Meta Learning

Seminar on Deep Neural Networks

Tobias Birchler

ETH Zürich

2021

Outline

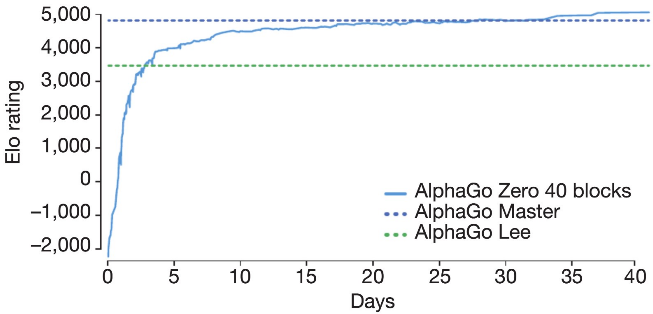
Motivation

Motivation

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM

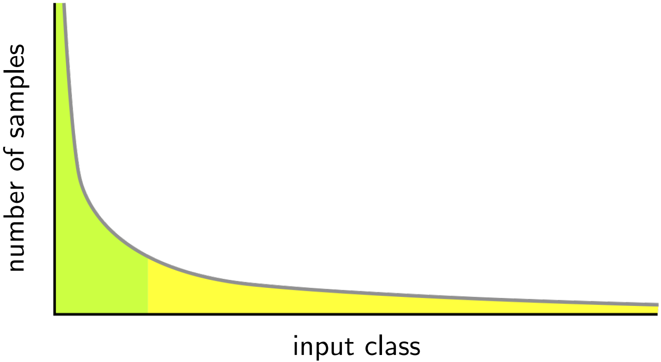
Motivation


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM


Elo rating
5,000
4,000
3,000
2,000
1,000
0
−1,000
−2,000
0
5
10
15
20
25
30
35
40
Days
AlphaGo Zero 40 blocks
AlphaGo Master
AlphaGo Lee

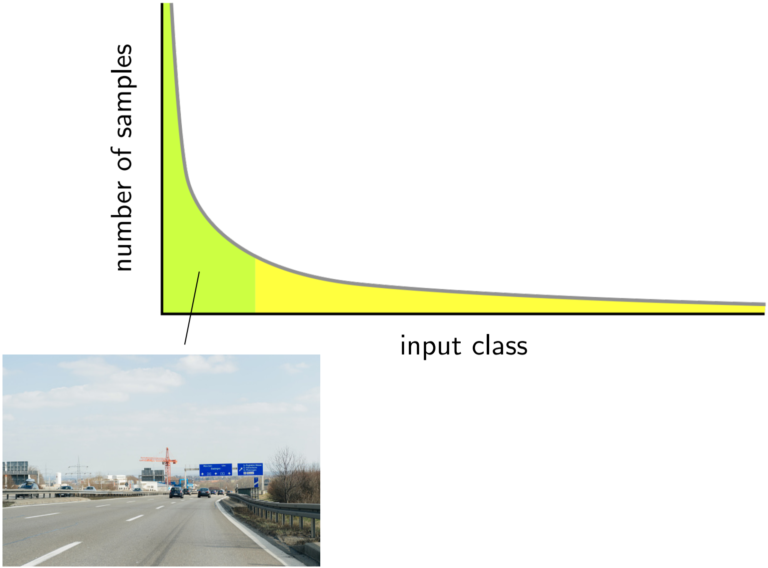
Motivation

Many RL applications have long tailed state distributions

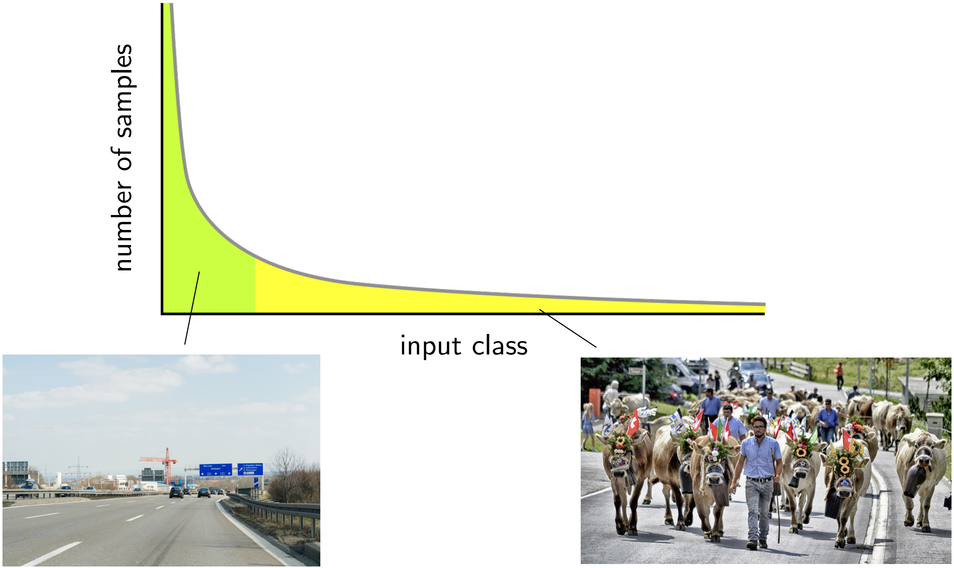


Motivation

Many RL applications have long tailed state distributions



Many RL applications have long tailed state distributions



Definition (meta)

referring to itself or to something of its own type (Camebridge Dictionary)

Motivation

Definition (meta)

referring to itself or to something of its own type (Camebridge Dictionary)

Remark

Meta Learning is also known as "Learning to Learn"

Motivation

training data

Braque Cezanne

test datapoint

By Braque or Cezanne?

