Mid-Term Project Evaluation

Model-Agnostic Meta-Learning (MAML)

Project Report

Suvan Arora

January 9, 2026

Contents

1 Introduction

1.1 Motivation and Problem Context

Deep learning models have achieved remarkable success in domains ranging from computer vision to natural language processing. However, a persistent limitation of standard deep learning is data hunger: these models typically require massive labeled datasets to generalize well. In contrast, human intelligence is characterized by the ability to learn new concepts and skills from a handful of examples by leveraging prior experience.

This project addresses this gap through the lens of **Few-Shot Learning (FSL)**. The motivating hypothesis is that rapid adaptation can be framed as an optimization problem: rather than training a model to solve a single task, we train it to "learn to learn." This approach, known as Meta-Learning, aims to extract transferable knowledge from a distribution of tasks, enabling an agent to adapt to a new task with minimal data and computational effort.

1.2 Project Roadmap and Trajectory

The work completed so far follows a logical progression designed to build rigorous mathematical and practical intuition:

1. **Foundations & Classical ML:** Understanding learning as an optimization problem, implementing regression and classification algorithms from scratch to grasp gradients, loss landscapes, and the bias-variance tradeoff.
2. **Deep Learning Primer:** Transitioning to hierarchical feature learning, understanding backpropagation, and studying CNN inductive biases.
3. **Transfer Learning Bridge:** Conducting experiments on MNIST to understand the role of initialization and the limitations of standard pre-training.
4. **Meta-Learning (MAML):** Studying the core focus of the project—Model-Agnostic Meta-Learning—including its bi-level optimization structure and implementation.

2 Foundations: Classical Machine Learning

Before approaching meta-learning, it was essential to establish a strong footing in classical machine learning. This phase focused less on achieving high accuracy and more on understanding *why* training behaves the way it does.

2.1 Learning as Optimization

Machine learning is fundamentally an optimization process. Unlike traditional programming, where rules are explicitly coded, ML models learn parameters θ (weights and biases) by minimizing a loss function $\mathcal{L}$.

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f_\theta(x), y) \tag{1}$$

We explored this through the implementation of Gradient Descent (GD) and Stochastic Gradient Descent (SGD). A key insight gained was the sensitivity of the learning rate η. A generic SGD update rule implemented was:

$$w_{t+1} = w_t - \eta \nabla_w \mathcal{L}(w_t; \mathcal{B}_t) \tag{2}$$

where $\mathcal{B}_t$ is the mini-batch at iteration t. We observed that while batch gradient descent yields smooth loss curves, SGD introduces gradient noise that can assist in escaping local minima, a concept that reappears in the stability analysis of meta-learning inner loops.

2.2 Regression and the Bias-Variance Tradeoff

We implemented **Linear Regression** and **Polynomial Regression** to predict continuous outputs. These experiments highlighted the **Bias-Variance Tradeoff**, a recurring theme in this project:

- **High Bias (Underfitting):** Occurs when the model (e.g., a linear model on curved data) is too simple to capture the underlying structure.
- **High Variance (Overfitting):** Occurs when the model (e.g., a high-degree polynomial) captures noise in the training set, leading to poor generalization.

In the context of Few-Shot Learning, this tradeoff is critical: adapting to a task with only $K = 5$ examples (5-shot) carries an extreme risk of overfitting (high variance) unless the model relies on a robust prior or initialization.

2.3 Classification and Decision Boundaries

We extended our study to discrete label prediction using **Logistic Regression**, **K-Nearest Neighbors (KNN)**, and **Support Vector Machines (SVM)**.

- **Logistic Regression:** Served as the building block for neural networks, introducing the sigmoid activation and cross-entropy loss.
- **KNN:** demonstrated instance-based learning. We observed that $K = 1$ leads to complex, jagged decision boundaries (overfitting), while large K over-smooths classes.
- **SVM Kernels:** The Kernel Trick $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$ introduced the concept of mapping data to high-dimensional feature spaces to make them linearly separable. This foreshadows deep learning, where the network learns the feature mapping $\phi(\cdot)$ explicitly.

