An Example for Model-Agnostic Meta-Learning Algorithm

1 Example: Single Neuron Neural Network

We use a single-neuron neural network as a simple example to illustrate the Model-Agnostic Meta-Learning (MAML) algorithm [FAL17]. The goal of MAML is to learn a good initialisation of w, the meta model's parameters. We define a dataset and aim to train the model to obtain a good initialisation that can quickly adapt to a new task using only few data samples and updates. This setting is also formalised as a few-shot learning problem.

Suppose that we have three datasets $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_{new}$. $\mathcal{D}_1$ and $\mathcal{D}_2$ will be used for training the meta model and $\mathcal{D}_{new}$, with only a few samples, will be used for testing the meta model. In the general case, the data points from $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_{new}$ can be drawn from different distributions.

Definition 1.1 (Dataset)**.** *Let $\mathcal{D}_j = \{S_j, Q_j\}$ be the j-th dataset, where the support (S_j) and query (Q_j) set are two subsets of $\mathcal{D}_j$. $S_j, Q_j \subset \mathcal{D}_j$ and $S_j \cap Q_j = \emptyset$.*

S_j and Q_j are used for task-specific (conventional training) update and meta update respectively. The loss over each sets are $\mathcal{L}_{S_j}$ and $\mathcal{L}_{Q_j}$, respectively.

Definition 1.2 (Single Neuron Model)**.** *Let $f_\theta : \mathbb{R} \to \mathbb{R}$ be a single neuron neural network model parameterised by $\theta \in \mathbb{R}$ and defined as,*

$$f_\theta(x) = \theta x$$

We build a meta-learning framework for a single neuron neural network model, as shown in Figure 1.

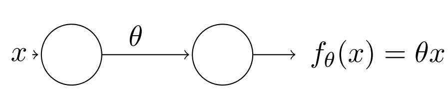

Figure 1: Single processing unit and its components.

The model is denoted by f_θ and applied on the input x of the unit to form its output $f_\theta(x) = \theta x$, where θ represents the weight of unit.

The single neuron neural network model computes a function $\hat{y}_i = f_\theta(x_i)$, where $\hat{y}_i$ is the predicted value of input sample x_i, and θ represents the model weights.

1.1 Conventional Supervised Learning

We first recap the process of conventional supervised learning. In supervised learning, a loss function (e.g., mean squared error (MSE)) measures the difference between the target values y_i and predicted output values $\hat{y}_i$ produced by the network model. In a simple case, the learning (i.e., model training) procedure aims at finding the best value of w that minimises the loss to its lowest value.

For regression tasks using MSE, given a general dataset, $\mathcal{D}$, the loss takes the form:

$$\mathcal{L}(f_w(x), y) = \sum_{(x_i, y_i) \in D} (y_i - w x_i)^2 \tag{1}$$

and the gradient of loss function is calculated as,

$$\frac{\partial \mathcal{L}(f_w)}{\partial w} = -2 \sum_{(x_i, y_i) \in D} x_i (y_i - w x_i) \tag{2}$$

We perform one-step update using an optimisation algorithm, such as Stochastic Gradient Descent (SGD):

$$\begin{aligned} w^{(1)}(w^{(0)}) &= w^{(0)} - \alpha \left. \frac{\partial \mathcal{L}(f_w)}{\partial w} \right|_{w = w^{(0)}} \\ w^{(1)}(w^{(0)}) &= w^{(0)} + 2 \sum_{(x_i, y_i) \in D} x_i (y_i - w^{(0)} x_i) \end{aligned} \tag{3}$$

Remark 1: Where $w^{(0)}$ is the initialisation weight and α is the learning rate. The updated weight is a function of initialisation weight, $w^{(1)}(w^{(0)})$ (in Equation 3).

