Model-Agnostic Meta-Learning (MAML) for Fast Adaptation of Deep Networks

from Chelsea Finn, Pieter Abbeel and Sergey Levine

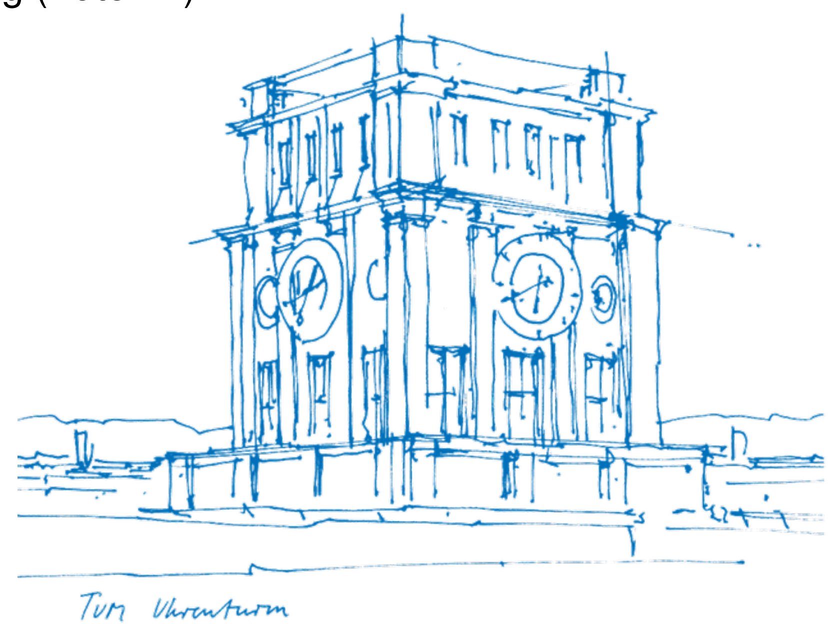
Philip Müller

Technical University Munich

Department of Informatics

Seminar Recent trends in Automated Machine Learning (AutoML)

Munich, 18th July 2019



Few-Shot Learning

(Normal) Learning:

- **Goal:** Learn some function or behaviour for one given task $\mathcal{T}$ that can be applied at test time
- **Given at training:** (Large) set of training samples/trajectories
- **Given at test:** One (or more) sample(s)

Few-Shot Learning

(Normal) Learning:

- **Goal:** Learn some function or behaviour for one given task $\mathcal{T}$ that can be applied at test time
- **Given at training:** (Large) set of training samples/trajectories
- **Given at test:** One (or more) sample(s)

Few-shot (K-shot) Learning:

- **Goal:** Learning to adapt quickly (at test time) to new task $\mathcal{T}_i$
  - After given K samples
  - Continue adaption when more samples ($> K$) are available
- **Given at training:** Set of tasks $\mathcal{T}_i$ from distribution $p(\mathcal{T})$
- **Given at adaption:** One task $\mathcal{T}_i$ with only K training samples
- **Given at test:** One (or more) sample(s)

How does a few-shot learning dataset look like?

