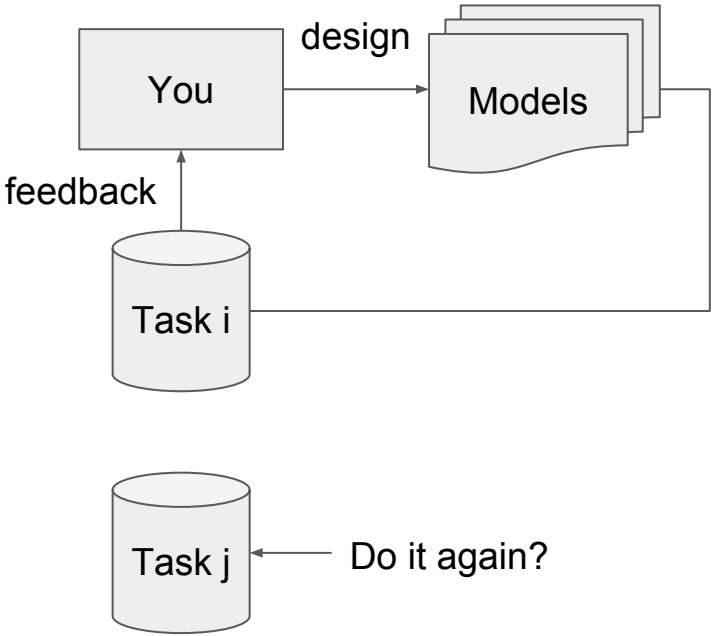
Meta Learning

Introduction

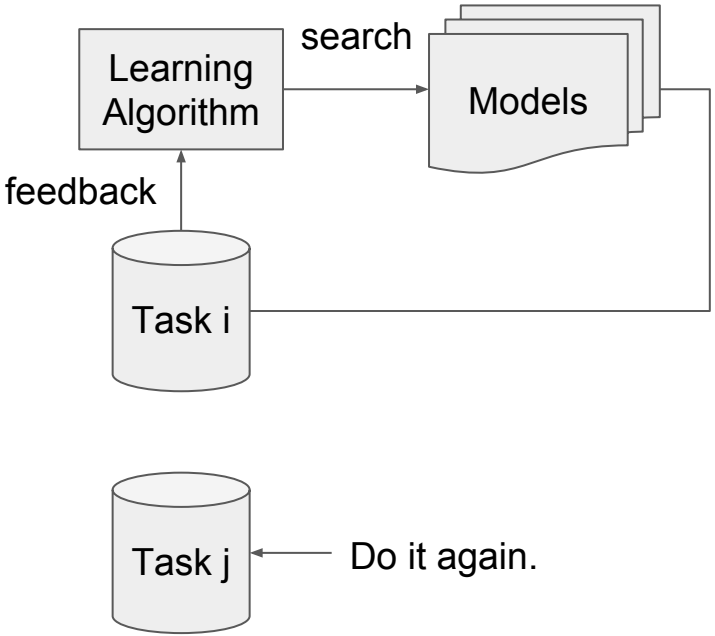
Then no *wonder* I can catch up with you so fast after you've had four years of biology." They had wasted all their time memorizing stuff like that, when it could be looked up in fifteen minutes.”

- Meta Learning: Learning to learn
- Big picture:
  - we want our learning algorithms to get better at learning with experience
- Towards AGI goal:
  - cannot just memorize task after task and start from scratch each time

Designing a Task Solver

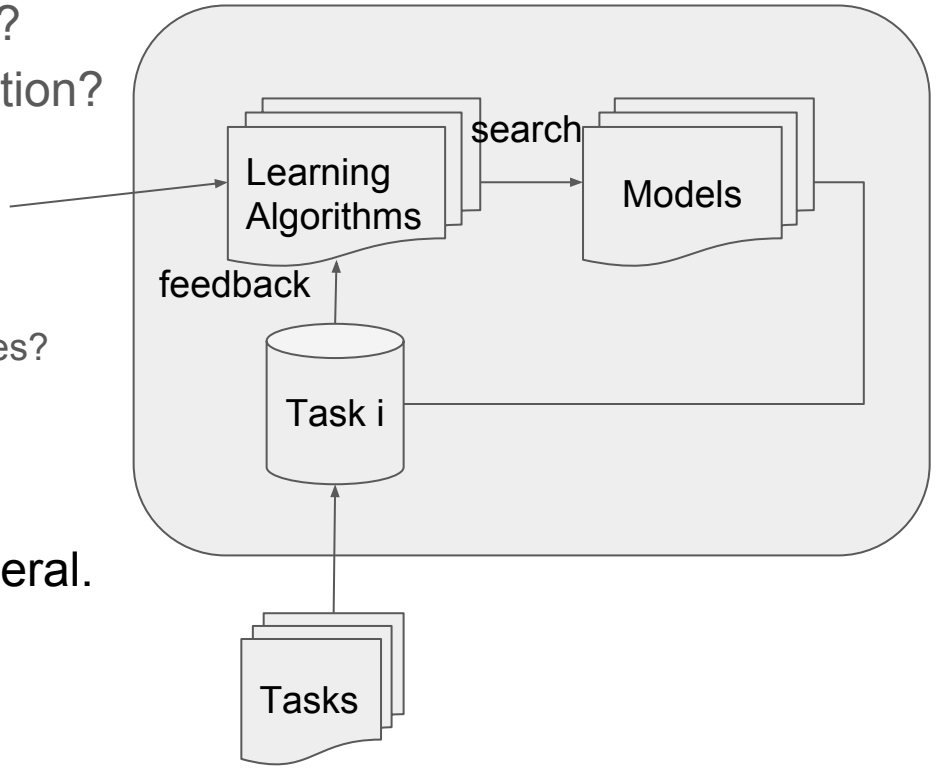


Problem Solved, right?



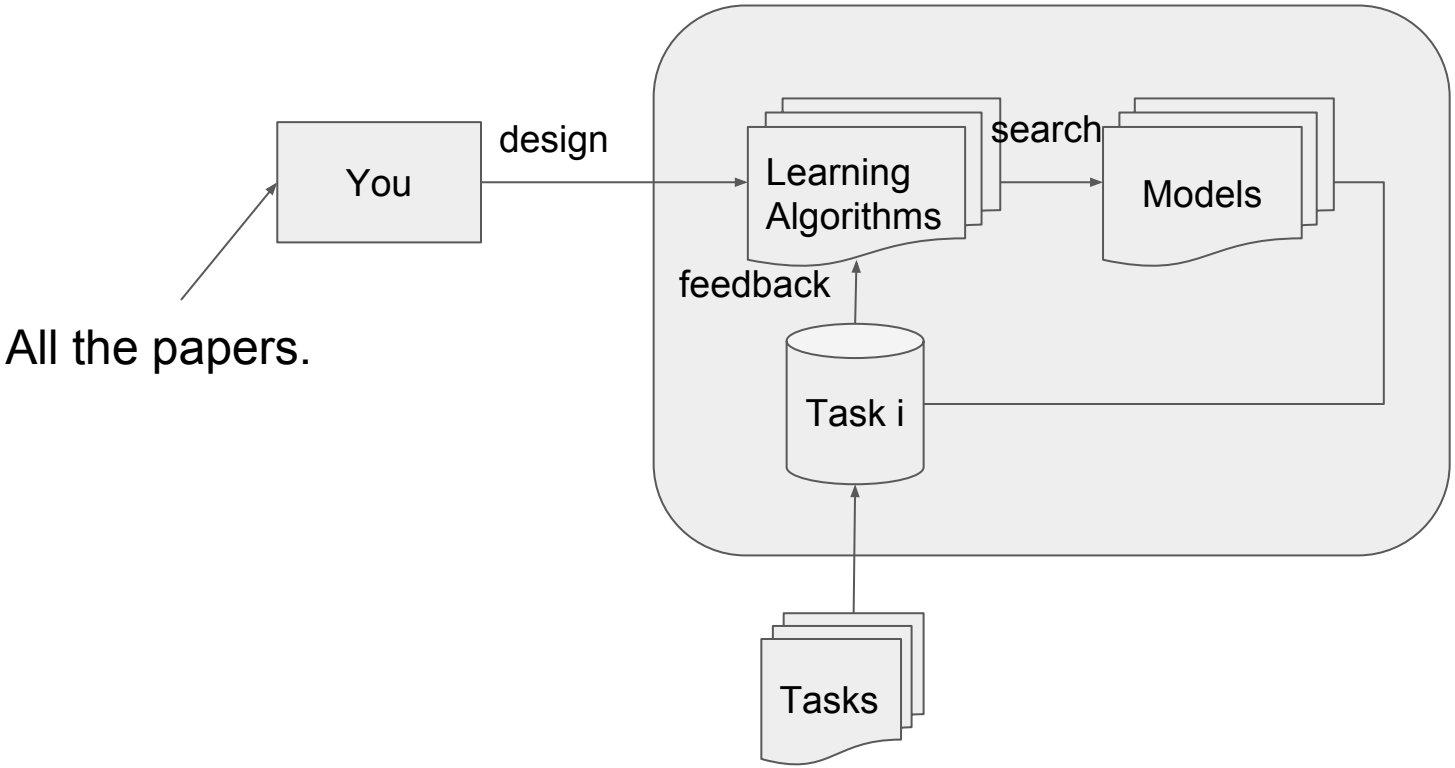
Problem Solved, right?