1.1.1 Numerical example

We define a dataset $\hat{\mathcal{D}}$ with 3 data points:

$\hat{\mathcal{D}}$	x	y	f_w
	1	2	1
	2	4	2
	3	1	3

Table 1: Dataset $\hat{\mathcal{D}}$

By initializing $w^{(0)} = 1$ and $\alpha = 0.1$, we perform one step update new weights w with equation 3 using the dataset $\hat{\mathcal{D}}$ in Table 1.

$$\begin{aligned} w^{(1)}(w^{(0)}) &= w^{(0)} + 2\alpha \sum_{i=1}^{3} x_i(y_i - w^{(0)}x_i) \\ w^{(1)}(1) &= 1 + 2 \times 0.1[1(2-1) + 2(4-2) + 3(1-3)] \\ &= 1 - 0.2 \\ &= 0.8 \end{aligned} \tag{4}$$

1.2 Meta Learning

MAML consists of deriving task-specific weights (via conventional training), followed by a meta update. We update θ during the meta-update through aggregating each task-specific weights φ_j. Figure 2 shows the workflow of MAML for two tasks.

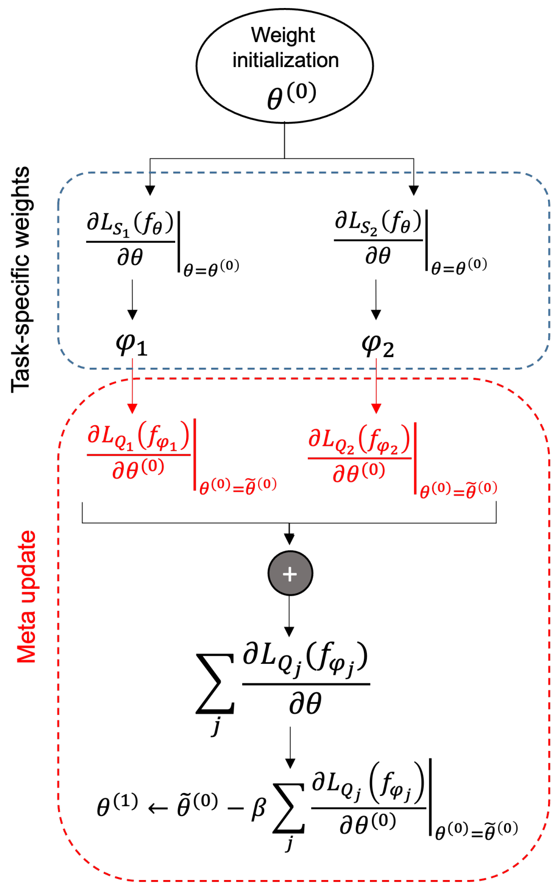

Figure 2: The workflow of key formulas used in MAML for two tasks.

1.2.1 Task-specific training

The task-specific training is conventional training on each j-th datasets $\mathcal{D}_j$, using data points from the support set S_j, except that we compute the task-specific model's weights φ_j instead of updating θ. In our example, φ_j is computed using one gradient update. We can rewrite the loss function and its gradients as:

$$\mathcal{L}_{\mathcal{S}_j}(f_\theta(x), y) = \sum_{(x_i^j, y_i^j) \in S_j} (y_i^j - \theta x_i^j)^2 \tag{5}$$

$$\frac{\partial \mathcal{L}_{\mathcal{S}_j}(f_\theta)}{\partial \theta} = -2 \sum_{(x_i^j, y_i^j) \in S_j} x_i^j (y_i^j - \theta x_i^j) \tag{6}$$

The task-specific weight with one gradient step is expressed as a function of initialization weight $\theta^{(0)}$ as:

$$\begin{aligned} \varphi_j(\theta^{(0)}) &= \theta^{(0)} - \alpha \frac{\partial \mathcal{L}_{S_j}(f_\theta)}{\partial \theta}\bigg|_{\theta=\theta^{(0)}} \\ &= \theta^{(0)} + 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} x_i^j (y_i^j - \theta^{(0)} x_i^j) \end{aligned} \tag{7}$$

where α is the task-specific learning rate and $\mathcal{L}_{S_j}$ is the loss on the support set of dataset $\mathcal{D}_j$.
Remark 2: For simplicity and readability of this report, we replace $\varphi_j(\theta^{(0)})$ with φ_j in the following sections.

1.2.2 Meta training

For meta training, the model f_θ is trained by optimising for the performance of f_{φ_j} with respect to θ *across all tasks*. Here,

$$f_{\varphi_j} = \varphi_j x \tag{8}$$

The meta-loss is calculated on the query set Q_j as the sum of task-specific losses after task-specific weight updates:

$$\sum_j \mathcal{L}_{Q_j}(f_{\varphi_j}) \tag{9}$$

and:

$$\mathcal{L}_{Q_j}(f_{\varphi_j}) = \sum_{(x_q^j, y_q^j) \in Q_j} (y_q^j - \varphi_j x_q^j)^2 \tag{10}$$

are the task-specific model and the MSE loss over query set Q_j.

