CS 330 Autumn 2023 Homework 4
Advanced Meta-Learning Topics

Due Wednesday, November 15th, 11:59 PM PT

SUNet ID:
Name:
Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Overview

This assignment will explore three different advanced meta-learning topics. You will:

1. Answer conceptual questions about whether memorization can occur in a few realistic meta-learning scenarios.
2. Derive an objective for a Bayesian meta-learning setting where, in addition to a dataset, metadata is available for all tasks.
3. Using provided code, run experiments comparing the computational costs of two different methods for learning initial parameters for a neural network.

The first two problems are purely conceptual and involve no programming at all. For the last question, you will use provided scripts to run experiments and report the results.

Grading policy: This assignment is *optional*, and accounts for up to 15% of your grade. Your grade for this homework will replace the grade of either one prior homework or part of the final project, whichever choice results in the highest final grade. Attempting this homework will therefore never harm your grade.

Submission: To submit your homework, submit one PDF report to Gradescope containing answers to the questions below. The PDF should also include your name and any students you talked to or collaborated with. **Any written responses or plots to the questions below must appear in your PDF submission**.

Code Submission: You are not required to submit code.

Problem 1: Memorization in Meta-Learning (6 Points)

In this problem, we will examine four task distributions in a meta-learning problem setting to determine whether or not memorization can occur. We consider a meta-learning setting

where each task consists of a labeled support set and a query set which the model must make predictions on. We adopt the notion of *complete meta-learning memorization* from [1]. A task distribution can suffer from complete memorization if a meta-learning algorithm can achieve perfect performance for all tasks in the meta-training set, solely based on the query set while ignoring the support set.

For each of the following meta-learning task settings, answer **two questions** and explain your reasoning for each answer. First, state whether or not complete memorization can occur. Second, state whether the performance on meta-test tasks will be (1) equal or worse than random guessing (2) worse than meta-train but better than random guessing (3) as good as meta-train. For each of these two questions, explain your reasoning in one or two sentences. You are not required to provide a formal mathematical proof.

Note: some of these questions may have multiple plausible answers depending on how you interpret the scenario and what additional assumptions you make. Explicitly state the problem setup you are considering, including any additional assumptions. Any reasonable and logically consistent answer will receive a full grade.

(a) **Regression tasks from basis functions.** Consider regression tasks where the input and output domains are $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \mathbb{R}$, respectively. Each task contains 10 train and 10 test datapoints, and is constructed from a finite set of basis functions $\phi_i : \mathbb{R} \to \mathbb{R}$ for $i \in \{1, \ldots, n\}$ that are linearly independent, i.e., there does not exist $w_1, \ldots, w_n$ such that $\sum_i w_i \phi_i = 0$. Let $p(w)$ be a distribution over $\mathbb{R}^n$. Each task is constructed by sampling $w \sim p(w)$ and then using $x \mapsto \sum_i w_i \phi_i(x)$ as the ground-truth function, where the inputs x are sampled from Unif$([0, 1])$. (2 points)

Your two-part answer goes here.

(b) **Medical image classification.** Consider a meta-learning dataset for medical imaging, where the goal is for a model to be able to recognize a novel disease given a small number of labeled cases. The input is a medical image, such as an X-ray or CT scan. The output is a binary label $y \in \{0, 1\}$ where 0 represents a sample from a healthy person and 1 represents a sample with a specific disease. Assume we have a large dataset of images corresponding to N different diseases, where N is a large number. To construct a meta-training task, we randomly sample one of the N diseases, then sample 5 healthy and 5 diseased images. Meta-testing tasks are constructed similarly from images corresponding to a set of M held-out diseases. (2 points)

Your two-part answer goes here.

