Meta-Learning

DRL Seminar

Philippe Blatter
Overview

- Introduction to Meta-Learning
- Model-Agnostic Meta-Learning (MAML)
- Optimization-based approaches
- Meta-Learning in RL
Supervised Learning Paradigm

- Large datasets
- Large models
- Long training time

Transformer
([1] Vaswani et al. 2017)
Possible Problems

Large datasets might not be available

General-purpose AI

Long-tailed data

Finn et al. 2017
Example

2-way

Braque

Cezanne

3 shots

Braque or Cezanne?

Can we learn to learn?

Problem Setting

$D^t_\text{tr}$

$D^t_\text{ts}$

$D$

$D_{\text{meta-train}}$

$D_1$

$D_2$

\[2\] Finn et al. 2017
Problem Setting

Supervised learning:

\[ \arg \max_{\phi} \log p(\phi|D) \]

Meta-learning:

\[ \arg \max_{\phi} \log p(\phi|D, D_{\text{meta-train}}) \]

\[ D = \{(x_1, y_1), \ldots, (x_k, y_k)\} \]

\[ D_{\text{meta-train}} = \{D_1, \ldots, D_n\} \]

\[ D_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\} \]

Meta-Learning Terminology

$D^{tr}$

$D^{ts}$

$D_1$

$D_2$

$D$

meta-training

$\theta^*$

use $\theta^*$ find $\phi^*$

Meta-Learning Problem

\[
\theta^* = \arg \max_{\theta} \log p(\theta | D_{\text{meta-train}})
\]


\[
\phi^* = \arg \max_{\phi} \log p(\phi | D, D_{\text{meta-train}}) = \arg \max_{\phi} \log p(\phi | D, \theta^*)
\]

adaptation
(Meta) Test-Time

Adaptation:

\[ \phi^* = \arg\max_{\phi} \log p(\phi|D, \theta^*) \]

\[ D^{tr} \]

\[ x^{tr}, y^{tr} \]

\[ M_{tr} \]

\[ D^{ts} \]

\[ x^{ts}, y^{ts} \]

\[ M_{ts} \]

(Meta) Training-Time

Meta-Learning:

\[ \theta^* = \arg \max_{\theta} \log p(\theta | D_{\text{meta-train}}) \]

Complete Meta-Learning Problem

Meta-learning: \[ \theta^* = \arg \max_{\theta} \log p(\theta | D_{\text{meta-train}}) \]

Adaptation: \[ \phi^* = \arg \max_{\phi} \log p(\phi | D, \theta^*) \]

Learn \( \theta \) such that \( \phi_i = f_\theta(D_i^{tr}) \) is good for \( D_i^{ts} \) for all tasks \( i \)

\[ \theta^* = \arg \max_{\theta} \sum_{i=1}^{n} \log p(\phi_i | D_i^{ts}) \]

where \( \phi_i = f_\theta(D_i^{tr}) \)

Model-Agnostic Meta-Learning (MAML)

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

\[ \theta \] parameter vector being meta-learned

\[ \phi_i^* \] optimal parameter vector for task i

Model-Agnostic Meta-Learning

“In our approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task.”

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

pre-trained parameters

training data for new task

Fine-tuning [test-time]

Meta-learning

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

\[ \phi_i \]

Understanding the Effectiveness of MAML

- Rapid Learning: large representational changes occur during adaptation to new task

- Feature Reuse: Meta-initialization already contains highly useful features that can be reused for new tasks

Freezing Layer Representations

Performance hardly changes.

-> Feature Reuse

\[ D^{tr} \rightarrow \theta^* \rightarrow \phi^* \rightarrow y^{ts} \]

\[ D^{ts} \rightarrow x^{ts} \]

Representational Similarity Experiments

- Measure changes in the latent representations learned by the NN during adaptation using Canonical Correlation Analysis (CCA)

- Highly similar representations in the body of the network
  - \( \text{CCA}(L1,L2) \)
  - \( \Rightarrow \) No functional change
  - \( \Rightarrow \) No rapid learning

ANIL Algorithm: Almost no Inner Loop (Adaptation)

- Similar Performance to MAML

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

\[ \phi_{i} \]

