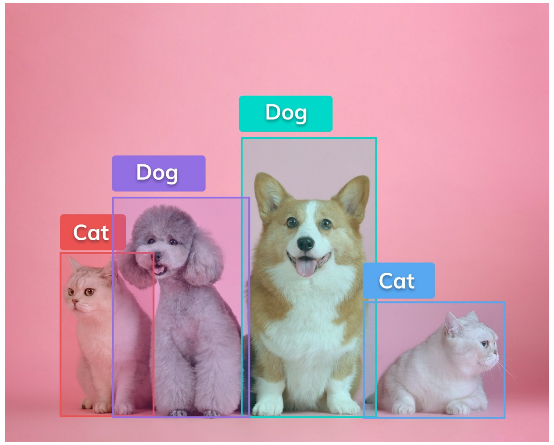
MAML: Model-Agnostic Meta-Learning for Fast Adaptation for Deep Networks

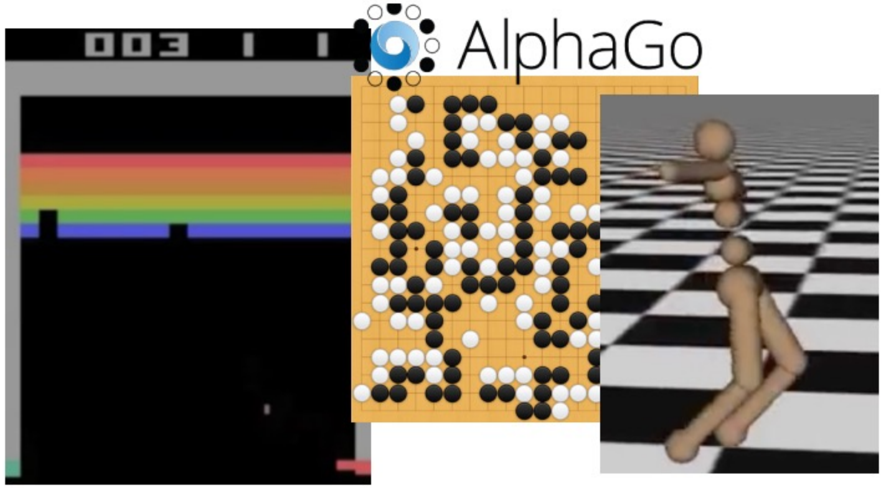
Chelsea Finn **Pieter Abbeel** **Sergey Levine**

Presented by Zixuan Huang

Motivation

- Today's machine learning accomplishes numerous challenging tasks
- Specialists
- Learn **one task** in **one environment** from **scratch**
  - Take long to master new tasks!





