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

Chelsea Finn Pieter Abbeel Sergey Levine

ICML 2017

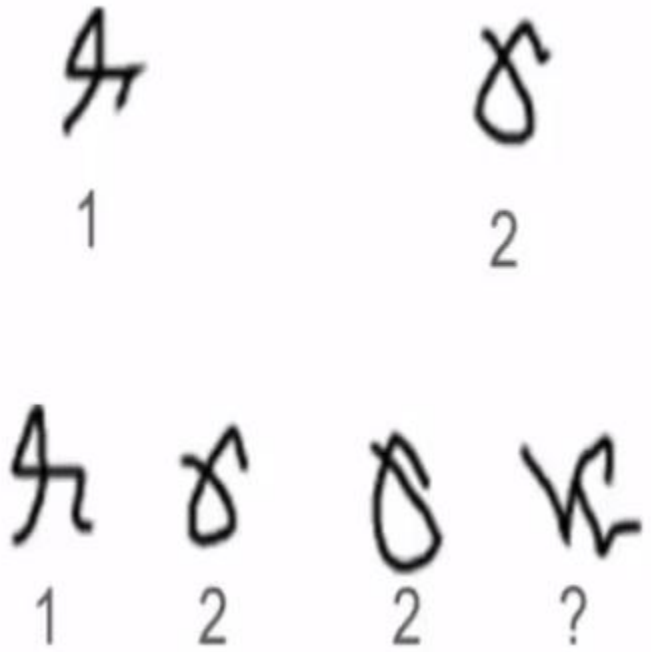
Outline

- Introduction
- Model-Agnostic Meta-learning
- Applications
  - Supervised Learning
  - Reinforcement Learning
- Experiments
- Conclusions

Introduction

One shot learning

- Human learn
  - a few examples
  - Learn quickly
- Approaches
  - Transfer Learning
  - Meta Learning
    - ◆ Model Agnostic Meta Learning



Model Agnostic Meta Learning

Intuition

- some internal representations are more transferrable than others
- maximizing the sensitivity of the loss functions of new tasks with respect to the parameters

Model Agnostic Meta Learning

- Model parameters that are sensitive to small changes in the task
- Small changes in the parameters will produce large improvements on the loss function

Model Agnostic Meta Learning

Meta Learning Problem Set-up

F : a model , that maps observations x to outputs a,
parameterized by θ

p(T) : a distribution over tasks

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

- L : a loss function
- q(x1) : a distribution over initial observations,
- q(xt+1|xt, at) : a transition distribution
- H : an episode length

K-shot Learning: K samples drawn from qi

A Model-Agnostic Meta-Learning Algorithm

Method:

For the Ti`s the model parameter θi:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

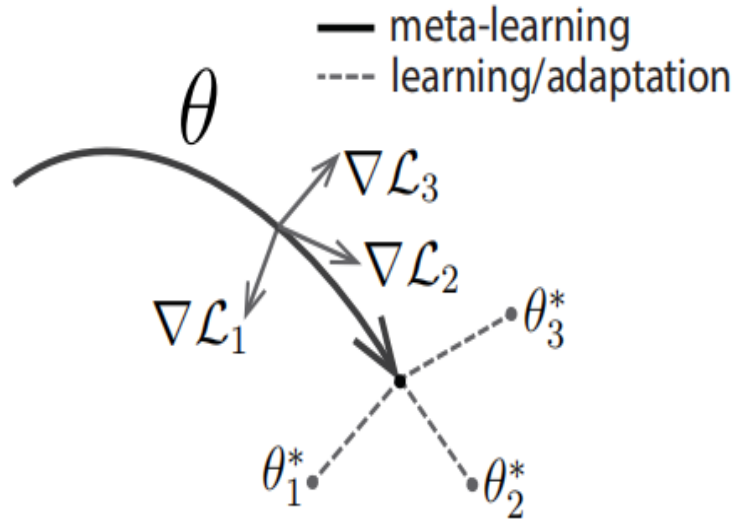


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Meta-objective:

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

A Model-Agnostic Meta-Learning Algorithm

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**

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

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

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Supervised Learning

Algorithm 2 MAML for Few-Shot Supervised 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 datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
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 2 or 3
11: **end while**