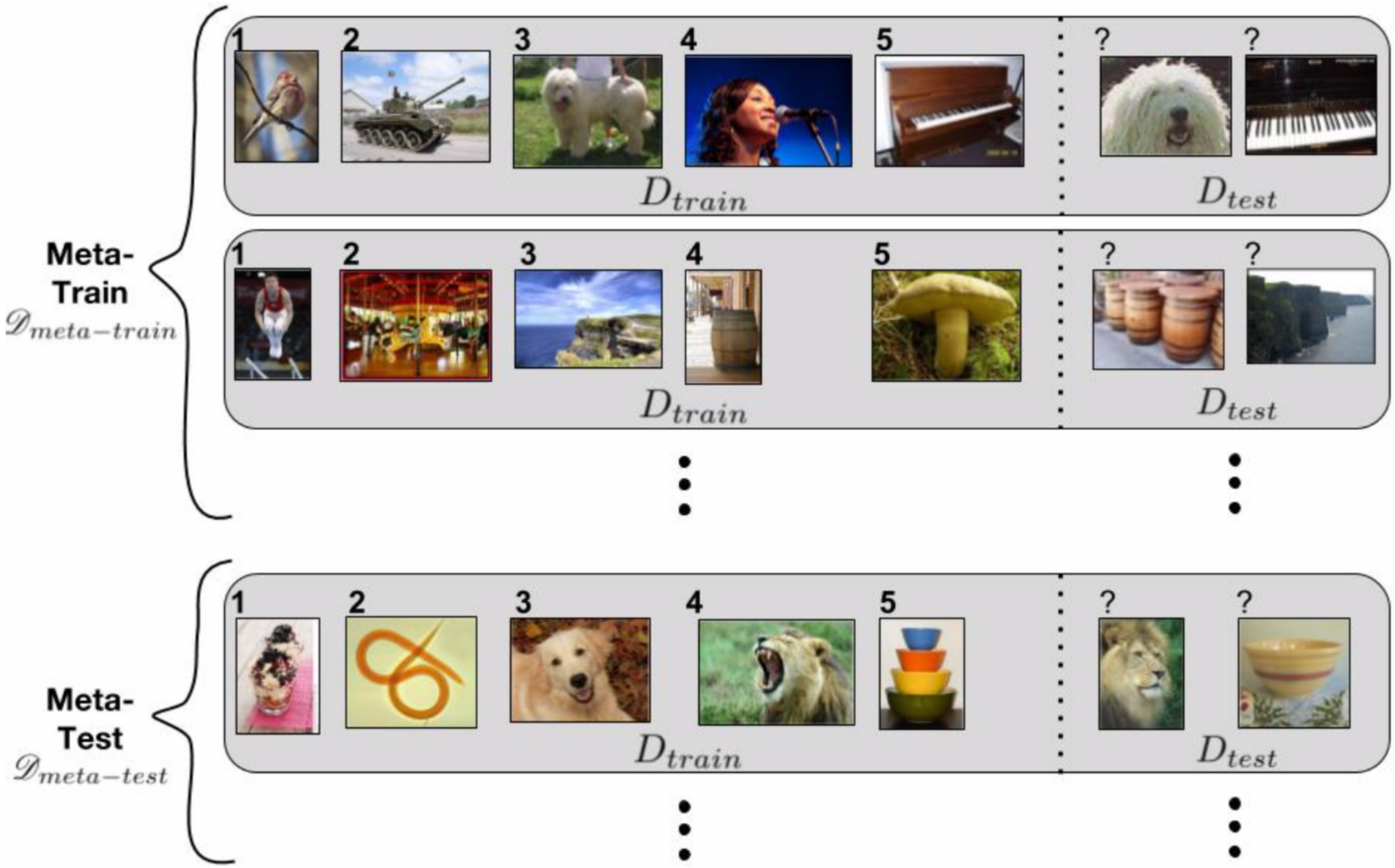


Figure: Meta-Training and Meta-Test dataset example for image classification, taken from [RL17]

How can we improve few-shot learning?

How can we improve few-shot learning?

Why are humans good at few-shot learning

- Integrate a lot of prior experience with the few samples of new information
  $\Rightarrow$ e.g. something Bayesian inference like
- Parameter initialization is crucial

How can we improve few-shot learning?

Why are humans good at few-shot learning

- Integrate a lot of prior experience with the few samples of new information
  $\Rightarrow$ e.g. something Bayesian inference like
- Parameter initialization is crucial

Challenges

- Bayesian inference in practice not feasible
- Gradient-based methods are not designed for constrained number of steps/samples
- How to integrate prior experience?
- Avoid overfitting to new data

Meta-Learning – Learning how to learn

Basics

- Meta-learner (agent) contains learning sub-system *(named: learner or model)*
- Meta-learner adapts/trains learner using experience

Meta-Learning – Learning how to learn

Basics

- Meta-learner (agent) contains learning sub-system *(named: learner or model)*
- Meta-learner adapts/trains learner using experience