- Weight Initialization?
- Network Regularization?
- Learning rate?
- Optimizer?
- Policy Gradient?
  - Advantage Baselines?
- Q-learning?

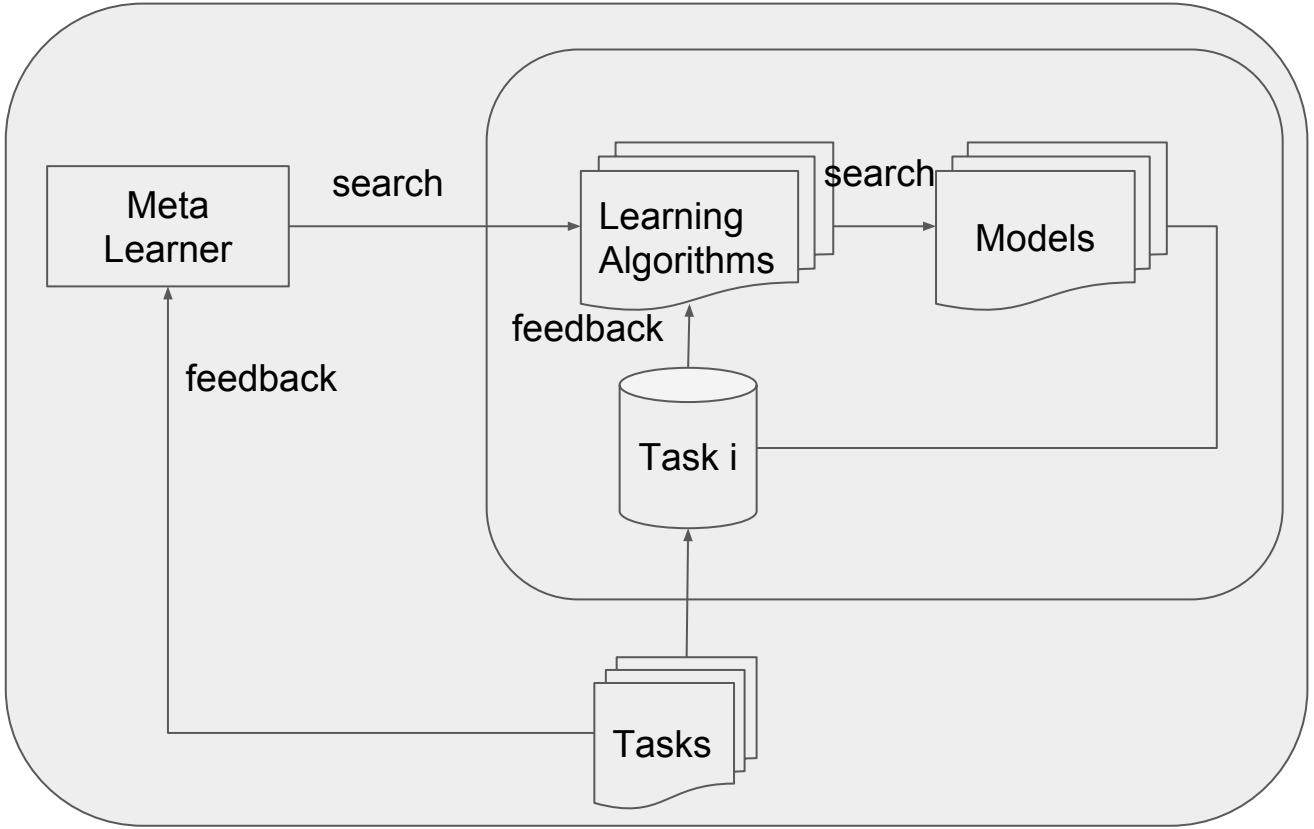


Needs to be fast, yet general.