Regression problem

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

Classification problem

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

Reinforcement Learning

Algorithm 3 MAML for Reinforcement Learning

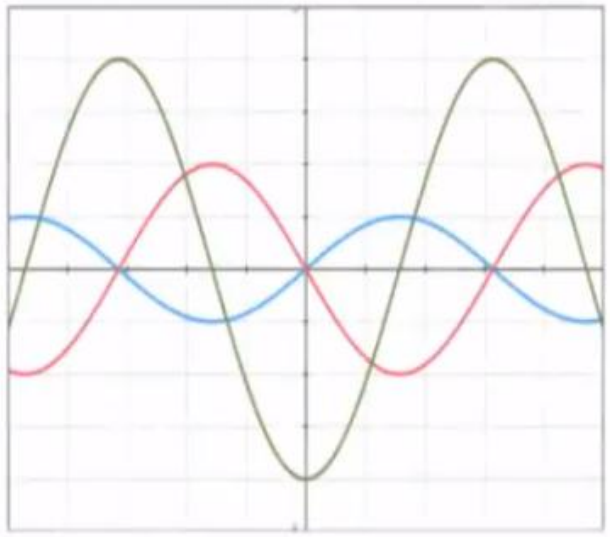
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ 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 $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ 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**

The loss of the RL Task

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

Regression Experiments

Regression： Sine wave



- **Meta Training**:
  - In p(T) is continuous
  - Amplitude varies within [0.1, 5.0]
  - Phase varies within [0, π]

During training and testing, data points x are sampled uniformly from [-5.0, 5.0]

The loss is the mean-squared error(MSE).

- **Model:**

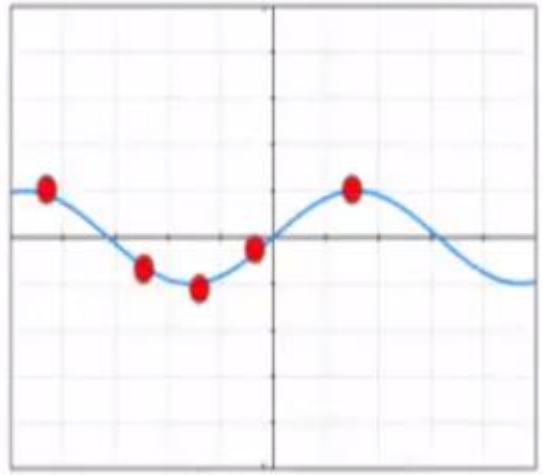
A neural network model with 2 hidden layers of size 40 with ReLU

- **Training:**

Use one gradient update with K = 10 examples with a fixed step size α = 0.01, and use Adam as the meta-optimizer

Regression Experiments

Regression： Sine wave



- **Meta Testing**:

  Fine-tune a single meta-learned model on varying numbers of K examples

- **Evaluation**:

  The mean-squared error(MSE) for the validation data points

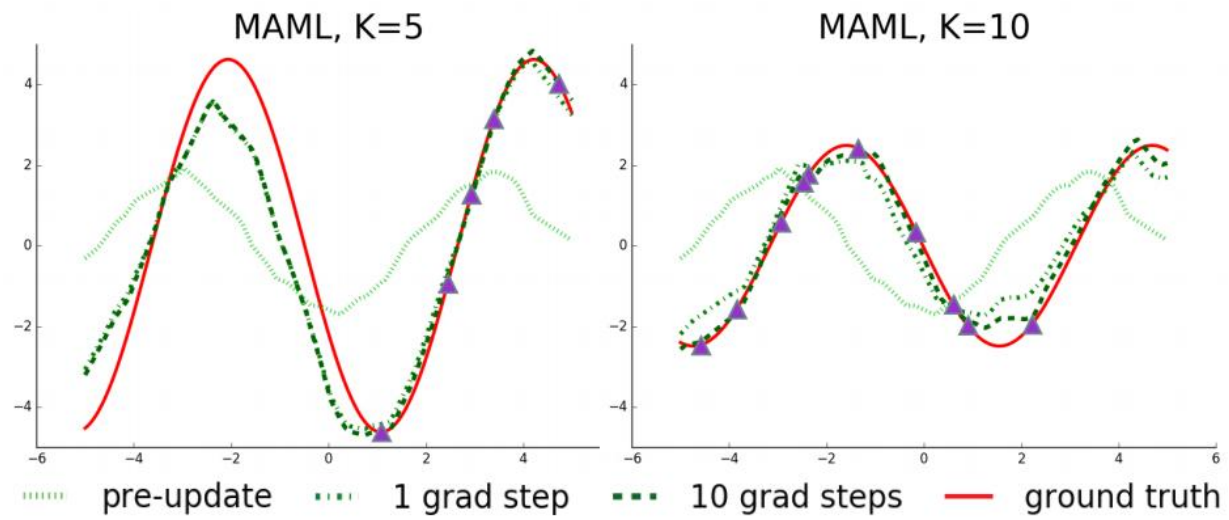
- **Two baselines**:

