Optimization-Based Meta-Learning and Non-Parametric Few-Shot Learning

CS 330

(finishing from last time)
Logistics

Homework 1 due, Homework 2 out this Wednesday

Fill out **poster presentation preferences**! (Tues 12/3 or Weds 12/4)

**Course project details & suggestions** posted
Proposal due Monday 10/28
Plan for Today

Optimization-Based Meta-Learning
- Recap & discuss advanced topics

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks

Properties of Meta-Learning Algorithms
- Comparison of approaches
Recap from Last Time

**MAML**

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

Optimizes for an effective initialization for fine-tuning.

**Fine-tuning**

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

[test-time]

Discussed: performance on extrapolated tasks, expressive power
Probabilistic Interpretation of Optimization-Based Inference

**Key idea:** Acquire $\phi_i$ through optimization.

Meta-parameters $\theta$ serve as a prior. One form of prior knowledge: initialization for fine-tuning

Meta-parameters

\[ \max_{\theta} \log \prod_i p(D_i | \theta) \]

\[ = \log \prod_i \int p(D_i | \phi_i) p(\phi_i | \theta) d\phi_i \quad \text{(empirical Bayes)} \]

\[ \approx \log \prod_i p(D_i | \hat{\phi}_i) p(\hat{\phi}_i | \theta) \]

MAP estimate

How to compute MAP estimate?

**Gradient descent with early stopping** = MAP inference **under Gaussian prior** with mean at initial parameters [Santos ’96]

(exact in linear case, approximate in nonlinear case)

\[ \text{MAML approximates hierarchical Bayesian inference.} \quad \text{Grant et al. ICLR ’18} \]
Optimization-Based Inference

**Key idea:** Acquire $\phi_i$ through optimization.

Meta-parameters $\theta$ serve as a prior. One form of prior knowledge: initialization for fine-tuning

**Gradient-descent + early stopping (MAML):** implicit Gaussian prior

$$\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}^{tr})$$

**Other forms of priors?**

**Gradient-descent with explicit Gaussian prior**

$$\phi \leftarrow \min_{\phi'} \mathcal{L}(\phi', \mathcal{D}^{tr}) + \frac{\lambda}{2}||\theta - \phi'||^2$$

Rajeswaran et al. implicit MAML ’19

**Bayesian linear regression on learned features**  Harrison et al. ALPaCA ’18

**Closed-form or convex optimization on learned features**

- ridge regression, logistic regression  Bertinetto et al. R2-D2 ’19
- support vector machine  Lee et al. MetaOptNet ‘19

Current **SOTA** on few-shot image classification
Optimization-Based Inference

**Key idea:** Acquire $\phi_i$ through optimization.

**Challenges**
How to choose architecture that is effective for inner gradient-step?

**Idea:** Progressive neural architecture search + MAML

(Kim et al. Auto-Meta)
- finds highly non-standard architecture (deep & narrow)
- different from architectures that work well for standard supervised learning

MinImagenet, 5-way 5-shot
MAML, basic architecture: 63.11%
MAML + AutoMeta: 74.65%
Optimization-Based Inference

**Key idea:** Acquire $\phi_i$ through optimization.

**Challenges**

Bi-level optimization can exhibit instabilities.

**Idea:** Automatically learn inner vector learning rate, tune outer learning rate

(Li et al. Meta-SGD, Behl et al. AlphaMAML)

**Idea:** Optimize only a subset of the parameters in the inner loop

(Zhou et al. DEML, Zintgraf et al. CAVIA)

**Idea:** Decouple inner learning rate, BN statistics per-step

(Antoniou et al. MAML++)

**Idea:** Introduce context variables for increased expressive power.

(Finn et al. bias transformation, Zintgraf et al. CAVIA)

**Takeaway:** a range of simple tricks that can help optimization significantly
Optimization-Based Inference

**Key idea**: Acquire $\phi_i$ through optimization.

**Challenges**
Backpropagating through many inner gradient steps is compute- & memory-intensive.

**Idea**: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity
(Finn et al. first-order MAML ‘17, Nichol et al. Reptile ‘18)

**Takeaway**: works for simple few-shot problems, but (anecdotally) not for more complex meta-learning problems.

Can we compute the meta-gradient *without differentiating through the optimization path*?