Atari

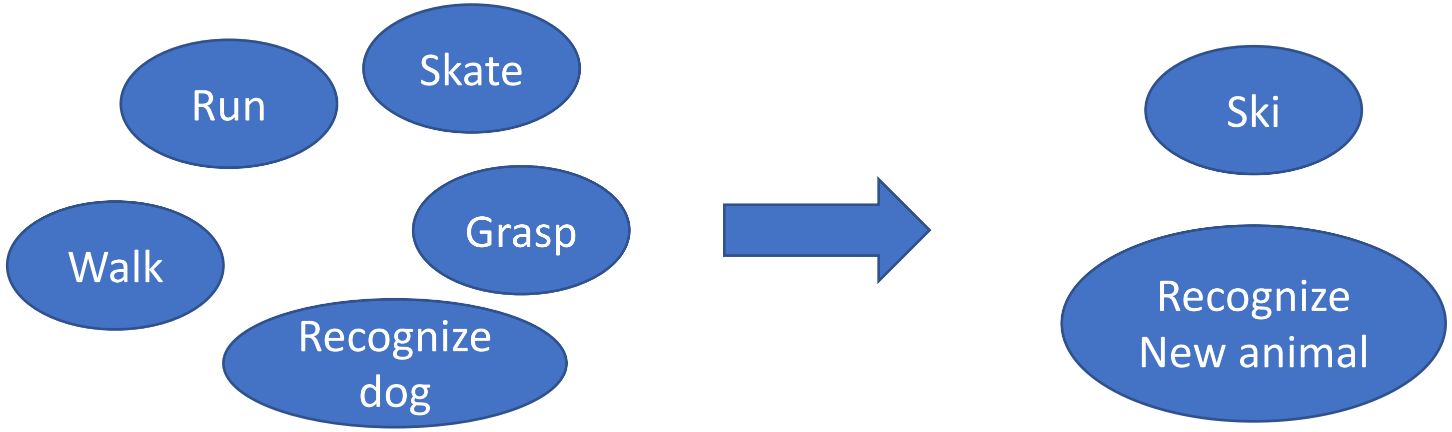
locomotion

Motivation

- Humans are generalists that learn and adapt quickly
- We’re able to
  - Learn new skills
  - Adapt to new environment
  - Recognize new objects
- In a few shots

Motivation

- With past experiences and prior knowledges, we figure out how to learn more efficiently
- Meta-learning: learning to learn
- How do we equip a ML model with such capability?



How does meta-learning work?
An example

Given 1 example of 5 classes:

training data $\mathcal{D}_{\text{train}}$

Classify new examples

test set $\mathbf{x}_{\text{test}}$

How does meta-learning work? An example

Can replace image classification with regression, skill learning, language generation and etc.

Problem setup | Meta Learning

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

Key assumption: meta-training tasks and meta-test task drawn i.i.d. from same task distribution

$$\mathcal{T}_1, \ldots, \mathcal{T}_n \sim p(\mathcal{T}), \mathcal{T}_j \sim p(\mathcal{T})$$

Like before, tasks must share structure.

Comparison to supervised learning

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$ Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

$$\mathbf{y} = g_\phi(\mathbf{x})$$

Meta Supervised Learning:

Inputs: $\underbrace{\mathcal{D}^{\text{tr}}}_{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}$ $\mathbf{x}^{\text{ts}}$ Outputs: $\mathbf{y}^{\text{ts}}$ Data: $\{\mathcal{D}_i\}$

$$\mathbf{y}^{\text{ts}} = f_\theta(\mathcal{D}^{\text{tr}}, \mathbf{x}^{\text{ts}})$$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

Why is this view useful?

Reduces the meta-learning problem to the design & optimization of f.

Prior works

- Learning an update function or update rule
  - LSTM optimizer (Learning to learn by gradient descent by gradient descent)
  - Meta LSTM optimizer (Optimization as a model for few-shot learning)
- Few shot (or meta) learning for specific tasks
  - Generative modeling (Neural Statistician)
  - Image classification (Matching Net., Prototypical Net.)
  - Reinforcement learning (Benchmarking deep reinforcement learning for continuous control)
- Memory-augmented model
  - Learning an RNN that ingests experience

MAML | Overview

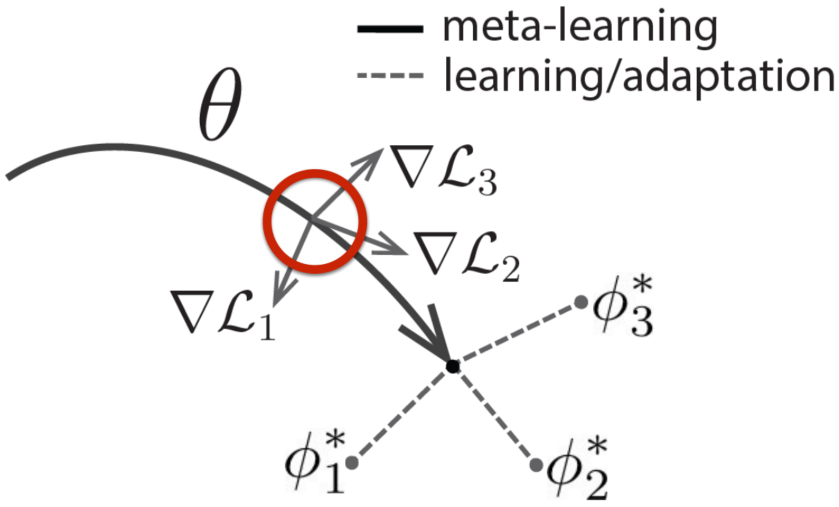
- Model-agnostic
  - Compatible with any model trained with gradient descent
- General
  - Applicable to a variety of different learning problems, including classification, regression, and reinforcement learning.
- Optimization-based
  - An explicit optimization procedure is embedded

