Meta-Learning and Universality

Conference paper at ICLR 2018

Chelsea Finn & Sergey Levine (US Berkeley)

Youngseok Yoon

Contents

- The Universality in Meta-Learning
- Model Construct (Pre-update)
- Single gradient step and Post-update
- Show universality from model
- Experiments
- Conclusion

The Universality in Meta-Learning

- What is the universality in Meta-Learning?
- Universality in Neural Network:

  Approximate any function $f: x \to y$
- Universality in Meta-Learning:

  Approximate any function $f: \left((x, y)_k, x^*\right) \to \hat{y}$

  Input: The training data $(x, y)_k$, and Test data x^*.

  In this presentation, only show about $k = 1$ case.
- Our model to verify Universality:

  MAML: $\hat{f}(\cdot;\theta) \to \hat{f}(\cdot;\theta')$, $\theta' = \theta - \alpha \cdot \nabla_\theta \mathcal{L}\left(target, \hat{f}(\cdot;\theta)\right)$

  Update model $f(\cdot;\theta)$ using gradient update

Model Construct (Pre-update)

- Our model: Deep neural network with ReLU, MAML training.

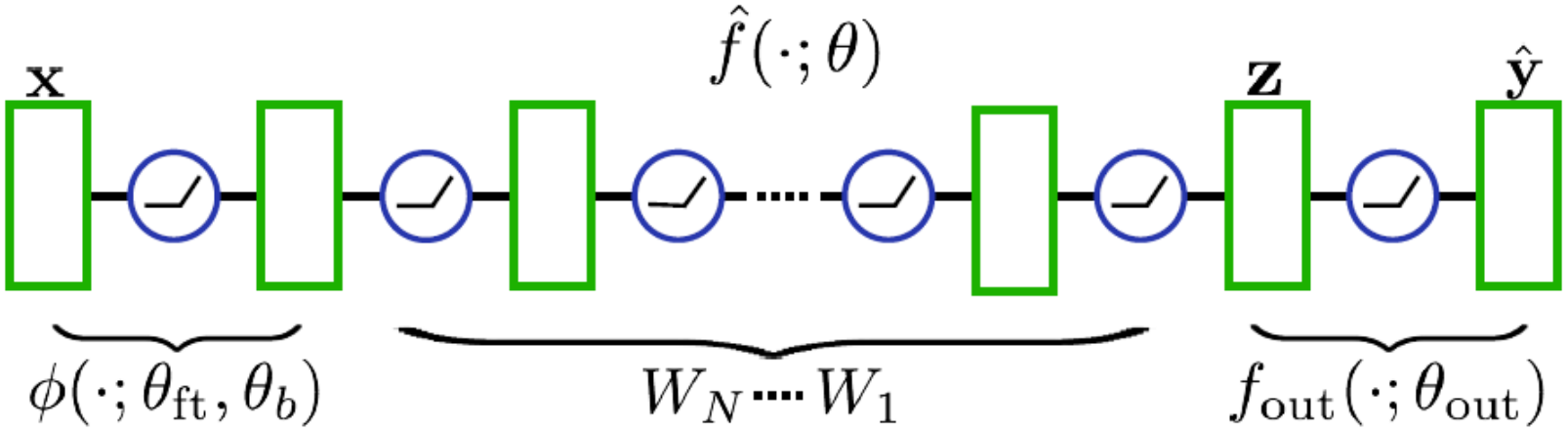
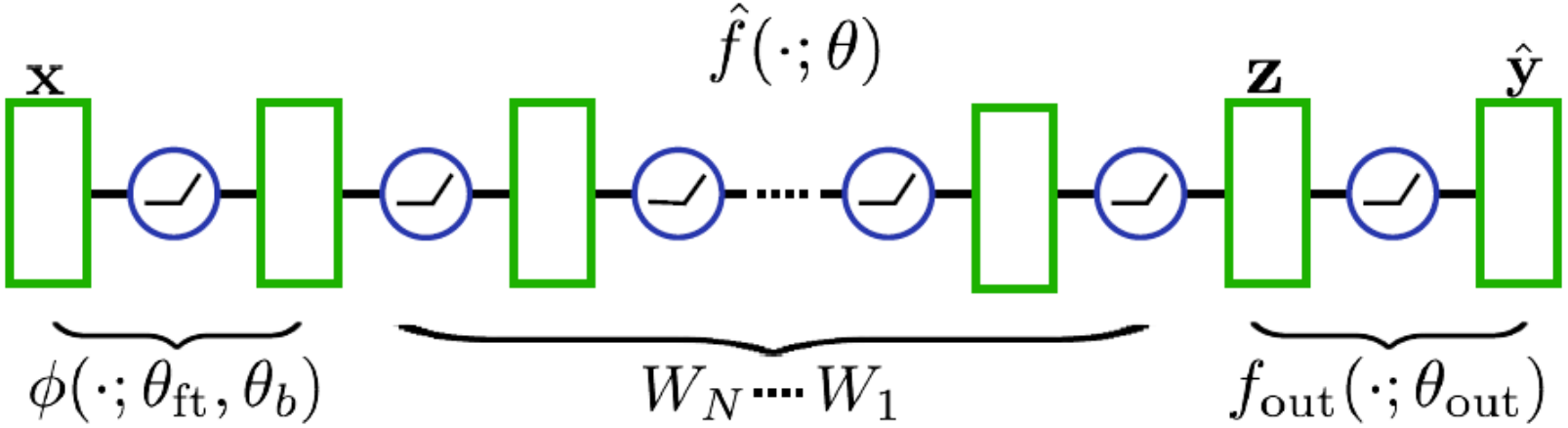


Figure 1: A deep fully-connected neural network with N+2 layers and ReLU nonlinearities. With this generic fully connected network, we prove that, with a single step of gradient descent, the model can approximate any function of the dataset and test input.

Model Construct (Pre-update)

- Construct model $\hat{f}(\cdot;\theta)$ with ReLU
- If we assume inputs and all pre-synaptic activations are non-negative, then deep ReLU act like deep linear networks. ⋯ (1)
- $\hat{f}(\cdot;\theta) = f_{out}\left(\left(\prod_{i=1}^{N} W_i\right)\phi(\cdot;\theta_{ft},\theta_b);\theta_{out}\right)$