**MAML**

\[ \theta^{*}_{T_{b}} = \left( \theta_{1} - \alpha \frac{\partial L_{T_{b}}(\theta)}{\partial \theta_{1}}, \theta_{2} - \alpha \frac{\partial L_{T_{b}}(\theta)}{\partial \theta_{2}}, \theta_{\text{head}} - \alpha \frac{\partial L_{T_{b}}(\theta)}{\partial \theta_{\text{head}}} \right) \]

\[ \text{Task}_{T_{b}} \quad \theta = (\theta_{1}, \theta_{2}, \theta_{\text{head}}) \]

\[ \theta^{*}_{T_{d}} \quad \text{Task}_{T_{d}} \]

\[ \theta^{*}_{T_{c}} \quad \text{Task}_{T_{c}} \]

**ANIL**

\[ \theta^{*}_{T_{b}} = \left( \theta_{1}, \theta_{2}, \theta_{\text{head}} - \alpha \frac{\partial L_{T_{b}}(\theta)}{\partial \theta_{\text{head}}} \right) \]

\[ \text{Task}_{T_{b}} \quad \theta = (\theta_{1}, \theta_{2}, \theta_{\text{head}}) \]

\[ \theta^{*}_{T_{d}} \quad \text{Task}_{T_{d}} \quad \theta^{*}_{T_{d}} \quad \text{Task}_{T_{c}} \]

Learning to learn by gradient descent by gradient descent

hand-designed features → learned features

hand-designed optimization algorithms → learned optimization algorithms

“Casting algorithm design as a learning problem”

Learning to learn by gradient descent

Learning to learn by gradient descent by gradient descent

hand-designed optimization algorithms

learned optimization algorithms

“Casting algorithm design as a learning problem”

\[ \theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi) \]

Learning to learn by gradient descent
by gradient descent

Learning to learn by gradient descent by gradient descent

Quadratics

Loss

- ADAM
- RMSprop
- SGD
- NAG
- LSTM

Step

MNIST

[MNIST, 200 steps]

Learning to learn by gradient descent

Andrychowicz et al. 2016

Meta-Learning in RL

Meta-Learning in RL

Reinforcement learning:

$$\theta^* = \arg \max_{\theta} E_{\pi_{\theta}(\tau)}[R(\tau)]$$

$$= f_{RL}(M) \quad M = \{S, A, P, r\}$$

\[\text{MDP}\]

Meta-reinforcement learning:

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

where $$\phi_i = f_{\theta}(M_i)$$

\[\text{MDP for task } i\]

"We view the learning process of the agent itself as an objective, which can be optimized using standard RL algorithms."

RL² – Fast RL via Slow RL

Policy is modeled by a RNN

RL$^2$ – Fast RL via Slow RL

Environment is modeled by a MDP

**RL² – Fast RL via Slow RL**

next state, action, reward and termination flag

---

RL$^2$ – Fast RL via Slow RL

**RL² – Fast RL via Slow RL**

Hidden state is kept

Diagram showing the flow of states and actions in two episodes:
- **Episode 1:** States $s_0$, $s_1$, $s_2$, $s_3$, $s_0$, $s_1$, $s_2$ with actions $a_0$, $a_1$, $a_2$, $a_0$, $a_1$, $a_0$, $a_1$ and rewards $r_0$, $d_0$, $r_1$, $d_1$, $r_2$, $d_2$, $r_0$, $d_0$, $r_1$, $d_1$.
- **Episode 2:** States $s_0$, $s_1$, $s_2$, $s_3$, $s_0$, $s_1$, $s_2$ with actions $a_0$, $a_1$, $a_2$, $a_0$, $a_1$, $a_0$, $a_1$ and rewards $r_0$, $d_0$, $r_1$, $d_1$, $r_2$, $d_2$, $r_0$, $d_0$, $r_1$, $d_1$.

MDP 1 (Trial 1): $s_0$, $s_1$, $s_2$, $s_3$, $s_0$, $s_1$, $s_2$

MDP 2 (Trial 2): $s_0$, $s_1$, $s_2$, $s_3$, $s_0$, $s_1$, $s_2$

RL² – Fast RL via Slow RL

Second trajectory is almost always shorter

Generalizes to larger mazes

Thought Experiment

We assumed that learning optimization algorithms was better than hand-designing optimization algorithms. But why do we think that hand-designing meta-learning algorithms is optimal and why don’t we meta-meta-learn them?

METACEPTION
References