Experience gained from Meta-knowledge:

- Previous episodes on same learning task and dataset
- **(Many) Different learning tasks with different datasets** (e.g. different problems or domains)

Meta-Learning – Learning how to learn

Basics

- Meta-learner (agent) contains learning sub-system *(named: learner or model)*
- Meta-learner adapts/trains learner using experience

Experience gained from Meta-knowledge:

- Previous episodes on same learning task and dataset
- **(Many) Different learning tasks with different datasets** (e.g. different problems or domains)

What can be meta-learned:

- Optimizer / Optimizer Parameters
- General features / metrics relevant for many tasks in task distribution
- **Initial parameters**

Model-Agnostic Meta-Learning (MAML) for Fast Adaptation of Deep Networks

from Chelsea Finn, Pieter Abbeel and Sergey Levine

Model-Agnostic Meta-Learning (MAML) for Fast Adaptation of Deep Networks

from Chelsea Finn, Pieter Abbeel and Sergey Levine

- **Adapt initialization:** Initial parameters are meta-learned

Model-Agnostic Meta-Learning (MAML) for Fast Adaptation of Deep Networks

from Chelsea Finn, Pieter Abbeel and Sergey Levine

- **Adapt initialization:** Initial parameters are meta-learned
- Optimize for model that can be fine-tuned fast for many tasks without overfitting
  - Not perfect for a single task
  - Maximum (average) performance on many tasks after short training (few samples, few gradient descent steps)

Model-Agnostic Meta-Learning (MAML) for Fast Adaptation of Deep Networks

from Chelsea Finn, Pieter Abbeel and Sergey Levine

- **Adapt initialization:** Initial parameters are meta-learned
- Optimize for model that can be fine-tuned fast for many tasks without overfitting
  - Not perfect for a single task
  - Maximum (average) performance on many tasks after short training (few samples, few gradient descent steps)
- Use gradient-based meta-training with gradient-based training in inner loop
  - **Loss:** task-specific
  - **Meta-Loss:** Performance of trained model on task-specific validation set

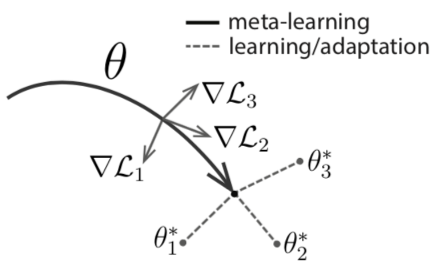


Figure: Parameter path, taken from [FAL17]

Algorithm [FAL17]

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Algorithm [FAL17]

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

$$
\begin{aligned}
& \nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \\
=& \nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot \nabla_{\boldsymbol{\theta}} \boldsymbol{\theta}_i' \\
=& \nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot \nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\text{train})}_{\mathcal{T}_i}(\boldsymbol{\theta})) \\
=& \nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot (\boldsymbol{I} - \alpha \nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{(\text{train})}_{\mathcal{T}_i}(\boldsymbol{\theta}))
\end{aligned}
$$

Algorithm [FAL17]

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

$$\begin{aligned}
&\nabla_{\theta} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \\
=&\nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot \nabla_{\boldsymbol{\theta}} \boldsymbol{\theta}_i' \\
=&\nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\text{train})}_{\mathcal{T}_i}(\boldsymbol{\theta})) \\
=&\nabla_{\boldsymbol{\theta}_i'} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \cdot (\boldsymbol{I} - \alpha \nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{(\text{train})}_{\mathcal{T}_i}(\boldsymbol{\theta}))
\end{aligned}$$

- Gradient of gradient (second derivative) involved
- Use Hessian-vector products:
  - Additional backward pass required (suppored out-of-the-box e.g. by Tensorflow)
  - Computational very expensive

Alternative: First-Order MAML (FOMAML)

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{(val)}}_{\mathcal{T}_i}(\boldsymbol{\theta}'_{\boldsymbol{i}}) \approx \nabla_{\boldsymbol{\theta}'_{\boldsymbol{i}}} \mathcal{L}^{\text{(val)}}_{\mathcal{T}_i}(\boldsymbol{\theta}'_{\boldsymbol{i}})$$