3 Transition to Deep Learning

Classical algorithms often fail on high-dimensional data (images, audio) because they lack hierarchical feature learning. This motivated the transition to Deep Neural Networks.

3.1 Neural Network Architecture

We studied Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). The core realization was that a neural network is a function approximator composed of differentiable layers.

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}) \tag{3}$$

where σ is a non-linear activation function (ReLU, Sigmoid). Without σ, the network collapses to a linear model.

3.2 Backpropagation

We implemented backpropagation to understand how gradients flow via the chain rule. For a layer l, the error term δ^l is computed recursively from the output layer backward:

$$\delta^l = (W^{l+1})^\top \delta^{l+1} \odot \sigma'(z^l) \tag{4}$$

Understanding this flow is a prerequisite for MAML, which involves differentiating *through* a gradient update (computing gradients of gradients).

4 Transfer Learning: The Bridge to Meta-Learning

To motivate MAML, we conducted a transfer learning experiment using the MNIST dataset.

4.1 Experimental Setup

We treated the digits dataset as a multi-task problem:

- **Task A (Source):** Classification of digits 0 vs. 1.
- **Task B (Target):** Classification of digits 2 vs. 3.

We compared three strategies:

1. **Random Initialization:** Training on Task B from scratch.
2. **Transfer ($A \to B$):** Pre-training on Task A, then fine-tuning on Task B.
3. **Generalist Pre-training:** Pre-training on all other digits, then fine-tuning.

4.2 Observations

The experiments demonstrated that starting from a pre-trained representation significantly accelerates convergence on the target task. However, we noted a critical limitation: standard transfer learning optimizes for performance on the *source* task, not for *adaptability* to the target task. This "greedy" approach can sometimes lead to features that are too specialized. This gap is precisely what MAML aims to fill by explicitly optimizing for adaptability.

5 Model-Agnostic Meta-Learning (MAML)

This section details the core focus of the project. MAML (Finn et al., 2017) is a framework for learning a network initialization that can adapt to a new task with a small number of gradient steps.

5.1 Problem Formulation

We define a distribution over tasks $p(\mathcal{T})$. Each task $\mathcal{T}_i$ consists of:

- A **Support Set** $\mathcal{D}_S$ (used for adaptation/training).
- A **Query Set** $\mathcal{D}_Q$ (used for evaluation/meta-update).

In an N-way K-shot setting, the support set contains N classes with K examples each.

5.2 The Algorithm: Bi-Level Optimization

MAML operates via two nested loops: an inner loop (task adaptation) and an outer loop (meta-update).

5.2.1 The Inner Loop

For a sampled task $\mathcal{T}_i$, we compute adapted parameters θ'_i by taking one (or more) gradient descent steps on the support set loss $\mathcal{L}_{\mathcal{T}_i}$:

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_S^{(i)}) \tag{5}$$

where α is the inner-loop learning rate.

5.2.2 The Outer Loop

The goal is to find the initialization θ that minimizes the loss of the *adapted* parameters θ'_i on the query set $\mathcal{D}_Q$. The meta-objective is:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}; \mathcal{D}_Q^{(i)}) \tag{6}$$

The meta-update is performed using Stochastic Gradient Descent on this objective with meta-learning rate β:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}; \mathcal{D}_Q^{(i)}) \tag{7}$$

Algorithm 1 Model-Agnostic Meta-Learning (MAML)

```
Require: Distribution over tasks $p(\mathcal{T})$
Require: Step size hyperparameters $\alpha, \beta$
 1: Initialize parameters $\theta$ randomly
 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:         Sample support set $\mathcal{D}_S$ and query set $\mathcal{D}_Q$
 6:         Compute adapted parameters with gradient descent:
 7:         $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta; \mathcal{D}_S)$
 8:     end for
 9:     Update $\theta$ using query sets:
10:     $\theta \leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}; \mathcal{D}_Q)$
11: end while
```