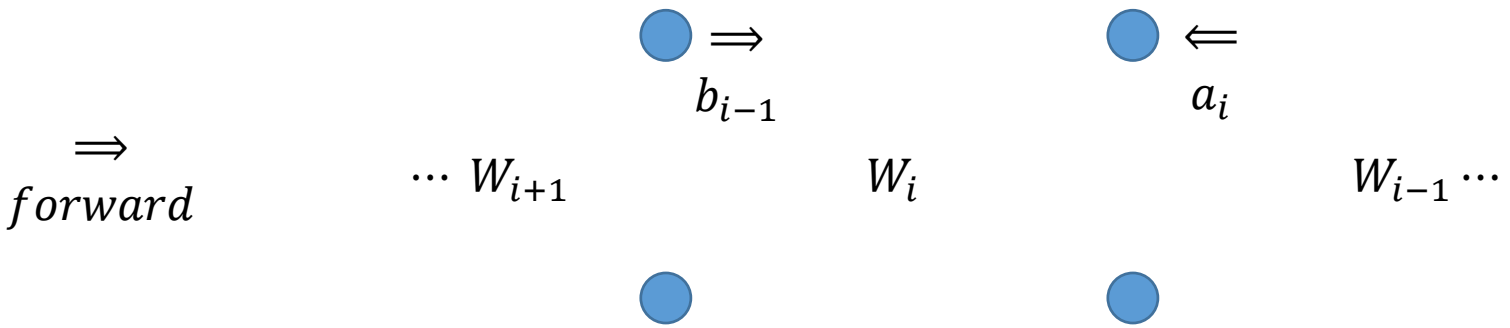


Single gradient step and Post-update

- $\nabla_{W_i}\mathcal{L} = a_i b_{i-1}^T$

  a_i: the error gradient with respect to the pre-synaptic activations at layer i

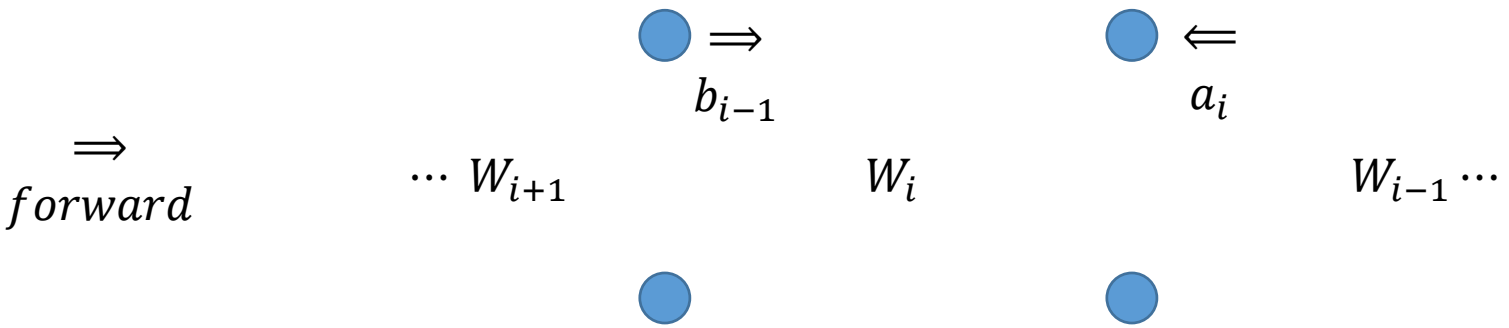
  b_{i-1}: the forward post-synaptic activations at layer $i-1$



Single gradient step and Post-update

- let $\boldsymbol{z} = \left(\prod_{i=1}^{N} W_i\right)\phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)$, $\nabla_{\boldsymbol{z}}\mathcal{L} = e(\boldsymbol{x}, \boldsymbol{y})$
- Then,

$$\begin{aligned}\nabla_{W_i}\mathcal{L} &= a_i b_{i-1}^T \\ &= W_{i-1}^T \cdots W_1^T e(\boldsymbol{x}, \boldsymbol{y}) \cdot \left(\left(\prod_{j=i+1}^{N} W_j\right)\phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)\right)^T \\ &= \left(\prod_{j=1}^{i-1} W_j\right)^T e(\boldsymbol{x}, \boldsymbol{y}) \cdot \phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T \left(\prod_{j=i+1}^{N} W_j\right)^T\end{aligned}$$



Single gradient step and Post-update

- let $\boldsymbol{z} = \left(\prod_{i=1}^{N} W_i\right)\phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)$, $\nabla_{\boldsymbol{z}}\mathcal{L} = e(\boldsymbol{x}, \boldsymbol{y})$
- Then,

$$\begin{aligned}\nabla_{W_i}\mathcal{L} &= a_i b_{i-1}^T \\ &= W_{i-1}^T \cdots W_1^T e(\boldsymbol{x}, \boldsymbol{y}) \cdot \left(\left(\prod_{j=i+1}^{N} W_j\right)\phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)\right)^T \\ &= \left(\prod_{j=1}^{i-1} W_j\right)^T e(\boldsymbol{x}, \boldsymbol{y}) \cdot \phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T \left(\prod_{j=i+1}^{N} W_j\right)^T\end{aligned}$$

- Therefore, post-update value of $\prod_{i=1}^{N} W_i' = \prod_{i=1}^{N}\left(W_i - \alpha \cdot \nabla_{W_i}\mathcal{L}\right)$

$$\prod_{i=1}^{N} W_i - \alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1} W_j\right)\left(\prod_{j=1}^{i-1} W_j\right)^T e(\boldsymbol{x}, \boldsymbol{y}) \cdot \phi\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T \left(\prod_{j=i+1}^{N} W_j\right)^T \left(\prod_{j=1}^{i-1} W_j\right) - O(\alpha^2)$$

Single gradient step and Post-update

- Post-update value of $\prod_{i=1}^{N} W_i' = \prod_{i=1}^{N}\left(W_i - \alpha \cdot \nabla_{W_i}\mathcal{L}\right)$

  $\prod_{i=1}^{N} W_i - \alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1} W_j\right)\left(\prod_{j=1}^{i-1} W_j\right)^T e(\boldsymbol{x},\boldsymbol{y}) \cdot \phi\left(\boldsymbol{x};\theta_{ft},\theta_b\right)^T \left(\prod_{j=i+1}^{N} W_j\right)^T \left(\prod_{j=1}^{i-1} W_j\right) - O(\alpha^2)$

- Now, post-update value of $\boldsymbol{z}^*$ when $\boldsymbol{x}^*$ input into $\hat{f}(\cdot;\theta')$

  $\boldsymbol{z}^* = \prod_{i=1}^{N} W_i' \,\phi\left(\boldsymbol{x}^*;\theta_{ft}',\theta_b'\right)$

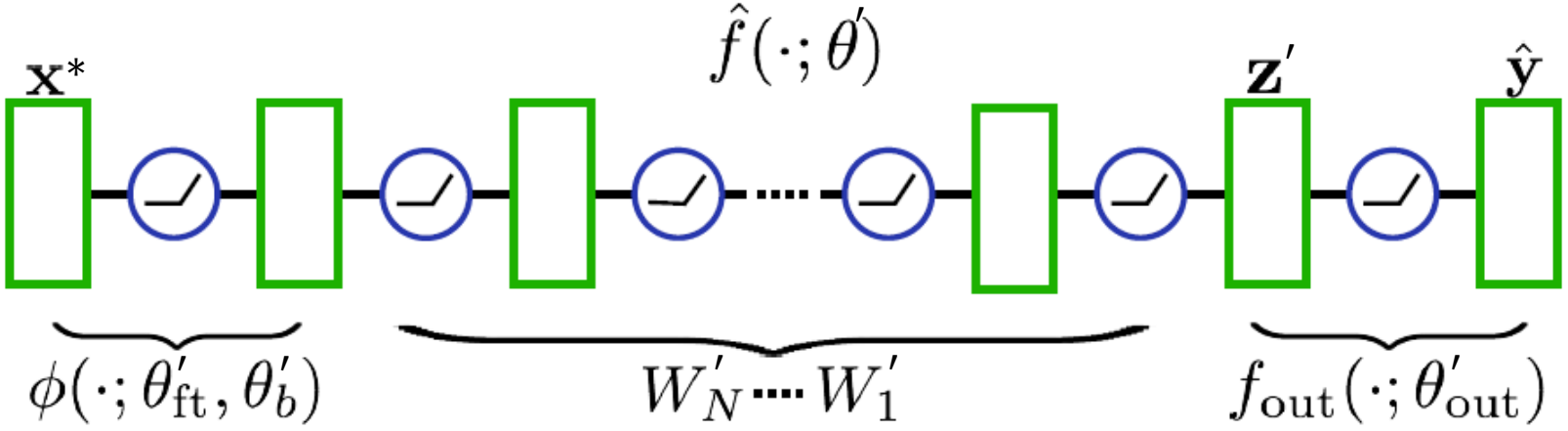
  $= \prod_{i=1}^{N} W_i \phi\left(\boldsymbol{x}^*;\theta_{ft}',\theta_b'\right)$

