like prototypical network)
- Attention-based inference from memory of base classes features
  - Base class features are like a dictionary
  - A memory matrix $M$
  - A key matrix $K$

$$w'_{att} = \frac{1}{N'} \sum_{i=1}^{N'} \sum_{b=1}^{K_{base}} Att(\phi_q \bar{z}'_i, k_b) \cdot \bar{w}_b$$
Target: \( p \left( \phi_i | \mathcal{D}_i^{\text{support}}, \theta \right) \)

What formations can \( \phi_i \) be?
- Contain task-specific information
- Contain only useful information

Solution 2: network weights
Meta Networks (Munkhdalai et al. ’17)

- Slow weight + Fast weight
- meta feature extractor $f_m$ (key embedding) with slow weight $M$, base feature extractor $f_b$ with slow weight $B$, meta weight generator $g_m$, base weight generator $g_b$
- Meta-information: gradients
Meta Networks (Munkhdalai et al. ’17)

- Generate fast weight for meta feature extractor:
  - Sample T examples \((x'_i, y'_i)\) in support set
  - For each i: \(\nabla_i \leftarrow \nabla_M \cdot L_{embed}(f_m(M, x'_i), y'_i)\)
  - \(M^* \leftarrow g_m(\{\nabla_i\})\) (then meta feature extractor \(f_m\) can use slow weight \(M + \) fast weight \(M^*\))

- Store support set base fast weights in NTM:
  - For all samples \((x'_i, y'_i)\) in support set:
    - \(\nabla_i \leftarrow \nabla_B \cdot L_{task}(f_b(B, x'_i), y'_i)\)
    - \(B_i^* \leftarrow g_b(\nabla_i)\) (slow weight for base extractor)
    - Store \(B_i^*\) in memory \(U(i)\)
    - \(r'_i \leftarrow f_m(M, M^*, x'_i)\) and store in key matrix \(K(i)\)

- Use fast weight from memory to extract features for query samples:
  - For all samples \((x_i, y_i)\) in query set:
    - \(r_i \leftarrow f_m(M, M^*, x_i)\) (key)
    - Access memory using \(r_i\) and get \(W_i^*\) (fast weight for \(f_b\))
    - Extract feature \(f_b(W, W_i^*, x_i)\) and compute total task loss

- Use total loss to update slow weights and weights for weight generators
Meta Networks (Munkhdalai et al. ’17)

▶ Remaining question: can we find a better meta-information? (in this work: gradients)
Model-Based Methods: Summary

- Meta Knowledge: Slow weight
- Task Knowledge: Generated weights & support set features

- Advantage:
  - Applicable to many learning problems

- Disadvantage:
  - Complicated model (model & architecture intertwined)
  - Difficult to optimize
Target: $p \left( \phi_i | \mathcal{D}_i^{\text{support}} , \theta \right)$

Since generating weights ($\phi_i$) is very computational cost, why not updating existing weights $\theta$?

fine-tune!
Problem Definition
Few-shot Learning

Approaches
Non-Parametric Methods (Metric Learning)
Model-Based Methods (Black-Box Adaptation)
Optimization-Based Methods
Other Methods

Summary
Optimization-Based Methods

- Idea:
  - Fine-tune to a specific task
  - At test time, given a task \( t \), \( \theta^t \leftarrow g(\theta, D^{support}) \). Then we have \( \hat{y} = f(\theta^t, \hat{x}) \)

- What can we do in fine-tune?
Take a look into Fine-Tune Process (Gradient Descent)

$$\theta' = \theta - \alpha \nabla_\theta L$$

- Compute a more effective “gradients”?
- Have a better initial parameters?
- Update only partial weights?
Optimization as a Model for Few-Shot Learning (Ravi et al. ’17)

▶ An observation:
  ▶ Gradient descent:
  \[ \theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t \]

▶ Cell update in LSTM:
  \[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \]

▶ If \( f_t = 1 \), \( i_t = a_t \) and \( \tilde{c}_t = \nabla_{\theta_{t-1}} \mathcal{L}_t \):
  \( c_t \) can be treated as \( \theta_t \)

▶ Dynamic \( f_t \) and \( \tilde{c}_t \):
  \[ i_t = \sigma \left( W_I \cdot \left[ \nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1} \right] + b_I \right) \]
  \[ f_t = \sigma \left( W_F \cdot \left[ \nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1} \right] + b_F \right) \]

