Alpha MAML: Adaptive Model-Agnostic Meta-Learning

Harkirat Singh Behl, Atilim Güneş Baydin, Philip H.S Torr
Department of Engineering Science, University of Oxford
{harkirat,gunes,phst}@robots.ox.ac.uk

UNIVERSITY OF OXFORD

Introduction

- **Shortcomings of MAML [1]:**
  - training very sensitive to learning rate values [2].
  - requires time-consuming hyperparameter tuning.
  - high training time.
- **Aim:**
  - make it possible to use MAML without or with significantly less parameter tuning.
  - make the algorithm converge in fewer iterations.
- **Contributions:**
  - an online hyperparameter adaptation scheme that eliminates the need to tune meta-learning and learning rates.
  - requires no extra gradient computations, only memory cost of storing previous gradient.
  - based on the hypergradient descent (HD) algorithm [3].

MAML: Model-Agnostic Meta-Learning

The MAML algorithm, given model parameters θ, aims to adapt to a new task $\mathcal{T}_t$ with SGD:

$$\theta'_t = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_{train(t)}}(f_\theta) \ , \quad (1)$$

where t is the task number, α is the learning rate, $T_{train(t)}$ and $T_{test(t)}$ denote the training and test set within task t. The meta-update equation is:

$$\theta_t = \theta - \beta \nabla_\theta \mathcal{L}_{\mathcal{T}_{test(t)}}(f_{\theta'_t}) \ , \quad (2)$$

where β is the meta step size. The full algorithm is shown in Algorithm 1.
MAML has two learning rates α and β, which require time-consuming hyperparameter tuning.

Alpha MAML: Adaptive Model-Agnostic Meta-Learning

We would like to derive update rules for the learning rates α and β as well.
We derive the partial gradient of $\mathcal{L}_{\mathcal{T}_{test(i-1)}}$ with respect to the learning rate α_{i-1}:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}})}{\partial \alpha} &= \nabla_{\theta'_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}})^T . \frac{\partial(\theta_{i-2} - \alpha_{i-1}\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{train(i-1)}}(f_{\theta_{i-2}}))}{\partial \alpha} \\ &= \nabla_{\theta'_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}}).(-\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{train(i-1)}}(f_{\theta_{i-2}}))\end{aligned} \quad (3)$$

We can estimate α^*_{i-1} as follows:

$$\alpha_i = \alpha_{i-1} + \alpha_{\text{hyperlr}} \nabla_{\theta'_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}}) . \nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{train(i-1)}}(f_{\theta_{i-2}}) \ , \quad (4)$$

where α_{hyperlr} is the hyper learning rate for α.
Similar to α, we derive the partial derivative of $\mathcal{L}_{\mathcal{T}_{test(i-1)}}$ with respect to the meta-learning rate β:

$$\frac{\partial \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i})}{\partial \beta} = \frac{\partial \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i})}{\partial \theta_{i-1}} . \frac{\partial \theta_{i-1}}{\partial \beta} = \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i}).(-\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}})) \quad (5)$$

We can estimate β^*_{i-1} as follows:

$$\beta_i = \beta_{i-1} + \beta_{\text{hyperlr}} \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i}).\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}}) \ , \quad (6)$$

where β_{hyperlr} is the hyper learning rate for β. The final Alpha MAML algorithm can thus be written down into just 4 update equations:

$$\begin{aligned} \theta'_i &= \theta_{i-1} - \alpha_i \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{train(i)}}(f_{\theta_{i-1}}) \\ \alpha_{i+1} &= \alpha_i + \alpha_{\text{hyperlr}} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i}).\nabla_{\theta_{i-1}}\mathcal{L}_{\mathcal{T}_{train(i)}}(f_{\theta_{i-1}}) \\ \beta_i &= \beta_{i-1} + \beta_{\text{hyperlr}} \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i}).\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_{i-1}}) \\ \theta_i &= \theta_{i-1} - \beta_i \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(i)}}(f_{\theta'_i}) \end{aligned} \quad (7)$$

requiring only the extra memory storage of previous gradient, and no extra gradient needs to be computed, The full algorithm is shown in Algorithm 2.