  $-\alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1} W_j\right)\left(\prod_{j=1}^{i-1} W_j\right)^T e(\boldsymbol{x},\boldsymbol{y}) \cdot \phi\left(\boldsymbol{x};\theta_{ft},\theta_b\right)^T \left(\prod_{j=i+1}^{N} W_j\right)^T \left(\prod_{j=1}^{i-1} W_j\right)\phi\left(\boldsymbol{x}^*;\theta_{ft}',\theta_b'\right)$

- And, $\hat{f}(\boldsymbol{x}^*;\theta') = f_{out}(\boldsymbol{z}^*;\theta_{out}')$

Show universality from model

- How to show universality from post-updated model?

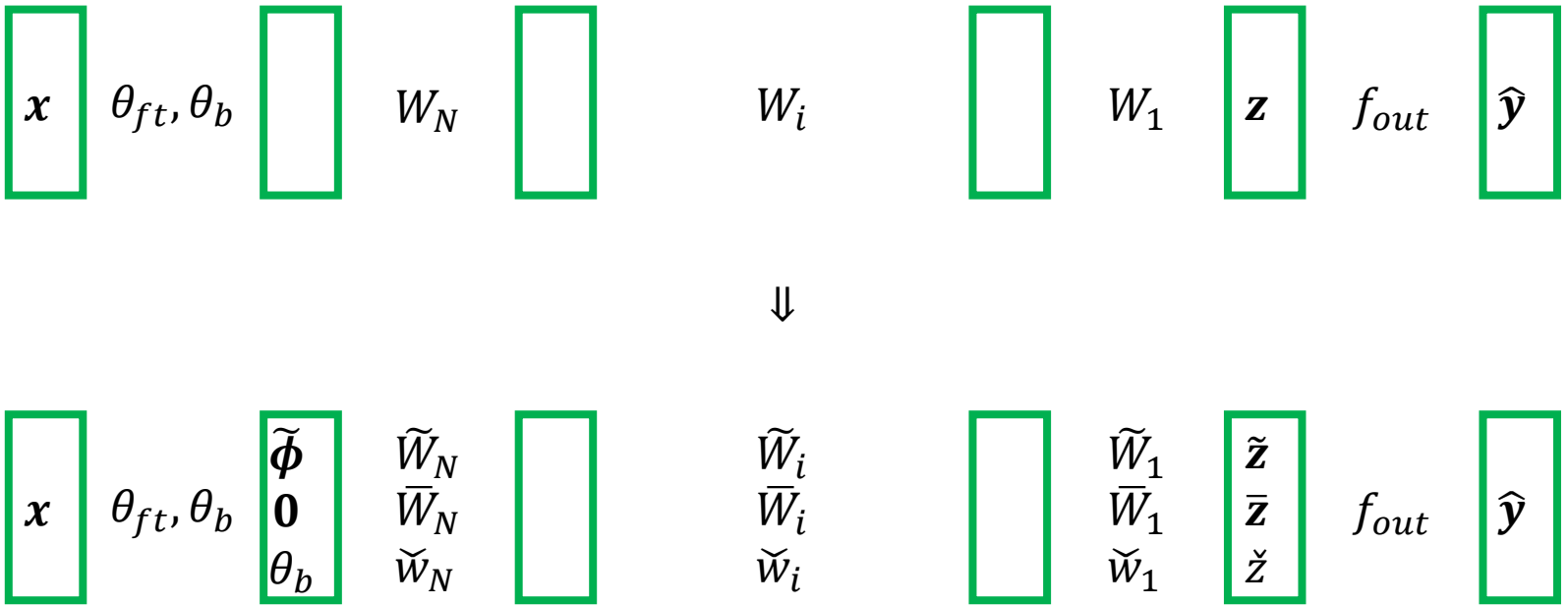


Show universality from model

- Use independently control information flow from $\boldsymbol{x}$, from $\boldsymbol{y}$, from $\boldsymbol{x}^*$ by multiplexing forward information from $\boldsymbol{x}$ and backward information from $\boldsymbol{y}$
- Decomposing W_i, ϕ and error gradient into three parts,

$$W_i \coloneqq \begin{bmatrix} \widetilde{W}_i & 0 & 0 \\ 0 & \overline{W}_i & 0 \\ 0 & 0 & \breve{w}_i \end{bmatrix}, \qquad \phi\left(\cdot; \theta_{ft}, \theta_b\right) \coloneqq \begin{bmatrix} \tilde{\phi}\left(\cdot; \theta_{ft}, \theta_b\right) \\ \mathbf{0} \\ \theta_b \end{bmatrix}, \qquad \nabla_z \mathcal{L}\left(y, \hat{f}(\boldsymbol{x}; \theta)\right) \coloneqq \begin{bmatrix} \mathbf{0} \\ \bar{e}(y) \\ \breve{e}(y) \end{bmatrix} \cdots (2)$$

Show universality from model



Show universality from model

- Then, before the gradient update,

$$\boldsymbol{z} = \begin{bmatrix} \tilde{\boldsymbol{z}} \\ \bar{\boldsymbol{z}} \\ \check{z} \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^{N} \widetilde{W}_i \, \tilde{\phi}(x; \theta_{ft}, \theta_b) \\ \mathbf{0} \\ 0 \end{bmatrix}$$

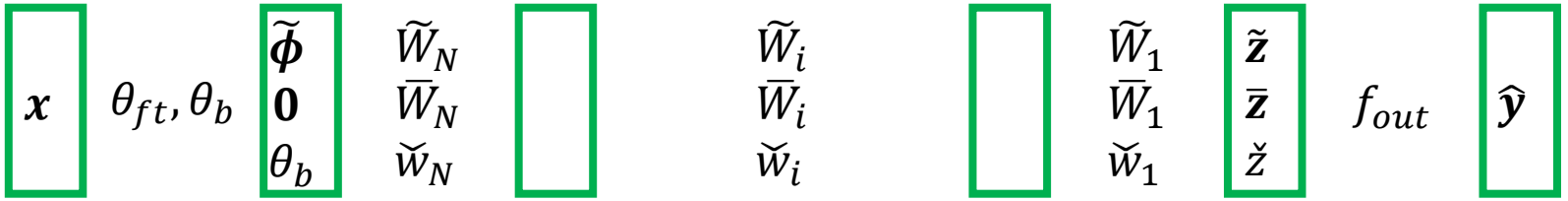
- Hence, represent the middle one of after gradient update, $\boldsymbol{z}^*$