MAML | Intuition

- Some internal representations are more transferrable than others.
- Desired model parameter set is θ such that:
  - Applying one (or a small # of) gradient step to θ on a new task will produce optimal behavior
- Find θ that commonly decreases loss of each task after adaptation.

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



MAML | Objective

pre-trained parameters

Fine-tuning [test-time] $$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

training data for new task

Meta-learning $$\min_\theta \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

Key idea: Over many tasks, learn parameter vector θ that transfers via fine-tuning

MAML | Algorithm

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

MAML | Second-order gradient

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

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

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

($\mathcal{L}$ is differentiable)

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

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(I - \alpha \nabla^2_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

Calculation of Hessian matrix is required.
→ MAML suggest 1st order approximation.

MAML | Second-order gradient

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

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

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

($\mathcal{L}$ is differentiable)

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

$$= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \boxed{(I - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta))} \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

In 1st order approximation,
we regard this as identity matrix *I*.

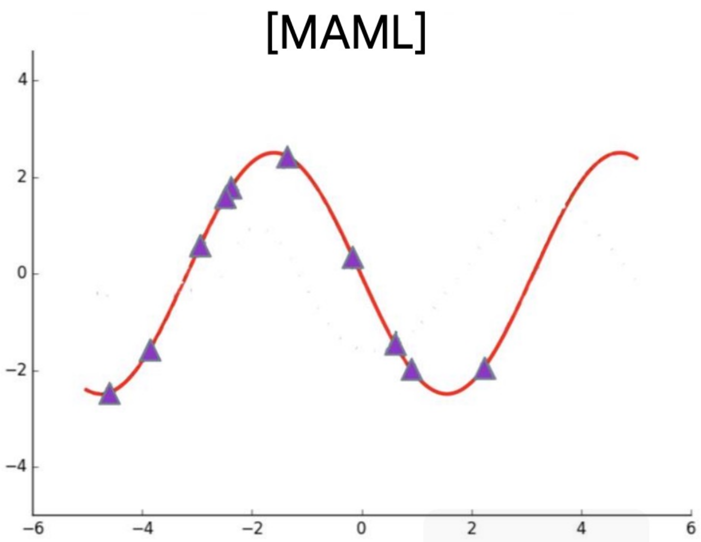
Experiments

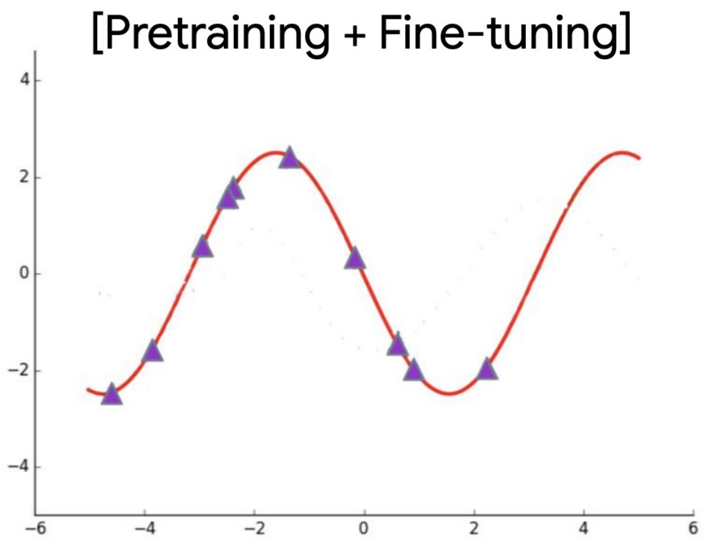
- Supervised regression
- Supervised classification
- Reinforcement Learning