The meta-optimisation across tasks is performed using SGD. The first step meta update are as follows:

$$
\begin{aligned}
\theta^{(1)} &= \tilde{\theta}^{(0)} - \beta \frac{\partial}{\partial \theta^{(0)}} \sum_j \mathcal{L}_{Q_j}(f_{\varphi_j}) \Big|_{\theta^{(0)}=\tilde{\theta}^{(0)}} \\
&= \tilde{\theta}^{(0)} - \beta \sum_j \underbrace{\frac{\partial \mathcal{L}_{Q_j}(f_{\varphi_j})}{\partial \theta^{(0)}}}_{*} \Big|_{\theta^{(0)}=\tilde{\theta}^{(0)}}
\end{aligned} \tag{11}
$$

where β is the meta learning rate and $\tilde{\theta}^{(0)}$ is the initialization constant. The part of Eq. 11 marked with $*$ can be calculated using the chain rule:

$$
\frac{\partial \mathcal{L}_{Q_j}(f_{\varphi_j})}{\partial \theta^{(0)}} = \underbrace{\frac{\partial \varphi_j}{\partial \theta^{(0)}}}_{\dagger} \underbrace{\frac{\partial \mathcal{L}_{Q_j}(f_{\varphi_j})}{\partial \varphi_j}}_{\ddagger} \tag{12}
$$

Then, the gradient marked with ($\ddagger$) is expressed by

$$
\frac{\partial \mathcal{L}_{Q_j}(f_{\varphi_j})}{\partial \varphi_j} = -2 \sum_{(x_q^j, y_q^j) \in Q_j} x_q^j \left(y_q^j - \varphi_j x_q^j \right) \tag{13}
$$

Using Eq. 7, the part of equation marked with $\dagger$ is calculated as follows:

$$
\begin{aligned}
\frac{\partial \varphi_j}{\partial \theta^{(0)}} &= \frac{\partial \theta^{(0)}}{\partial \theta^{(0)}} + 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \frac{\partial}{\partial \theta^{(0)}} \; x_i^j (y_i^j - \theta^{(0)} x_i^j) \\
&= 1 - 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \left(x_i^j \right)^2
\end{aligned} \tag{14}
$$

Using Eq. 14 and 13, the Eq. 12 becomes as follows:

$$
\frac{\partial \mathcal{L}_{Q_j}(f_{\varphi_j})}{\partial \theta^{(0)}} = \left(1 - 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \left(x_i^j \right)^2 \right) \left(-2 \sum_{(x_q^j, y_q^j) \in Q_j} x_q^j (y_q^j - \varphi_j x_q^j) \right) \tag{15}
$$

Remark 3: Eq. 14 is computed using the support set, since φ_j is derived using the support set.

1.3 Numerical example walkthrough

Here, we put our formulas derived using a numerical example.

Table 1.3 shows the data points from $\mathcal{D}_1$ and $\mathcal{D}_2$.

$\mathcal{D}_1$	x	y	f_θ
Q_1	1	2	-
S_1	2	4	2
	3	1	3

(a) Dataset $\mathcal{D}_1$

$\mathcal{D}_2$	x	y	f_θ
Q_2	4	1	-
S_2	5	3	5
	6	0	6

(b) Dataset $\mathcal{D}_2$

Table 2: Two dataset $\mathcal{D}_1$ and $\mathcal{D}_2$.

In our defined datasets, $\mathcal{D}_1$ consists of support set $S_1 = \{(2,4),(3,1)\}$ and query set $Q_1 = \{(1,2)\}$. Likewise, $S_2 = \{(5,3),(6,0)\}$ and $Q_2 = \{(4,1)\}$.