$$\boldsymbol{z}^* = \prod_{i=1}^{N} \begin{bmatrix} \widetilde{W}_i & 0 & 0 \\ 0 & \bar{W}_i & 0 \\ 0 & 0 & \check{w}_i \end{bmatrix} \begin{bmatrix} \tilde{\phi}(\boldsymbol{x}; \theta_{ft}, \theta_b) \\ \mathbf{0} \\ \theta_b \end{bmatrix} - \alpha \Sigma_{i=1}^{N} \left(\prod_{j=1}^{i-1} \begin{bmatrix} \widetilde{W}_j & 0 & 0 \\ 0 & \bar{W}_j & 0 \\ 0 & 0 & \check{w}_j \end{bmatrix} \right) \left(\prod_{j=1}^{i-1} \begin{bmatrix} \widetilde{W}_j & 0 & 0 \\ 0 & \bar{W}_j & 0 \\ 0 & 0 & \check{w}_j \end{bmatrix} \right)^T \begin{bmatrix} \mathbf{0} \\ \bar{e}(y) \\ \check{e}(y) \end{bmatrix} \cdot$$
$$\begin{bmatrix} \tilde{\phi}(\boldsymbol{x}; \theta_{ft}, \theta_b) \\ \mathbf{0} \\ \theta_b \end{bmatrix}^T \left(\prod_{j=i+1}^{N} \begin{bmatrix} \widetilde{W}_j & 0 & 0 \\ 0 & \bar{W}_j & 0 \\ 0 & 0 & \check{w}_j \end{bmatrix} \right)^T \left(\prod_{j=1}^{i-1} \begin{bmatrix} \widetilde{W}_j & 0 & 0 \\ 0 & \bar{W}_j & 0 \\ 0 & 0 & \check{w}_j \end{bmatrix} \right) \begin{bmatrix} \tilde{\phi}(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b) \\ \mathbf{0} \\ \theta'_b \end{bmatrix}$$

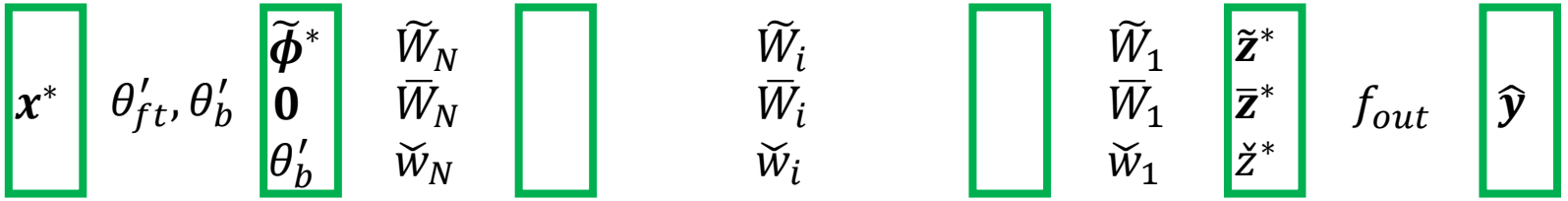
$$\therefore \bar{\boldsymbol{z}}^* = -\alpha \Sigma_{i=1}^{N} \left(\prod_{j=1}^{i-1} \bar{W}_j \right) \left(\prod_{j=1}^{i-1} \bar{W}_j \right)^T \bar{e}(\boldsymbol{y}) \cdot \tilde{\phi}(\boldsymbol{x}; \theta_{ft}, \theta_b)^T \left(\prod_{j=i+1}^{N} \widetilde{W}_j \right)^T \left(\prod_{j=1}^{i-1} \widetilde{W}_j \right) \tilde{\phi}(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b)$$

Show universality from model

- $\hat{y} = f_{out}(\boldsymbol{z}; \theta_{out})$ and $\bar{\boldsymbol{z}} = 0$



- $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1}\bar{W}_j\right)\left(\prod_{j=1}^{i-1}\bar{W}_j\right)^T \bar{e}(\boldsymbol{y}) \cdot \tilde{\phi}\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T \left(\prod_{j=i+1}^{N}\widetilde{W}_j\right)^T \left(\prod_{j=1}^{i-1}\widetilde{W}_j\right)\tilde{\phi}\left(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b\right)$
  $= -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y})\tilde{\phi}\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T B_i^T B_i \tilde{\phi}\left(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b\right)^T \quad \cdots (3)$

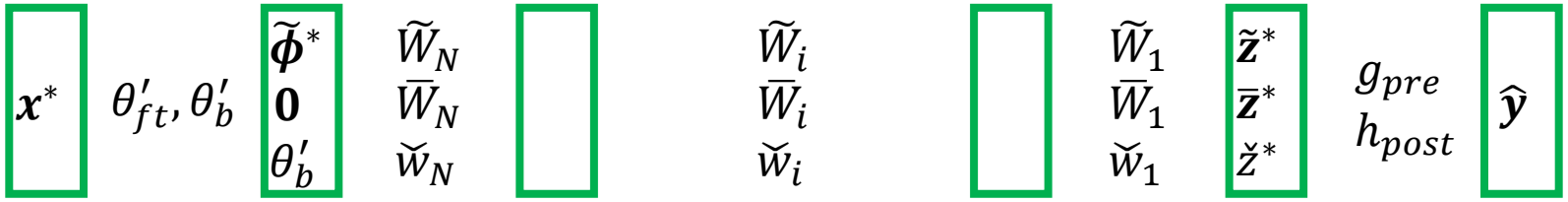


Show universality from model

- Path 1)
  - Define f_{out}
- Path 2)
  - Show $\bar{\boldsymbol{z}}^*$ as new vector v, which containing informations of $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{x}^*$
- At last,
  - Universality with Vector v

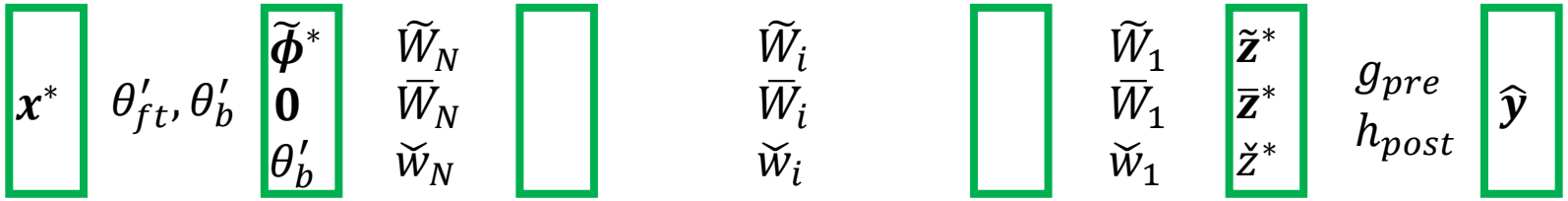
Show universality from model

- Define f_{out} as a neural network that approximates the following multiplexer function and its derivatives
  - $f_{out}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}\\ \bar{\boldsymbol{z}}\\ \check{z}\end{bmatrix};\theta_{out}\right)=\mathbf{1}_{\bar{z}=0}g_{pre}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}\\ \bar{\boldsymbol{z}}\\ \check{z}\end{bmatrix};\theta_g\right)+\mathbf{1}_{\bar{z}\neq 0}h_{post}(\bar{\boldsymbol{z}};\theta_h),\ \cdots(4)$
  - Where g_{pre} is a linear function with parameters θ_g such that $\nabla_{\boldsymbol{z}}\mathcal{L}=e(\boldsymbol{y})\cdots(2)$
  - Where h_{post} is a neural network with one or more hidden layers.
- Then, after pose-update,
  - $f_{out}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}^*\\ \bar{\boldsymbol{z}}^*\\ \check{z}^*\end{bmatrix};\theta'_{out}\right)=h_{post}(\bar{\boldsymbol{z}}^*;\theta_h),\cdots(5)^*$



Show universality from model

- Change the form of $\bar{\boldsymbol{z}}^*$ with kernel form
  - $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y})\tilde{\phi}\left(\boldsymbol{x};\theta_{ft},\theta_b\right)^T B_i^T B_i \tilde{\phi}\left(\boldsymbol{x}^*;\theta'_{ft},\theta'_b\right)^T = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x},\boldsymbol{x}^*)$
- $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x},\boldsymbol{x}^*)$ has the all information of $\boldsymbol{x}, \boldsymbol{y}$ and $\mathbf{x}^*$.
- Assume $\bar{e}(\boldsymbol{y})$ is a linear function of $\boldsymbol{y}$ which can extract the original information $\boldsymbol{y}$.

