INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

COURSE PROJECT REPORT

# Bayesian Meta Learning

*Author:*
Aakash LAHOTI
Asad KARIM
Pardhu CHOWDARY
Raghukul RAMAN
Sudhanshu BANSAL
Vibhor PORWAL

*Supervisor:*
Dr. Piyush RAI

*A report submitted in fulfillment of the requirements*
*for Course Project (CS771)*

Department of Computer Science and Engineering

## Abstract

Our project is on..

### Subsection (if any) Name

....


## 1 Literature Review

MAML has been designed to enable fast adaptation to unseen tasks by training on statistically related tasks. The algorithm is Model-Agnostic which means that it can be used on any model trained through gradient-descent.

### 1.1 Problem Formulation

The paper presents a generic formulation applicable to tasks like regression, classification and reinforcement learning. Let $f$ denote the common model for all tasks with $\theta$ being the central parameter to be learned. It maps input $x$ to output $a$. Each Task is denoted by

$$\mathcal{T} : \{L(x_1, a_1, ..., x_H, a_H), q(x_1), q(x_{t+1}|x_t, a_t), H\}$$

here L is the task specific loss function, $q(x_1)$ is the distribution of inputs, $q(x_{t+1}|x_t, a_t)$ is the transition probability for all states , and episode length is H.
H is specifically for non-iid data, for example the unfolding of reinforcement learning training sample. For the supervised iid classification and regression experiments, H=1.

### 1.2 Algorithm

$p(\mathcal{T})$ is the distribution over tasks over which we would train and test the meta-learning model. We will follow a K-Shot learning setting. First, a batch of tasks is sampled from $p(\mathcal{T})$ and for each sample task $\mathcal{T}_i$, $K$ inputs are sampled from $q_i$. These sampled inputs are used to learn task specific $\theta_i$s which were initialized to the central parameter $\theta$. Then, we sample $K$ more inputs to evaluate that particular task with the $\theta_i$s. The sum of the loss incurred in all the tasks is used as the meta-loss function to find the optimal $\theta$.

---

**Input:** $p(\mathcal{T})$ : Distribution over tasks, $\alpha, \beta$ : Step size hyper parameters
**Output:** Parameters for this model
randomly initialize $\theta$
**while** *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \approx p(\mathcal{T})$
  **for** *all* $\mathcal{T}_i$ **do**
    Sample $K$ data points $D = \{x^{(j)}, y^{(j)}\}$ from $\mathcal{T}_i$
    Compute $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $K$ samples
    Compute adapted parameters with gradient descent:
    $\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample data points $D_i^{'} = \{x^{(j)}, y^{(j)}\}$ from $\mathcal{T}_i$ for the meta update
  **end**
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \approx p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_(\theta_i^{'}))$ using each $D_i^{'}$ and $\mathcal{L}_{\mathcal{T}_i}$.
**end**
**return** $\theta$

---

Formally, for each $\mathcal{T}_i$, we calculate $\theta_i'$ using the $K$ samples by the following gradient descent update.

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

Here $\alpha$ is the step size and is a hyperparameter for the model. The meta objective is minimizing the sum of loss incurred in testing the sampled tasks against $D_i$s.

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

Finally, we perform the meta optimization step using gradient descent for the central parameter $\theta$ on the loss function defined above.

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

### 1.3 Algorithm for supervised learning (Regression and classification)

To formalize in the context of meta learning, the loss function used for regression is the mean squared error loss as shown below

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} ||f_\phi(x^{(j)}) - y^{(j)}||_2^2$$

where $\phi$ is the model parameter. For classification, the loss function is given as

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} y^{(j)} \log f_\phi(x^{(j)}) + (1 - y^{(j)}) \log(1 - f_\phi(x^{(j)}))$$

## Algorithm Review

....

## Bayesian ML

....

## Reinforcement Learning

....

# 2 Experiments

Most of the experiments carried out required automated differentiation through gradient update, for which we used TensorFlow. A signification computation was required in finding the second derivative, in the meta gradient update step, which involved backpropogating the meta-gradient through gradient calculation in meta gradient objective ($\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \approx p(T)} L_{T_i}(f_{\theta_i'})$).

We have tested the MAML algorithm for Regression and Classfication problem. Now let's discuss each of these experiment in details.

## 2.1 Regression

We tested our model on the task of regressing from the input to output of a sine wave, where each task had a different amplitude and phase, where amplitude varies within $[0.1, 5.0]$ and phase varies within $[0, \pi]$. Note that input and output in our problem are just $1$ dimensional. Datapoints are sampled uniformly from $[-5.0, 5.0]$ and the loss function is simply the mean sqaured error between prediction and the ground truth.

### 2.1.1 Architecture

Our regressor is a neural network model which has $2$ hidden layers of $40$ hidden units, and each each containing activation function as ReLU.
While training hyperparameter $\alpha = 0.01$, and our meta optimizer is Adam [4].

....

### Classification

....

## Conclusion

....

### Subsection (if any) Name

....

## Future Work

....

### Subsection (if any) Name

....

#### Acknowledgments

....

# References

[1] Chelsea Finn, Pieter Abbeel and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.* 2017.

[2] Taesup Kim, Jaesik Yoon, Ousmane Dia1, Sungwoong Kim, Yoshua Bengio and Sungjin Ahn. *Bayesian Model-Agnostic Meta-Learning.* Jun 2018.

[3] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, Thomas Griffiths. *Recasting Gradient-based Meta-Learning as hierarchical Bayes.* Jan 2018.

[4] Maclaurin, Dougal, Duvenaud, David, and Adams, Ryan. *Gradient-based hyperparameter optimization through reversible learning.* In International Conference on Machine Learning (ICML), 2015.