By initializing $\tilde{\theta}^{(0)} = 1$ and $\alpha = 0.1$, we can substitute data points (x_i^1, y_i^1) in S_1 and (x_i^2, y_i^2) in S_2 into equation 7, and compute task-specific weights for $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively:

$$\begin{aligned}
\varphi_1(\tilde{\theta}^{(0)}) &= \tilde{\theta}^{(0)} + 2\alpha \sum_{(x_i^1, y_i^1)\in S_1} x_i^1(y_i^1 - \tilde{\theta}^{(0)} x_i^1) \\
\varphi_1(1) &= 1 + 2 * 0.1(2(4-2) + 3(1-3)) \\
&= 1 - 0.4 \\
&= 0.6
\end{aligned} \tag{16}$$

$$\begin{aligned}
\varphi_2(\tilde{\theta}^{(0)}) &= \tilde{\theta}^{(0)} + 2\alpha \sum_{(x_i^2, y_i^2)\in S_2} x_i^2(y_i^2 - \tilde{\theta}^{(0)} x_i^2) \\
\varphi_2(1) &= 1 + 2 * 0.1(5(3-5) + 6(0-6)) \\
&= 1 - 9.2 \\
&= -8.2
\end{aligned} \tag{17}$$

In our regression problem, the Eq. 15 can be solved by replacing the data points in Q_1 and Q_2 of each tasks as follows:

$$\left.\frac{\partial \mathcal{L}_{Q_1}(f_{\varphi_1})}{\partial \theta^{(0)}}\right|_{\varphi_1 = 0.6} = (1 - 2(0.1)(2^2 + 3^2)) \cdot (-2(1)(2 - 1(0.6))) = 4.48 \tag{18}$$

$$\left.\frac{\partial \mathcal{L}_{Q_2}(f_{\varphi_2})}{\partial \theta^{(0)}}\right|_{\varphi_2 = -8.2} = (1 - 2(0.1)(5^2 + 6^2)) \cdot (-2(4)(1 - 4(-8.2))) = 3028.48 \tag{19}$$

By setting $\beta = 0.5$ and $\tilde{\theta}^{(0)} = 1$, the new weight of model according to Eq. 11 becomes:

$$\begin{aligned}
\theta^{(1)} &= \tilde{\theta}^{(0)} - \beta \left(\left.\frac{\partial \mathcal{L}_{Q_1}(f_{\varphi_1})}{\partial \theta^{(0)}}\right|_{\varphi_1 = 0.6} + \left.\frac{\partial \mathcal{L}_{Q_2}(f_{\varphi_2})}{\partial \theta^{(0)}}\right|_{\varphi_2 = -8.2} \right) \\
\theta^{(1)} &= 1 - 0.5(4.48 + 3028.48) \\
&= -1515.48
\end{aligned} \tag{20}$$

The meta-update calculation gives us a large value of -295.32. For our simple problem statement, we can accept this result, since the data $\mathcal{D}_1$ and $\mathcal{D}_2$ are selected randomly and has no underlying assumption of the data following any distribution. For future verification, we may choose to select data points that follow a distribution.

2 General Formulation for Few Gradient Steps

In this section, we consider the case of performing k task-specific gradient steps, $k \geq 1$. Starting with the initial model weight $\theta^{(0)}$, the equation 7 is refined as:

$$\begin{aligned}
\varphi_j^{(1)} &= \theta^{(0)} - \alpha \frac{\partial \mathcal{L}_{S_j}(f_\theta)}{\partial \theta}\bigg|_{\theta=\theta^{(0)}} \\
\varphi_j^{(2)}(\varphi_j^{(1)}) &= \varphi_j^{(1)} - \alpha \frac{\partial \mathcal{L}_{S_j}(f_\theta)}{\partial \theta}\bigg|_{\theta=\varphi_j^{(1)}} \\
&\vdots \\
\varphi_j^{(k)}(\varphi_j^{(k-1)}) &= \varphi_j^{(k-1)} - \alpha \frac{\partial \mathcal{L}_{S_j}(f_\theta)}{\partial \theta}\bigg|_{\theta=\varphi_j^{(k-1)}}
\end{aligned} \tag{21}$$

where k denotes the iteration number and $\varphi_j^{(k)}$ is the weight of task j updated after k gradient steps. We can also express Equation 21 recursively as a composite function:

$$W_j^{(k)}(\theta^{(0)}) = \varphi_j^{(k)} \circ \varphi_j^{(k-1)} \circ \ldots \circ \varphi_j^{(2)} \circ \varphi_j^{(1)}(\theta^{(0)}) \tag{22}$$

Then in the meta-learning, we need to update the meta-learning model weights.