-> whiteboard

**Idea**: Derive meta-gradient using the implicit function theorem
(Rajeswaran, Finn, Kakade, Levine. Implicit MAML ’19)
Optimization-Based Inference

Can we compute the meta-gradient *without differentiating through the optimization path?*

**Idea:** Derive meta-gradient using the implicit function theorem  
(Rajeswaran, Finn, Kakade, Levine. Implicit MAML)

### Memory and computation trade-offs

![Graph showing GPU Memory (Normalized) and Compute Time (sec/iter)]

**Algorithm**                      | 5-way 1-shot | 5-way 5-shot | 20-way 1-shot | 20-way 5-shot |
---                                |             |             |               |               |
MAML [15]                          | 98.7 ± 0.4% | 99.9 ± 0.1% | 95.8 ± 0.3%   | 98.9 ± 0.2%   |
first-order MAML [15]              | 98.3 ± 0.5% | 99.2 ± 0.2% | 89.4 ± 0.5%   | 97.9 ± 0.1%   |
Reptile [43]                       | 97.68 ± 0.04% | 99.48 ± 0.06% | 89.43 ± 0.14% | 97.12 ± 0.32% |
iMAML, GD (ours)                   | 99.16 ± 0.35% | 99.67 ± 0.12% | 94.46 ± 0.42% | 98.69 ± 0.1%   |
iMAML, Hessian-Free (ours)         | **99.50 ± 0.26%** | 99.74 ± 0.11% | **96.18 ± 0.36%** | **99.14 ± 0.1%** |

A very recent development (NeurIPS ’19)  
(thus, all the typical caveats with recent work)
Optimization-Based Inference

**Key idea:** Acquire $\phi_i$ through optimization.

**Takeaways:** Construct *bi-level optimization* problem.
- positive inductive bias at the start of meta-learning
- consistent procedure, tends to extrapolate better
- maximally expressive with sufficiently deep network
- model-agnostic (easy to combine with your favorite architecture)
  - typically requires second-order optimization
  - usually compute and/or memory intensive

Can we embed a learning procedure *without* a second-order optimization?
Plan for Today

Optimization-Based Meta-Learning
- Recap & discuss advanced topics

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks

Properties of Meta-Learning Algorithms
- Comparison of approaches
So far: Learning parametric models.

In low data regimes, **non-parametric** methods are simple, work well.

During **meta-test time**: few-shot learning <-> low data regime

During **meta-training**: still want to be parametric

Can we use **parametric meta-learners** that produce effective **non-parametric learners**?

Note: some of these methods precede parametric approaches
Non-parametric methods

**Key Idea**: Use non-parametric learner.

Compare test image with training images

In what space do you compare? With what distance metric?

pixel space, $l_2$ distance?
In what space do you compare? With what distance metric?

pixel space, $l_2$ distance?
Non-parametric methods

Key Idea: Use non-parametric learner.

Compare test image with training images

In what space do you compare? With what distance metric?

pixel-space, $l_2$-distance?

Learn to compare using meta-training data!
Non-parametric methods

**Key Idea**: Use non-parametric learner.

Train Siamese network to predict whether or not two images are the same class.
Non-parametric methods

**Key Idea:** Use non-parametric learner.

Train Siamese network to predict whether or not two images are the same class.
Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class

Koch et al., ICML ‘15
Non-parametric methods

**Key Idea:** Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class

Meta-test time: compare image $\mathbf{X}_{\text{test}}$ to each image in $\mathcal{D}_j^{\text{tr}}$

Meta-training: Binary classification

Meta-test: N-way classification

Can we **match** meta-train & meta-test?
Non-parametric methods

**Key Idea:** Use non-parametric learner.

Can we **match** meta-train & meta-test?

Nearest neighbors in learned embedding space

\[
\hat{y}^{ts} = \sum_{x_k, y_k \in D_{tr}} f_\theta(x^{ts}, x_k) y_k
\]

Trained end-to-end. Meta-train & meta-test time match.

Vinyals et al. Matching Networks, NeurIPS ‘16
Non-parametric methods

Key Idea: Use non-parametric learner.

General Algorithm:

Amortized approach  Non-parametric approach (matching networks)

1. Sample task \( T_i \)  (or mini batch of tasks)
2. Sample disjoint datasets \( D_{i}^{\text{tr}}, D_{i}^{\text{test}} \) from \( D_i \)
3. Compute \( \phi_i \leftarrow f_{\theta}(D_{i}^{\text{tr}}) \)  Compute \( \hat{y}_{ts} = \sum_{x_k, y_k \in D_{i}^{\text{tr}}} f_{\theta}(x_{ts}, x_k, y_k) \)
4. Update \( \theta \) using \( \nabla_{\theta} \mathcal{L}(\phi_i, D_{i}^{\text{test}}) \)  Update \( \theta \) using \( \nabla_{\theta} \mathcal{L}(\hat{y}_{ts}, y_{ts}) \)

What if >1 shot?  Matching networks will perform comparisons independently
Can we aggregate class information to create a prototypical embedding?
Non-parametric methods

Key Idea: Use non-parametric learner.

\[
c_n = \frac{1}{K} \sum_{(x,y) \in D_i^{tr}} 1(y = n) f_\theta(x)
\]

\[
p_\theta(y = n | x) = \frac{\exp(-d(f_\theta(x), c_n))}{\sum_{n'} \exp(d(f_\theta(x), c_{n'}))}
\]

\(d: \) Euclidean, or cosine distance
Non-parametric methods

**So far:** Siamese networks, matching networks, prototypical networks

Embed, then nearest neighbors.