- First-Order Approximation, drop second derivative
- Only compute gradient at position $\boldsymbol{\theta}'_{\boldsymbol{i}}$ – after the update
- Less computational expensive (about 33% speedup)

Alternative: First-Order MAML (FOMAML)

$$\nabla_{\theta}\mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i') \approx \nabla_{\boldsymbol{\theta}_i'}\mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}_i')$$

- First-Order Approximation, drop second derivative
- Only compute gradient at position $\boldsymbol{\theta}_i'$ – after the update
- Less computational expensive (about 33% speedup)

Results:

- Achieves similar (almost as good) results
- Most power of MAML comes from gradient of objective after update of parameters – not from derivative through gradient update

Alternative: First-Order MAML (FOMAML)

$$\nabla_{\theta} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}'_i) \approx \nabla_{\boldsymbol{\theta}'_i} \mathcal{L}^{(\text{val})}_{\mathcal{T}_i}(\boldsymbol{\theta}'_i)$$

- First-Order Approximation, drop second derivative
- Only compute gradient at position $\boldsymbol{\theta}'_i$ – after the update
- Less computational expensive (about 33% speedup)

Results:

- Achieves similar (almost as good) results
- Most power of MAML comes from gradient of objective after update of parameters – not from derivative through gradient update

Explanation:

- ReLU NNs locally almost linear $\Rightarrow$ second derivative near 0

Inner Loop vs. Outer loop

Learning of Task $\mathcal{T}_i$

Meta-Learning [FAL17]

Inner Loop vs. Outer loop

Learning of Task $\mathcal{T}_i$		Meta-Learning [FAL17]
	Data Set	
$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \dots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$		$\{\mathcal{T}_i\}_{i=1}^{N}$

Inner Loop vs. Outer loop

Learning of Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	
$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \dots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	
$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \vert \boldsymbol{x}_t, \boldsymbol{a}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$

Inner Loop vs. Outer loop

	Learning of Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \dots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	$\mathcal{L}_{\mathcal{T}_i}(f_\theta)$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_{\boldsymbol{\theta}})}\right)$

Inner Loop vs. Outer loop

	Learning of Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \dots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \| \boldsymbol{x}_t, \boldsymbol{a}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	$\mathcal{L}_{\mathcal{T}_i}(f_\theta)$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)}\right)$
Optimization	$\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}_i'})$

Inner Loop vs. Outer loop

Learning of Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	
$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \ldots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^K$	$\{\mathcal{T}_i\}_{i=1}^N$
Data Distribution	
$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \vert \boldsymbol{x}_t, \boldsymbol{a}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	
$\mathcal{L}_{\mathcal{T}_i}(f_\theta)$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)}\right)$
Optimization	
$\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'})$
Forward Pass	
Normal forward-pass	Several model training steps (of task)

Inner Loop vs. Outer loop

Learning of Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	
$\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \ldots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	
$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \| \boldsymbol{x}_t, \boldsymbol{a}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	
$\mathcal{L}_{\mathcal{T}_i}(f_\theta)$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'})$ $= \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)}\right)$
Optimization	
$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)$	$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'})$
Forward Pass	
Normal forward-pass	Several model training steps (of task)
Backward Pass	
Normal backward-pass (backpropagation)	Backprop gradient of meta-loss through loss on validation data and training process to initial weights

