

1703.03400 - Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

- **Yunqiu Xu**
- Other reference:
 - <https://www.jiqizhixin.com/articles/2017-07-20-4>
 - <http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/>
 - <https://github.com/cbfinn/maml>

1. Introduction

- The goal of meta-learning: train a model with some learning tasks, then it can solve new tasks with only a few samples
- Our work:
 - The initial parameters of the model are explicitly trained first
 - Then this model can be generated to new task with a small number of training samples / gradient steps on that task
- Do not use following meta-learning methods:
 - Learn an update function or learning rules
 - Expand the number of learned parameters
 - Place constraints on the model architecture
- Our advantage:
 - General and model-agnostic: can be directly applied to any learning problem and model with a few samples / GD steps
 - Can easily handle different architectures / problem settings / loss functions with minimal modification
 - Easy and fast to fine-tune

2. MAML

2.1 Meta-Learning Problem Set-Up

- Goal: Treat entire tasks as training examples, then train a model with these learning tasks, thus it can solve new tasks with only a few samples / steps
- Model $f: x \rightarrow a$
 - x : observations
 - a : outputs
 - During meta-learning, the model is trained to be able to adapt to a large or infinite number of tasks.
- $p(T)$ is a distribution over tasks, each task $T = \{L(x_1, a_1, \dots, x_H, a_H), q(x_1), q(x_{t+1}|x_t, a_t), H\}$:
 - $L \rightarrow R$: loss function
 - $q(x_1)$: distribution over initial observations
 - $q(x_{t+1}|x_t, a_t)$: transition distribution
 - H : episode length, model may generate samples of length H by choosing an output a_t at each time t
- K-shot meta-learning:
 - Sample a new task T_i from $p(T)$
 - Train model with K samples from q_i
 - Generate feedback L_{T_i} from T_i
 - Test on new samples from T_i
 - Treat the test error on sampled tasks T_i as the training error of meta-learning process
 - At the end of meta-learning, sample new tasks from $p(T)$
 - Meta-performance: model f 's performance after learning from K samples

2.2. A MAML Algorithm

- How to encourage the emergence of general-purpose representations:
 - Learn a model that gradient-based learning rule can make rapid progress on new tasks drawn from $p(T)$ without overfitting
 - Find model parameters that are sensitive to changes of the task \rightarrow small

changes lead to large improvement

- Model f_θ
 - No assumption on the form of model
 - Only assume that the model is parameterized by params θ , the loss function can be optimized by gradient through these params
 - When adapting to a new task T_i , θ becomes θ'_i by gradient updating
$$\theta'_i = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$$
 - Step size α : fixed as a hyperparameter or meta-learned.

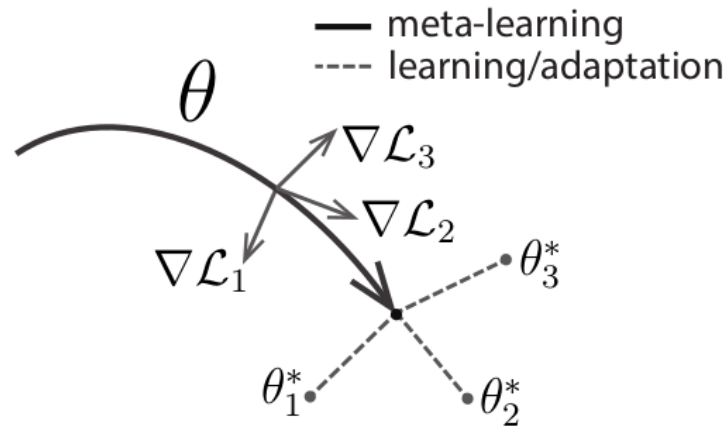


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

- Meta-objective:

$$\min_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta'_i}) = \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta})})$$

- Meta-optimization across tasks:

- Performed over params θ
- The objective is computed using updated params θ'
- Update θ using SGD, β is meta stepsize

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta'_i})$$

- Entire algorithm

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β : step size hyperparameters

```
1: randomly initialize  $\theta$ 
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
```

2.3 Species of MAML

- Two kinds of MAML are shown below, the only difference is loss function
- MAML for few-shot supervised learning: MSE or cross validation
- MAML for RL: reward function, as we need to maximize reward, in loss function we multiply "-1" to minimize the value

$$L_{\mathcal{T}_i}(f_{\psi}) = -E_{x_t, a_t \sim f_{\psi, \mathcal{T}_i}} \left[\sum_{t=1}^H R_i(x_t, a_t) \right] \quad (4)$$

Algorithm 3 MAML for Reinforcement 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: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, \dots, \mathbf{x}_H)\}$ using f_θ in \mathcal{T}_i
 - 6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using \mathcal{D} and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
 - 7: Compute adapted parameters with gradient descent:
 $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
 - 8: Sample trajectories $\mathcal{D}'_i = \{(\mathbf{x}_1, \mathbf{a}_1, \dots, \mathbf{x}_H)\}$ using $f_{\theta'_i}$ in \mathcal{T}_i
 - 9: **end for**
 - 10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each \mathcal{D}'_i and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
 - 11: **end while**
-

