INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

COURSE PROJECT REPORT

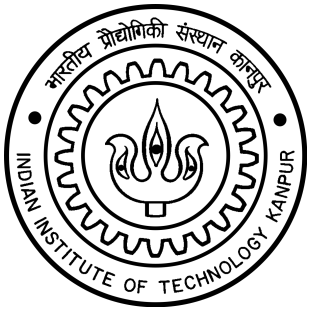
Bayesian Meta Learning

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



1 Abstract

Machine learning algorithms generally require a lot of data to perform significantly well on any task.In contrast humans have this inherent ability to learn any task with limited experience.For example even a kid can recognize an object by looking at it only once.This is possible due to abilities such as shape bias. We will present a machine learning algorithm which tries to accomplish this task.
This domain of machine learning is known as meta-learning.The word "Meta" is used to indicate some concept about a concept . So meta-learning is basically learning about learning.Our goal is to train a model that has the ability to learn a new task with little amount of training.We will show that our approach works well on variety of supervised learning problems such as sinusoidal curve fitting(regression) and classification of hand written letters on omniglot dataset.

2 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.

2.1 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 θ 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.

2.2 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 θ_is which were initialized to the central parameter θ. Then, we sample *K* more inputs to evaluate that particular task with the θ_is. The sum of the loss incurred in all the tasks is used as the meta-loss function to find the optimal θ.

Input: $p(\mathcal{T})$: Distribution over tasks, α, β : Step size hyper parameters
Output: Parameters for this model
randomly initialize θ
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 θ

Formally, for each $\mathcal{T}_i$, we calculate θ_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 α 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_is.

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

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

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

2.3 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 ϕ 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

....

3 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.

3.1 Regression

3.1.1 Data Set

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.

3.1.2 Architecture

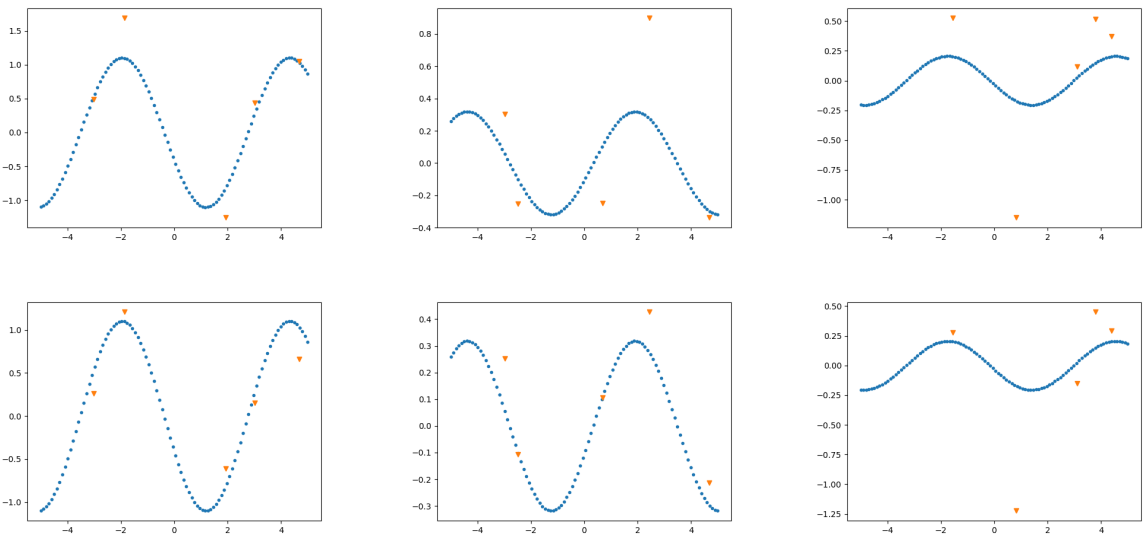
Our regressor is a neural network model which has 2 hidden layers of 40 hidden units, and each each containing activation function as ReLU.
While training hyperparameter $\alpha = 0.01$, and our meta optimizer is Adam [4].

