When MAML Can Adapt Fast and How to Assist When It Cannot

APPENDIX

A  Theoretical Analysis of Linear Models for Meta-Learning

This section provides more details to §4 in the main text.

A.1 Analytic Solution of 1D Linear Regression

We use \( \theta_\alpha = \theta - \alpha \nabla \ell_\tau(\theta) \) as a short-hand. The gradient of the MAML loss \( L^{\text{MAML}} \) is given by

\[
\frac{\partial L^{\text{MAML}}}{\partial \theta} = \mathbb{E}_\tau \left( I - \alpha H_\tau(\theta) \right) \frac{\partial \ell_\tau}{\partial \theta} \bigg|_{\theta - \alpha \nabla \ell_\tau} = \mathbb{E}_\tau \left( I - \alpha H_\tau(\theta) \right) \nabla \ell_\tau(\theta_\alpha) \tag{28}
\]

We setup the problem as follows. Let the task parameter \( \theta_\tau \sim N(0, 1) \) be a normal distributed scalar. Let the covariate \( x \sim N(0, 1) \) be a normal distributed scalar and the outcome be \( y \sim N(\theta_\tau x, 1) \). We investigate two models for their meta-learning performance:

**SHALLOW:** \( \hat{y} = cx \),  \**DEEP:** \( \hat{y} = abx \)

Note that the “deep” model is overparameterized. For each task, we use the least-square loss

\[
\ell_\tau = \frac{1}{2} \mathbb{E}_{p(x, y | \theta_\tau)} (y - cx)^2, \quad \text{or} \quad \ell_\tau = \frac{1}{2} \mathbb{E}_{p(x, y | \theta_\tau)} (y - abx)^2 \tag{30}
\]

A.1.1 Shallow Model

For the shallow model, the gradient is

\[
\nabla \ell_\tau = \mathbb{E}_{p(x, y | \theta_\tau)} (cx - y)x = c - \theta_\tau \tag{31}
\]

Thus, the loss for the \( \tau \)-th task is given by

\[
\ell_\tau(c - \alpha \nabla \ell_\tau) = \mathbb{E}_{p(x, y | \theta_\tau)} (y - [(1 - \alpha)c + \alpha \theta_\tau]x)^2 \tag{32}
\]

\[
= [(1 - \alpha)c + \alpha \theta_\tau]^2 - 2[(1 - \alpha)c + \alpha \theta_\tau]\theta_\tau \tag{33}
\]

Simplifying and completing the square on the \( \theta_\tau^2 \) term with constants, we arrive at

\[
\ell_\tau(c - \alpha \nabla \ell_\tau) = (1 - \alpha)^2(c - \theta_\tau)^2 + \text{CONST} \tag{34}
\]

Taking the expectation of the loss with respect to \( p(\tau) \), we have

\[
L^{\text{MAML}}_{\text{SHALLOW}} = 2(1 - \alpha)^2c^2 + \text{CONST} \tag{35}
\]

A.1.2 Deep Model

The gradients of the linear regression loss w.r.t \( a, b \) are given by:

\[
\frac{\partial \ell_\tau}{\partial a} = ab^2 - b \theta_\tau \tag{36}
\]

\[
\frac{\partial \ell_\tau}{\partial b} = a^2b - a \theta_\tau \tag{37}
\]
The post-adaptation parameters are obtained by taking one step of gradient descent with learning rate $\alpha$:

\[
\begin{align*}
a &\leftarrow a - \alpha a (ab - \theta_r) \\
b &\leftarrow b - \alpha a (ab - \theta_r)
\end{align*}
\]

This gives us the expression for the MAML loss:

\[
L_{\text{DEEP}}^{\text{MAML}} = \frac{1}{2} \mathbb{E} \mathbb{E}_{x,y} \left[ y - (a - \alpha a (ab - \theta_r)) \cdot (b - \alpha a (ab - \theta_r)) \right]^2
\]

(38)