Inner Loop vs. Outer loop

	Supervised Regression Task $\mathcal{T}_i$	**Meta-Learning [FAL17]**
Data Set	K i.i.d. obervations $(H = 1)$: $\{(\boldsymbol{x}^{(\mathcal{T}_i,j)}, \boldsymbol{y}^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	$(\boldsymbol{x}, \boldsymbol{y}) \sim q_{\mathcal{T}_i}(\boldsymbol{x}, \boldsymbol{y})$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	$\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}) = \sum_{(\boldsymbol{x}^{(\mathcal{T}_i,j)}, \boldsymbol{y}^{(\mathcal{T}_i,j)}) \sim q_{\mathcal{T}_i}} \lVert f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(\mathcal{T}_i,j)}) - \boldsymbol{y}^{(\mathcal{T}_i,j)} \rVert_2^2$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_{\boldsymbol{\theta}})}\right)$
Optimization	$\boldsymbol{\theta}'_i \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_{\boldsymbol{\theta}})$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}'_i})$
Forward Pass	Normal forward-pass	Several model training steps (of task)
Backward Pass	Normal backward-pass (backpropagation)	Backprop gradient of meta-loss through loss on validation data and training process to initial weights

Inner Loop vs. Outer loop

Supervised Classification Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	
K i.i.d. obervations $(H = 1)$: $\{(\boldsymbol{x}^{(\mathcal{T}_i,j)}, \boldsymbol{y}^{(\mathcal{T}_i,j)})\}_{j=1}^{K}$	$\{\mathcal{T}_i\}_{i=1}^{N}$
Data Distribution	
$(\boldsymbol{x}, \boldsymbol{y}) \sim q_{\mathcal{T}_i}(\boldsymbol{x}, \boldsymbol{y})$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	
$\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}) = \sum_{(\boldsymbol{x}^{(\mathcal{T}_i,j)}, \boldsymbol{y}^{(\mathcal{T}_i,j)}) \sim q_{\mathcal{T}_i}} \boldsymbol{y}^{(\mathcal{T}_i,j)} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(\mathcal{T}_i,j)})$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_{\boldsymbol{\theta}})}\right)$
Optimization	
$\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_{\boldsymbol{\theta}})$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\boldsymbol{\theta}_i'})$
Forward Pass	
Normal forward-pass	Several model training steps (of task)
Backward Pass	
Normal backward-pass (backpropagation)	Backprop gradient of meta-loss through loss on validation data and training process to initial weights

Inner Loop vs. Outer loop

Reinforcement Learning Task $\mathcal{T}_i$	Meta-Learning [FAL17]
Data Set	
K rollouts of episode length H using policy f_θ : $\{(\boldsymbol{x}_1^{(\mathcal{T}_i,j)}, \boldsymbol{a}_1^{(\mathcal{T}_i,j)}, \ldots, \boldsymbol{x}_H^{(\mathcal{T}_i,j)}, \boldsymbol{a}_H^{(\mathcal{T}_i,j)})\}_{j=1}^K$	$\{\mathcal{T}_i\}_{i=1}^N$
Data Distribution	
$\boldsymbol{x}_1 \sim q_{\mathcal{T}_i}(\boldsymbol{x}_1), \boldsymbol{x}_{t+1} \sim q_{\mathcal{T}_i}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{a}_t), \boldsymbol{a}_t \sim f_\theta(\boldsymbol{a}_t \mid \boldsymbol{x}_t)$	$\mathcal{T}_i \sim p(\mathcal{T})$
Loss	
$\mathcal{L}_{\mathcal{T}_i}(f_\theta) = -\mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{a}_t \sim q_{\mathcal{T}_i}, f_\theta}\left(\sum_{t=1}^H R_{\mathcal{T}_i}(\boldsymbol{x}_t, \boldsymbol{a}_t)\right)$	$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}\left(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)}\right)$
Optimization	
$\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{(\text{train})}(f_\theta)$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{(\text{val})}(f_{\theta_i'})$
Forward Pass	
Normal forward-pass	Several model training steps (of task)
Backward Pass	
Normal backward-pass (backpropagation)	Backprop gradient of meta-loss through loss on validation data and training process to initial weights

Experiments

Classification

- **Tasks:** Few-shot image recognition/classification: given a single image, classify it
- **Meta-Training:** each task with *N* unseen classes, sampled from larger set of classes
- **Training:** using *K* different instances for each of the *N* classes of the task in each gradient step