Algorithm 1 MAML

Input: $p(\mathcal{T})$: distribution over tasks.
Input: α, β : learning rates
randomly initialize θ.
while not done **do**
  Sample batch of tasks $\mathcal{T}_t \sim p(\mathcal{T})$
  for all $\mathcal{T}_t$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_{train(t)}}(f_\theta)$ with respect to K examples.
    Compute adapted parameters with gradient descent: $\theta'_t = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_{train(t)}}(f_\theta)$
  end for
  Update $\theta = \theta - \beta \nabla_\theta \sum_{\mathcal{T}_t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_{test(t)}}(f_{\theta'_t})$
end while

Algorithm 2 Alpha MAML

Input: $p(\mathcal{T})$: distribution over tasks.
Input: α_0, β_0 : initial learning rates
Input: $\alpha_{\text{hyperlr}}, \beta_{\text{hyperlr}}$: hypergradient learning rates
randomly initialize θ.
while not done **do**
  Sample batch of tasks $\mathcal{T}_t \sim p(\mathcal{T})$
  for all $\mathcal{T}_t$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_{train(t)}}(f_\theta)$ with respect to K examples.
    Compute adapted parameters with gradient descent: $\theta'_t = \theta - \alpha_i \nabla_\theta \mathcal{L}_{\mathcal{T}_{train(t)}}(f_\theta)$
  end for
  $\alpha_{i+1} = \alpha_i + \alpha_{\text{hyperlr}} \sum_{\mathcal{T}_t \sim p(\mathcal{T})} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_{test(t)}}(f_{\theta'_t}).\nabla_{\theta_{i-1}}\mathcal{L}_{\mathcal{T}_{train(t)}}(f_{\theta_{i-1}})$
  $\beta_i = \beta_{i-1} + \beta_{\text{hyperlr}} \sum_{\mathcal{T}_t \sim p(\mathcal{T})} \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(t)}}(f_{\theta'_t}).\nabla_{\theta_{i-2}}\mathcal{L}_{\mathcal{T}_{test(i-1)}}(f_{\theta'_t})$
  $\theta_i = \theta_{i-1} - \beta_i \sum_{\mathcal{T}_t \sim p(\mathcal{T})} \nabla_{\theta_{i-1}} \mathcal{L}_{\mathcal{T}_{test(t)}}(f_{\theta'_t})$
end while

Behaviour of MAML vs Alpha-MAML

By performing online updates with hypergradients, Alpha-MAML:

- **adapts α_t and β_t on-the-fly within the main optimization loop, and**
- **significantly reduces the need to tune the initial learning rates α_0 and β_0.**

Here we choose:

- a good case (where the MAML user picked a good pair of α_0 and β_0, i.e., the tuned case)
- a bad case (where the user picked a bad pair of α_0 and β_0, i.e., an untuned case)