We now take a closer look at the term $\mathbb{E} f^2$:

\[
\begin{align*}
f^2(a, b, \theta_r) &= \left[ ab - \alpha a^2 (ab - \theta_r) - ab^2 (ab - \theta_r) + \alpha^2 ab (ab - \theta_r)^2 \right]^2 \\
&= \left[ ab - \alpha (a^2 + b^2)(ab - \theta_r) + \alpha^2 (ab - \theta_r)^2 \right]^2 \\
&= \left[ ab - \alpha (a^2 + b^2)ab + \alpha^2 a^3 b^3 + (\alpha(a^2 + b^2) - 2\alpha^2 a^2 b^2)\theta_r + \alpha^2 ab\theta_r^2 \right]^2 \\
&= \left[ p_1 + p_2\theta_r + p_3\theta_r^2 \right]^2
\end{align*}
\]

(40)

where we let:

\[
p_1 = ab - \alpha (a^2 + b^2)ab + \alpha^2 a^3 b^3, \quad p_2 = \alpha (a^2 + b^2) - 2\alpha^2 a^2 b^2, \quad p_3 = \alpha^2 ab
\]

(41)

We then obtain

\[
\mathbb{E} f^2(a, b, \theta_r) = p_1^2 + p_2^2 + 3p_3^2 + 2p_1 p_3
\]

(42)

and where the last equality is obtained by remembering the expectation properties of $\theta_r$:

\[
\mathbb{E}_{p(\tau)} \theta_r^2 = 1, \quad \mathbb{E}_{p(\tau)} \theta_r^3 = 0, \quad \mathbb{E}_{p(\tau)} \theta_r^4 = 3.
\]

(43)

Furthermore, we obtain

\[
\mathbb{E} \theta_r f(a, b, \theta_r) = p_2
\]

(44)

This gives us a final expression for the MAML loss:

\[
L_{\text{DEEP}}^{\text{MAML}} = \frac{1}{2} (1 + p_1^2 + p_2^2 + 3p_3^2 + 2p_1 p_3 - 2p_2).
\]

(45)

We use the Matlab Symbolic Toolbox to help us simplify the rest of the calculation. The code is given below.

% We use sa, sb, salpha as the "symbolic version" of a, b, and alpha
sym sa sb salpha

pp1 = sa*sb*(1- salpha*(sa^2+sb^2)+ salpha^2*sa^2*sb^2);
pp2 = salpha*(sa^2+sb^2) - 2 *salpha^2 *sa^2*sb^2;
pp3 = salpha^2*sa*sb;

L = pp1*pp1 + pp2*pp2 + 3*pp3*pp3 + 2*pp1*pp3 -2 *pp2;
diffa = diff(L, sa);
diffb = diff(L, sb);

H = [diff(diffa, sa) diff(diffa, sb); diff(diffb, sa) diff(diffb, sb)];
Stationary Points  The gradients are complicated polynomials in $a$ and $b$. Instead, we give the specific cases:

$$\frac{\partial L_{\text{MAML}}}{\partial a} \bigg|_{b=0} = 4\alpha a(a^2 - 1), \quad \frac{\partial L_{\text{MAML}}}{\partial b} \bigg|_{b=0} = 0$$ \hspace{1cm} (54)

We immediately can derive that there are at least 5 stationary points:

- $(a = 0, b = 0)$, with Hessian
  $$H = \begin{bmatrix} -4\alpha & 0 \\ 0 & -4\alpha \end{bmatrix}$$ \hspace{1cm} (55)

- $(a = \pm \frac{1}{\sqrt{\alpha}}, 0)$, with Hessian
  $$H = \begin{bmatrix} 8\alpha & 0 \\ 0 & 6\alpha^3 \end{bmatrix}$$ \hspace{1cm} (56)

- $(a = 0, b = \pm \frac{1}{\sqrt{\alpha}})$, with Hessian
  $$H = \begin{bmatrix} 6\alpha^3 & 0 \\ 0 & 8\alpha \end{bmatrix}$$ \hspace{1cm} (57)

B Supplementary Experiments

This section provides additional experimental evidence to complement the evidence presented in the main text.

B.1 Binary Logistic Regression

To resonate more with our experiments of using (multinominal) logistic regression models on Omniglot, CIFAR and MNIST datasets, we also analyze binary logistic regression models on synthetic data.