Few-shot regression

- Sinusoid function:
  - Amplitude (A) and phase (ϕ) are varied between tasks
    - A in [0.1, 0.5]
    - ϕ in $[0, \pi]$
  - x in [-5.0, 5.0]
- Loss function: Mean Squared Error (MSE)
- Regressor: 2 hidden layers with 40 units and ReLU
- Training
  - Use **only 1 gradient step** for learner
  - K = 5 or 10 example (5-shot learning or 10-shot learning)
  - Fixed step size (α=0.01) for Adam optimizer.

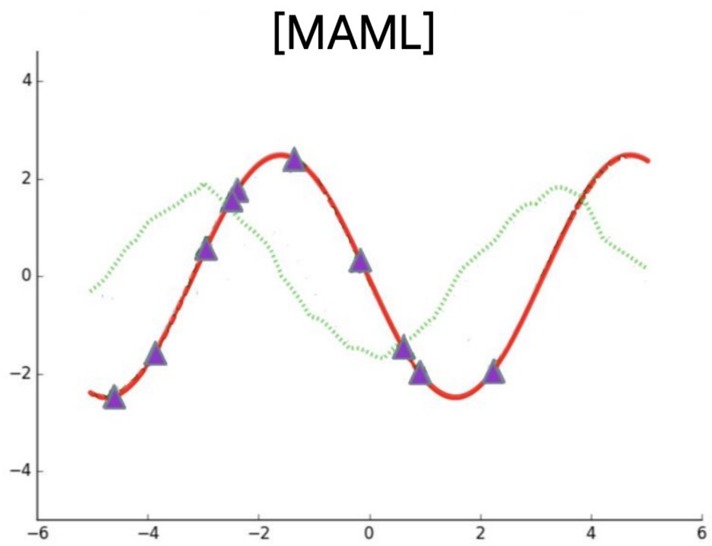
Few-shot regression

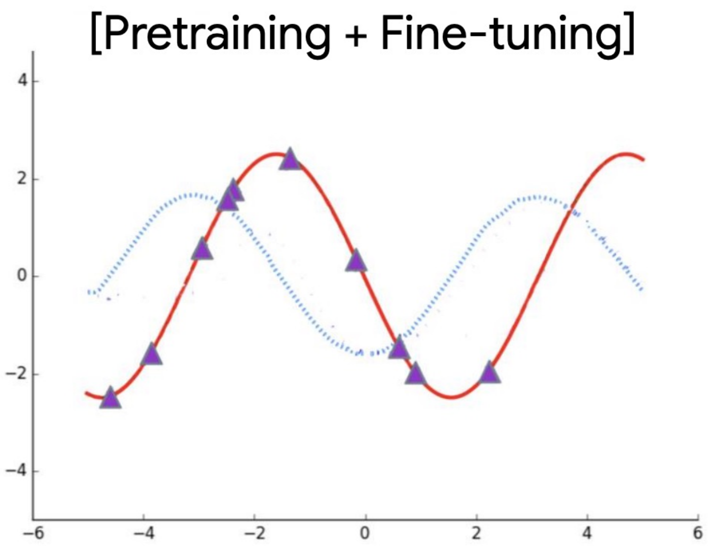




The red line is ground truth.
Fit this sine function with only few (10) samples.