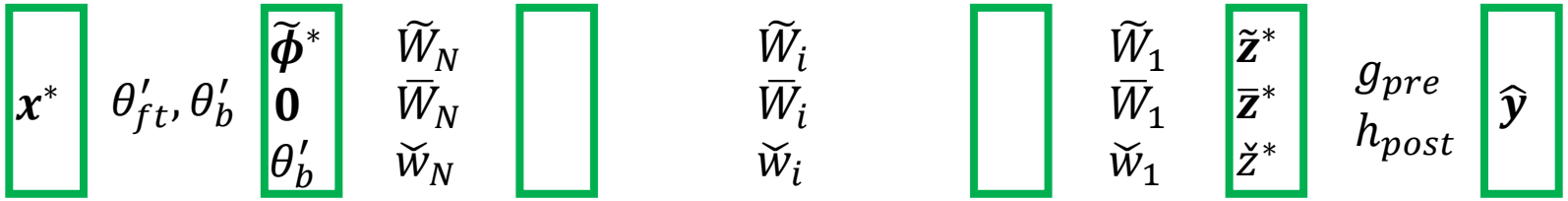


Show universality from model

- Change the form of $\bar{\boldsymbol{z}}^*$ with kernel form
  - $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y})\tilde{\phi}(\boldsymbol{x};\theta_{ft},\theta_b)^T B_i^T B_i \tilde{\phi}(\boldsymbol{x}^*;\theta'_{ft},\theta'_b)^T = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x},\boldsymbol{x}^*)$
- $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x},\boldsymbol{x}^*)$ has the all information of $\boldsymbol{x}, \boldsymbol{y}$ and $\mathbf{x}^*$.
- Assume $\bar{e}(\boldsymbol{y})$ is a linear function of $\boldsymbol{y}$ which can extract the original information $\boldsymbol{y}$.
- **Idea 1)**
  - Decompose index $i = 1, \dots, N$ to a product of $j = 0, \dots, J-1$ and $l = 0, \dots, L-1$, and then assign those to $\boldsymbol{x}$ and $\boldsymbol{x}^*$
- **Idea 2)**
  - Discretize $\boldsymbol{x}$ and $\boldsymbol{x}^*$ to j and l by kernel $k(\cdot,\cdot)$
- **Idea 3)**
  - Make new vector v, which contain all information of $\boldsymbol{x}, \boldsymbol{x}^*$ and $\boldsymbol{y}$ from $A_i, \bar{e}(\boldsymbol{y})$ and k_i

Show universality from model

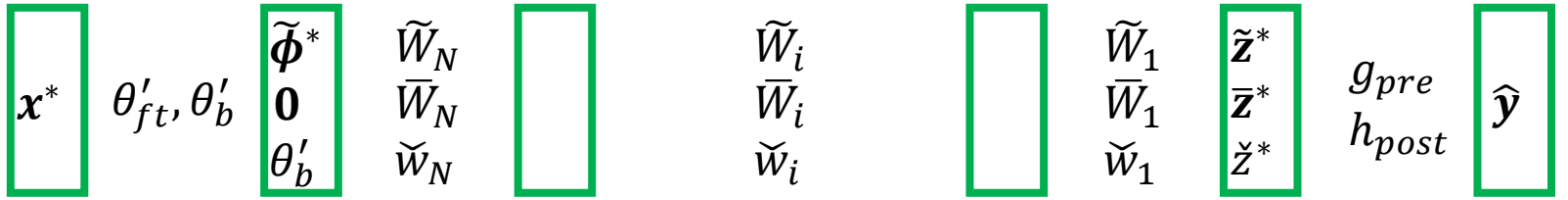
- **Idea 1)**
  - Decompose index $i = 1, \dots, N$ to a product of $j = 0, \dots, J-1$ and $l = 0, \dots, L-1$, and then assign those to $\boldsymbol{x}$ and $\boldsymbol{x}^*$
  - $\bar{\boldsymbol{z}}^* = -\alpha \Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x}, \boldsymbol{x}^*) = -\alpha \Sigma_{j=0}^{J-1} \Sigma_{l=0}^{L-1} A_{jl} \bar{e}(\boldsymbol{y}) k_{jl}(\boldsymbol{x}, \boldsymbol{x}^*)$
- **Idea 2)**
  - Discretize $\boldsymbol{x}$ and $\boldsymbol{x}^*$ to j and l by kernel $k(\cdot,\cdot)$
  - $k_{jk}(\boldsymbol{x}, \boldsymbol{x}^*) = \begin{cases} 1 & if\ discr(\boldsymbol{x}) = e_j\ and\ discr(\boldsymbol{x}^*) = e_l \\ 0 & otherwise \end{cases} \quad \cdots (6)$



Show universality from model

- **Idea 3)**
  - Make new vector v, which contain all information of $\boldsymbol{x}, \boldsymbol{x}^*$ *and* $\boldsymbol{y}$ from $A_i, \bar{e}(\boldsymbol{y})$ *and* k_i
  - Choose $\bar{e}(\boldsymbol{y})$ to be a linear function that outputs $\mathrm{J} * L$ stacked copies of $\boldsymbol{y}$
  - Define A_{jl} to be a select $\boldsymbol{y}$ at the position of $(j, l) = j + J * L$, $A_{jl} = \begin{bmatrix} \epsilon & \cdots & \epsilon \\ \vdots & 1+\epsilon & \vdots \\ \epsilon & \cdots & \epsilon \end{bmatrix}$, $1+\epsilon$ at position (j, l)
- As a result,
  - $\bar{\boldsymbol{z}}^* = -\alpha \Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x}, \boldsymbol{x}^*) = -\alpha \Sigma_{j=0}^{J-1} \Sigma_{l=0}^{L-1} A_{jl} \bar{e}(\boldsymbol{y}) k_{jl}(\boldsymbol{x}, \boldsymbol{x}^*)$

$= v(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{y}) \sim \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{y} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