![Graph](image)

Figure B.1: Meta-learning of linear models on synthetic data. For linear separable data, MAML fails on logistic regression but succeed on logistic regression augmented with 2 or 3 linear layers. (Left) Meta-train accuracies. (Right) Meta-test accuracies.

B.1.1 MAML on Logistic Regression and Logistic Regression with Linear Layers

We randomly sample a set of 2-dimensional task parameters $\theta_\tau \in \mathbb{R}^2$ from a standard multivariate spherical Gaussian and use each of them to define a linear decision boundary of a binary classification task. We sample inputs from a 2-dimensional multivariate spherical Gaussian and the binary outputs for each task are sampled from $y \sim \sigma(\theta_\tau^T x)$, where $\sigma()$ is the sigmoid function.

By construction, a logistic regression (LR) is sufficient to achieve very high accuracy on any task. But can MAML learn a linear classifier from a randomly sampled subset of training tasks that is able to adapt quickly to the test
tasks? It is intuitive to see the minimizer for the MAML loss $\mathcal{L}_{\text{MAML}}$ is the origin due to the rotation invariances of both the task parameters and the inputs. The origin thus provides the best initialization to adapt to new tasks by not favoring any particular task.

Fig. B.1 (Left) reports the 1-step post-adaptation accuracy on the test tasks for the meta-learned logistic regression model. Equally surprising as the previous results, the model fails to perform better than chance. However, adding linear layers “rescues” the meta-learning; LR with 2 or 3 linear layers attains nearly perfect classification on any task.

B.2 Logistic Regression Failure Modes

As argued in §5.1, shallow logistic regression models fail to meta-learn on Omniglot and CIFAR-FS. Fig. B.2 displays this same failure mode on the mini-ImageNet dataset. To simplify computations, we downscaled the original mini-ImageNet images (84x84 pixels, with RGB channels) to the same size as CIFAR-FS. (32x32 pixels, with RGB channels)

Figure B.2: Meta-training logistic regression models with MAML on Omniglot, CIFAR-FS, and mini-ImageNet led to poor performances. Adding linear nets improves meta-learning significantly, without changing the model’s capacity.

B.3 Linear Layers Cannot Be Collapsed

Our previous results have shown that when MAML cannot meta-train well, adding multiple linear layers helps to meta-learn, cf. Fig. B.1 and Fig. 1 of the main text.

Figure B.3: Collapsing multiple linear layers into a single layer before adapting to new tasks (green curves) leads to poor adaptation results.

But after meta-training, can we collapse those multiple linear layers into a single linear layer so as to reduce the model size?
Fig. B.3 shows that collapsing erases the model’s ability to adapt to new tasks. This suggests that the solutions identified in models with additional linear layers need to stay in the overparameterized space to be effective and they cannot be collapsed before adaptation.

B.4 Does SGD-induced implicit regularization for linear networks matter for meta-learning?

Implicit regularization refers to the bias towards solutions with different generalization abilities when stochastic gradient descent (SGD) is used to optimize overparameterized models [Gidel et al. 2019].

One might want to argue that implicit regularization explains why additional linear layers in our experiments (Fig. B.1) help to meta-learn. This is an interesting hypothesis, though we do not think it can fully explain what is observed.

First, as those experiments have shown, for models augmented with linear nets, there is very little difference between training accuracies and meta-testing accuracies. Moreover, for models without linear nets, there is very little difference between the two phases either. Thus, while SGD could lead to solutions with different generalization abilities in regular supervised learning settings, our experiments do not show there is an overfitting issue in the experiments here for meta-learning.

The difference between parsimonious and overparameterized models (with linear layers) could be due to the difference in solutions. However, the solutions are in space of different dimensionality. Fig. B.3 shows that after we collapse the overparameterize models such that the solutions are in the same space, the solutions from the collapsing becomes as ineffective (in terms of leading to poor adaptation results) as the original models. This suggests that SGD is not identifying a solution in the overparameterized space that could have been identified in the original space. In other words, SGD is not biasing towards any solution in the overparameterized space that could have made difference in the original space.

