MAML and Reptile

Seoul Artificial Intelligence
Martin Kersner

People can learn fast. They don't need many examples to learn from in order to distinguish between different objects.

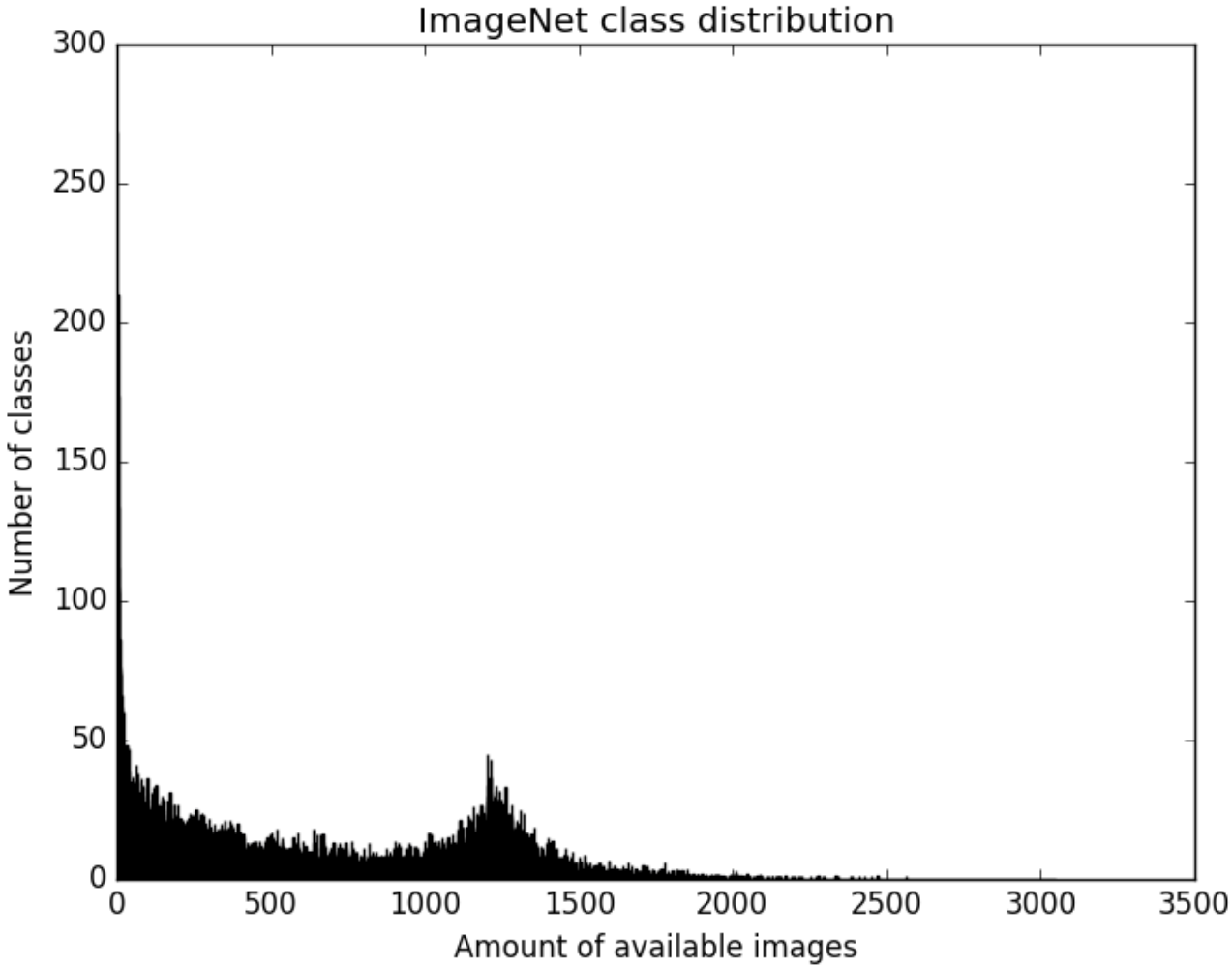
WHY?