3.1.3 Experiment Protocol

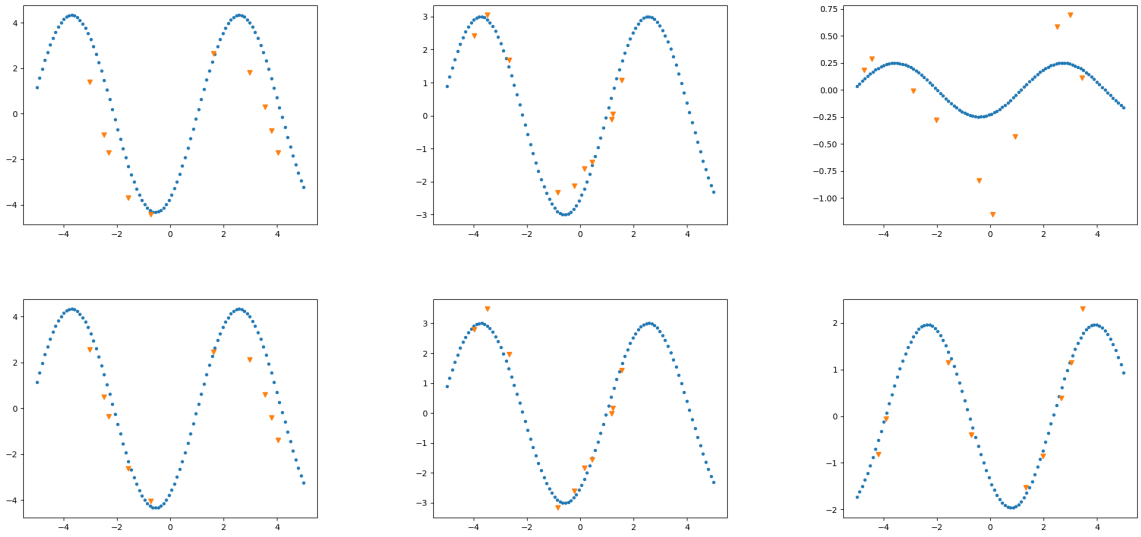
We performed K shot learning for $K = 5, 10, 20$.

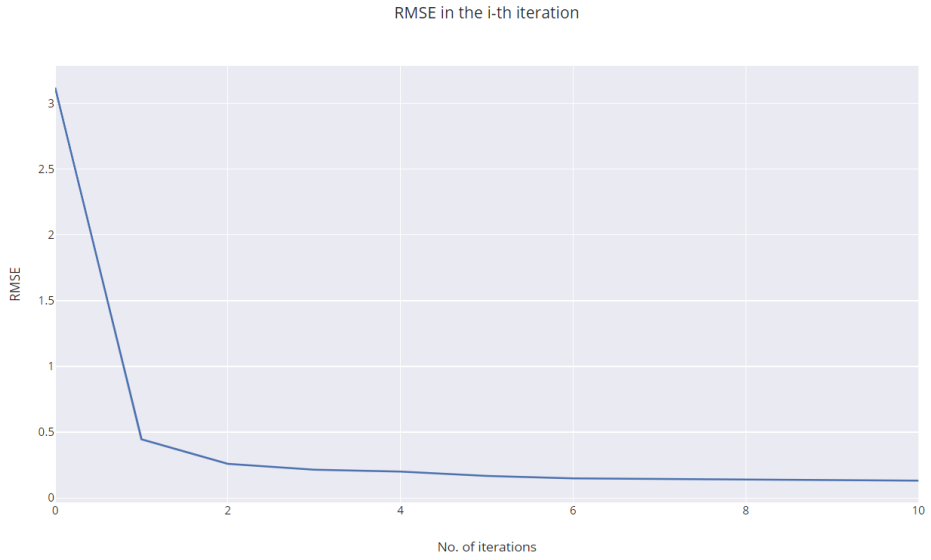
3.1.4 Results

Results for $K = 5$



Results for $K = 10$





3.2 Classification

3.2.1 Data Set

3.2.2 Architecture

3.2.3 Experiment Protocol

3.2.4 Results

5 **Shot Learning**

Update	Update 0	Update 1	Update 2	Update 3	Update 4	update 5	Update 6	Update 7	Update 8	Update 9	Update 10
Accuracy	19.1	96.9	97.2	97.2	97.2	97.2	97.2	97.2	97.3	97.3	97.3
Standard Deviation	0.151	0.083	0.081	0.081	0.081	0.081	0.081	0.081	0.081	0.081	0.082
Confidence Interval	0.012	0.0052	0.0054	0.0059	0.0059	0.0058	0.0059	0.0059	0.0059	0.0058	0.0058

20 **Shot Learning**

Update	Update 0	Update 1	Update 2	Update 3	Update 4	update 5	Update 6	Update 7	Update 8	Update 9	Update 10
Accuracy	4.8	83.9	87.2	88.3	88.4	88.5	88.5	88.6	88.6	88.7	88.7
Standard Deviation	0.11	0.08	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
Confidence Interval	0.006	0.006	0.006	0.0059	0.0059	0.0058	0.0059096827	0.0059	0.0059	0.0058	0.0058

4 Conclusion and Future Work

We found that the MAML performs reasonably well on sinusoidal curve fitting and also on the ominiglot dataset .So from this we infer that it is a pretty good method for gradient based models.So we can use it for various other problems also.
The model provided in the paper is simple as it works on any gradient based model.One of the possibilities could be to generalize the model we studied to other variety of problems.We tested this method using neural networks , it would be great if we could try the method on other type of gradient based models like Kernelised ridge regression, SVMs etc. More work could be done so that multi-task initialization becomes a standard step in reinforcement learning and deep learning. Currently the method does not scale appreciably on reinforcement learning problems.So we should focus on this area also.

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.