**Challenge**
What if you need to reason about more complex relationships between datapoints?

**Idea:** Learn non-linear relation module on embeddings

Sung et al. Relation Net

**Idea:** Learn infinite mixture of prototypes.

Allen et al. IMP, ICML ‘19

**Idea:** Perform message passing on embeddings

Garcia & Bruna, GNN
Plan for Today

Optimization-Based Meta-Learning
- Recap & discuss advanced topics

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks

Properties of Meta-Learning Algorithms
- Comparison of approaches

How can we think about how these methods compare?
Black-box vs. Optimization vs. Non-Parametric

**Computation graph perspective**

**Black-box**

\[ y^{ts} = f_\theta(D^\text{tr}_i, x^{ts}) \]

**Optimization-based**

\[ y^{ts} = f_{\text{MAML}}(D^\text{tr}_i, x^{ts}) = f_{\phi_i}(x^{ts}) \]

where \( \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D^\text{tr}_i) \)

**Non-parametric**

\[ y^{ts} = f_{\text{PN}}(D^\text{tr}_i, x^{ts}) = \text{softmax}\left( -d\left(f_\theta(x^{ts}), c_n\right) \right) \]

where \( c_n = \frac{1}{K} \sum_{(x,y) \in D^\text{tr}_i} \mathbb{I}(y = n)f_\theta(x) \)

**Note**: (again) Can mix & match components of computation graph

- Gradient descent on relation net embedding.
- MAML, but initialize last layer as ProtoNet during meta-training.

Both condition on data & run gradient descent.

Jiang et al. CAML ’19

Rusu et al. LEO ’19

Triantafillou et al. Proto-MAML ’19
Black-box vs. Optimization vs. Non-Parametric

*Algorithmic properties perspective*

**Expressive power**
- the ability for $f$ to represent a range of learning procedures
  - *Why?* scalability, applicability to a range of domains

**Consistency**
- learned learning procedure will solve task with enough data
  - *Why?* reduce reliance on meta-training tasks, good OOD task performance

Recall:

These properties are important for most applications!
<table>
<thead>
<tr>
<th>Black-box</th>
<th>Optimization-based</th>
<th>Non-parametric</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ complete expressive power</td>
<td>+ consistent, reduces to GD</td>
<td>+ expressive for most architectures</td>
</tr>
<tr>
<td>- not consistent</td>
<td>~ expressive for very deep models*</td>
<td>~ consistent under certain conditions</td>
</tr>
<tr>
<td>+ easy to combine with variety of learning problems (e.g. SL, RL)</td>
<td>+ positive inductive bias at the start of meta-learning</td>
<td>+ entirely feedforward</td>
</tr>
<tr>
<td>- challenging optimization (no inductive bias at the initialization)</td>
<td>+ handles varying &amp; large K well</td>
<td>+ computationally fast &amp; easy to optimize</td>
</tr>
<tr>
<td>- often data-inefficient</td>
<td>+ model-agnostic</td>
<td>- harder to generalize to varying K</td>
</tr>
<tr>
<td></td>
<td>+ second-order optimization</td>
<td>- hard to scale to very large K</td>
</tr>
<tr>
<td></td>
<td>- usually compute and memory intensive</td>
<td>- so far, limited to classification</td>
</tr>
</tbody>
</table>

Generally, well-tuned versions of each perform **comparably** on existing few-shot benchmarks! (likely says more about the benchmarks than the methods)

Which method to use depends on your **use-case**.

*for supervised learning settings*
Black-box vs. Optimization vs. Non-Parametric

\textit{Algorithmic properties} perspective

- **Expressive power**: the ability for $f$ to represent a range of learning procedures
  \textit{Why?} scalability, applicability to a range of domains

- **Consistency**: learned learning procedure will solve task with enough data
  \textit{Why?} reduce reliance on meta-training tasks, good OOD task performance

- **Uncertainty awareness**: ability to reason about ambiguity during learning
  \textit{Why?} active learning, calibrated uncertainty, RL principled Bayesian approaches

\textbf{We’ll discuss this next time!}
Reminders

Homework 1 due, Homework 2 out this Wednesday

Fill out poster presentation preferences! (Tues 12/3 or Weds 12/4)

Course project details & suggestions posted
Proposal due Monday 10/28