1. Pretraining on all of the tasks,on all samples，at test-time, fine-tuning with gradient descent on the K provided points, using an automatically tuned step size
2. An oracle which receives the true amplitude and phase as input

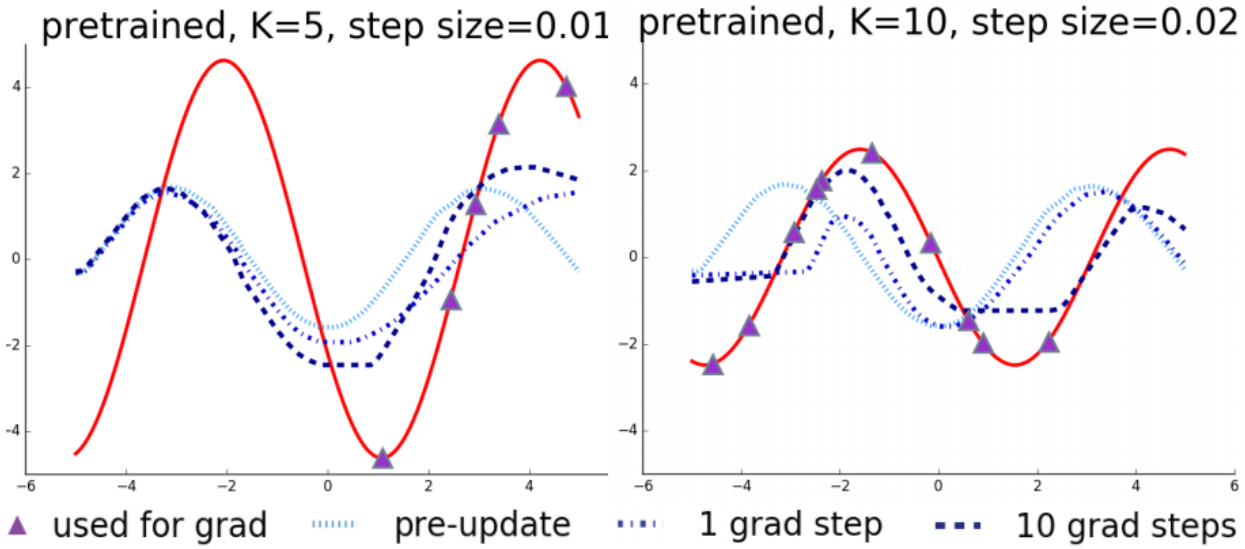
Regression Experiments Results

Regression： Sine wave

- **MAML**:

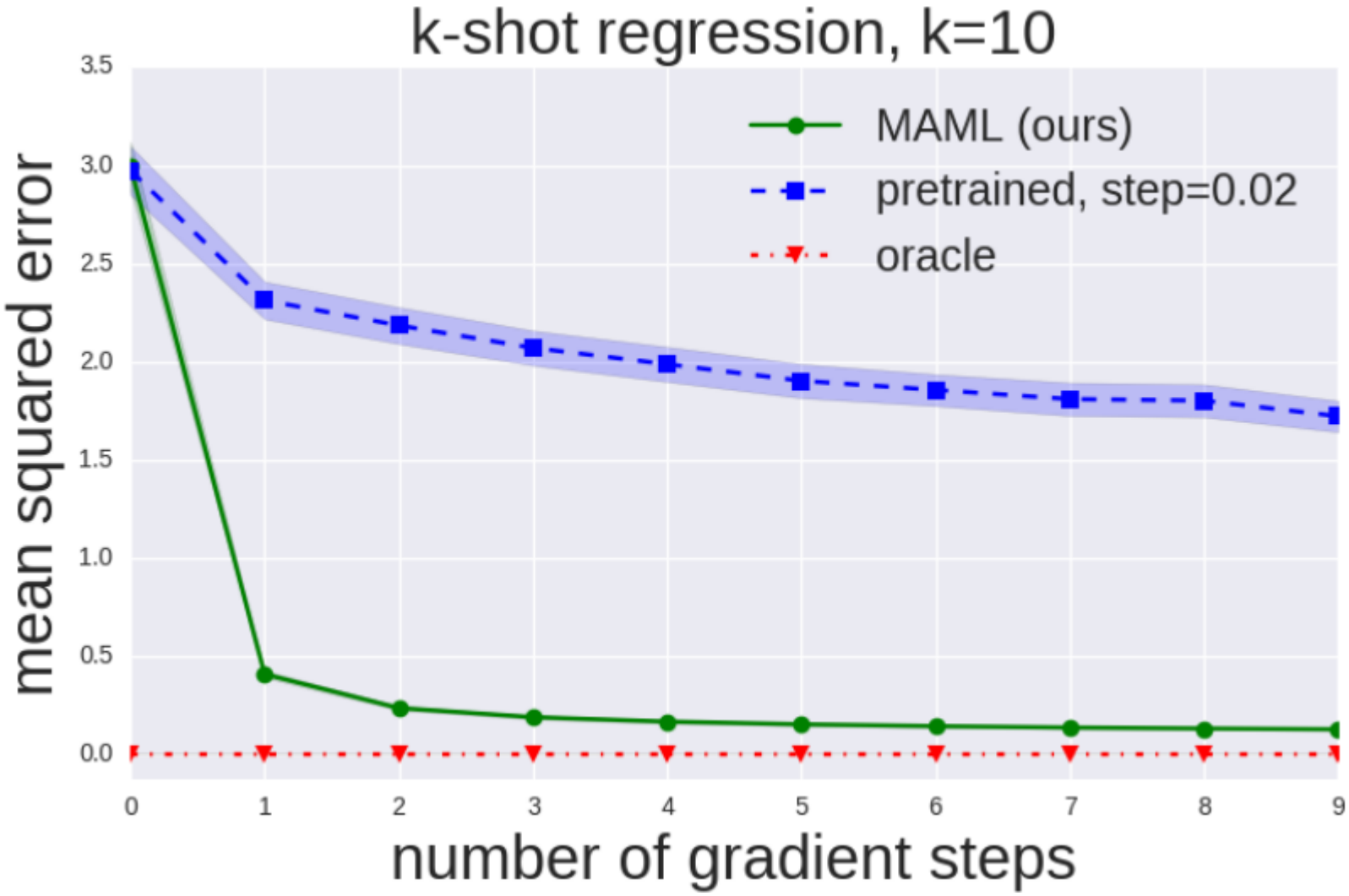


- **Pretrained**:



Regression Experiments Results

Regression： Sine wave



Classification Experiments

Classification： few-shot image recognition

- **The problem setup**:
  - select N unseen classes, provide the model with K different instances of each of the Nclasses
  - evaluate the model`s ability to classify new instances within the N classes
- **Network Architecture(Model):**

  which has 4 modules with a 3 × 3 convolutions and 64 filters
  a ReLU nonlinearity, and 2 × 2 max-pooling
- **Data Set:**

  Omniglot dataset consists of 20 instances of 1623 characters from 50 different alphabets

  MiniImagenet dataset involves 64 training classes, 12 validation classes, and 24 test classes.

Classification Experiments Results

Classification： few-shot image recognition

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%	–	–
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%**

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy 1-shot	5-way Accuracy 5-shot
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%**

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

Reinforcement Learning Experiments

RL： 2D Navigation

- **RL problem**:

  a set of tasks where a point agent must move to different goal positions in 2D

  MAML to maximize performance after 1 policy gradient update using 20 trajectories

- **Evaluation**:

  The reward is the negative squared distance to the goal

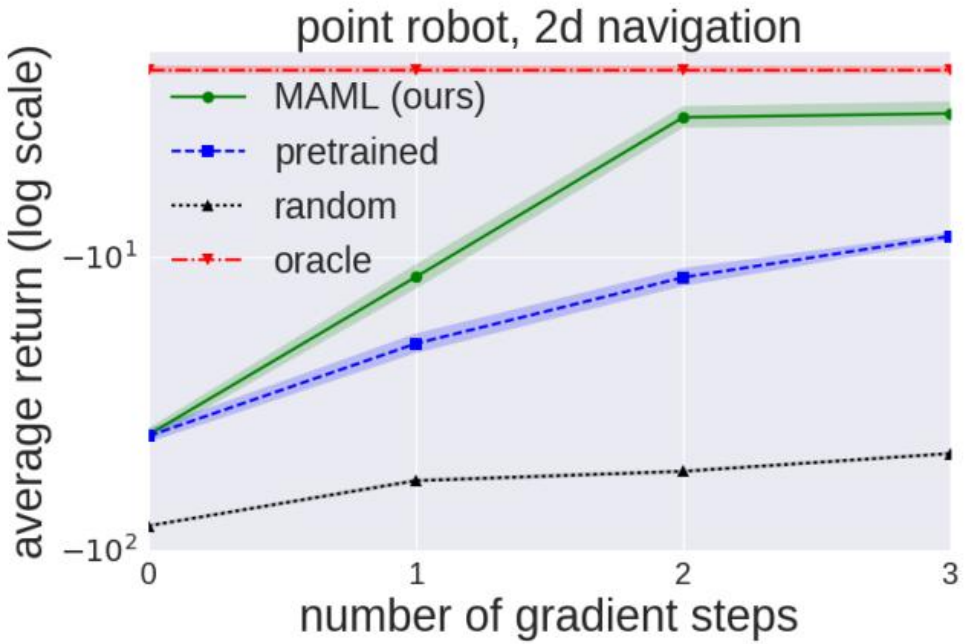
  Adaptation to a new task with up to 4 gradient updates, each with 40 samples