Mountain Bluebird, Eastern Bluebird, Superb fairywren

Eastern Bluebird, Superb fairywren, Mountain Bluebird

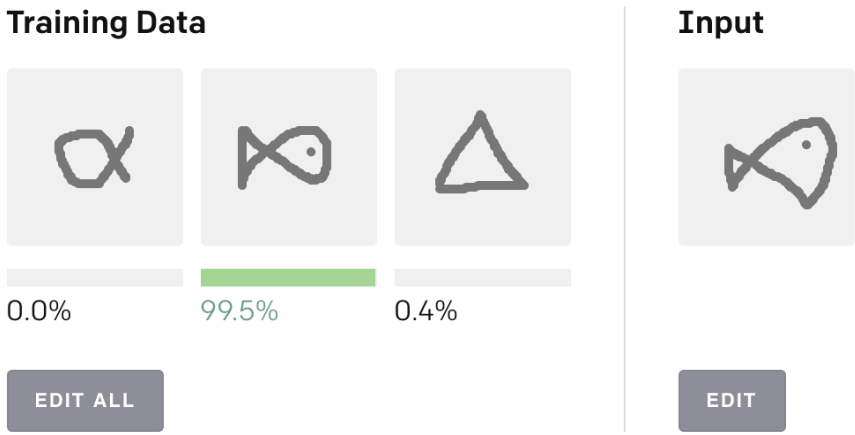
It is estimated that a child has learned almost all of the 10-30 thousand object categories in the world by the age of six.

Irving Biederman: Recognition-by-Components: a theory of human understanding


ImageNet class distribution
Number of classes
Amount of available images
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000
3500

Few-shot learning

The key motivation for the few-shot learning technique is that systems, like humans, can use **prior knowledge** about object categories to classify new objects.



https://blog.openai.com/reptile/

Few-shot meta-learning

The goal of few-shot meta-learning is to train a model that can **quickly adapt to a new task** using only a few data points and training iterations.

This kind of fast and flexible learning is challenging, since the learner must **integrate its prior experience with a small amount of new information**, while avoiding overfitting to the new data.

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

Chelsea Finn [1] **Pieter Abbeel** [1 2] **Sergey Levine** [1]

Abstract

We propose an algorithm for meta-learning that is model-agnostic, in the sense that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning. The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. In our approach, the parameters of the model are explicitly trained such that a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task. In effect, our method trains the model to be easy to fine-tune. We demonstrate that this approach leads to state-of-the-art performance on two few-shot image classification benchmarks, produces good results on few-shot regression, and accelerates fine-tuning for policy gradient reinforcement learning with neural network policies.

1. Introduction

the form of computation required to complete the task.

In this work, we propose a meta-learning algorithm that is general and model-agnostic, in the sense that it can be directly applied to any learning problem and model that is trained with a gradient descent procedure. Our focus is on deep neural network models, but we illustrate how our approach can easily handle different architectures and different problem settings, including classification, regression, and policy gradient reinforcement learning, with minimal modification. In meta-learning, the goal of the trained model is to quickly learn a new task from a small amount of new data, and the model is trained by the meta-learner to be able to learn on a large number of different tasks. The key idea underlying our method is to train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task. Unlike prior meta-learning methods that learn an update function or learning rule (Schmidhuber, 1987; Bengio et al., 1992; Andrychowicz et al., 2016; Ravi & Larochelle, 2017), our algorithm does not expand the number of learned parameters nor place constraints on the model architecture (e.g. by requiring a recurrent model (Santoro et al., 2016) or a Siamese network (Koch, 2015)), and it can be readily combined with fully connected, convolutional, or recurrent neu-

Model-Agnostic Meta-Learning

- **any learning problem** that is trained with a **gradient descent**
- no constraints on the model architecture
- variety of loss functions
- does not expand the number of learned parameters
- train model's **initial parameters**
- easy to **fine-tune**

Feature learning standpoint

If the **internal representation** is suitable to many tasks, simply fine-tuning the parameters slightly can produce good results.

Fine-tuning from pretrained models is standard technique to **achieve good results fast**.

Dynamical system standpoint

Maximizing the sensitivity of the loss functions of new tasks with respect to the parameters.

When the **sensitivity** is high, **small local changes** to the parameters can lead to **large improvements** in the task loss.

Problem setup

Train model *f*, that maps observations *x* to outputs *a*.