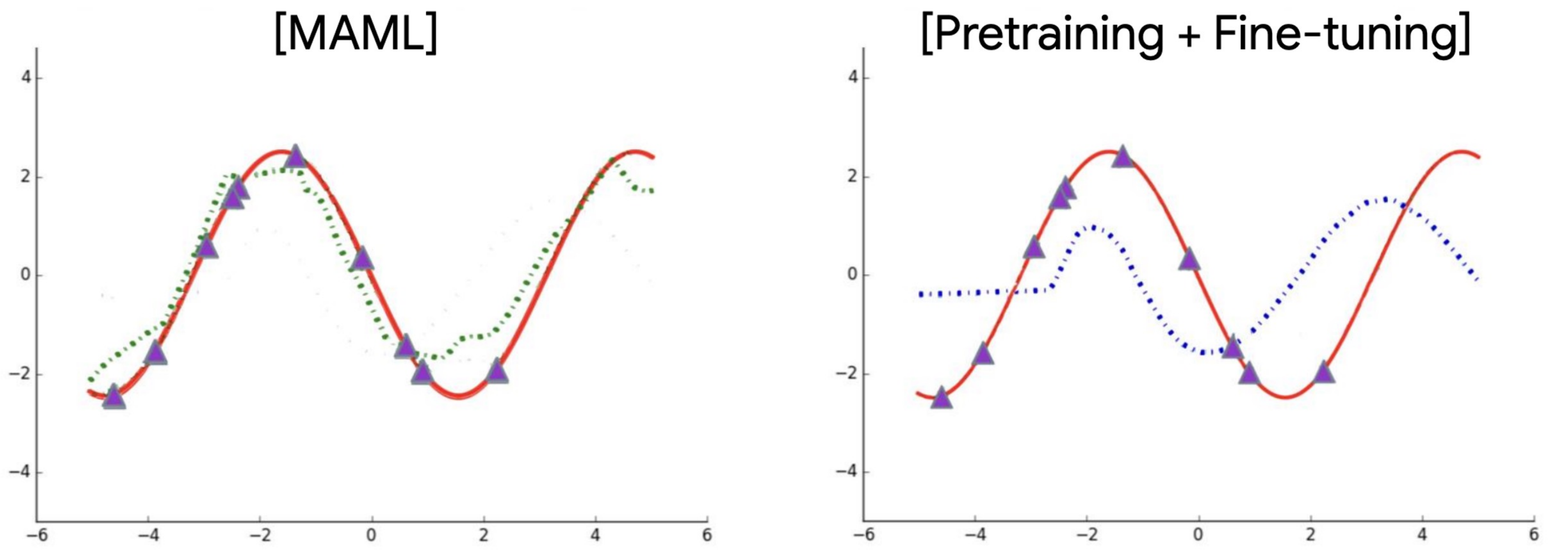
Few-shot regression





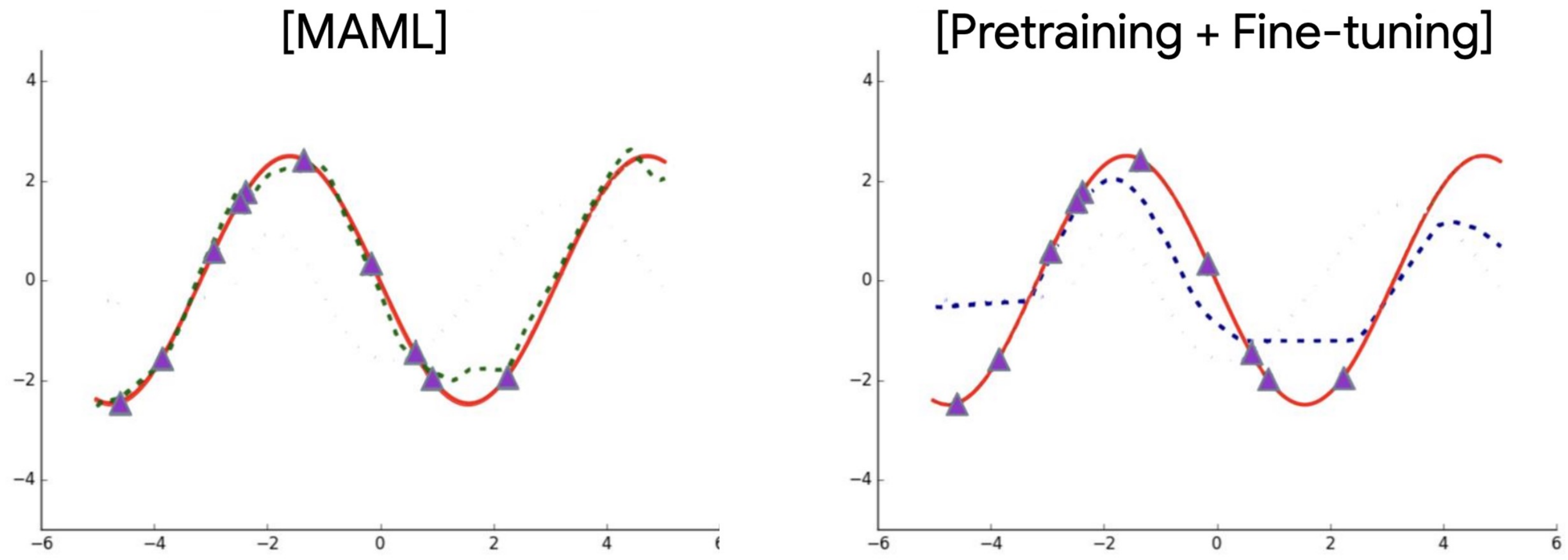
Above plots are the pre-trained function of two models.
(The prediction of meta-parameter of MAML,