(c) **Robotic grasping.** Consider a simplified robotic grasping task, where the goal is to predict the three-dimensional coordinates ($\in \mathbb{R}^3$) of a target object to grasp. The robot is a movable arm firmly attached to a desk. Many different objects are placed on the desk, and each task consists of grasping a specific object. The meta-testing tasks correspond to held-out objects on the desk. The input is an image that shows everything in the robot's field of view, including multiple objects on a desk and a

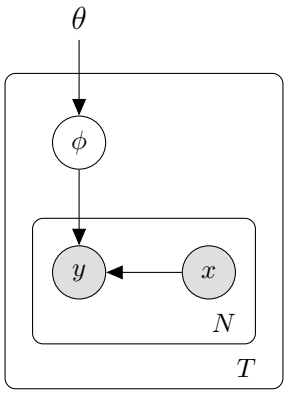

Figure 1: Plate notation diagram of a simple black-box Bayesian meta-learning model.

computer monitor. The target object for each task is written on a computer monitor, inside the camera's field of view. As task-specific adaptation, the robot is allowed $K = 5$ attempts to predict the grasp location; i.e. the support set includes 5 grasp attempts. During each attempt, the robot makes object coordinate predictions and receives a reward signal based on how close its prediction was to the object. At meta-test time, we test the robot on novel objects with the monitor turned off, and measure the average grasping success rate after the initial 5 trials. (2 points)

Your two-part answer goes here.

Problem 2: Bayesian Meta-Learning (4 Points)

In this problem, we consider a Bayesian black-box meta-learning method with observable task metadata as side information. The novel component in our setup is *task metadata* which succinctly summarizes aspects of each task. We assume that m is available for all tasks.

Our probabilistic model involves two latent variables $\phi_1 \in \mathbb{R}^D$ and $\phi_2 \in \mathbb{R}^D$ and an observable task metadata variable $m \in \mathbb{R}^D$. We assume a prior $p(\phi_1)$ over the top-level variable, and that ϕ_2 is sampled according to the conditional distribution $p(\phi_1|\phi_2, m)$. The label corresponding to an example x is predicted as $p(y|\phi_2, x)$. We use networks $q(\phi_2|x_{1:N}, y_{1:N})$ and $q(\phi_1|\phi_2, m)$ to perform amortized inference of the two hidden variables ϕ_1 and ϕ_2.

(a) Draw a plate notation diagram for this model. Your diagram should include all variables $\theta, \phi_1, \phi_2, x, y, m$ with appropriate node colors reflecting whether each node is hidden or observable, and plates representing the T tasks and N datapoints in each task. You can hand-draw a diagram or modify the provided `tikz` diagram for a simpler black-box Bayesian meta-learning model (Figure 1). (1 points)

Your diagram goes here.

(b) Complete the following derivation of an evidence lower bound (ELBO) for this model, given the dataset $(x_{1:N}, y_{1:N})$ and metadata m from each task. Your derivation can use standard lemmas such as Jensen's inequality without proof. (3 points)

Hint: introducing new symbols such as $X = x_{1:N}$ and $Y = y_{1:N}$ can simplify your equations.

Your derivation goes here, and should likely start with something like:

$$\log p(y_{1:N}|x_{1:N}, m) \geq \dots \tag{1}$$

Problem 3: Scaling of Meta-Learning Algorithms (5 Points)

In this problem, you will investigate how the computational costs, both runtime and memory usage, of two meta-learning algorithms scale with different hyperparameters. We will consider Model-Agnostic Meta-Learning (MAML) and Evolution Strategies (ES). Understanding the computational behavior of these algorithms is important for understanding their practicality in real-world applications, where computational resources are often a limiting factor.

Your primary objective is to profile the computational costs of MAML and ES under different hyperparameter configurations. You will be provided with code to run the experiments. We will not be considering the actual performance of the meta-learning methods; we will focus exclusively on their computational costs. To measure the computational costs with the provided code, **you must run the experiments on a GPU**, since we use total allocated GPU memory as a proxy for memory usage.

We will be using a simple 1D regression task, where the input and output are both scalars. Each task corresponds to a random sinusoidal function with a random phase, bias, and amplitude. The dataset for each task consists of 10 training examples and 100 test examples. The meta-learner is trained on many tasks of this type, and is expected to generalize to new tasks by learning a good initialization for model parameters which quickly adapts to new tasks based on the 10 training examples. The provided codebase automatically generates a random dataset for each task. The model is a fully-connected neural network, and the default architecture uses 2 hidden layers of 128 hidden units each.