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

- Loss function *L*
- Distribution over initial observations $q(x_1)$
- Transition distribution $q(x_{t+1}|x_t, a_t)$
- Episode length *H*

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**

Meta-objective

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

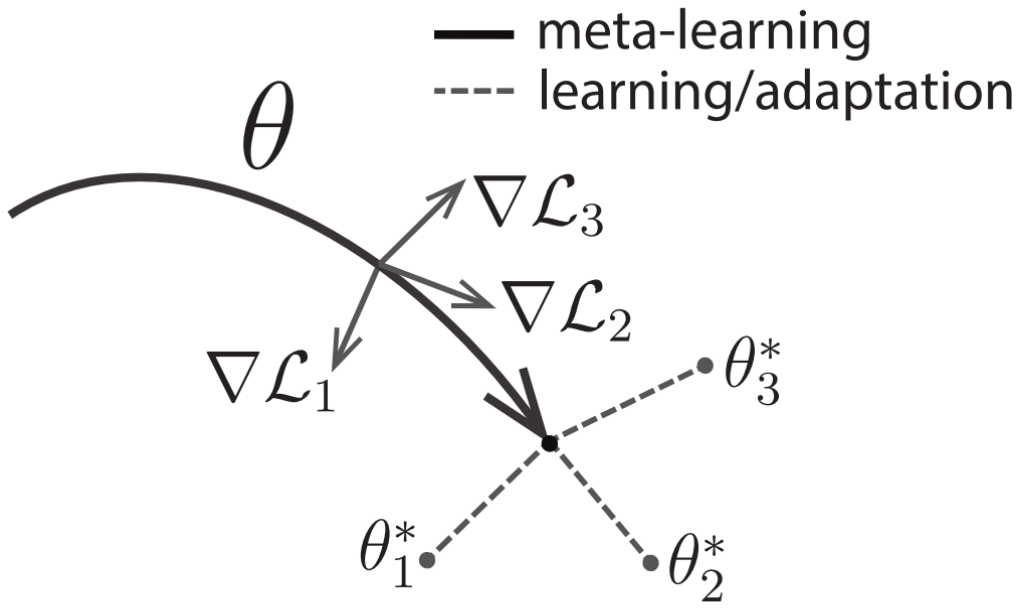


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Cons

A significant **computational expense** in MAML comes from the **use of second derivatives** when backpropagating the meta-gradient through the gradient operator in the meta-objective.

Solution?

Don't compute the second derivatives!

"Species" of MAML

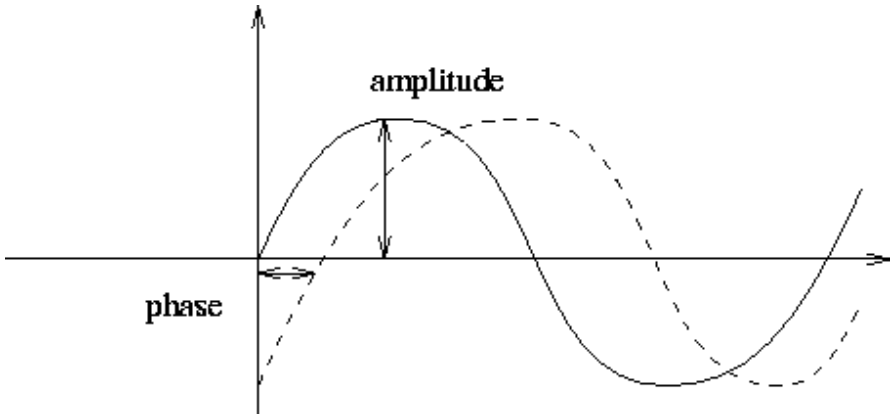
The domains differ in the form of loss function and in how data is generated by the task and presented to the model, but the same basic adaptation mechanism can be applied in all cases.

