1 Information from the video

Seminar

1. Understand the Bayesian formulation of meta-learning
2. Understand Bayesian extensions of MAML (Model-Agnostic Meta-Learning) [1,2,3]

Practicum

1. implement Bayesian meta-learning algorithms
2. apply to various benchmark tasks
3. compare performance

Paper

1. Finn et al., "Model-agnostic Meta-Learning for Fast Adaption of Neural Networks"
2. Finn et al., "Probabilistic Model-Agnostic Meta-Learning"
3. Kim et al., "Bayesian Meta-Learning"
4. Yin et al., "Meta-learning without Memorization"

2 Model-agnostic Meta-Learning for Fast Adaption of Neural Networks

2.1 Abstract and Introduction

Goal of Meta-Learning

- The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples.

Advantage Compared to Previous Methods

- does not expand the number of learned parameters nor places constraints on the model architecture, unlike prior methods.
- compares favorably to state-of-the-art one-shot learning methods designed specifically for supervised classification while using fewer parameters, but can also be readily applied to regression and can accelerate reinforcement learning in the presence of task variability, substantially outperforming direct pretraining as initialization.

2.1.1 REINFORCE

1. Initialize random θ and obtain $f_\theta(\boldsymbol{a}_t|\boldsymbol{x}_t)$
2. Sample K trajectories $\tau_j = \{(x_0, a_0), ..., (x_T, a_T)\}_j \sim q_\theta(\boldsymbol{\tau})$

$$q_\theta(\boldsymbol{\tau}) = q(\boldsymbol{x}_1) \prod_{t=1}^{H} q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t) f_\theta(\boldsymbol{a}_t|\boldsymbol{x}_t). \tag{1}$$

3. Approximate expected return $J(\theta)$ as average episode reward

$$J(\theta) = \frac{1}{K} \sum_{j=1}^{K} \sum_{t=1}^{H} \gamma^t r(x_{j,t}, a_{j,t}). \tag{2}$$

4. Update parameters θ with gradient ascent

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{3}$$

where $\nabla J(\theta)$ is approximated with the log ratio trick

$$\nabla J(\theta) \approx \frac{1}{K} \sum_{j=1}^{K} \left(\sum_{t=1}^{H} \nabla_\theta \log f_\theta(a_{j,t}|x_{j,t}) \right) \left(\sum_{t=1}^{H} \gamma^t r(x_{j,t}, a_{j,t}) \right). \tag{4}$$

This involves evaluation of and gradient of the neural network

2.1.2 Loss function

In the algorithm:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta) \tag{5}$$

For REINFORCE case we would have:

$$\mathcal{L}(f_\theta) = -J(\theta) \tag{6}$$

Loss function equals negative return. Input parameter f_θ or θ implicitly says we need to use f_θ to sample trajectories τ from this policy and use the samples to calculate the loss/return

2.2 Meta-Learning Problem Setup Theory

$\mathcal{T} = \{\mathcal{L}(f_\theta), q(\boldsymbol{x}_1), q(\boldsymbol{x}_{t+1}\|\boldsymbol{x}_t, \boldsymbol{a}_t), H\} \sim p(\mathcal{T})$	Task and Task distribution
$\boldsymbol{x}$	State/Observation
$\boldsymbol{a}$	Action/Output
H	Episode Length
$q(\boldsymbol{x_1})$ and $q(\boldsymbol{x}_{t+1}\|\boldsymbol{x}_t, \boldsymbol{a}_t)$	Initial State Distribution and Transition Distribution
$\mathcal{L}(\boldsymbol{x}_1, \boldsymbol{a}_1, ..., \boldsymbol{x}_H, \boldsymbol{a}_H)$ or $\mathcal{L}(f_\theta) \to \mathbb{R}$	Loss Function
$f_\theta(\boldsymbol{a}_t\|\boldsymbol{x}_t)$	Parameterized Model/Policy

Meta-Training (optimize θ)

1. Start with random parameters θ
2. Sample n_{train} tasks $\mathcal{T}_i \sim p(\mathcal{T})$
3. Draw K samples each of length H $\{(x_1, a_1, ..., x_H, a_H)\}_{j=1...K}$ from $q_i(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{a}_t \sim f_\theta(\cdot|\boldsymbol{x_t})$
4. Adapt f_θ to f_{θ_i} by minimizing the loss function $\mathcal{L}_i$

$$\theta_i = \underset{\theta}{\operatorname{argmin}} \sum_{j=1}^{K} \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j). \tag{7}$$

   Also sample more samples of length H $\{(x_1, a_1, ..., x_H, a_H)\}_{j=1...L}$ for testing where $\boldsymbol{a}_t \sim f_{\theta_i}(\cdot|\boldsymbol{x_t})$.
   Do steps 3 and 4 with all tasks $\mathcal{T}_i$.