4. Related Work of Meta-Learning

	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
Omniglot (Lake et al., 2011)				
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%

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

- RNNs as learners: MANN
 - Search space includes all conceivable ML algorithms
 - Moves the burden of innovation to RNNs
 - Ignores advances achieved in ML by humans
 - The results are not good
- Metric learning: Siamese nets, matching nets
 - Learn a metric in input space
 - Specialized to one/few-shot classification
 - Can't use in other problems
- Optimizer learning: meta-learner LSTM
 - Learn parameter update given gradients (search space includes SGD, RMSProp, Adam etc)
 - Applicable to any architecture / task
 - But we can achieve better performance with MAML

5. Experimental Evaluation

- Questions need to be answered:
 - Can MAML enable fast learning of new tasks
 - Can MAML be used for meta-learning in multiple different domains

- Can a model learned with MAML continue to improve with additional gradient updates and/or examples
- Classification result: see 4
- RL result:

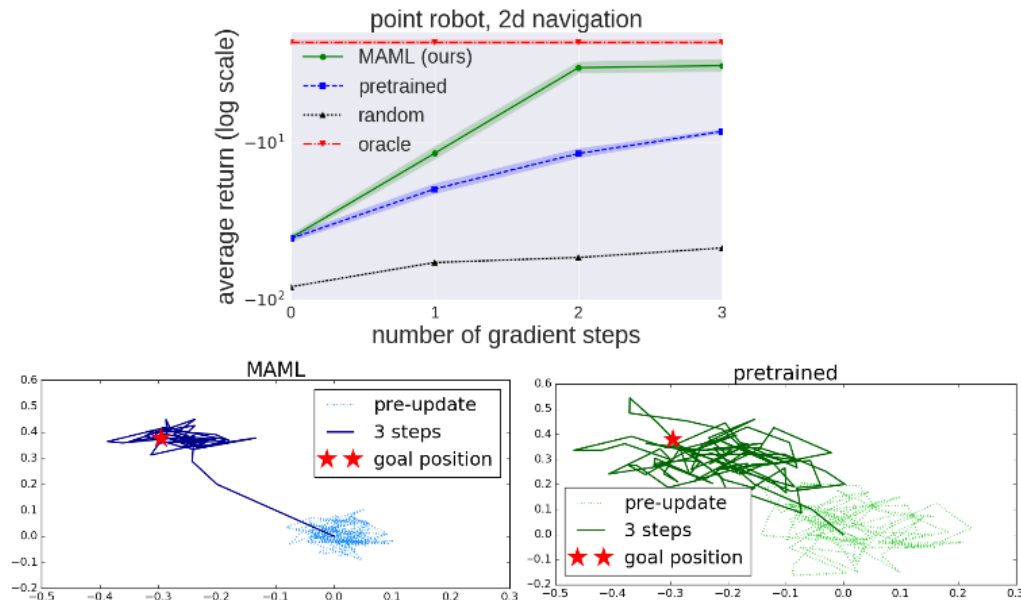


Figure 4. Top: quantitative results from 2D navigation task, Bottom: qualitative comparison between model learned with MAML and with fine-tuning from a pretrained network.

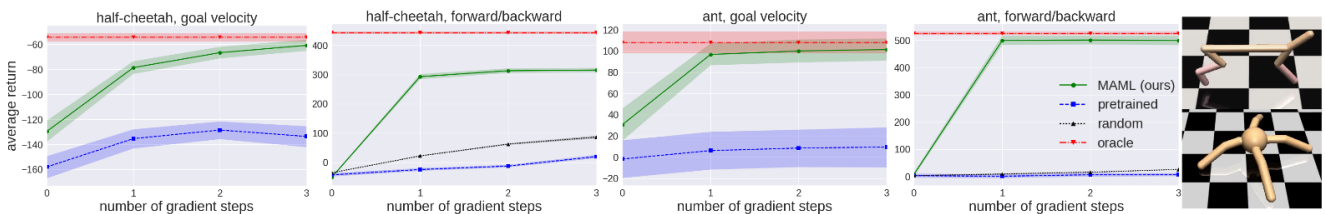


Figure 5. Reinforcement learning results for the half-cheetah and ant locomotion tasks, with the tasks shown on the far right. Each gradient step requires additional samples from the environment, unlike the supervised learning tasks. The results show that MAML can adapt to new goal velocities and directions substantially faster than conventional pretraining or random initialization, achieving good performs in just two or three gradient steps. We exclude the goal velocity, random baseline curves, since the returns are much worse (< -200 for cheetah and < -25 for ant).

6. Discussion and Future Work

- Benefits:
 - Simple, does not introduce any learned parameters for meta-learning
 - Any gradient-based model representation / differentiable objective
 - Adaptation on new tasks
- Future work:

- Generalize meta-learning technique to apply to any problem and any model
- Make multitask initialization a standard ingredient in DL and RL