- Supervised regression
- Supervised classification
- Reinforcement learning

Regression

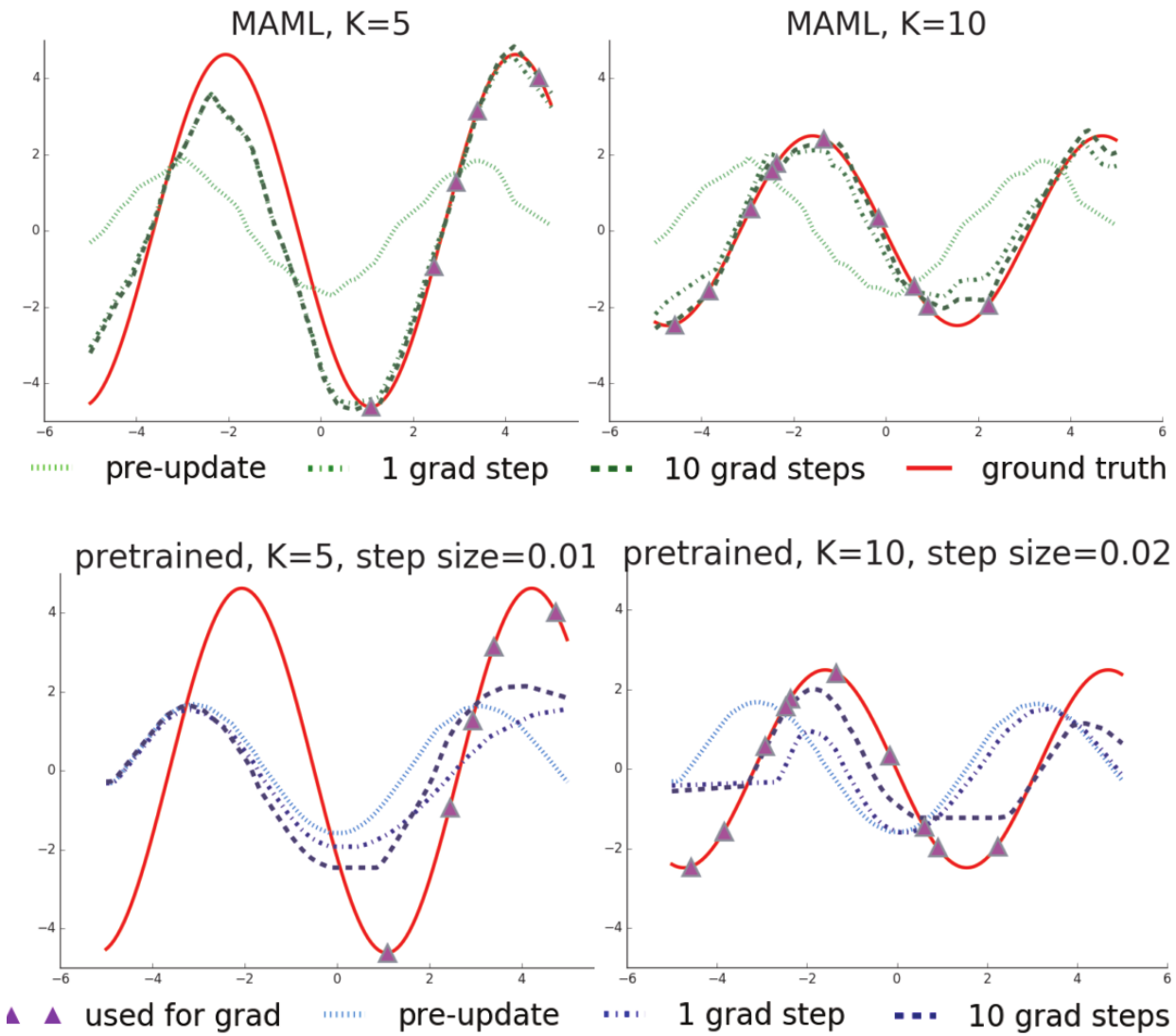
Sinusoid regression

- amplitude within *[0.1, 5.0]*
- phase within *[0, π]*
- input and output both have a dimensionality of *1*
- data points sampled uniformly from *[−5.0, 5.0]*



http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT4/node2.html

network: 2 hidden layers of size 40 with ReLU
loss: mean-squared error



N-way classification

Omniglot

20 instances of 1623 characters from 50 different alphabets
1200 for training, the rest for testing

525 character concepts

Brenden M. Lake, Ruslan Salakhutdinov, Joshua B. Tenenbaum: Human-level concept learning through probabilistic program induction