5. Use test set samples from $f_{\theta_1}, ..., f_{\theta_{n_{train}}}$ to optimize θ

$$\theta_{new} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n_{train}} \sum_{j=1}^{L} \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j). \tag{8}$$

Meta-Testing (check Meta-Performance ,i.e., if θ_{new} works on new tasks)

1. Sample n_{test} new tasks $\mathcal{T}_i \sim p(\mathcal{T})$
2. Draw a Support Set with K samples $\mathcal{D}_i^S = \{(x_1, a_1, ..., x_H, a_H)\}_{j=1...K}$ from $q_i(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{a}_t \sim f_{\theta_{new}}(\cdot|\boldsymbol{x_t})$
3. Adapt $f_{\theta_{new}}$ to f_{θ_i} by minimizing the loss function $\mathcal{L}_i$

$$\theta_i = \underset{\theta}{\operatorname{argmin}} \sum_{j=1}^{K} \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j). \tag{9}$$

   Also sample more samples of length H (Query Set) $\mathcal{D}_i^Q = \{(x_1, a_1, ..., x_H, a_H)\}_{j=1...L}$ for testing where $\boldsymbol{a}_t \sim f_{\theta_i}(\cdot|\boldsymbol{x_t})$.
   Do steps 3 and 4 with all testing tasks $\mathcal{T}_i$.

4. Use test set samples from $f_{\theta_1}, ..., f_{\theta_{n_{train}}}$ to evaluate the Meta-Loss $\mathcal{L}$

$$Loss = \sum_{i=1}^{n_{test}} \sum_{j=1}^{L} \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j). \tag{10}$$

Neural Network Evaluations Supervised Learning (H=1)

- Training 3: $K \cdot n_{train}$ (we evaluate f_θ on K inputs for each Task)
- Training 4: $L \cdot n_{train}$ (we evaluate each f_{θ_i} on L test samples)
- Testing 2: $K \cdot n_{test}$
- Testing 3: $L \cdot n_{test}$

2.3 MAML Objective

$$\theta_i = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta, \mathcal{D}_i^S)$$
$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\theta_i, \mathcal{D}_i^Q)$$

2.4 Model-Agnostic Meta-Learning (MAML) Algorithm

Meta-Training (optimize θ)

1. Start with random parameters θ
2. Sample n_{train} tasks $\mathcal{T}_i \sim p(\mathcal{T})$
3. Draw K samples each of length H $\{(x_1, a_1, ..., x_H, a_H)\}_{j=1...K}$ from $q_i(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{a}_t \sim f_\theta(\cdot|\boldsymbol{x_t})$
4. Adapt $f_{\theta_{new}}$ to f_{θ_i} by minimizing the loss function $\mathcal{L}_i$

$$\theta_i \leftarrow \theta - \alpha \frac{1}{K} \sum_{j=1}^{K} \nabla_\theta \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j) \tag{11}$$

   Also sample more samples of length H $\{(x_1, a_1, ..., x_H, a_H)\}_{j=1...L}$ for testing where $\boldsymbol{a}_t \sim f_{\theta_i}(\cdot|\boldsymbol{x_t})$.
   Do steps 3 and 4 with all testing tasks $\mathcal{T}_i$.
5. Use test set samples from $f_{\theta_1}, ..., f_{\theta_{n_{train}}}$ to optimize θ

$$\theta \leftarrow \theta - \beta \sum_{i=1}^{n_{train}} \sum_{j=1}^{L} \nabla_\theta \mathcal{L}_i((x_1, a_1, ..., x_H, a_H)_j) \tag{12}$$

Meta-Testing (check Meta-Performance ,i.e., if θ works on new tasks)

1. Sample n_{test} new tasks $\mathcal{T}_i \sim p(\mathcal{T})$
2. Draw K-Samples $(\boldsymbol{x}_1, \boldsymbol{a}_1, ..., \boldsymbol{x}_K, \boldsymbol{a}_K)$ from $q_i(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{a}_t \sim f_\theta(\cdot|\boldsymbol{x_t})$
3. train f_θ to optimize Loss $\mathcal{L}$ and obtain f_{θ_i}
4. Sample more samples $(\boldsymbol{x}_{K+1}, \boldsymbol{a}_{K+1}, ...)$ for testing on $\mathcal{L}$ and obtain loss.