2.1 Meta Training

In the meta training phase, the model f_θ (see Definition 1.2) is trained by optimising the kth updated of task-specific weights, $W_j^{(k)}(\theta^{(0)})$ obtained from equation 21. Therefore, the equations 8 and 10 are changed accordingly:

$$f_{W_j^{(k)}(\theta^{(0)})} = W_j^{(k)}(\theta^{(0)})x \tag{23}$$

The meta-loss is calculated on the query set Q_j as the sum of task-specific losses after kth task-specific weight updates:

$$\mathcal{L}_{Q_j}(f_{W_j^{(k)}(\theta^{(0)})}) = \sum_{(x_q^j, y_q^j) \in Q_j} (y_q^j - W_j^{(k)}(\theta^{(0)})x_q^j)^2 \tag{24}$$

In order to update the meta weights based on the task-specific weights, the equation 11 is refined as follows:

$$
\begin{aligned}
\theta^{(1)} &= \tilde{\theta}^{(0)} - \beta \frac{\partial}{\partial \theta^{(0)}} \sum_j \mathcal{L}_{Q_j}\left(f_{W_j^{(k)}(\theta^{(0)})}\right)\Bigg|_{\theta^{(0)}=\tilde{\theta}^{(0)}} \\
&= \tilde{\theta}^{(0)} - \beta \sum_j \underbrace{\frac{\partial \mathcal{L}_{Q_j}\left(f_{W_j^{(k)}(\theta^{(0)})}\right)}{\partial \theta^{(0)}}}_{*}\Bigg|_{\theta^{(0)}=\tilde{\theta}^{(0)}}
\end{aligned}
\tag{25}
$$

The equations 12, 13, 14 are changed as follows:

$$
\frac{\partial \mathcal{L}_{Q_j}\left(f_{W_j^{(k)}(\theta^{(0)})}\right)}{\partial \theta^{(0)}} = \underbrace{\frac{\partial W_j^{(k)}(\theta^{(0)})}{\partial \theta^{(0)}}}_{\dagger} \underbrace{\frac{\partial \mathcal{L}_{Q_j}\left(f_{W_j^{(k)}(\theta^{(0)})}\right)}{\partial W_j^{(k)}(\theta^{(0)})}}_{\ddagger}
\tag{26}
$$

Then, the gradient marked with (‡) is expressed by

$$
\frac{\partial \mathcal{L}_{Q_j}\left(f_{W_j^{(k)}(\theta^{(0)})}\right)}{\partial W_j^{(k)}(\theta^{(0)})} = -2 \sum_{(x_q^j, y_q^j) \in Q_j} x_q^j \left(y_q^j - W_j^{(k)}(\theta^{(0)}) x_q^j\right)
\tag{27}
$$

Using Eq. 23, the part of equation marked with † can be broken down further using the chain rule until we get a derivative wrt $\theta^{(0)}$:

$$
\begin{aligned}
\frac{\partial W_j^{(k)}(\theta^{(0)})}{\partial \theta^{(0)}} &= \frac{\partial \varphi_j^{(k)}(\varphi_j^{(k-1)})}{\partial \varphi_j^{(k-1)}} \cdot \frac{\partial \varphi_j^{(k-1)}(\varphi_j^{(k-2)})}{\partial \varphi_j^{(k-2)}} \cdot \ldots \cdot \underbrace{\frac{\partial \varphi_j^{(1)}}{\partial \theta^{(0)}}}_{\diamond} \\
&= \prod_{\tau=1}^{k} \left(\frac{\partial \varphi_j^{(\tau)}(\varphi_j^{(\tau-1)})}{\partial \varphi_j^{(\tau-1)}} \right) \\
&= \left(1 - 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \left(x_i^j\right)^2 \right)^k
\end{aligned}
\tag{28}
$$

We expand $\diamond$ below:

$$
\begin{aligned}
\frac{\partial \varphi_j^{(1)}}{\partial \theta^{(0)}} &= \frac{\partial \theta^{(0)}}{\partial \theta^{(0)}} + 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \frac{\partial}{\partial \theta^{(0)}} \, x_i^j (y_i^j - \theta^{(0)} x_i^j) \\
&= 1 - 2\alpha \sum_{(x_i^j, y_i^j) \in S_j} \left(x_i^j\right)^2
\end{aligned}
\tag{29}
$$

Appendix I: Glossary

Table 3: Notations for MAML simple example.