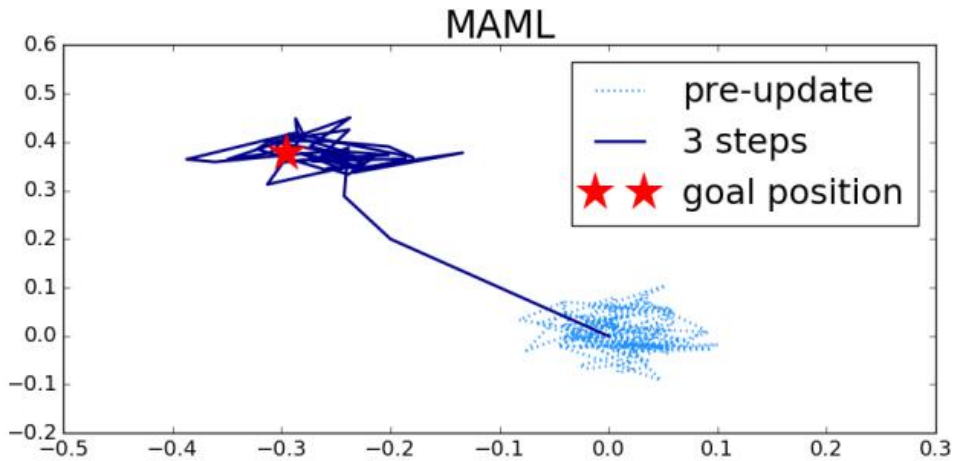
- **Two baselines**:

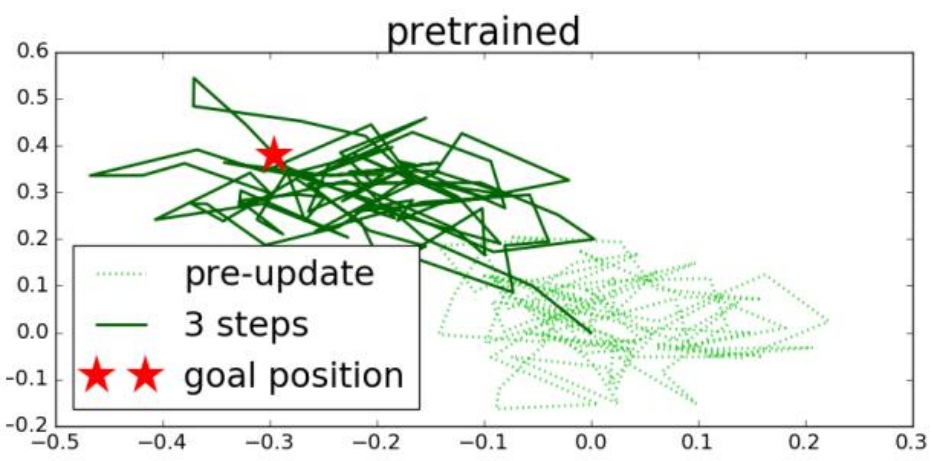
  1. Conventional pretraining on the same set of tasks, random initialization
  2. An oracle policy that receives the goal position as input

RL Experiments Results

RL： 2D Navigation







Conclusion

The advantages

- It is simple
- Does not introduce any learned parameters for meta-learning
- Applicable to any models representation that is amenable to gradient-based training
- Adaptation can be performed with any amount of data and any number of gradient steps

The disadvantages

- 需要计算二次梯度，训练会很慢，很难支持大型深度神经网络
- 无监督数据处理效果不好