The prediction of co-learned parameter of vanilla multi-task learning)

Few-shot regression



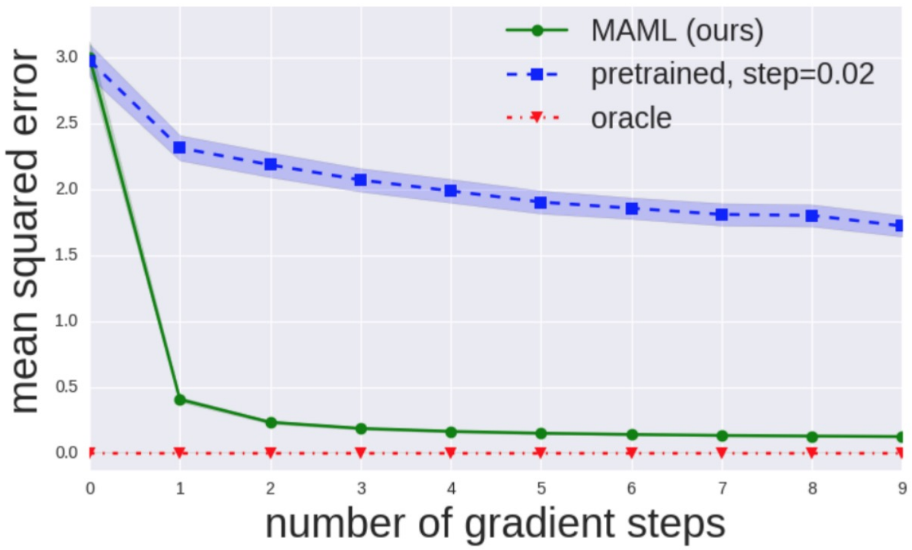
After 1 gradient step update.