Omniglot evaluation

no conv: Fc+BN+ReLU: 256 -> 128 -> 64 -> 64 -> linear -> softmax
conv: (3×3×64 strided conv + BatchNorm + ReLU) * 4 -> softmax

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%**

MiniImagenet

60,000 color images of size 84×84 with 100 classes (600 per class)
64 training classes, 12 validation classes, and 24 test classes

$$L_{train} =$$

n01614925, n01632777, n01641577, n01664065, n01687978, n01695060, n01729322, n01773157, n01833805, n01871265, n01877812, n01978455, n01986214, n02013706, n02066245, n02071294, n02088466, n02090379, n02091635, n02096437, n02097130, n02099429, n02108089, n02108915, n02109047, n02109525, n02111889, n02115641, n02123045, n02129165, n02167151, n02206856, n02264363, n02279972, n02342885, n02346627, n02364673, n02454379, n02481823, n02486261, n02494079, n02655020, n02793495, n02804414, n02808304, n02837789, n02895154, n02909870, n02917067, n02966687, n03000684, n03014705, n03041632, n03045698, n03065424, n03180011, n03216828, n03355925, n03384352, n03424325, n03452741, n03482405, n03494278, n03594734, n03599486, n03630383, n03649909, n03676483, n03690938, n03742115, n03868242, n03877472, n03976467, n03976657, n03998194, n04026417, n04069434, n04111531, n04118538, n04200800

$$L_{test} =$$

n04201297, n04204347, n04239074, n04277352, n04370456, n04409515, n04456115, n04479046, n04487394, n04525038, n04591713, n04599235, n07565083, n07613480, n07695742, n07714571, n07717410, n07753275, n10148035, n12768682

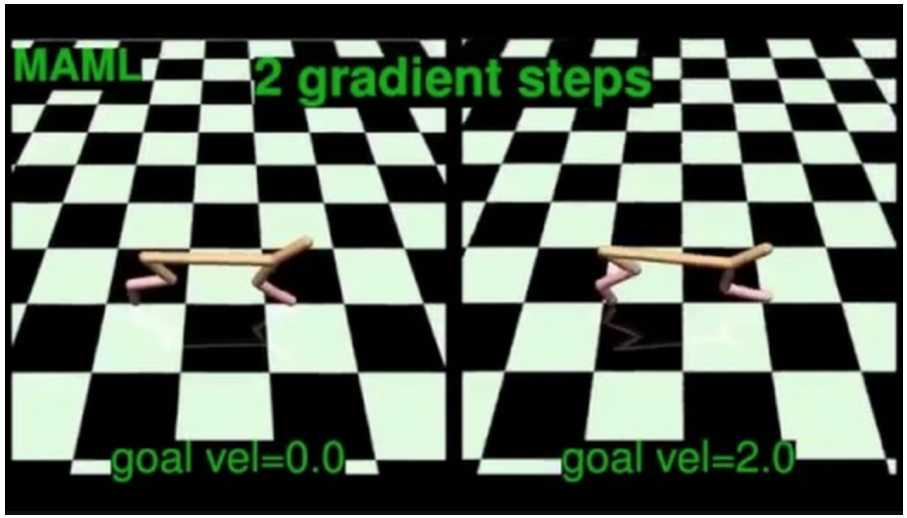
Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, Daan Wierstra: Matching Networks for One Shot Learning