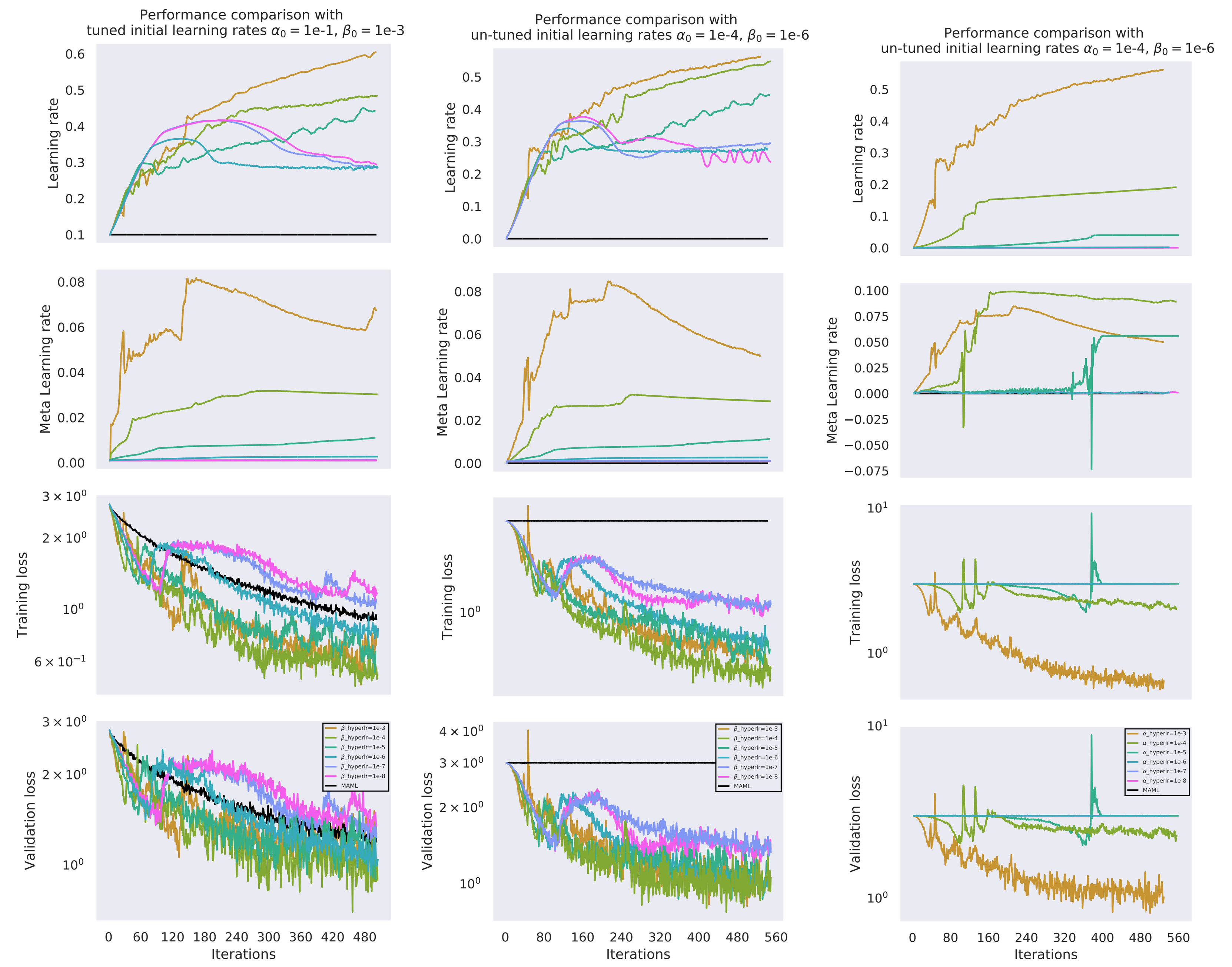

Figure: Convergence comparison of Alpha-MAML and MAML, with and without tuned initial learning rate hyper-parameters α_0 and β_0.
Columns: *left:* tuned initial learning rates; *middle:* untuned initial learning rates, with varying β_{hyperlr} for Alpha-MAML; *right:* untuned initial learning rates, with varying α_{hyperlr} for Alpha-MAML.
Rows: *first:* evolution of the learning rate α; *second:* evolution of the meta learning rate β; *third:* training loss; *fourth:* validation loss.

For given untuned initial learning rates, Alpha-MAML consistently brings the loss trajectory closer to the optimal one that would be attained by the baseline algorithm with tune initial learning rates.

Insensitivity with Respect to Hyperparameter Choices

We present experiments to study the effect of initial learning rate values on the number of iterations needed for the algorithms to reach a particular chosen loss threshold, i.e., the grid search for tuning the learning rate hyper-parameters.