Few-shot regression



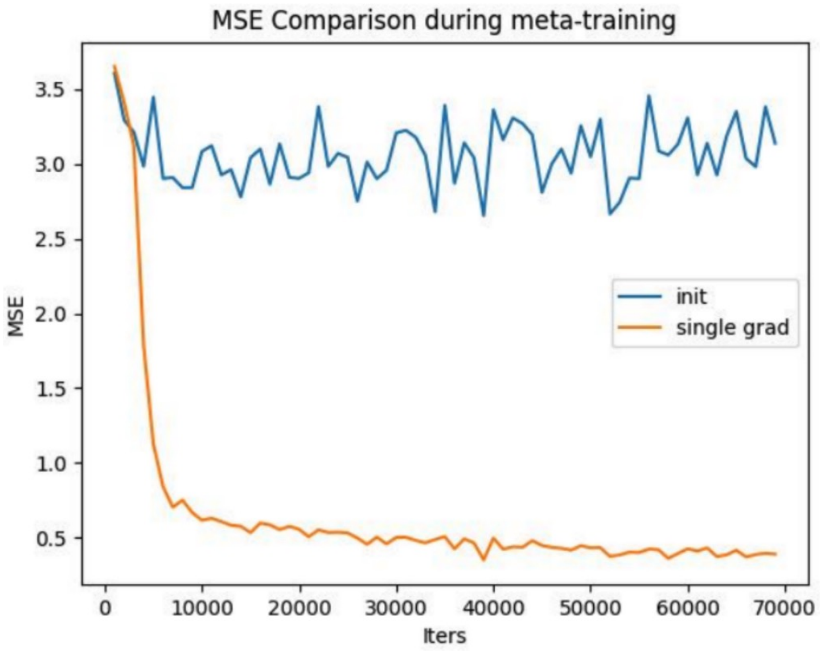
After 10 gradient step update.

MAML only requires 1 gradient step



Vanilla pretrained model adapted slowly,
but, the MAML method quickly adapted **even in one gradient step**.

Performance of meta model



- The performance of the-meta parameters was not improved much in training.
- However, the performance of the single gradient updated parameters
started on meta-parameters improved as training progressed.

Few-shot classification

- Omniglot (Lake et al., 2012)
  - 50 different alphabets, 1623 characters.
  - 20 instances for each characters were drawn by 20 different people.
  - 1200 for training, 423 for test.
- Mini-Imagenet (Ravi & Larochelle, 2017)
  - Classes for each set: train=64, validation=12, test=24.

Few-shot classification

MAML outperforms methods that are specially designed for this task

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%**

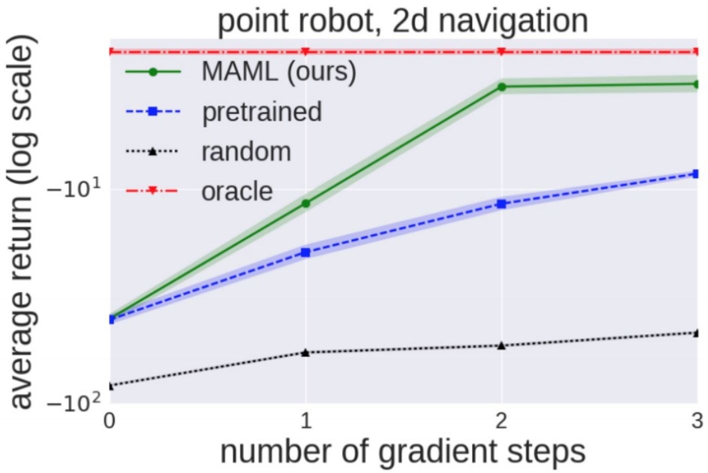
Reinforcement learning

- rllab benchmark suite
- Neural network policy with two hidden layers of size 100 with ReLU
- Gradients updates are computed using vanilla policy gradient (REINFORCE)
  and trust-region policy (TRPO) optimization as meta-optimizer.