Omniglot (Lake et al., 2011)	5-way Accuracy 1-shot	5-way Accuracy 5-shot	20-way Accuracy 1-shot	20-way Accuracy 5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MANN, no conv (ours)	**89.7 $\pm$ 1.1%**	**97.5 $\pm$ 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 $\pm$ 0.4%**	**99.9 $\pm$ 0.1%**	**95.8 $\pm$ 0.3%**	**98.9 $\pm$ 0.2%**

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy 1-shot	5-way Accuracy 5-shot
fine-tuning baseline	28.86 $\pm$ 0.54%	49.79 $\pm$ 0.79%
nearest neighbor baseline	41.08 $\pm$ 0.70%	51.04 $\pm$ 0.65%
matching nets (Vinyals et al., 2016)	43.56 $\pm$ 0.84%	55.31 $\pm$ 0.73%
meta-learner LSTM (Ravi & Larochelle, 2017)	43.44 $\pm$ 0.77%	60.60 $\pm$ 0.71%
MAML, first order approx. (ours)	**48.07 $\pm$ 1.75%**	**63.15 $\pm$ 0.91%**
MAML (ours)	**48.70 $\pm$ 1.84%**	**63.11 $\pm$ 0.92%**

Figure: Results of image classification, taken from [FAL17]

Experiments

Regression

- **Tasks:** Sine-wave regression: given x regress y of task-specific sine wave
- **Meta-Training:** each task is sine-wave with different amplitude and phase
- **Training:** using same K samples on unseen wave in each gradient step

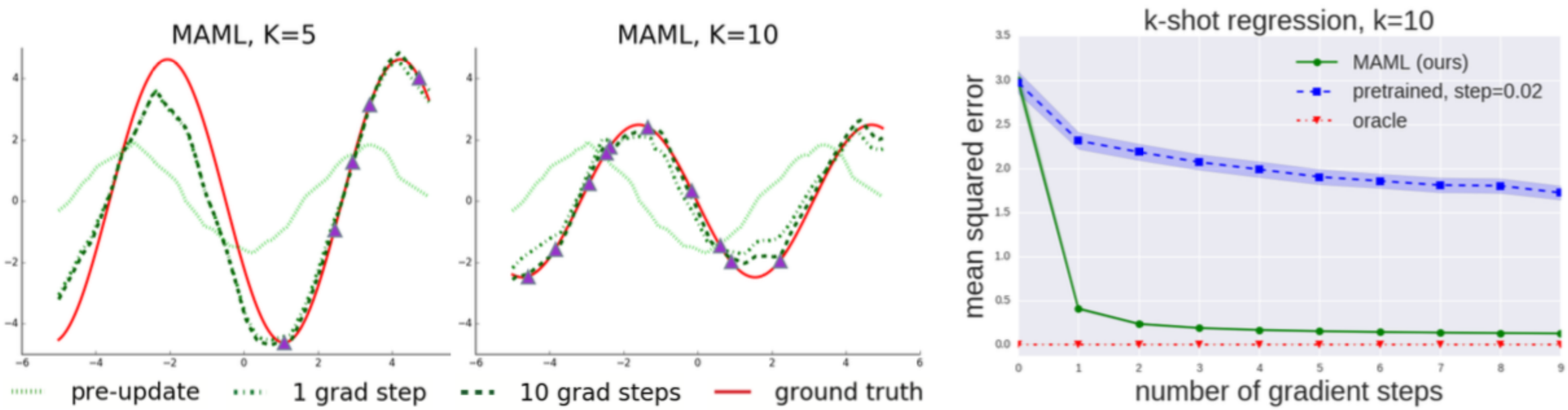


Figure: Results of sine-wave regression, taken from [FAL17]

Experiments

Reinforcement Learning

- **Optimizer:** REINFORCE
- **Meta-Optimizer:** Trut-region policy optimization (TRPO) – with finite differences for Hessian-vector in TRPO (to not compute third derivatives)
- **2D-navigation Tasks:** move to task-specific goal position by controlling velocity
- **Locomotion Tasks:** simulated robot learns to run into task-specific direction with task-specified velocity
- **Meta-Training:** each task with different parameters: goal postition / target direction + velocity
- **Training:** using K rollouts on specific task