B.5 ANIL with Meta-Optimizers

In §5.3, we showed that MAML augmented with meta-optimizers was able to meta-learn in shallow non-linear networks. Table B.4 shows similar results when using ANIL as the meta-learning algorithm. (Note: this table omits T-Nets, as it is not compatible with ANIL.) We observe that ANIL always benefits from the combination with a meta-optimizer, and specifically that it benefits most from KFO.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ANIL</th>
<th>ANIL w/ MSGD</th>
<th>ANIL w/ MC</th>
<th>ANIL w/ KFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omniglot</td>
<td>91.00</td>
<td>92.47</td>
<td>92.67</td>
<td><strong>94.40</strong></td>
</tr>
<tr>
<td>CIFAR-FS</td>
<td>66.10</td>
<td>67.55</td>
<td>67.43</td>
<td><strong>68.81</strong></td>
</tr>
<tr>
<td>mini-ImageNet</td>
<td>56.42</td>
<td>57.45</td>
<td>59.07</td>
<td><strong>62.65</strong></td>
</tr>
</tbody>
</table>

B.6 Effect of Number of Layers

Figure B.4 complements the results in §5.3 when varying the number of layers in the convolutional network. (We include again results on Omniglot and CIFAR-FS for convenience.) As we observed in the main text, shallower models greatly benefit from the additional meta-optimization parameters which alleviates the burden of learning how to adapt. But, this benefit dampens as we increase the number of convolutional layers – the gap between META-KFO and MAML shrinks – since the additional upper layers implicitly transform the gradient of lower layers, thus enabling successful meta-learning.

C Details on KFO

C.1 Other factorization schemes

A technical issue arises when expressing the meta-optimizer $U_T$ as a neural network: the dimensionality of modern (model) network architectures ranges in the tens of thousands, if not more. To address those computational
and memory issues, we learn one meta-optimizer network per matrix parameter in the model network and express the weights of the optimizer neural network as a Kronecker factorization such that for each weight $W \in \mathbb{R}^{m \times n}$: $W = R^T \otimes L$, where $R \in \mathbb{R}^{m \times m}$ and $L \in \mathbb{R}^{n \times n}$.

Many matrix factorization schemes could be used and would result in different modeling and computational trade-offs. For example, using a low-rank Cholesky factorization $A = LL^T$ where $L \in \mathbb{R}^{k \times r}$ allows to interpolate between computational complexity and decomposition rank by tuning the additional hyper-parameter $r$. The Cholesky decomposition might be preferable to the Kronecker one in memory-constrained applications, since $r$ can be used to control the memory requirements of the meta-optimizer. Moreover, such a decomposition imposes symmetry and positiveness on $A$, which might be desirable when approximating the Hessian or its inverse.

In this work, we preferred the Kronecker decomposition over alternatives for three reasons: (1) the computational and memory cost of the Kronecker-product are acceptable, (2) $R^T \otimes L$ is full-rank whenever $L, R$ are full-rank, and (3) the identity matrix lies in the span of Kronecker-factorized matrices. In particular, this last motivation allows meta-optimizers to recover the gradient descent update by letting $R, L$ be the identity.

### C.1.1 Schematics and Pseudo-code

Pseudo-code for meta-optimizers is provided in Algorithm 1 and a schematic of the model-optimizer loop in Figure C.5.

![Figure C.5](image)

**Algorithm 1 Meta-Learning with Meta-Optimizers**

**Require:** Fast learning rate $\alpha$, Initial parameters $\theta^{(\text{init})}$ and $\xi^{(\text{init})}$, Optimizer $\text{Opt}$

1. **while** $\theta^{(\text{init})}, \xi^{(\text{init})}$ not converged **do**
2. 
3. 
4. **for** step $t = 1, \ldots, T$ **do**
5. 
6. 
7. 
8. 
9. 
10. **Update model and meta-optimizer initializations**
11. 
12. 