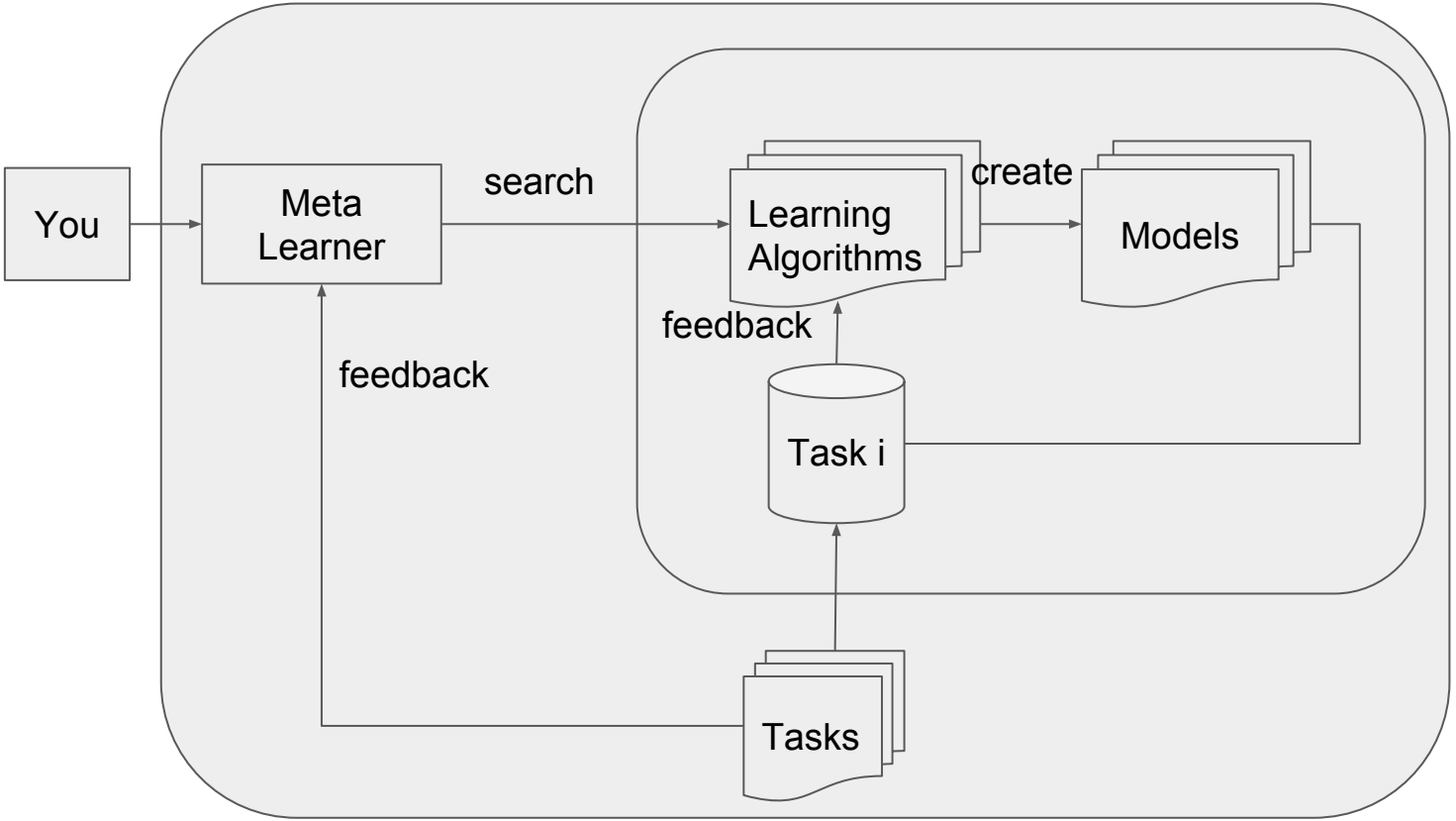
Designing a Tasks Learner



Learning a Tasks Learner

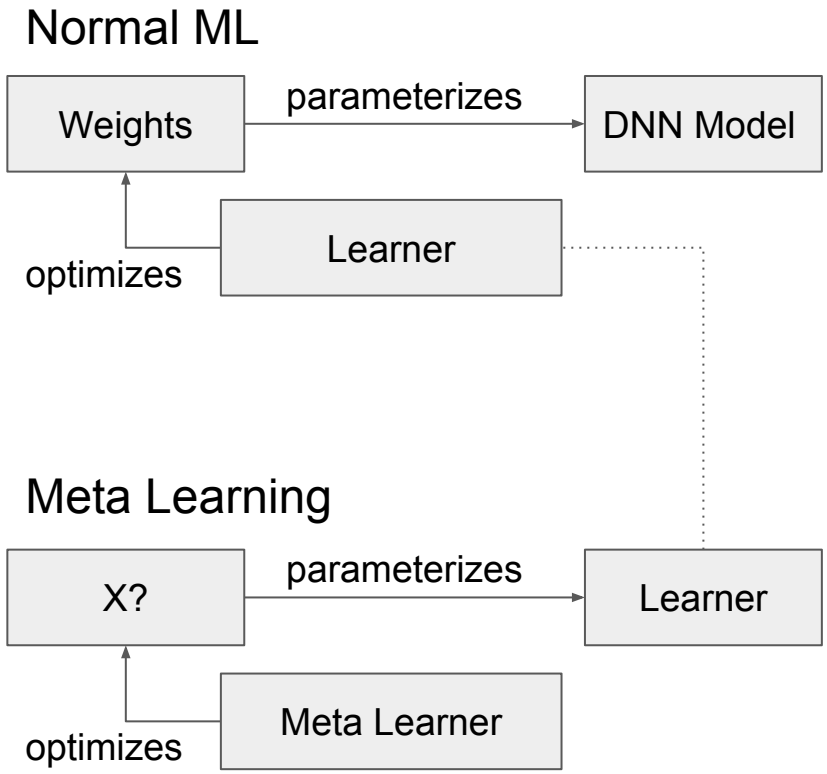


Designing a Tasks Learner Learner

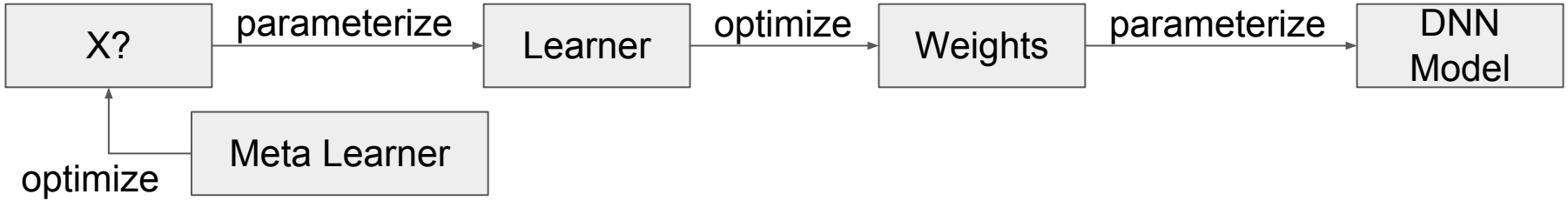


Approaches

- Parameterize the Learner Algorithm with some parameters X
- Meta Learner must optimize the learner with respect to X



Meta Learning Variants

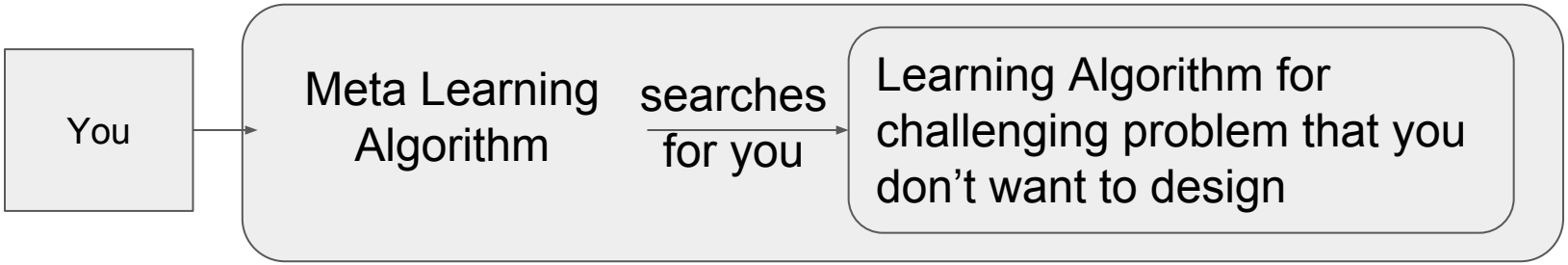


Method	**Learner**	**X**	**Meta Learner**
MAML	*SGD*	*Initial Weights*	*SGD*
L to L by GD by GD	LSTM Weight Updater	Weights of LSTM	SGD
L Transf. Architectures	SGD	Architecture	RNN Controller with RL

Applications

Learning to Learn X:

- Learning to do Fewshot Learning
- Learning Algorithms for Active Learning (Bachman et. al, 2017)
- Meta Learning Shared Hierarchies (Frans et. al, 2017)
- One-shot visual imitation learning via meta-learning (Finn et. al, 2017)
- Deep Online Learning Via Meta-Learning (Nagabandi et. al, 2019)



Meta-Learning problem set-up

- We have a model **f** with parameters **θ**

Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**

Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**
- We have a distribution over tasks: **p(T)**

Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**
- We have a distribution over tasks: **p(T)**
- Formal definition of each task:

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**
- We have a distribution over tasks: **p(T)**
- Formal definition of each task:

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

- K-shot learning setting: **p(T)** → $\mathbf{T_i}$ → $\mathbf{q_i}$ → **K** samples → $\mathbf{L_{Ti}}$ → training

Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**
- We have a distribution over tasks: **p(T)**
- Formal definition of each task:

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

- K-shot learning setting: **p(T)** → $\mathbf{T_i}$ → $\mathbf{q_i}$ → **K** samples → $\mathbf{L_{Ti}}$ → training
- Meta-Training error: the test error on sampled tasks $\mathbf{T_i}$

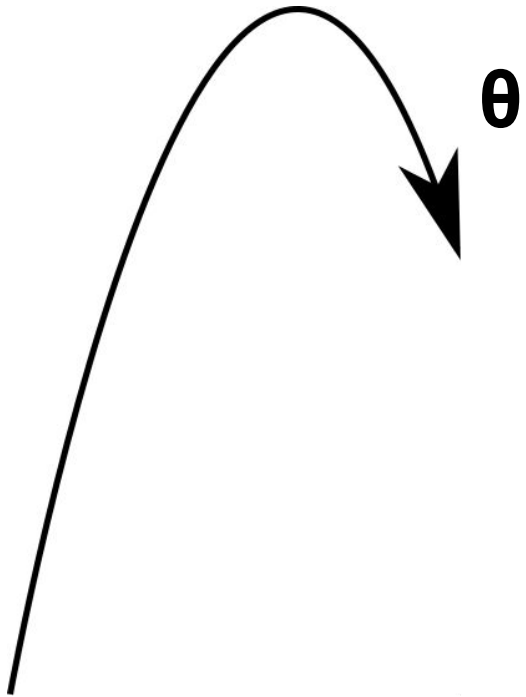
Meta-Learning problem set-up

- We have a model **f** with parameters **θ**
- It maps **x** to **a**: **f(x)** → **a**
- We have a distribution over tasks: **p(T)**
- Formal definition of each task:

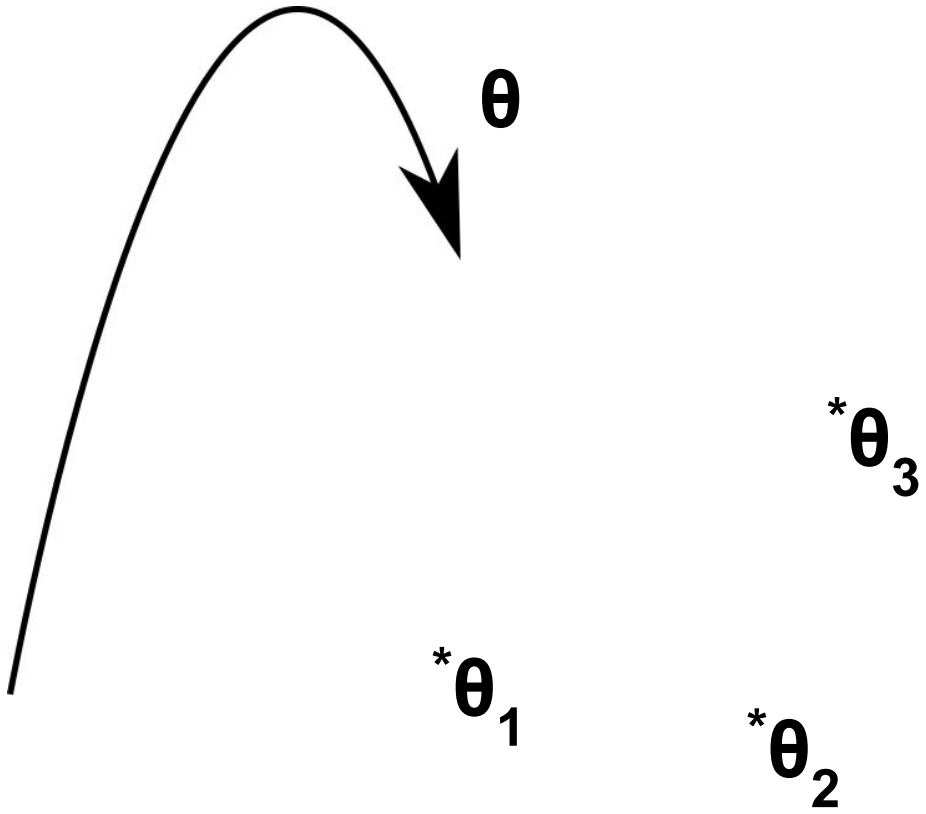
$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

- K-shot learning setting: **p(T)** → $\mathbf{T_i}$ → $\mathbf{q_i}$ → **K** samples → $\mathbf{L_{Ti}}$ → training
- Meta-Training error: the test error on sampled tasks $\mathbf{T_i}$
- Meta-Test error: the test error on tasks that were held out during training