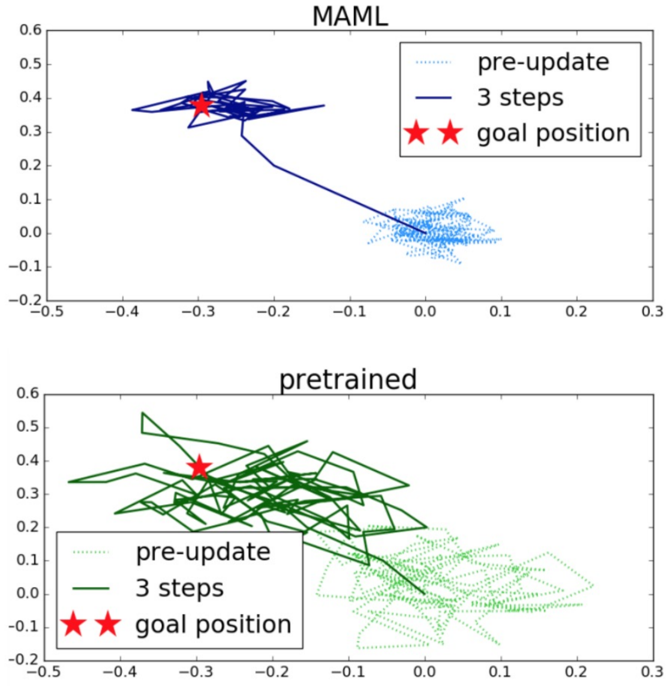
- Comparison
  - Pretraining one policy on all of the tasks and fine-tuning
  - Training a policy from randomly initialized weights
  - Oracle policy

Reinforcement learning

- 2d navigation

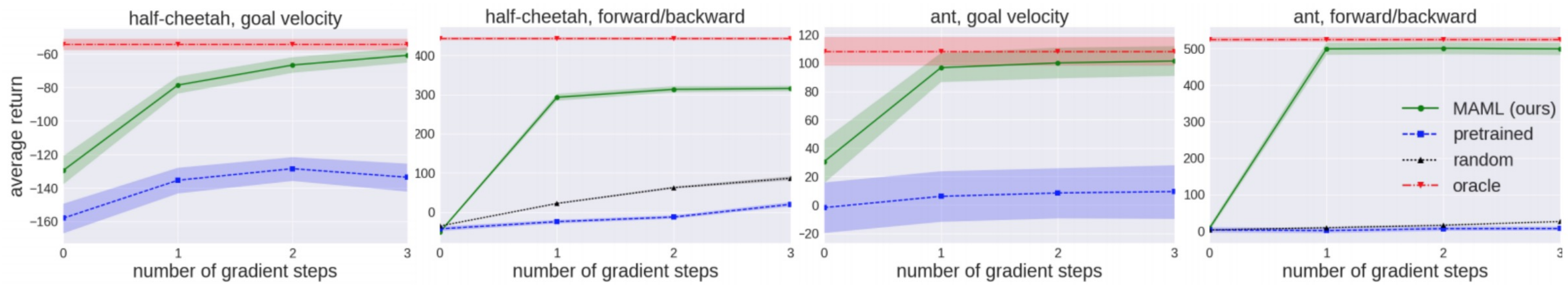


num. grad steps	0	1	2	3
context vector	−42.42	−13.90	−5.17	**−3.18**
MAML (ours)	−40.41	**−11.68**	**−3.33**	−3.23



Reinforcement learning

- Locomotion
  - High-dimensional locomotion tasks with the MuJoCo simulator



num. grad steps	0	1	2	3
context vector	−40.49	−44.08	−38.27	−42.50
MAML (ours)	−50.69	**293.19**	**313.48**	**315.65**

Conclusion

1. MAML is a meta learning technique that reuses past experiences to achieve fast adaptation on new tasks.
2. It's simple, model-agnostic, and generally applicable to many tasks such as classification, regression and RL.

3. It can be viewed from:
    1. **Feature learning standpoint**: building an internal representation that is broadly suitable for many tasks
    2. **Dynamical system standpoint**: Maximizing the sensitivity of loss function with respect to the parameters.

Discussion

- **Multi-task Learning** vs **Meta Learning**.
  - Why don't we learn a single set of weights that are applicable to many tasks?
- **Assumption of Meta Learning.**
  - Meta Learning assumes the tasks during training and test are drawn from the same distribution. But in reality, it's inevitable that we encounter tasks that are out-of-distribution. In this case, is MAML still going to work?
- **Continuous setting**
  - MAML assumes access to an offline training dataset
  - What if the training data come in sequentially?
  - How to fight against catastrophic forgetting?