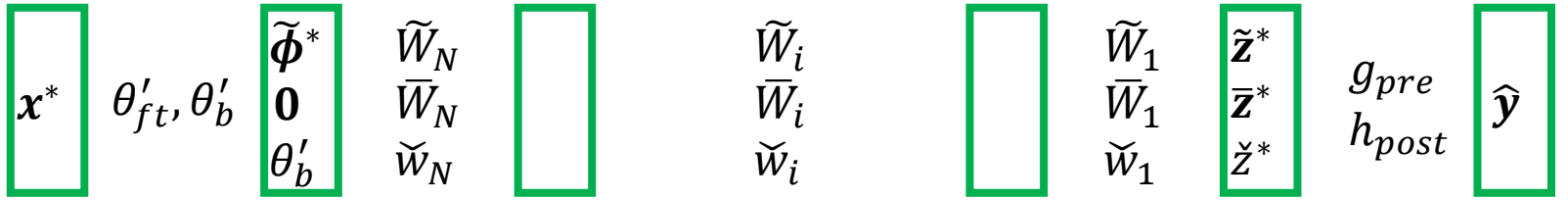


Show universality from model

- As a result,
  - $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y}) k_i(\boldsymbol{x}, \boldsymbol{x}^*) = -\alpha\Sigma_{j=0}^{J-1}\Sigma_{l=0}^{L-1} A_{jl} \bar{e}(\boldsymbol{y}) k_{jl}(\boldsymbol{x}, \boldsymbol{x}^*) = -\alpha v(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{y})$,

where $v(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{y}) \sim \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{y} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

- $\therefore \hat{f}(\boldsymbol{x}^*; \theta') \sim h_{post}(-\alpha v(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{y}); \theta_h)$
- It is a universal function approximator with respect to its inputs $(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{y})$



Experiments

- Q1) Is there empirical benefit to using one meta-learning approach versus another, and in which case?
- A1) Empirically show the inductive bias of gradient-based vs recurrent meta-learners
  - Explore the differences between gradient-based vs recurrent
    - A learner trained with MAML, improve or start to overfit after additional gradient steps.
    - Better few-shot learning performance on tasks outside of the training distribution?

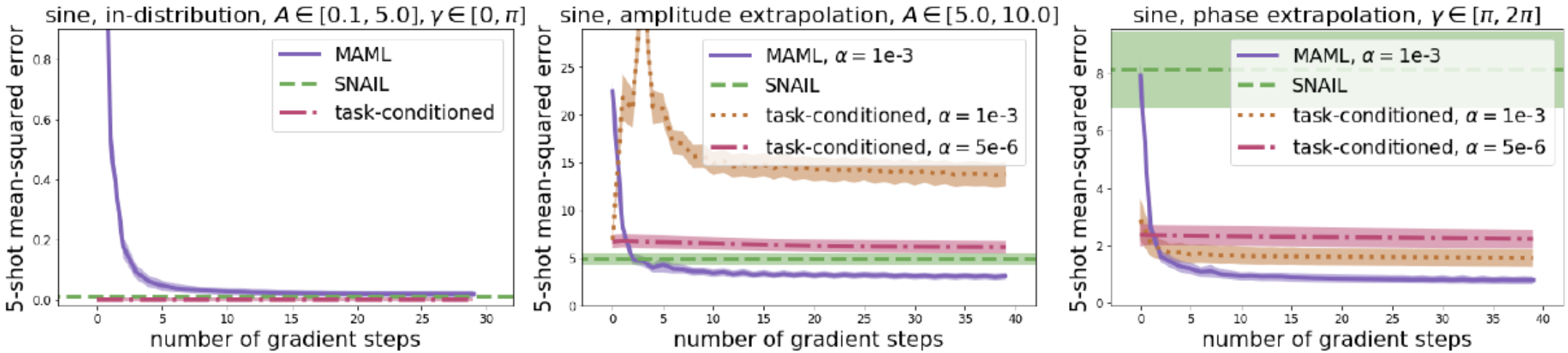


Figure 2: The effect of additional gradient steps at test time when attempting to solve new tasks. The MAML model, trained with 5 inner gradient steps, can further improve with more steps. All methods are provided with the same data – 5 examples – where each gradient step is computed using the same 5 datapoints.

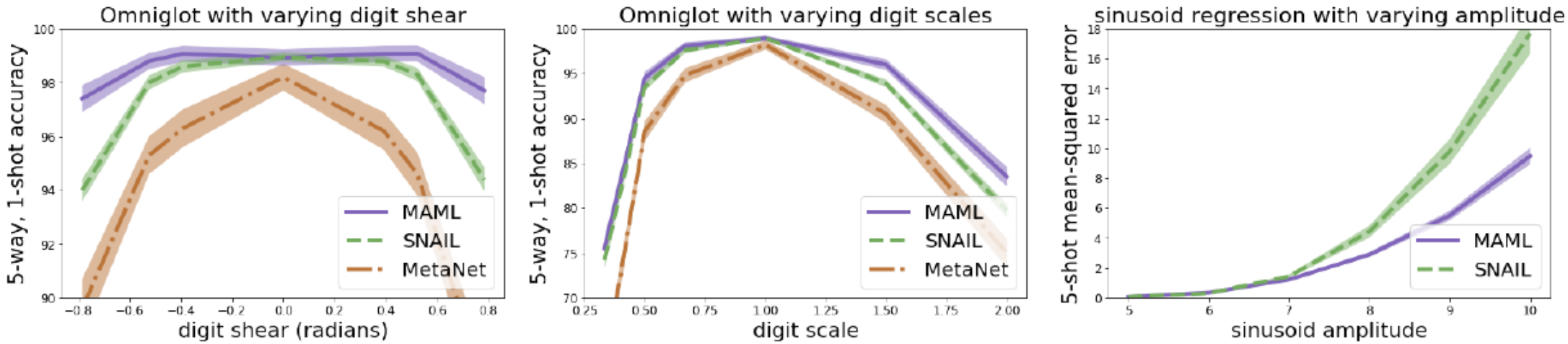


Figure 3: Learning performance on out-of-distribution tasks as a function of the task variability. Recurrent meta-learners such as SNAIL and MetaNet acquire learning strategies that are less generalizable than those learned with gradient-based meta-learning.

Experiments

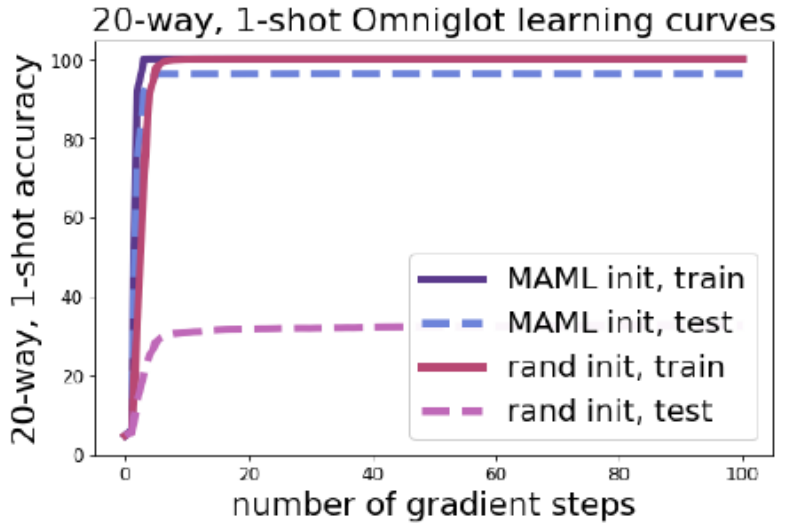


Figure 4: Comparison of finetuning from a MAML-initialized network and a network initialized randomly, trained from scratch. Both methods achieve about the same training accuracy. But, MAML also attains good test accuracy, while the network trained from scratch overfits catastrophically to the 20 examples. Interestingly, the MAML-initialized model does not begin to overfit, even though meta-training used 5 steps while the graph shows up to 100.