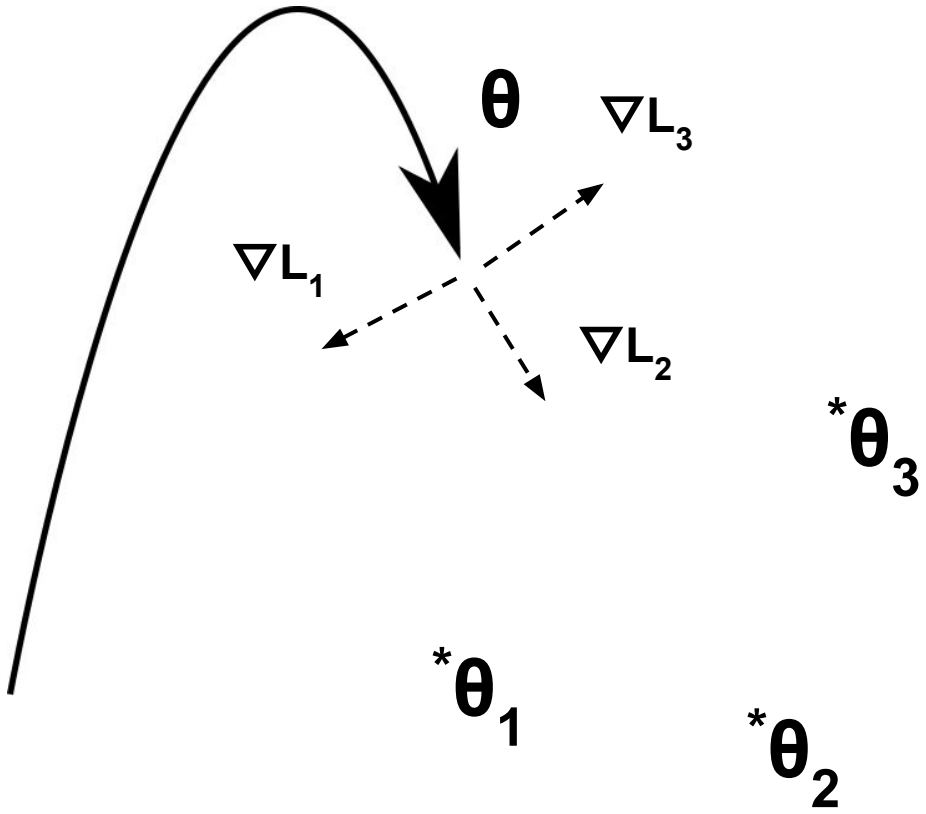
General idea



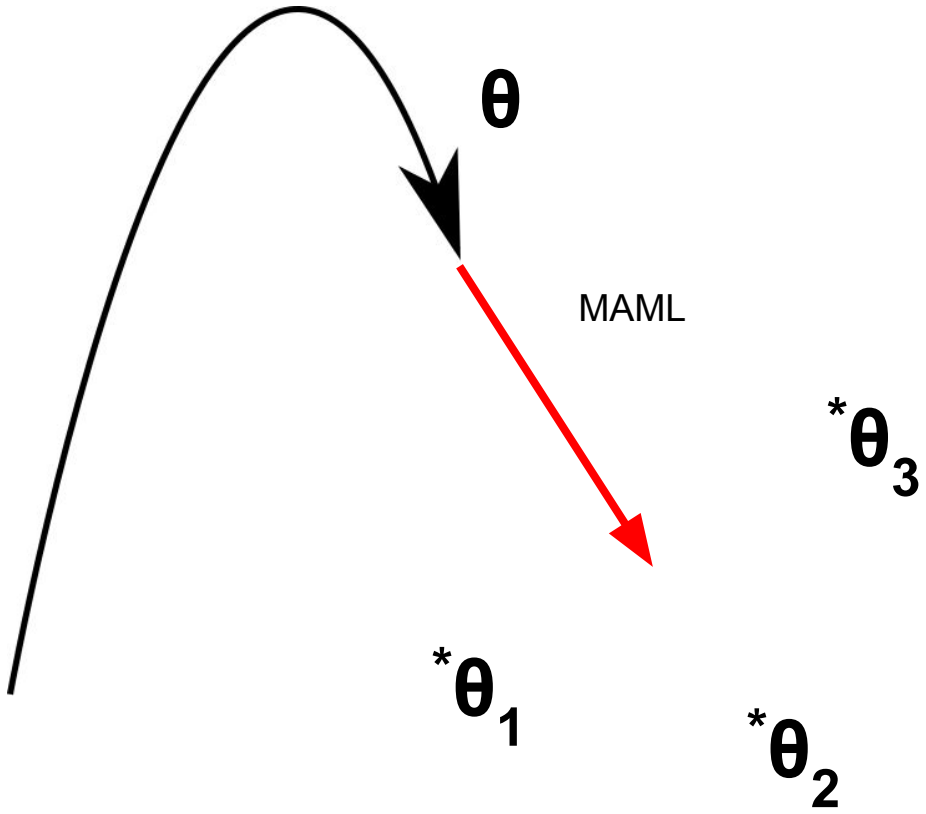
General idea



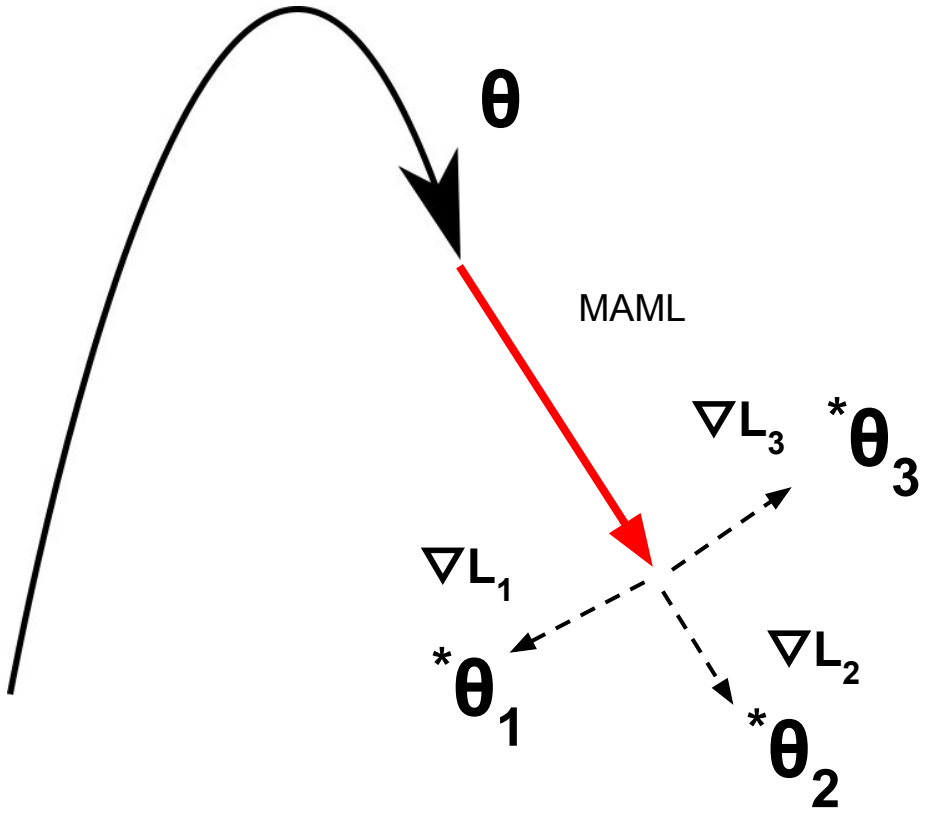
General idea



General idea



General idea



Actual algorithm

1. Randomly initialize $\boldsymbol{\theta}$

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**
3. Draw **K** examples from $\mathbf{q_i}$

Actual algorithm

1. Randomly initialize $\boldsymbol{\theta}$
2. Sample task $\mathbf{T_i}$ from $\mathbf{p(T)}$
3. Draw $\mathbf{K}$ examples from $\mathbf{q_i}$
4. Evaluate $\boldsymbol{\nabla_\theta L_{Ti}(f_\theta)}$ with respect to $\mathbf{K}$ examples

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**
3. Draw **K** examples from $\mathbf{q_i}$
4. Evaluate $\nabla_{\theta}\mathbf{L}_{\mathbf{Ti}}(\mathbf{f}_{\theta})$ with respect to **K** examples
5. Compute adapted parameters: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**
3. Draw **K** examples from $\mathbf{q_i}$
4. Evaluate $\nabla_{\theta}\mathbf{L}_{\mathbf{Ti}}(\mathbf{f}_{\theta})$ with respect to **K** examples
5. Compute adapted parameters: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

...

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**
3. Draw **K** examples from $\mathbf{q_i}$
4. Evaluate $\nabla_{\theta} \mathbf{L_{Ti}}(\mathbf{f_{\theta}})$ with respect to **K** examples
5. Compute adapted parameters: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

   ...

6. After sampling several tasks: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

Actual algorithm

1. Randomly initialize **θ**
2. Sample task $\mathbf{T_i}$ from **p(T)**
3. Draw **K** examples from $\mathbf{q_i}$
4. Evaluate $\nabla_{\theta} \mathbf{L_{Ti}}(\mathbf{f_{\theta}})$ with respect to **K** examples
5. Compute adapted parameters: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

   ...