5.3 Mathematics of the Meta-Update

A key theoretical insight explored in the mid-term study is the presence of second-order derivatives. To minimize the meta-loss, we differentiate through the inner update. Using the chain rule on $\mathcal{L}(f_{\theta'_i})$:

$$\nabla_\theta \mathcal{L}(f_{\theta'_i}) = \nabla_{\theta'_i} \mathcal{L}(f_{\theta'_i}) \cdot \nabla_\theta(\theta - \alpha \nabla_\theta \mathcal{L}_S) \tag{8}$$

This expands to:

$$\nabla_\theta \mathcal{L}(f_{\theta'_i}) = \nabla_{\theta'_i} \mathcal{L}(f_{\theta'_i}) \cdot (I - \alpha \nabla^2_\theta \mathcal{L}_S) \tag{9}$$

The term $\nabla^2_\theta \mathcal{L}_S$ is the Hessian matrix. This confirms that MAML optimizes the initialization by explicitly accounting for the curvature of the loss landscape.

5.4 First-Order Approximation (FOMAML)

Calculating the Hessian is computationally expensive for deep networks. We reviewed the First-Order MAML approximation, which assumes $(I - \alpha\nabla^2) \approx I$. This simplifies the update and, surprisingly, often yields comparable performance, suggesting that the gradient direction at the adapted point is often sufficient for meta-learning.

6 Comparison with Metric-Based Methods

To contextualize MAML, we briefly studied **Metric-Based** few-shot learning, specifically **Siamese Networks**.

Siamese networks utilize a contrastive loss to pull same-class examples together and push different-class examples apart:

$$L = \sum \frac{1}{2}(1 - Y)D^2 + \frac{1}{2}Y\{\max(0, m - D)\}^2 \tag{10}$$

While effective for classification, metric-based methods lack the versatility of MAML, which can be applied to Reinforcement Learning and Regression tasks seamlessly.

Optimization-Based (MAML)	**Metric-Based (Siamese)**
Learns an initialization θ optimized for fast fine-tuning.	Learns an embedding space for similarity-based inference.
Requires task-time adaptation (gradient steps).	Often no gradient updates at test time (Nearest Neighbor).
Model-agnostic (works for RL/Regression).	Typically tailored to classification/retrieval.
Computationally heavier (bi-level gradients).	Inference is lightweight.

Table 1: Comparison of Meta-Learning Approaches

7 Experimental Evaluation

7.1 Sinusoid Regression

We replicated the canonical MAML experiment: regressing sine waves $y = A\sin(x + \phi)$ where amplitude A and phase ϕ vary per task.

- **Setup:** Each task has $K = 5$ points.
- **Baseline:** A standard neural network pre-trained on all tasks simply learns the "average" sine wave (a flat line at $y = 0$) because it cannot adapt to the phase shift.
- **MAML:** The MAML-trained model learns a frequency representation such that seeing just 5 points allows it to snap the phase and amplitude into place almost instantly.

7.2 Implementation Tools

The implementation was conducted using Python and the PyTorch framework.

- **NumPy:** Used for data manipulation and creating the sinusoid datasets.
- **PyTorch:** Used for building the computational graphs. We utilized ‘torch.autograd‘ to handle the higher-order derivatives required for the outer loop update.
- **Matplotlib:** Used to visualize the "pre-adaptation" vs. "post-adaptation" curves.

8 Conclusion and Future Work

At this mid-term stage, the project has established a clear conceptual pipeline:

1. **Foundations:** We have verified that learning is an optimization process and that geometry (initialization) dictates training dynamics.
2. **Deep Learning:** We have implemented the architectures necessary for high-dimensional representation learning.

3. **Meta-Learning:** We have successfully parsed the MAML algorithm, understanding it not as magic, but as a bi-level optimization problem that trades immediate performance for *adaptability.*

The MAML framework provides a clean formalization of "learning to learn." By optimizing the initialization such that the gradient vector points toward the task solution, MAML bridges the gap between the data-hungry nature of deep learning and the few-shot capabilities of human intelligence.