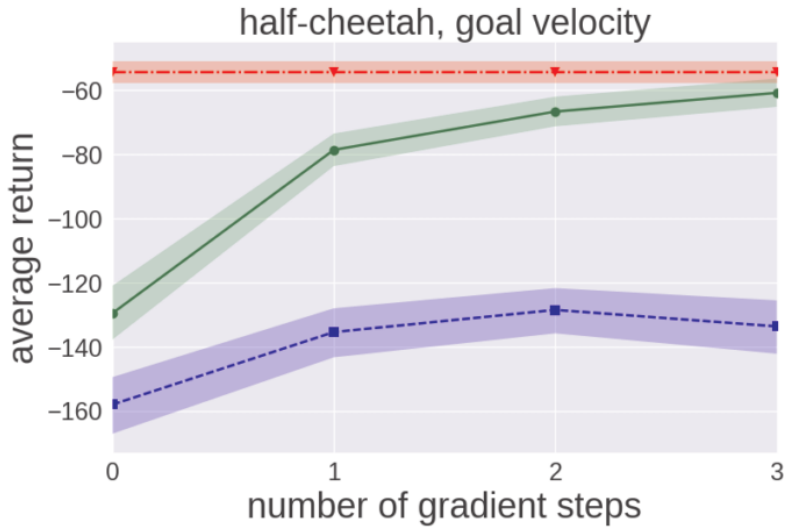
MiniImagenet evaluation

(3×3×32 conv + BatchNorm + ReLU + 2×2 MaxPool) * 4 -> softmax

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)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Reinforcement learning





https://sites.google.com/view/maml

MAML implementation

https://github.com/cbfinn/maml

vs

https://github.com/martinkersner/maml

References

https://arxiv.org/abs/1703.03400
https://sites.google.com/view/maml
https://github.com/cbfinn/maml
http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/
https://towardsdatascience.com/model-agnostic-meta-learning-maml-8a245d9bc4ac
http://people.eecs.berkeley.edu/~cbfinn/
http://rail.eecs.berkeley.edu/nips_demo.html
https://github.com/cbfinn/maml_rl

Reptile: a Scalable Metalearning Algorithm

Alex Nichol and John Schulman
OpenAI
{alex, joschu}@openai.com

Abstract

This paper considers metalearning problems, where there is a distribution of tasks, and we would like to obtain an agent that performs well (i.e., learns quickly) when presented with a previously unseen task sampled from this distribution. We present a remarkably simple metalearning algorithm called Reptile, which learns a parameter initialization that can be fine-tuned quickly on a new task. Reptile works by repeatedly sampling a task, training on it, and moving the initialization towards the trained weights on that task. Unlike MAML, which also learns an initialization, Reptile doesn't require differentiating through the optimization process, making it more suitable for optimization problems where many update steps are required. We show that Reptile performs well on some well-established benchmarks for few-shot classification. We provide some theoretical analysis aimed at understanding why Reptile works.

1 Introduction

While machine learning systems have surpassed humans at many tasks, they generally need far more data to reach the same level of performance. For example, Schmidt et al. [Sch09; STT12] showed that human subjects can recognize new object categories based on a few example images.

Performs **stochastic gradient descent** (SGD) on each task in a standard way — it does not unroll a computation graph or calculate any second derivatives.

Algorithm 2 Reptile, batched version

Initialize ϕ
for iteration $= 1, 2, \ldots$ **do**
  Sample tasks $\tau_1, \tau_2, \ldots, \tau_n$
  for $i = 1, 2, \ldots, n$ **do**
    Compute $W_i = \mathrm{SGD}(L_{\tau_i}, \phi, k)$
  end for
  Update $\phi \leftarrow \phi + \epsilon \frac{1}{k} \sum_{i=1}^{n} (W_i - \phi)$
end for

Evaluation

Algorithm	1-shot 5-way	5-shot 5-way	1-shot 20-way	5-shot 20-way
MAML + Transduction	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
1^{st}-order MAML + Transduction	$98.3 \pm 0.5\%$	$99.2 \pm 0.2\%$	$89.4 \pm 0.5\%$	$97.9 \pm 0.1\%$
Reptile	$95.32 \pm 0.05\%$	$98.87 \pm 0.02\%$	$88.27 \pm 0.30\%$	$97.07 \pm 0.12\%$
Reptile + Transduction	$97.97 \pm 0.08\%$	$99.47 \pm 0.04\%$	$89.36 \pm 0.20\%$	$97.47 \pm 0.10\%$

Table 2: Results on Omniglot

Algorithm	1-shot 5-way	5-shot 5-way
MAML + Transduction	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
1^{st}-order MAML + Transduction	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
Reptile	$45.79 \pm 0.44\%$	$61.98 \pm 0.69\%$
Reptile + Transduction	$48.21 \pm 0.69\%$	$66.00 \pm 0.62\%$

Table 1: Results on Mini-ImageNet

demo

https://gist.github.com/joschu/f503500cda64f2ce87c8288906b09e2d

References

https://arxiv.org/abs/1803.02999
https://blog.openai.com/reptile/