6. After sampling several tasks: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

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

Experiments - Regression

- Draw **K** examples from $\mathbf{q_i}$ - data points $x^{(j)}$, $y^{(j)}$

Experiments - Regression

- Draw **K** examples from $\mathbf{q_i}$ - data points $x^{(j)}$, $y^{(j)}$
- Loss function $\mathbf{L_{Ti}(f_\theta)}$ with respect to **K** examples:

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)},\mathbf{y}^{(j)} \sim \mathcal{T}_i} \| f_\phi(\mathbf{x}^{(j)}) - \mathbf{y}^{(j)} \|_2^2$$

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]
- **K** points are sampled from [-5.0, 5.0]

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]
- **K** points are sampled from [-5.0, 5.0]
- The regressor - neural network, 2 hidden layers with 40 units, ReLU

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]
- **K** points are sampled from [-5.0, 5.0]
- The regressor - neural network, 2 hidden layers with 40 units, ReLU
- Training with MAML with K = 10;

Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]
- **K** points are sampled from [-5.0, 5.0]
- The regressor - neural network, 2 hidden layers with 40 units, ReLU
- Training with MAML with K = 10;
- For comparison:
    - Baseline (pretrained on all possible sine wave tasks)

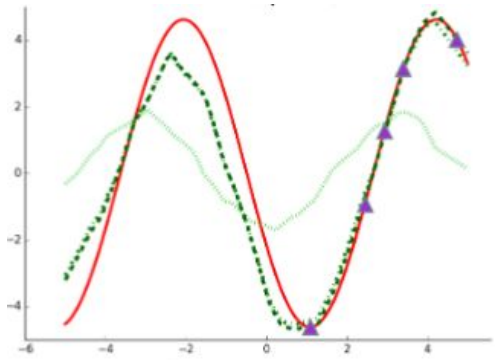
Experiments - Regression

- Tasks - regress from the input to the output of a sine wave;
- **p(T)** - continuous over sinusoids;
- Amplitude [0.1,0.5]
- Phase [0, π]
- **K** points are sampled from [-5.0, 5.0]
- The regressor - neural network, 2 hidden layers with 40 units, ReLU
- Training with MAML with K = 10;
- For comparison:
  - Baseline (pretrained on all possible sine wave tasks)
  - Oracle

Experiments - Regression

K=5

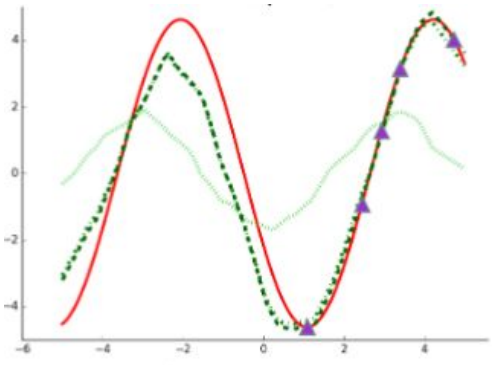
MAML



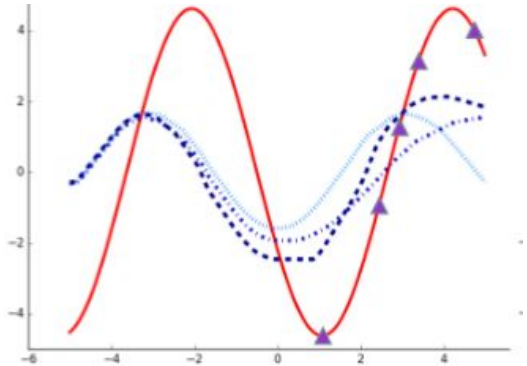
Experiments - Regression

K=5

MAML



Pretrained

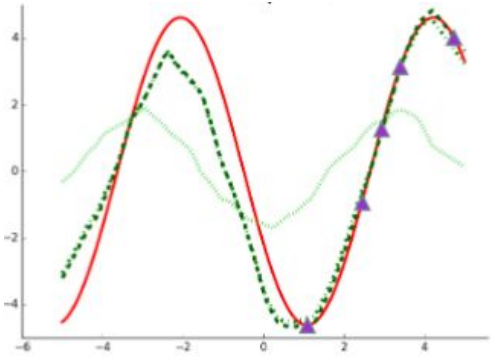


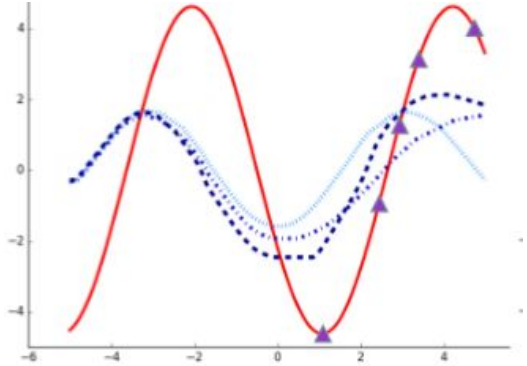
Experiments - Regression

MAML

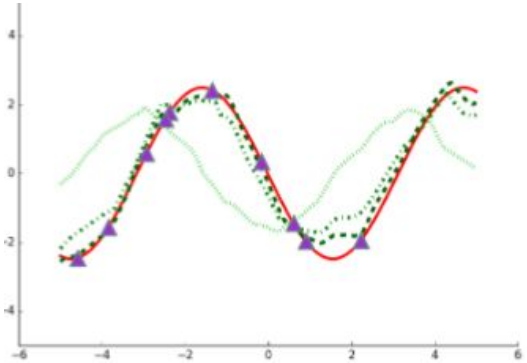
Pretrained

K=5





K=10

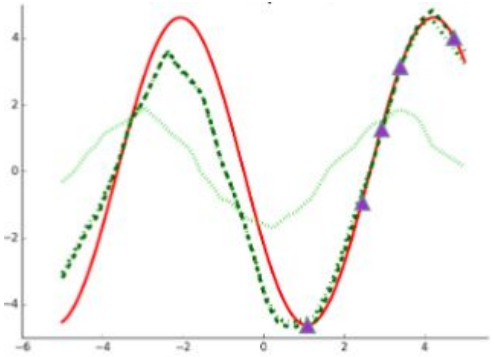


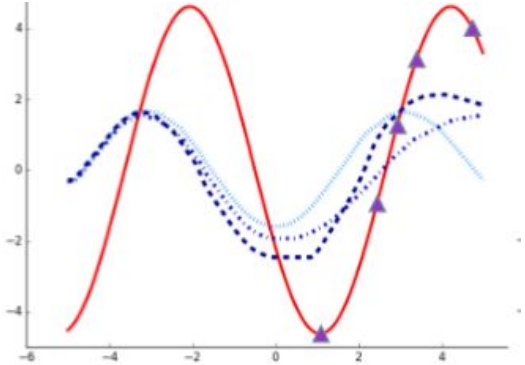
Experiments - Regression

MAML

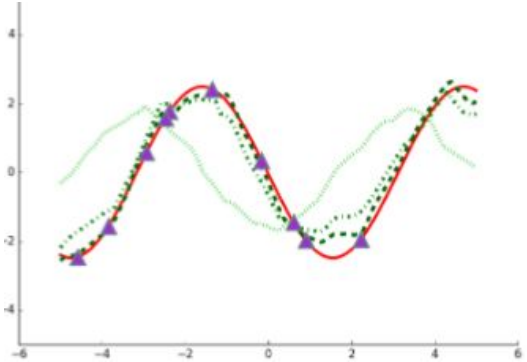
Pretrained

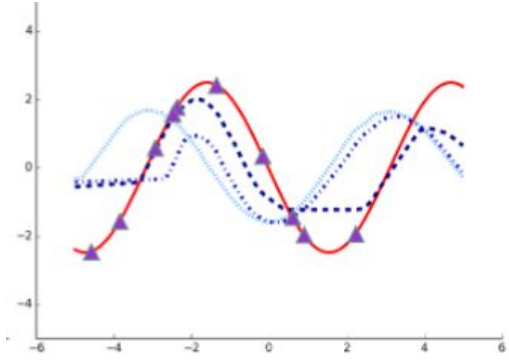
K=5



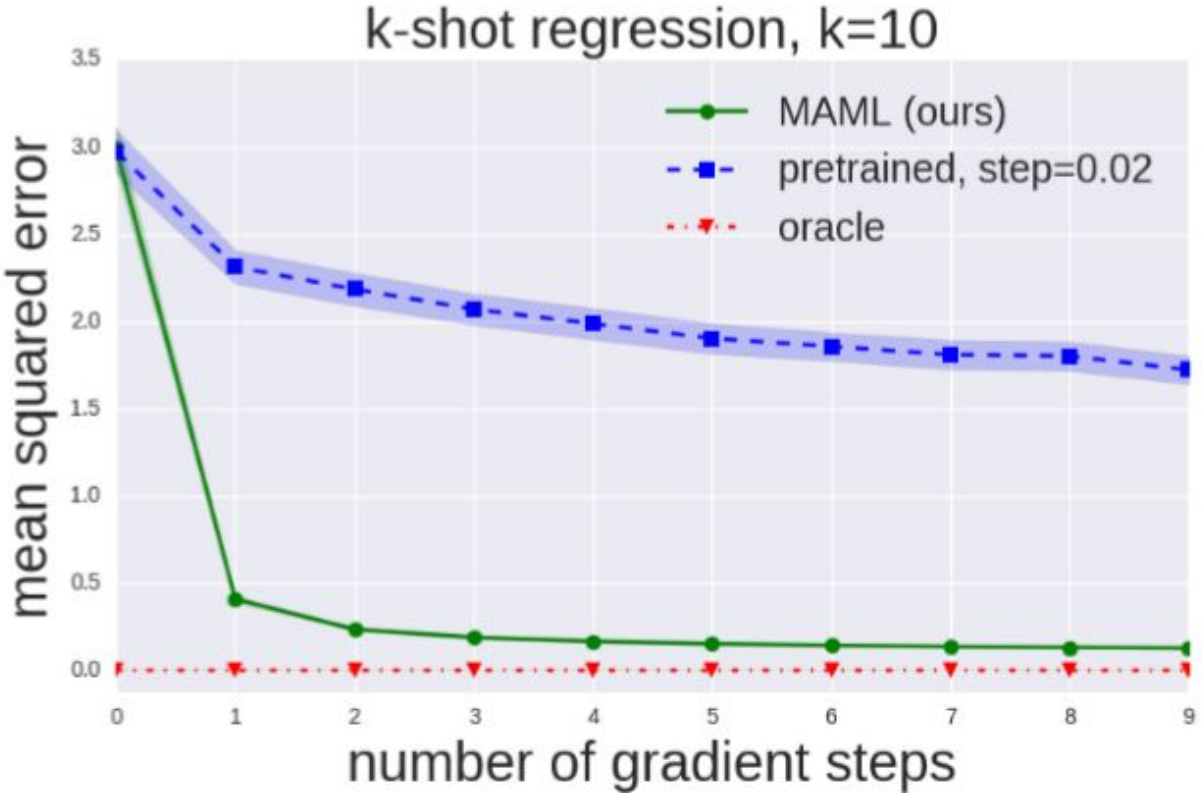


K=10





Experiments - Regression



Experiments - Classification

- *N*-way image classification with *K* shots
  - Trained with only *K* examples from each of the *N* classes