We will be comparing two meta-learning methods: Model-Agnostic Meta-Learning (MAML) and Evolution Strategies (ES), which we briefly review here. We use either method to learn the initial parameters of a neural network that is then adapted to a new task using gradient descent. Denote the initial parameters as θ, the adapted parameters as ϕ, and the dataset for a task $\mathcal{T}_i$ as $\mathcal{D}_i = (\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}})$. Inner-loop adaptation is performed by taking L

gradient descent steps on the support data $\mathcal{D}_i^{\text{tr}}$:

$$\begin{aligned}
\phi^1 &= \phi^0 - \alpha \nabla_{\phi^0} \mathcal{L}(\phi^0, \mathcal{D}_i^{\text{tr}}) \\
\phi^2 &= \phi^1 - \alpha \nabla_{\phi^1} \mathcal{L}(\phi^1, \mathcal{D}_i^{\text{tr}}) \\
&\vdots \\
\phi^L &= \phi^{L-1} - \alpha \nabla_{\phi^{L-1}} \mathcal{L}(\phi^{L-1}, \mathcal{D}_i^{\text{tr}}).
\end{aligned} \tag{2}$$

In this assignment, both MAML and ES optimize the initial parameters $\theta = \phi^0$ by taking gradient steps with respect to the meta-objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T}), (\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}}) \sim \mathcal{T}_i} \left[\mathcal{L}(\phi^L, \mathcal{D}_i^{\text{ts}}) \right]. \tag{3}$$

The two methods differ in how they compute the meta-objective gradient $\nabla_\theta \mathcal{J}(\theta)$. MAML performs backpropagation through all L unrolled inner-loop gradient steps:

$$\begin{aligned}
\nabla_\theta \mathcal{J}(\theta) &= \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T}), (\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}}) \sim \mathcal{T}_i} \left[\nabla_\theta \mathcal{L}(\phi^L, \mathcal{D}_i^{\text{ts}}) \right] \\
&= \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T}), (\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}}) \sim \mathcal{T}_i} \left[\nabla_\theta \mathcal{L}(\phi^{L-1} - \alpha \nabla_{\phi^{L-1}} \mathcal{L}(\phi^{L-1}, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}}) \right] \\
&= \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T}), (\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}}) \sim \mathcal{T}_i} \left[\nabla_\theta \mathcal{L}(\phi^{L-2} - \alpha_{\phi^{L-2}} \cdots \right]
\end{aligned} \tag{4}$$

In contrast, ES uses a finite-difference estimator:

$$\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\mathcal{J}(\theta + \epsilon \delta_i) - \mathcal{J}(\theta - \epsilon \delta_i)}{2\epsilon} \delta_i, \tag{5}$$

where δ_i is a random vector that has the same dimension as θ and is sampled from a standard normal distribution. This sort of finite-difference estimator is commonly used in evolutionary algorithms; its use for optimizing initial parameters has been explored in [2]. In the context of this assignment, we are comparing the computational costs of unrolled backpropagation (MAML) and finite-difference estimation (ES) for the same optimization problem rather than comparing the performance of the two methods.

We will be comparing the computational costs of MAML and ES while varying three hyperparameters:

- **Inner steps:** the number of gradient descent steps to take on the support set.
- **Layers:** the number of hidden layers in the model.
- **Hidden units:** the number of hidden units per layer.

(a) **Plots [2 points]** Compare the runtime and memory cost of MAML and ES while varying the number of inner steps, layers, and hidden units. Show results in six plots, one for each combination of metric (runtime or memory) and experiment (inner steps, layers, hidden units).

Your plots go here.

(b) **Analysis [3 points]** Comment on your plots, and explain the trends you observe. For which experiment do you see the largest qualitative difference between MAML and ES, and why do you think this is the case?

Your analysis goes here.

References

[1] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, Chelsea Finn. *Meta-Learning without Memorization* https://arxiv.org/abs/1912.03820

[2] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, Yunhao Tang *ES-MAML: Simple Hessian-Free Meta Learning* https://arxiv.org/abs/1910.01215