Experiments

- Q2) Theory suggest that deeper networks lead to increased expressive power for representing different learning procedures. Is it right?
- A1) Investigate the role of model depth in gradient-based meta-learning

Experiments

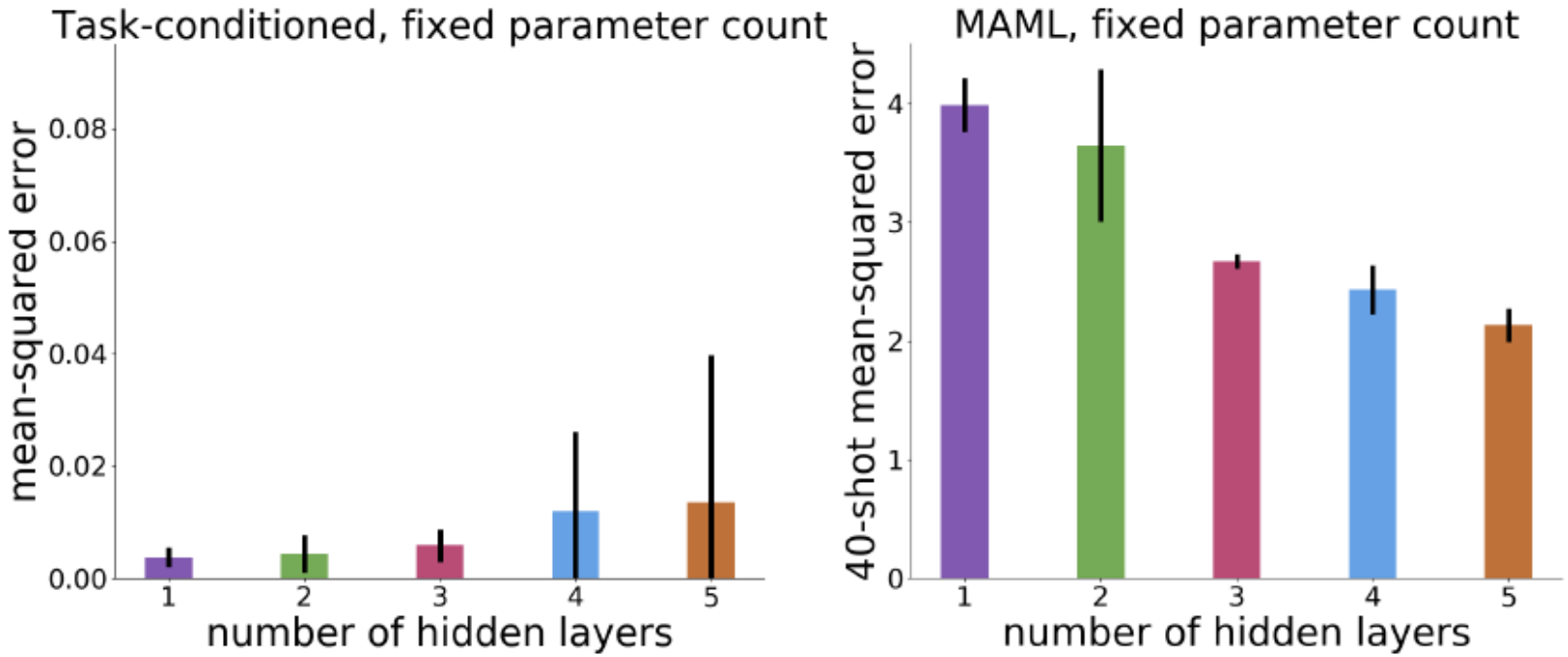


Figure 5: Comparison of depth while keeping the number of parameters constant. Task-conditioned models do not need more than one hidden layer, whereas meta-learning with MAML clearly benefits from additional depth. Error bars show standard deviation over three training runs.

Conclusion

- Meta-learners that use standard gradient descent with a sufficiently deep representation can approximate any learning procedure, and are equally expressive as recurrent learners.
- In experiments, MAML is more successful when faced with out-of-domain tasks compared to recurrent model
- We formalize what it means for a meta-learner to be able to approximate any learning algorithm in terms of its ability to represent functions of the dataset and test inputs.

Appendix

- A
  - we assume inputs and all pre-synaptic activations are non-negative, then deep ReLU act like deep linear networks. $\cdots (1)$
  - $\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1}\bar{W}_j\right)\left(\prod_{j=1}^{i-1}\bar{W}_j\right)^T \bar{e}(\boldsymbol{y}) \cdot \tilde{\phi}\left(\boldsymbol{x};\theta_{ft},\theta_b\right)^T \left(\prod_{j=i+1}^{N}\widetilde{W}_j\right)^T \left(\prod_{j=1}^{i-1}\widetilde{W}_j\right)\tilde{\phi}\left(\boldsymbol{x}^*;\theta'_{ft},\theta'_b\right)$
    $= -\alpha\Sigma_{i=1}^{N} A_i \bar{e}(\boldsymbol{y})\tilde{\phi}\left(\boldsymbol{x};\theta_{ft},\theta_b\right)^T B_i^T B_i \tilde{\phi}\left(\boldsymbol{x}^*;\theta'_{ft},\theta'_b\right)^T \quad \cdots (3)$
- B
  - $\nabla_z \mathcal{L}\left(y,\hat{f}(\boldsymbol{x};\theta)\right) \coloneqq \begin{bmatrix}\mathbf{0}\\ \bar{e}(y)\\ \check{e}(y)\end{bmatrix} \cdots (2)$
  - $f_{out}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}\\ \bar{\boldsymbol{z}}\\ \check{z}\end{bmatrix};\theta_{out}\right) = \mathbf{1}_{\bar{z}=0} g_{pre}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}\\ \bar{\boldsymbol{z}}\\ \check{z}\end{bmatrix};\theta_g\right) + \mathbf{1}_{\bar{z}\neq 0} h_{post}(\bar{\boldsymbol{z}};\theta_h), \cdots (4), \qquad f_{out}\left(\begin{bmatrix}\tilde{\boldsymbol{z}}^*\\ \bar{\boldsymbol{z}}^*\\ \check{z}^*\end{bmatrix};\theta'_{out}\right) = h_{post}(\bar{\boldsymbol{z}}^*;\theta_h), \cdots (5)^*$
- C
  - $k_{jk}(\boldsymbol{x},\boldsymbol{x}^*) = \begin{cases}1 & if\ discr(\boldsymbol{x}) = e_j\ and\ discr(\boldsymbol{x}^*) = e_l\\ 0 & otherwise\end{cases} \cdots (6)$

Appendix A

$$\bar{\boldsymbol{z}}^* = -\alpha\Sigma_{i=1}^{N}\left(\prod_{j=1}^{i-1}\bar{W}_j\right)\left(\prod_{j=1}^{i-1}\bar{W}_j\right)^T \bar{e}(\boldsymbol{y})\cdot\tilde{\phi}(\boldsymbol{x};\theta_{ft},\theta_b)^T\left(\prod_{j=i+1}^{N}\widetilde{W}_j\right)^T\left(\prod_{j=1}^{i-1}\widetilde{W}_j\right)\tilde{\phi}(\boldsymbol{x}^*;\theta'_{ft},\theta'_b)$$

$$= -\alpha\Sigma_{i=1}^{N}A_i\bar{e}(\boldsymbol{y})\tilde{\phi}(\boldsymbol{x};\theta_{ft},\theta_b)^T B_i^T B_i\tilde{\phi}(\boldsymbol{x}^*;\theta'_{ft},\theta'_b)^T$$