Experiments - Classification

- *N*-way image classification with *K* shots
  - Trained with only *K* examples from each of the *N* classes
- *p(T)*: distribution over all possible *N*-way tasks
  - Sample *N* classes
  - Sample *K* instances from each class

Experiments - Classification

- *N*-way image classification with *K* shots
  - Trained with only *K* examples from each of the *N* classes
- *p(T)*: distribution over all possible *N*-way tasks
  - Sample *N* classes
  - Sample *K* instances from each class
- Loss function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)}, \mathbf{y}^{(j)} \sim \mathcal{T}_i} \mathbf{y}^{(j)} \log f_\phi(\mathbf{x}^{(j)}) + (1 - \mathbf{y}) \log (1 - f_\phi(\mathbf{x}^{(j)}))$$

Experiments - Classification

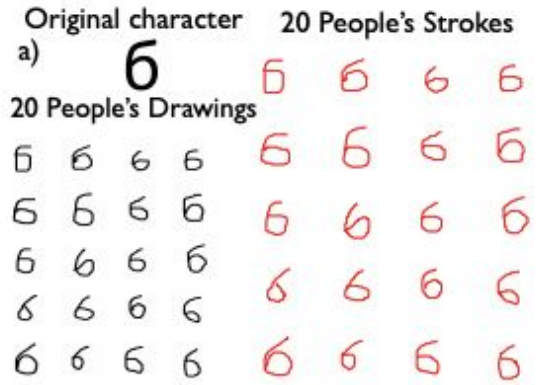
- *N*-way image classification with *K* shots
  - Trained with only *K* examples from each of the *N* classes
- *p(T)*: distribution over all possible *N*-way tasks
  - Sample *N* classes
  - Sample *K* instances from each class
- Loss function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)}, \mathbf{y}^{(j)} \sim \mathcal{T}_i} \mathbf{y}^{(j)} \log f_\phi(\mathbf{x}^{(j)}) + (1 - \mathbf{y}) \log (1 - f_\phi(\mathbf{x}^{(j)}))$$

- Datasets
  - Omniglot
  - MiniImageNet

Experiments - Few-Shot Image Recognition

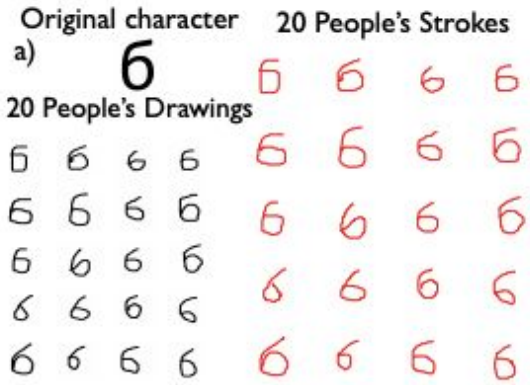
- Omniglot: 1623 characters with 20 instances each



Source: One shot learning of simple visual concepts (Lake et al.)

Experiments - Few-Shot Image Recognition

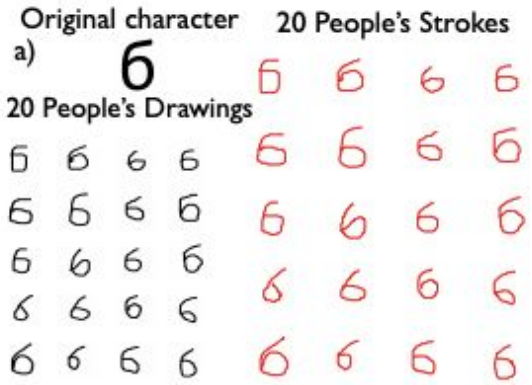
- Omniglot: 1623 characters with 20 instances each
- Randomly sample 1200 characters for training MAML, and use the rest for evaluation



Source: One shot learning of simple visual concepts (Lake et al.)

Experiments - Few-Shot Image Recognition

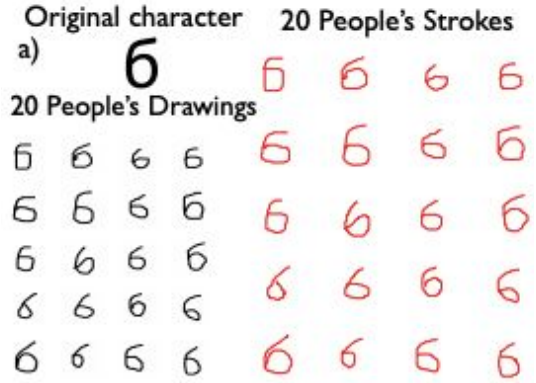
- Omniglot: 1623 characters with 20 instances each
- Randomly sample 1200 characters for training MAML, and use the rest for evaluation
- N = 5, 20
- K = 1, 5

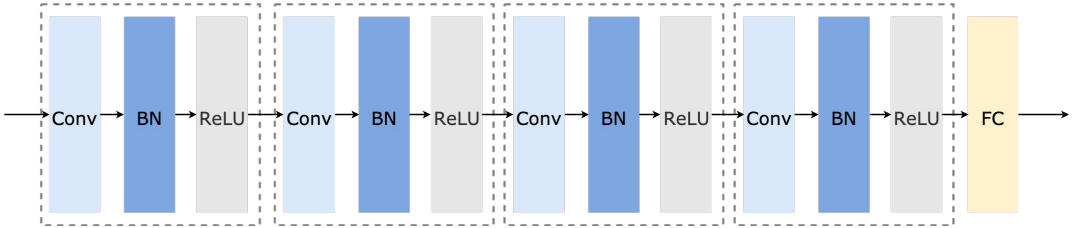


Source: One shot learning of simple visual concepts (Lake et al.)

Experiments - Few-Shot Image Recognition

- Omniglot: 1623 characters with 20 instances each
- Randomly sample 1200 characters for training MAML, and use the rest for evaluation
- N = 5, 20
- K = 1, 5
- A CNN-based classifier





Source: One shot learning of simple visual concepts (Lake et al.)

Experiments - Few-Shot Image Recognition

Omniglot (Lake et al., 2011)	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 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 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

Experiments - Few-Shot Image Recognition

- MiniImagenet
  - 64 training classes
  - 12 validation classes
  - 24 test classes
  - 600 samples of 84x84 colour images per class

Experiments - Few-Shot Image Recognition

- MiniImagenet
  - 64 training classes
  - 12 validation classes
  - 24 test classes
  - 600 samples of 84x84 colour images per class
- N = 5
- K = 1, 5

Experiments - Few-Shot Image Recognition

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	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)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Experiments - Reinforcement Learning

- Enable an agent to quickly acquire a new policy for a new task through a small amount of interactions

Experiments - Reinforcement Learning

- Enable an agent to quickly acquire a new policy for a new task through a small amount of interactions
- Each task is an MDP with horizon H
  - $q_i(x_1)$: initial state distribution
  - $q(x_{t+1} \mid x_t, a_t)$: transition probabilities
  - Task-specific loss corresponds to the negative reward function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{\mathbf{x}_t, \mathbf{a}_t \sim f_\phi, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} R_i(\mathbf{x}_t, \mathbf{a}_t) \right]$$

Experiments - Reinforcement Learning

- Enable an agent to quickly acquire a new policy for a new task through a small amount of interactions
- Each task is an MDP with horizon H
  - $q_i(x_1)$: initial state distribution
  - $q(x_{t+1} \mid x_t, a_t)$: transition probabilities
  - Task-specific loss corresponds to the negative reward function

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{\mathbf{x}_t, \mathbf{a}_t \sim f_\phi, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} R_i(\mathbf{x}_t, \mathbf{a}_t) \right]$$