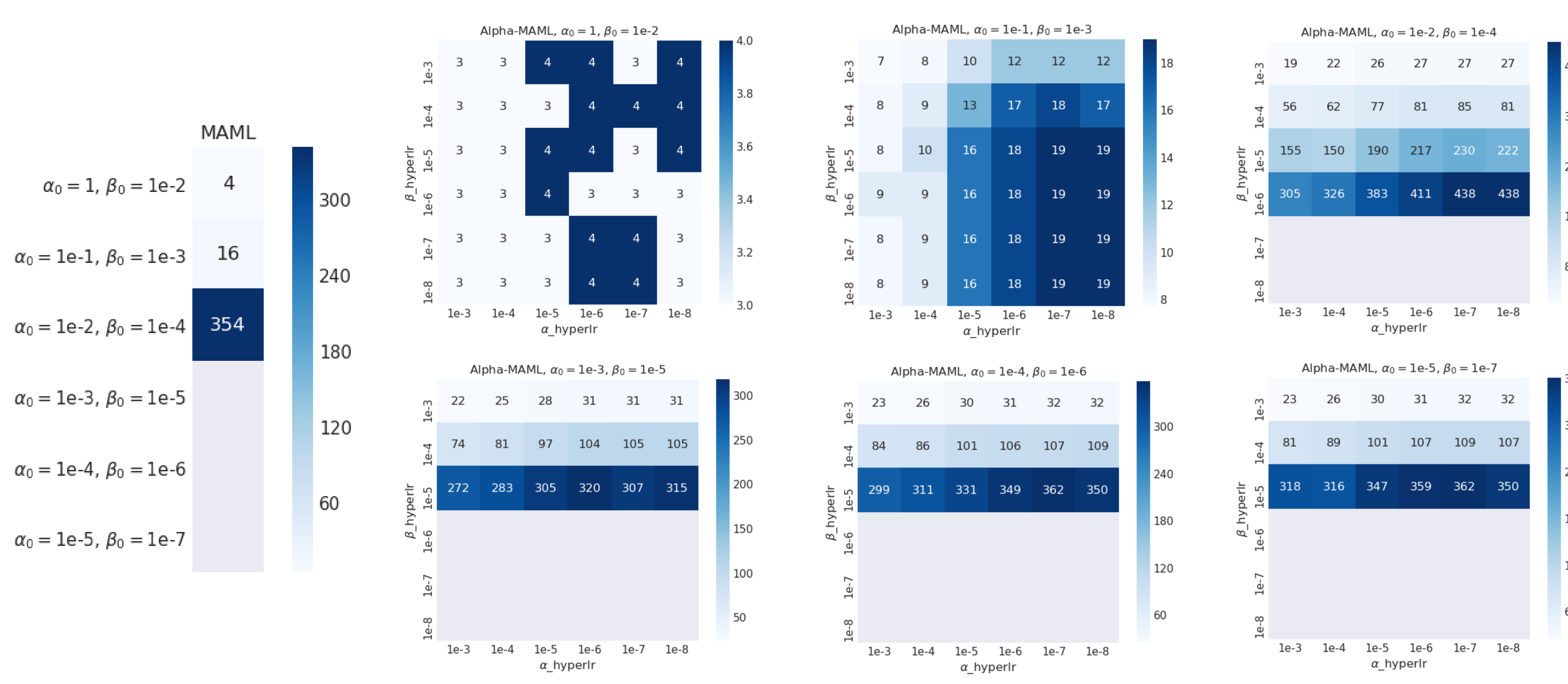

Figure: Iterations to converge to a training loss of 2.5 for Omniglot dataset. blank cells: > 500 iterations.

We observe:

- MAML shows very slow convergence for a range of initial learning rates In comparison, Alpha MAML shows comparatively faster convergence for these initial learning rates also, for a wide range of α_{hyperlr} and β_{hyperlr}.
- For cases where MAML shows fast convergence, Alpha MAML also shows fast convergence for all values of α_{hyperlr} and β_{hyperlr}.

No matter which values one chooses for the inital learning and meta-learning rates α_0 and β_0, Alpha MAML always shows faster, or in the worst case the same, convergence as MAML.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019.

[3] Atılım Güneş Baydin, Robert Cornish, David Martínez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *Sixth International Conference on Learning Representations (ICLR), Vancouver, Canada, April 30 – May 3, 2018*, 2018.