- Choose all $\widetilde{W}_i$ and $\bar{W}_i$ to be a square and full rank.
- Set $\widetilde{W}_i = \widetilde{M}_i\widetilde{M}_{i+1}^{-1}$ and $\bar{W}_i = \bar{M}_{i-1}^{-1}\bar{M}_i$ then $\prod_{j=i+1}^{N}\widetilde{W}_j = \widetilde{M}_{i+1}$, $\prod_{j=1}^{i-1}\bar{W}_j = \bar{M}_{i-1}$ ($\widetilde{M}_{N+1} = \boldsymbol{I}$, $\bar{M}_0 = \boldsymbol{I}$)
- Set $A_1 = \boldsymbol{I}, A_i = \bar{M}_{i-1}\bar{M}_{i-1}^T$ and $B_i = \widetilde{M}_{i+1}, B_N = \boldsymbol{I}$

Appendix A

- All inputs are non-negative:
  - Input ϕ consist of three term, (discretization, constant 0, 0 before update and not used afterward)
  - All inputs are non-negative before and after update.
- All pre-synaptic activations are non-negative:
  - It is sufficient to show that products of W_i, $\prod_{i=j}^{N} W_i \ (1 \leq j \leq N)$ are positive semi-definite.
  - So, show that $\prod_{i=j}^{N} \widetilde{W}_i$, $\prod_{i=j}^{N} \bar{W}_i$, $\prod_{i=j}^{N} \breve{w}_i \ (1 \leq j \leq N)$ are positive semi-definite.
  - We define $\prod_{i=j}^{N} \widetilde{W}_i = \widetilde{M}_{j+1} = B_i$, and B_i is set to be a positive definite.
  - We define $\bar{W}_i = \bar{M}_{i-1}^{-1} \bar{M}_i$ where $A_i = \bar{M}_{i-1} \bar{M}_{i-1}^{T}$, so each $\bar{M}_i$ is also symmetric positive definite and $\bar{W}_i$ is positive definite.
  - Purpose of $\breve{w}_i$ is to provide nonzero gradient to the input θ_b, thus positive value for each $\breve{w}_i$ will suffice.

Appendix B

$$\nabla_z \mathcal{L}\left(y, \hat{f}(\boldsymbol{x};\theta)\right) \coloneqq \begin{bmatrix} \mathbf{0} \\ \bar{e}(y) \\ \check{e}(y) \end{bmatrix} \cdots (2) \qquad \nabla_z \mathcal{L}\left(y, \hat{f}(\boldsymbol{x};\theta)\right) = \nabla_z \hat{f}(\boldsymbol{x};\theta) \cdot \nabla_{\hat{y}} \mathcal{L}(y,\hat{y}) = \begin{bmatrix} \mathbf{0} \\ \bar{e}(y) \\ \check{e}(y) \end{bmatrix}$$

$$\hat{f}(\boldsymbol{x};\theta) = \hat{f}_{out}(\boldsymbol{z};\theta_{out}) = g_{pre}(\boldsymbol{z};\theta_g)$$

- Because we assume g_{pre} as a linear function of θ_g,
  let $g_{pre}(\boldsymbol{z};\theta_g) = \begin{bmatrix} \widetilde{W}_g & \bar{W}_g & \check{w}_g \end{bmatrix} \boldsymbol{z} = \widetilde{W}_g \tilde{\boldsymbol{z}} + \bar{W}_g \bar{\boldsymbol{z}}_g + \check{w}_g \check{z}$
- To make top element of $e(\boldsymbol{x}, \boldsymbol{y})$ to be $\mathbf{0}$, let $\widetilde{W}_g = 0$, which make $\hat{y} = 0$.
- Then, $\bar{e}(y) = \bar{W}_g^T \cdot \nabla_{\hat{y}} \mathcal{L}(y, 0)$
- For any linear loss function for y, $\bar{e}(y) = Ay$,
  to extract all information of y, A has to be invertible.
- Sufficient loss function: standard mean-squared error or softmax cross entropy.

Appendix C

$$k_{jk}(\boldsymbol{x}, \boldsymbol{x}^*) = \tilde{\phi}(\boldsymbol{x}; \theta_{ft}, \theta_b)^T B_{jl}^T B_{jl} \tilde{\phi}(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b)^T = \begin{cases} 1 & if\ discr(\boldsymbol{x}) = e_j\ and\ discr(\boldsymbol{x}^*) = e_l \\ 0 & otherwise \end{cases}$$

- Choose $\tilde{\phi}$ and B_{jl}:

$$\tilde{\phi}(\cdot; \theta_{ft}, \theta_b) \coloneqq \begin{cases} \begin{bmatrix} discr(\cdot) \\ 0 \end{bmatrix} if\ \theta_b = 0 \\ \begin{bmatrix} 0 \\ discr(\cdot) \end{bmatrix} otherwise \end{cases} \qquad B_{jl} = \begin{bmatrix} E_{jj} & E_{jl} \\ E_{lj} & 0 \end{bmatrix} + \epsilon I$$

where E_{ik} denote the matrix a 1 at (i, k) and 0 otherwise

- Then,

$$\tilde{\phi}(\boldsymbol{x}; \theta_{ft}, \theta_b)^T B_{jl}^T \sim \begin{cases} [e_j \quad 0] & if\ discr(\boldsymbol{x}) = e_j \\ [0 \quad 0] & otherwise \end{cases} \qquad B_{jl} \tilde{\phi}(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b)^T \sim \begin{cases} [e_l \quad 0]^T & if\ discr(\boldsymbol{x}^*) = e_l \\ [0 \quad 0]^T & otherwise \end{cases}$$

Appendix C

$$k_{jk}(\boldsymbol{x}, \boldsymbol{x}^*) = \tilde{\phi}\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T B_{jl}^T B_{jl} \tilde{\phi}\left(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b\right)^T = \begin{cases} 1 & if\ discr(\boldsymbol{x}) = e_j\ and\ discr(\boldsymbol{x}^*) = e_l \\ 0 & otherwise \end{cases}$$

$$\tilde{\phi}(\cdot; \theta_{ft}, \theta_b) \coloneqq \begin{cases} \begin{bmatrix} discr(\cdot) \\ 0 \end{bmatrix} if\ \theta_b = 0 \\ \begin{bmatrix} 0 \\ discr(\cdot) \end{bmatrix} otherwise \end{cases} \qquad B_{jl} = \begin{bmatrix} E_{jj} & E_{jl} \\ E_{lj} & 0 \end{bmatrix} + \epsilon I$$

$$\tilde{\phi}\left(\boldsymbol{x}; \theta_{ft}, \theta_b\right)^T B_{jl}^T \sim \begin{cases} [e_j \quad 0] & if\ discr(\boldsymbol{x}) = e_j \\ [0 \quad 0] & otherwise \end{cases} \qquad B_{jl} \tilde{\phi}\left(\boldsymbol{x}^*; \theta'_{ft}, \theta'_b\right)^T \sim \begin{cases} [e_l \quad 0]^T & if\ discr(\boldsymbol{x}) = e_l \\ [0 \quad 0]^T & otherwise \end{cases}$$

- So,

$$k_{jk}(\boldsymbol{x}, \boldsymbol{x}^*) \sim \begin{cases} [e_j \quad 0] \begin{bmatrix} e_l \\ 0 \end{bmatrix} & if\ discr(\boldsymbol{x}) = e_j\ and\ discr(x^*) = e_l \\ 0 & otherwise \end{cases}$$

Thank you!