- *K*-shot: sample *K* rollouts from the current policy for each task

Experiments - Reinforcement Learning

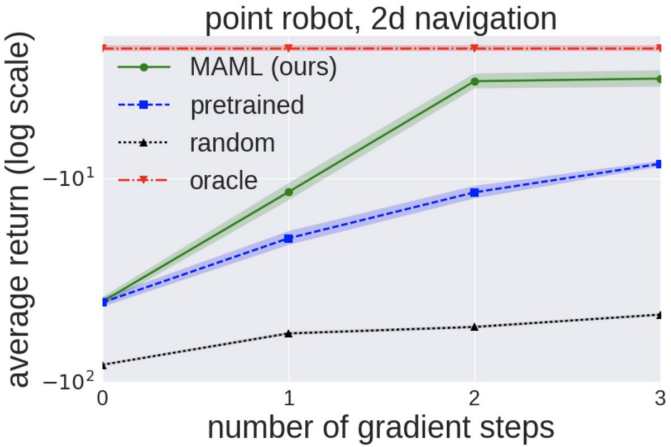
- Policy: MLP with 2 hidden layers of size 100 with ReLU non-linearities
- Update: REINFORCE
- Meta-update: TRPO

Experiments - Reinforcement Learning

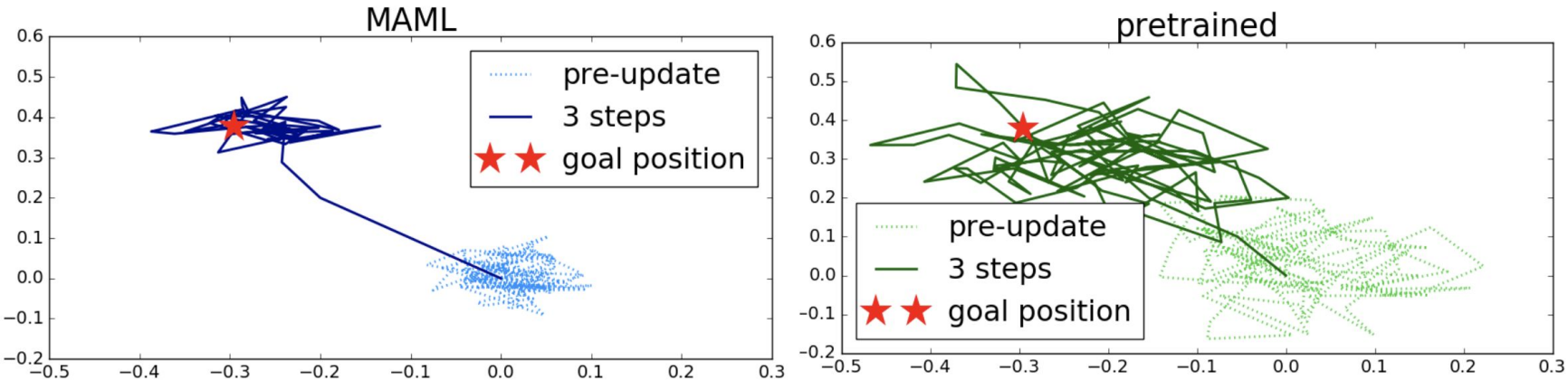
- Policy: MLP with 2 hidden layers of size 100 with ReLU non-linearities
- Update: REINFORCE
- Meta-update: TRPO
- Baselines models for comparisons
  - `pretrained`: pretrained on all tasks, fine-tuned on the specific task
  - `random`: training a policy from randomly initialized weights
  - `oracle`: an oracle policy which receives the parameters of the task as input

Experiments - Reinforcement Learning

- 2D navigation
  - Observation: current 2D coordinates
  - Goal: navigate to a target location
  - Reward: negative squared distance to the target location

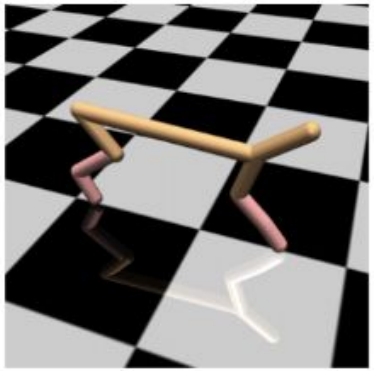


Experiments - Reinforcement Learning



Experiments - Reinforcement Learning

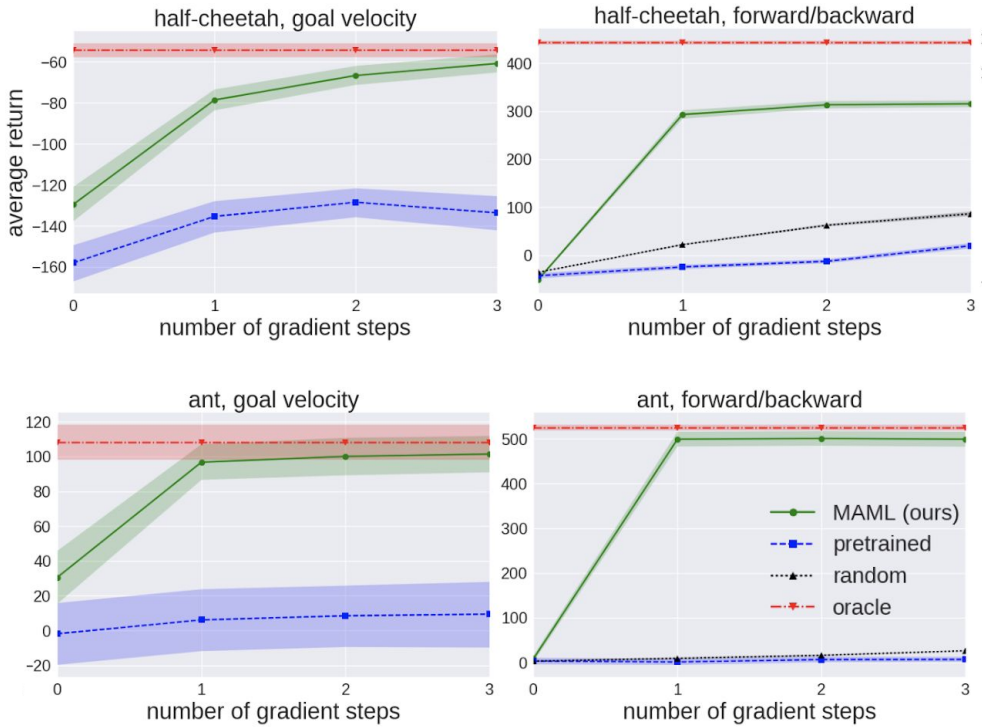
- Continuous control tasks in MuJoCo
  - Agent: ant, half cheetah
  - Observation: torques
  - Goal: run in a particular direction or at a particular velocity
  - Reward:
    - The magnitude of the velocity
    - The negative absolute value between the current velocity of the agent and a goal



Half Cheetah

Ant

Experiments - Reinforcement Learning



Video results can be found at https://sites.google.com/view/maml

PAML - Proactive and Adaptive meta-learning.

- L.-Y. Gui et al modified the MAML algorithm in their paper: Few-Shot Human Motion Prediction via Meta-learning
- The paper combined the MAML algorithm with a model regression network (MRN) to predict outcomes of human motion in with only a few examples.
- PAML was developed out of the need to develop a more flexible meta-learning algorithm to learn a more complicated tasks quickly.

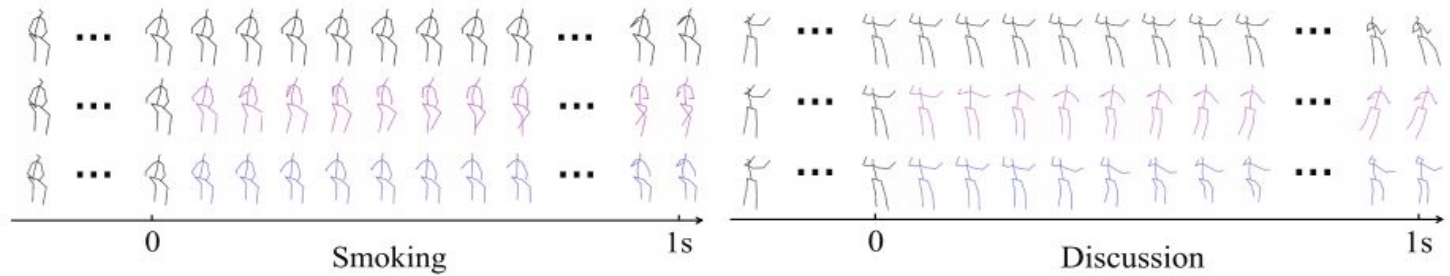
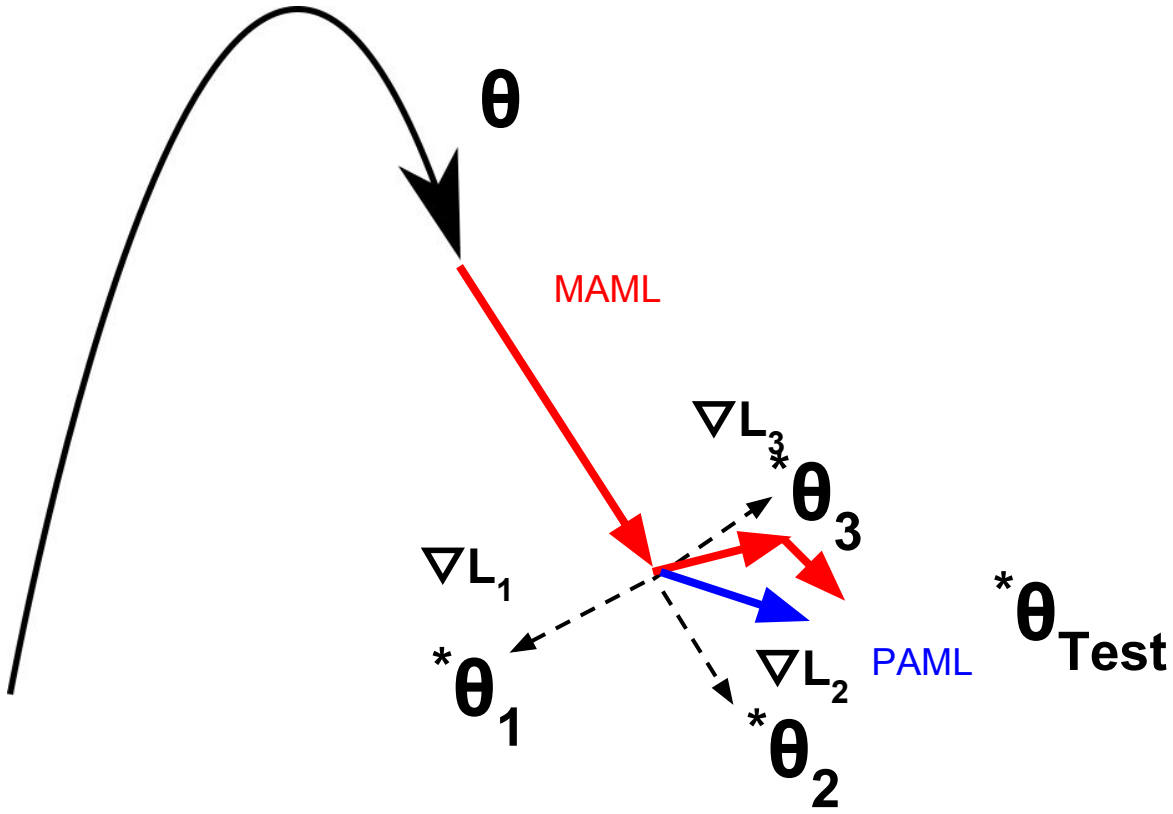