▶ Weight Sharing
Optimization as a Model for Few-Shot Learning (Ravi et al. '17)
Utilize Loss ($\mathcal{L}_t$), Gradient ($\nabla_{\theta_{t-1}} \mathcal{L}_t$), network weight ($\theta_i$) and previous state ($i_{t-1} \& f_{t-1}$) as meta-information.
Take a look into Fine-Tune Process (Gradient Descent)

\[ \theta' = \theta - \alpha \nabla_{\theta} L \]

- Compute a more effective “gradients”?
- Have a better initial parameters?
- Update only partial weights?
Learn **initial parameters** that is easy to adapt to all tasks in the task distribution

- Initial parameters: prior knowledge (meta knowledge)
- Easy: one step or few steps
Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (Finn et al. '17)

Algorithm 1 Model-Agnostic Meta-Learning

Require: \( p(\mathcal{T}) \): distribution over tasks
Require: \( \alpha, \beta \): step size hyperparameters

1: randomly initialize \( \theta \)
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
Does optimal point in parameter space exist?
Take a look into Fine-Tune Process (Gradient Descent)

\[
\theta' = \theta - \alpha \nabla_{\theta} L
\]

- Compute a more effective “gradients”?
- Have a better initial parameters?
- Update only partial weights?
Fast Context Adaptation via Meta-Learning (Zintgraf et al. ’19)

- Meta parameters $\phi$ and task-specific parameters $\theta$

- Divide parameters to meta and task-specific, similar to LSTM meta-learner (Ravi et al. ’17) and Meta Networks (Munkhdalai et al. ’17)

---

**Algorithm 1** Model-Agnostic Meta-Learning

**Require:** $p(\mathcal{T})$: distribution over tasks

**Require:** $\alpha, \beta$: step size hyperparameters

1. randomly initialize $\theta$
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
## Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>1-shot</th>
<th>5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching Nets (Vinyals et al., 2016)</td>
<td>46.6%</td>
<td>60.0%</td>
</tr>
<tr>
<td>Meta LSTM (Ravi &amp; Larochelle, 2017)</td>
<td>43.44 ± 0.77%</td>
<td>60.60 ± 0.71%</td>
</tr>
<tr>
<td>Prototypical Networks (Snell et al., 2017)</td>
<td>46.61 ± 0.78%</td>
<td>65.77 ± 0.70%</td>
</tr>
<tr>
<td>Meta-SGD (Li et al., 2017)</td>
<td>50.47 ± 1.87%</td>
<td>64.03 ± 0.94%</td>
</tr>
<tr>
<td>REPTILE (Nichol &amp; Schulman, 2018)</td>
<td>49.97 ± 0.32%</td>
<td>65.99 ± 0.58%</td>
</tr>
<tr>
<td>MT-NET (Lee &amp; Choi, 2018)</td>
<td><strong>51.70 ± 1.84%</strong></td>
<td>-</td>
</tr>
<tr>
<td>VERSA (Gordon et al., 2018)</td>
<td><strong>53.40 ± 1.82%</strong></td>
<td><strong>67.37 ± 0.86</strong></td>
</tr>
<tr>
<td>MAML (32) (Finn et al., 2017a)</td>
<td>48.07 ± 1.75%</td>
<td>63.15 ± 0.91%</td>
</tr>
<tr>
<td>MAML (64)</td>
<td>44.70 ± 1.69%</td>
<td>61.87 ± 0.93%</td>
</tr>
<tr>
<td>CAVIA (32)</td>
<td>47.24 ± 0.65%</td>
<td>59.05 ± 0.54%</td>
</tr>
<tr>
<td>CAVIA (128)</td>
<td>49.84 ± 0.68%</td>
<td>64.63 ± 0.54%</td>
</tr>
<tr>
<td>CAVIA (512)</td>
<td><strong>51.82 ± 0.65%</strong></td>
<td>65.85 ± 0.55%</td>
</tr>
<tr>
<td>CAVIA (512, first order)</td>
<td>49.92 ± 0.68%</td>
<td>63.59 ± 0.57%</td>
</tr>
</tbody>
</table>
Optimization-Based Methods: Summary