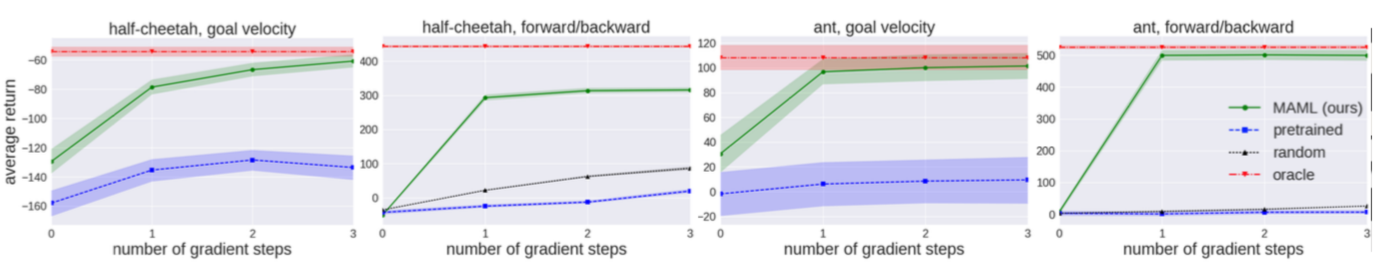


Figure: Results on different locomotion tasks, taken from [FAL17]

Discussion and Conclusion

Benefits of Meta-Learning

- Small datasets and computational costs for training meta-trained model

Discussion and Conclusion

Benefits of Meta-Learning

- Small datasets and computational costs for training meta-trained model

Benefits of MAML

- Model Agnostic:
  - Few constraints on model/method: parameterized and smooth enough loss
  - Many different learning tasks: supervised, reinforcement, . . .
  - No constraints on model-architecture: fully connected, convolutional, recurrent, . . .
- No additional learned parameters
- Use of gradient-descent like optimizers for meta-learning

Discussion and Conclusion

Benefits of Meta-Learning

- Small datasets and computational costs for training meta-trained model

Benefits of MAML

- Model Agnostic:
  - Few constraints on model/method: parameterized and smooth enough loss
  - Many different learning tasks: supervised, reinforcement, ...
  - No constraints on model-architecture: fully connected, convolutional, recurrent, ...
- No additional learned parameters
- Use of gradient-descent like optimizers for meta-learning

Drawbacks of Meta-Learning

- Very large training sets and computational costs for meta-training

Discussion and Conclusion

Benefits of Meta-Learning

- Small datasets and computational costs for training meta-trained model

Benefits of MAML

- Model Agnostic:
  - Few constraints on model/method: parameterized and smooth enough loss
  - Many different learning tasks: supervised, reinforcement, . . .
  - No constraints on model-architecture: fully connected, convolutional, recurrent, . . .
- No additional learned parameters
- Use of gradient-descent like optimizers for meta-learning

Drawbacks of Meta-Learning

- Very large training sets and computational costs for meta-training

Drawbacks of MAML

- Computational expensive (second derivative)

References (1)

[Dou18] Firdaouss Doukkali. *What is Model-Agnostic Meta-learning (MAML)?* Jan. 24, 2018. URL: https://towardsdatascience.com/model-agnostic-meta-learning-maml-8a245d9bc4ac (visited on 05/08/2019).

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017).

[LBG15] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. "Metalearning: a survey of trends and technologies". In: *Artif Intell Rev* 44.1 (2015), pp. 117–130.

[NAS18] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *CoRR* abs/1803.02999 (2018).

[RL17] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *ICLR* (2017).

[Wen18] Lilian Weng. *Meta-Learning: Learning to Learn Fast*. Nov. 30, 2018. URL: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html (visited on 05/08/2019).

[Wol18] Thomas Wolf. *From zero to research — An introduction to Meta-learning*. Apr. 3, 2018. URL: https://medium.com/huggingface/from-zero-to-research-an-introduction-to-meta-learning-8e16e677f78a (visited on 05/08/2019).