Figure from L.-Y. Gui et al. where the top is the ground truth of motion, the pink are the baseline predictions and the bottom is the prediction form PAML.

General idea - PAML vs MAML



PAML - Algorithm changes

Algorithm 1: PAML Meta-Training for k-Shot Human Motion Prediction

Require: Learner: motion predictor model $\mathcal{P}_\theta$ with parameters θ;
MRN adaptation meta-network: $\mathcal{H}_\phi$ with parameters ϕ
Require: $p(\mathcal{T})$: distribution over prediction tasks
Require: α, β, γ: learning or meta-learning rate hyper-parameters;
λ: trade-off hyper-parameter

Randomly initialize θ and ϕ
while *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for *all* $\mathcal{T}_i$ **do**
    Learn (or retrieve) θ_i^* from the original large set of annotated past and future sequence pairs of the corresponding action class
    Sample k action-specific past and future sequence pairs $\mathcal{D}_{\text{train}} = \left\{\left(\mathbf{X}_u, \mathbf{Y}_u^{gt}\right)\right\}, u = 1, \ldots, k$ from $\mathcal{T}_i$
    Evaluate $\mathcal{L}_{\mathcal{T}_i}\left(\mathcal{P}_\theta\right)$ on $\mathcal{D}_{\text{train}}$
    Compute adapted parameters using Eq. (5), *i.e.*, performing SGD updates then applying adaptation $\mathcal{H}$: $\theta_i' = \mathcal{H}_\phi\left(\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\left(\mathcal{P}_\theta\right)\right)$
    Sample $\mathcal{D}_{\text{test}} = \left\{\left(\mathbf{X}_v, \mathbf{Y}_v^{gt}\right)\right\}$ from $\mathcal{T}_i$ for the meta-update
    Evaluate $\widetilde{\mathcal{L}}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right) = \mathcal{L}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right) + \frac{1}{2}\lambda \left\|\theta_i' - \theta_i^*\right\|_2^2$ on $\mathcal{D}_{\text{test}}$ using Eq. (6)
  end for
  Update θ and ϕ by performing SGD:
  $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \widetilde{\mathcal{L}}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right)$, $\phi \leftarrow \phi - \gamma \nabla_\phi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \widetilde{\mathcal{L}}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right)$
end while

- The main difference is using a feed forward network to perform the parameter update for the k shot learning.
- We also must solve for parameters from many shot learning before hand and use it in the regularization term.
- We then update two sets of parameters: the learners parameters, and the meta learners parameters.
- The idea is the meta-network will learn how to adjust parameters to allow it to generalize well.

$$\min_{\theta,\phi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \widetilde{\mathcal{L}}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right) = \min_{\theta,\phi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(\mathcal{P}_{\theta_i'}\right) + \frac{1}{2}\lambda \left\|\theta_i' - \theta_i^*\right\|_2^2,$$

Figures from L.-Y. Gui et al.

Experimental results - sanity check.

Method	5-way accuracy	
	1-shot	5-shot
Matching networks [61]	$43.56\% \pm 0.84\%$	$55.31\% \pm 0.73\%$
MAML [14]	$48.70\% \pm 1.84\%$	$63.11\% \pm 0.92\%$
Meta-learner LSTM [43]	$43.44\% \pm 0.77\%$	$60.60\% \pm 0.71\%$
Prototypical networks [51]	$46.61\% \pm 0.78\%$	$65.77\% \pm 0.70\%$
Meta networks [34]	$49.21\% \pm 0.96\%$	$--$
PAML (Ours)	$\mathbf{53.26\% \pm 0.52\%}$	$\mathbf{68.19\% \pm 0.61\%}$

Table from L.-Y. Gui et al.

- L.-Y Gui et al performed a sanity check using 1-shot and 5-shot learning on classification tasks on mini-ImageNet.
- Results showed their accuracy to be the best.
- I don’t think this should be surprising. It’s more flexible method and there is no shortage of data.

Experimental results - Human motion prediction

		Walking						Eating					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\text{Scratch}_{\text{spec}}$	1.90	1.95	2.16	2.18	1.99	2.00	2.33	2.31	2.30	2.30	2.31	2.34
	$\text{Scratch}_{\text{agn}}$	1.78	1.89	2.20	2.23	2.02	2.05	2.27	2.16	2.18	2.27	2.25	2.31
	$\text{Transfer}_{\text{ots}}$	0.60	0.75	0.88	0.93	1.03	1.26	0.57	0.70	0.91	1.04	1.19	1.58
	Multi-task	0.57	0.71	0.79	0.85	0.96	1.12	0.59	0.68	0.83	0.93	1.12	1.33
	$\text{Transfer}_{\text{ft}}$	0.44	0.55	0.85	0.95	0.74	1.03	0.61	0.65	0.74	0.78	0.86	1.19
Meta-learning (Ours)	PAML	**0.35**	**0.47**	**0.70**	**0.82**	**0.80**	**0.83**	**0.36**	**0.52**	**0.65**	**0.70**	**0.71**	**0.79**
		Smoking						Discussion					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\text{Scratch}_{\text{spec}}$	2.88	2.86	2.85	2.83	2.80	2.99	3.01	3.13	3.12	2.95	2.62	2.99
	$\text{Scratch}_{\text{agn}}$	2.53	2.61	2.67	2.65	2.71	2.73	2.77	2.79	2.82	2.73	2.82	2.76
	$\text{Transfer}_{\text{ots}}$	0.70	0.84	1.18	1.23	1.38	2.02	0.58	0.86	1.12	1.18	1.54	2.02
	Multi-task	0.71	0.79	1.09	1.20	1.25	1.23	0.53	0.82	1.02	1.17	1.33	1.97
	$\text{Transfer}_{\text{ft}}$	0.87	1.02	1.25	1.30	1.45	2.06	0.57	0.82	1.11	1.11	1.37	2.08
Meta-learning (Ours)	PAML	**0.39**	**0.66**	**0.81**	**1.01**	**1.03**	**1.01**	**0.41**	**0.71**	**1.01**	**1.02**	**1.09**	**1.12**

Figures from L.-Y. Gui et al.

- Dataset is human 3.6M, which is benchmarked often.
- Used 5 shot learning.
- Training was done on 11 action classes: directions, greeting, phoning, posing, purchases, sitting, sitting down, taking photo, waiting, walking dog, and walking together
- Testing was done on the four shown here.
- The numbers correspond to mean angle error, so lower is better.

Discussion - MAML

Positives

- MAML is a simple way to get a good network initialization.
- It is also Model agnostic, so it is easy to implement.
- It is great for building pretrained models for others to use without having to spend weeks of training time.

Negatives

- My not be flexible enough for other certain applications, but is a great start.

Discussion - PAML

Positives

- Adds more flexibility to the MAML architecture.
- Trained model can learn more complex tasks in a few shots.
- Needed when an agent needs to learn from few examples.
- L.-Y. Gui et al. found would be useful for robots to learn interactions with humans quickly.

Negatives

- We need to train a second network which adds time and complexity.
- Does appear to outperform MAML, but we need to train more hyper parameters with also adds more time for cross validation.
- A lot of hyper parameters mean we have to be careful to not overfit the validation set.

Conclusion

- Meta Learning is learning how to learn.
- Main goal: we want our learning algorithms to get better at learning with experience
- We presented two major meta learning frameworks.
- Maml to find a good parameter initialization
- PAML which adds more flexibility to learn in less examples, with the trade off of more training time and more hyper parameter tuning.
- Thank you for listening, Questions?