Indices:	
i	Sample index in support set
q	Sample index in query set
j	Task index
k	Task specific updates
τ	Task specific update index
Data and Dataset:	
$\mathcal{D}$	All data space
$\mathcal{D}_j$	Dataset of task j
Q_j	Query set of task j
S_j	Support set of task j
(x_i^j, y_i^j)	i-th input and target of j-th task in support set
(x_q^j, y_q^j)	q-th input and target of j-th task in query set
$\mathcal{D}_{new}$	Dataset of new task with a few samples
Function:	
f_w	Single neuron model parameterised by w
f_θ	Single neuron model parameterised by θ
Variables:	
w	Conventional SL model weights
θ	MAML model weights
$\theta^{(0)}$	MAML Initialisation variable
φ_j	MAML task-specific weights for j-th dataset
Constants:	
$w^{(0)}$	Conventional SL Initialisation weight
$\tilde{\theta}^{(0)}$	MAML Initialisation weight
α	Task-specific learning rate
β	Meta model learning rate

Appendix II: Numerical Walkthrough for 2 task-specific gradient steps

This is the extension of work from the main section. The main purpose of this pen-and-paper exercise is to verify that the outputs from our code implementation is performing the correct computation. Following the example from Eq. 16 and 17, we derive the 2nd step gradient update as:

$$
\begin{aligned}
\varphi_1^{(2)}(\varphi_1^{(1)}) &= \varphi_1^{(1)} + 2\alpha \sum_{(x_i^1, y_i^1) \in S_1} x_i^1 (y_i^1 - \varphi_1^{(1)} x_i^1) \\
\varphi_1^{(2)}(0.6) &= 0.6 + 2 * 0.1(2(4 - 0.6(2)) + 3(1 - 0.6(3))) \\
&= 0.6 + 0.6399 \\
&= 1.24
\end{aligned} \tag{30}
$$

and

$$
\begin{aligned}
\varphi_2^{(2)}(\varphi_2^{(1)}) &= \varphi_2^{(1)} + 2\alpha \sum_{(x_i^2, y_i^2) \in S_2} x_i^2 (y_i^2 - \varphi_2^{(1)} x_i^2) \\
\varphi_2^{(2)}(-8.19) &= -8.19 + 2 * 0.1(5(3 - (-8.19)(5)) + 6(0 - (-8.19)(6))) \\
&= (-8.19) + 103.04 \\
&= 94.83
\end{aligned} \tag{31}
$$

following Eq. 21.

We can then proceed to compute the loss over query, and its gradients, using $\varphi_1^{(2)}$ and $\varphi_2^{(2)}$ respectively:

$$
\left.\frac{\partial \mathcal{L}_{Q_1}(f_{\varphi_1^{(2)}})}{\partial \theta^{(0)}}\right|_{\varphi_1^{(2)}=1.24} = (1 - 2(0.1)(2^2 + 3^2))^2 \cdot (-2(1)(2 - 1(1.24))) = -3.8912 \tag{32}
$$

$$
\left.\frac{\partial \mathcal{L}_{Q_2}(f_{\varphi_2^{(2)}})}{\partial \theta^{(0)}}\right|_{\varphi_2^{(2)}=94.83} = (1 - 2(0.1)(5^2 + 6^2))^2 \cdot (-2(4)(1 - 4(94.83))) = 379651.69 \tag{33}
$$

Then, the new meta update becomes:

$$
\begin{aligned}
\theta^{(1)} &= \tilde{\theta}^{(0)} - \beta \left(\left.\frac{\partial \mathcal{L}_{Q_1}(f_{\varphi_1^{(2)}})}{\partial \theta^{(0)}}\right|_{\varphi_1^{(2)}=1.24} + \left.\frac{\partial \mathcal{L}_{Q_2}(f_{\varphi_2^{(2)}})}{\partial \theta^{(0)}}\right|_{\varphi_2^{(2)}=-94.83} \right) \\
\theta^{(1)} &= 1 - 0.5(-3.8912 + 379651.69) \\
&= -189822.90
\end{aligned} \tag{34}
$$

References

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.