- Meta Knowledge: Outer loop optimization
- Task Knowledge: Inner loop optimization

- Advantages:
  - Applicable to many kinds of tasks
  - Model-agnostic

- Disadvantages:
  - Second-order optimization (first-order MAML & Reptile)
<table>
<thead>
<tr>
<th></th>
<th>Metric-Based</th>
<th>Model-Based</th>
<th>Optimization-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applicable Tasks</strong></td>
<td>Classification or verification</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td><strong>Applicable Models</strong></td>
<td>All feature extractors</td>
<td>Designed Model</td>
<td>All BP-based models</td>
</tr>
<tr>
<td><strong>Computational Cost</strong></td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Easy</td>
<td>Hard</td>
<td>Hard</td>
</tr>
<tr>
<td><strong>Meta Information</strong></td>
<td>Feature space &amp; distance metric</td>
<td>Slow weight</td>
<td>Outer loop optimized weights</td>
</tr>
<tr>
<td><strong>Task Information</strong></td>
<td>None</td>
<td>Generated features &amp; weights</td>
<td>Inner loop optimized weights</td>
</tr>
</tbody>
</table>
Problem Definition
Few-shot Learning

Approaches
Non-Parametric Methods (Metric Learning)
Model-Based Methods (Black-Box Adaptation)
Optimization-Based Methods
Other Methods

Summary
MetaGAN: An Adversarial Approach to Few-Shot Learning (Zhang et al. ’18)

\[ L_D = \mathbb{E}_{x \sim Q_T^y} \log p_D(y \leq N | x) + \mathbb{E}_{x \sim p_G} \log p_D(N + 1 | x) \]

Explanation:

- Enrich task samples with outliers
- cat & car > cat & dog → better decision boundary

<table>
<thead>
<tr>
<th>Model</th>
<th>1-shot</th>
<th>5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototypical Nets</td>
<td>49.42 ± 0.78</td>
<td>68.20 ± 0.66</td>
</tr>
<tr>
<td>MAML (5 gradient steps)</td>
<td>48.70 ± 1.84</td>
<td>63.11 ± 0.92</td>
</tr>
<tr>
<td>MAML (5 gradient steps, first order)</td>
<td>48.07 ± 1.75</td>
<td>63.15 ± 0.91</td>
</tr>
<tr>
<td>MAML (1 gradient step, first order)</td>
<td>43.64 ± 1.91</td>
<td>58.72 ± 1.20</td>
</tr>
<tr>
<td>Ours: MetaGAN + MAML (1 step, first order)</td>
<td>46.13 ± 1.78</td>
<td>60.71 ± 0.89</td>
</tr>
<tr>
<td>Relation Net</td>
<td>50.44 ± 0.82</td>
<td>65.32 ± 0.7</td>
</tr>
<tr>
<td>Ours: MetaGAN + RN</td>
<td>52.71 ± 0.64</td>
<td>68.63 ± 0.67</td>
</tr>
</tbody>
</table>
A Simple Neural Attentive Meta-Learner (Mishra et al. '18)

Temporal Generation:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_{t-1})$$

Causal Temporal Convolutional Layers:

Dilated Causal Convolutional Layers (van den Oord et al. '16):
A Simple Neural Attentive Meta-Learner (Mishra et al. ’18)

\[ p \left( y_{test} | x_{test}, X_{support}, Y_{support} \right) \]

Only **Coarse** access of previous inputs (support set)
A Simple Neural Attentive Meta-Learner (Mishra et al. ´18)

Attention is all you need!
Problem Definition

Few-shot Learning

Approaches

Non-Parametric Methods (Metric Learning)
Model-Based Methods (Black-Box Adaptation)
Optimization-Based Methods
Other Methods

Summary
Summary: Common Ideas

- Knowledge Design
  - Meta-knowledge
  - Task-specific knowledge (support set)

- Network Design
  - Discriminate meta-knowledge and task-specific knowledge
  - Combine meta-knowledge and task-specific knowledge

- Training Strategy
  - Joint training or mimicking the test case?
  - Outloop and innerloop for updating different knowledge
Summary: Challenges

- Overfit
  - Sample-level overfit
  - Meta-overfit

- Optimization

- Ambiguity
Thanks for your listening!