Figure B.4: Complementing Fig. we vary the number of convolutional layers in the model. As the model size increases the performance of both methods improves, which can be attributed to the model’s increased capacity to learn the target task. However, the net gain from meta-KFO over MAML has diminishing returns as the number of layers increases since the benefit of the external meta-optimizer reduces as the upper layers of the larger models have more capacity to meta-learn to control their own bottom layers.
Table D.5: Adaptation learning rates for the linear model experiments in the main text and the Suppl. Material

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Meta learning rate</th>
<th>Adaptation learning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>LR</td>
<td>0.01</td>
<td>0.900</td>
</tr>
<tr>
<td>Synthetic</td>
<td>LR+LinNet(2)</td>
<td>0.1</td>
<td>0.900</td>
</tr>
<tr>
<td>Synthetic</td>
<td>LR+LinNet(3)</td>
<td>0.1</td>
<td>0.900</td>
</tr>
<tr>
<td>Omniglot</td>
<td>LR</td>
<td>0.0005</td>
<td>1.0</td>
</tr>
<tr>
<td>Omniglot</td>
<td>LR+LinNet</td>
<td>0.0005</td>
<td>0.08</td>
</tr>
<tr>
<td>CIFAR-FS</td>
<td>LR</td>
<td>0.0005</td>
<td>0.01</td>
</tr>
<tr>
<td>CIFAR-FS</td>
<td>LR+LinNet</td>
<td>0.0001</td>
<td>0.02</td>
</tr>
<tr>
<td>mini-ImageNet</td>
<td>LR</td>
<td>0.0005</td>
<td>0.01</td>
</tr>
<tr>
<td>mini-ImageNet</td>
<td>LR+LinNet</td>
<td>0.0005</td>
<td>0.01</td>
</tr>
</tbody>
</table>

D  Hyperparameters and details for reproducibility

We report the hyper-parameter values for the experiments of the main text. Each experimental setup is tuned independently based on post-adaptation accuracies obtained on validation tasks.

All experiments are implemented on top of PyTorch [Paszke et al., 2019] and learn2learn [Arnold et al., 2020]. Moreover, our work relied on tools from the Python scientific ecosystem [Van Rossum and Drake, 1995], including numpy [Hunter, 2007], matplotlib [Harris et al., 2020], and scipy [Virtanen et al., 2020]. We provide example implementations of Meta-KFO at: https://github.com/Sha-Lab/kfo

Linear Models  On the 1D linear regression, both shallow and deep models use a fast-adaptation learning rate set to 0.1. Their parameters are optimized by SGD with a meta learning rate set to 0.01, and a momentum term set to 0.9. On the vision datasets (Omniglot, CIFAR-FS, mini-ImageNet), the meta and adaptation learning rates are given in Table D.5. For those experiments, the added linear network consists of 256-128-64-64 hidden units. To simplify computations on mini-ImageNet, we downsample images to 32x32 pixels.

Nonlinear Models  All methods in Tables 1, 2, and 3 are tested on the standard 5-ways 5-shots setting. For Omniglot and CIFAR-FS, the networks are the original CNN from [Finn et al., 2017] with 32 hidden units, while for mini-ImageNet we used 64 hidden units. We use Adam to optimize the networks, with default values of (0.9, 0.999) for $\beta$ and 1e-8 for $\epsilon$, and process tasks by batches of 16. Meta and adaptation learning rates for all methods are reported in Tables D.6 and D.7.
Table D.6: **Adaptation learning rates** for the two-layer CNN experiments in the main text and the Suppl. Material

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAML w/</th>
<th>ANIL w/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(null)</td>
<td>LinNet</td>
</tr>
<tr>
<td>Omniglot</td>
<td>0.08</td>
<td>0.05</td>
</tr>
<tr>
<td>CIFAR-FS</td>
<td>0.07</td>
<td>0.01</td>
</tr>
<tr>
<td>mini-ImageNet</td>
<td>0.001</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table D.7: **Meta learning rates** for the two-layer CNN experiments in the main text and the Suppl. Material

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAML w/</th>
<th>ANIL w/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(null)</td>
<td>LinNet</td>
</tr>
<tr>
<td>Omniglot</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>CIFAR-FS</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>mini-ImageNet</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
</tbody>
</table>
When MAML Can Adapt Fast and How to Assist When It Cannot

References


Sébastien M. R. Arnold, Shariq Iqbal, Fei Sha