Problem Definition - Supervised Learning

"Normal" supervised learning:

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)]$$

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data.

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data.

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

"Meta" supervised learning:

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y) = L_T(M_{A(D^{\text{train}})}, D^{\text{test}})$$

"Meta" supervised learning:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{T\sim P(T)}[L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})]$$

Problem Definition - Supervised Learning

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y) = L_T(M_{A(D^{\text{train}})}, D^{\text{test}})$$

"Meta" supervised learning:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{T\sim P(T)}[L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})] \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})$$

"Normal" supervised learning:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{(x,y)\sim P(x,y)}[L(M_\phi(x), y)] \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

We solve this using a learning algorithm A and the training data. The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y) = L_T(M_{A(D^{\text{train}})}, D^{\text{test}})$$

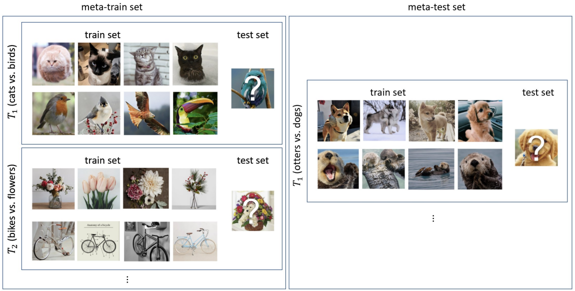
"Meta" supervised learning:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{T\sim P(T)}[L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})] \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$

Problem Definition - Supervised Learning



"Meta" supervised learning:

$$\theta^* = \arg\min_\theta \mathbb{E}_{T\sim P(T)}[L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})] \approx \arg\min_\theta \sum_{T \in \mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$

Problem Definition - Generic Learning

"Normal" supervised learning:

"Normal" generic learning:

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

$$L_T(M_{A(T^{\text{tr}})})$$

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

$$L_T(M_{A(T^{\text{tr}})})$$

"Meta" supervised learning:

"Meta" generic learning:

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

$$L_T(M_{A(T^{\text{tr}})})$$

"Meta" supervised learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})$$

"Meta" generic learning:

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_{\phi}(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_{\phi})$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

$$L_T(M_{A(T^{\text{tr}})})$$

"Meta" supervised learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_{\theta}(D_T^{\text{train}})}, D_T^{\text{test}})$$

"Meta" generic learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_{\theta}(T^{\text{tr}})})$$

Problem Definition - Generic Learning

"Normal" supervised learning:

$$\phi^* \approx \arg\min_{\phi} \sum_{(x,y)\in D^{\text{test}}} L(M_\phi(x), y)$$

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources.
The resulting loss is

$$\sum_{(x,y)\in D^{\text{test}}} L(M_{A(D^{\text{train}})}(x), y)$$

$$L_T(M_{A(T^{\text{tr}})})$$

"Meta" supervised learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(D_T^{\text{train}})}, D_T^{\text{test}})$$

"Meta" generic learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T\in\mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(T^{\text{tr}})})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$

Problem Definition - Generic Learning

"Normal" generic learning:

$$\phi^* \approx \arg\min_{\phi} L_T(M_\phi)$$

We solve this using a learning algorithm A and the given training resources. The resulting loss is

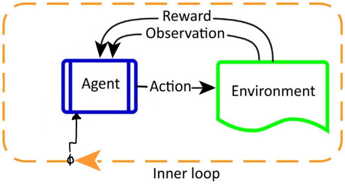
$$L_T(M_{A(T^{\text{tr}})})$$

"Meta" generic learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T \in \mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(T^{\text{tr}})})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$

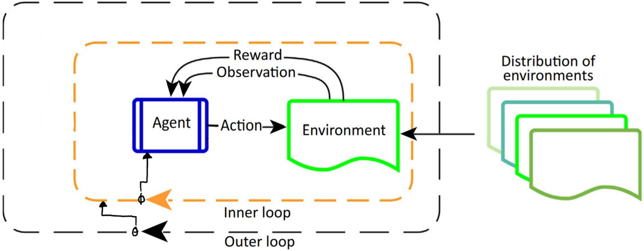


"Meta" generic learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T \in \mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(\mathcal{T}^{\text{tr}})})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$



"Meta" generic learning:

$$\theta^* \approx \arg\min_{\theta} \sum_{T \in \mathcal{T}^{\text{meta-test}}} L_T(M_{A_\theta(T^{\text{tr}})})$$

We solve this using a meta learning algorithm f and the meta-training data

$$\theta^* \approx f(\mathcal{T}^{\text{meta-train}})$$

Models - RL^2

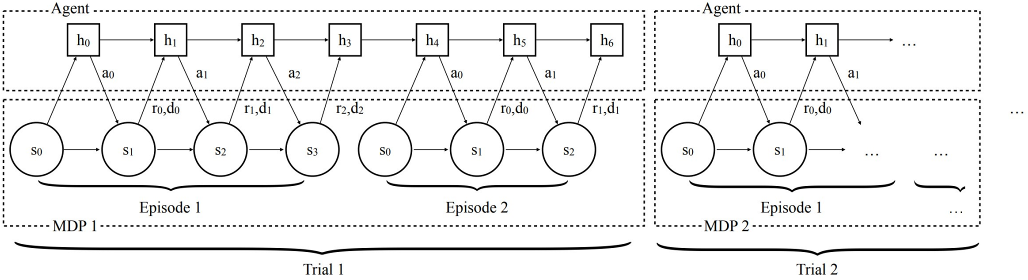
RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

Yan Duan[†‡], John Schulman[†‡], Xi Chen[†‡], Peter L. Bartlett[†], Ilya Sutskever[‡], Pieter Abbeel[†‡]

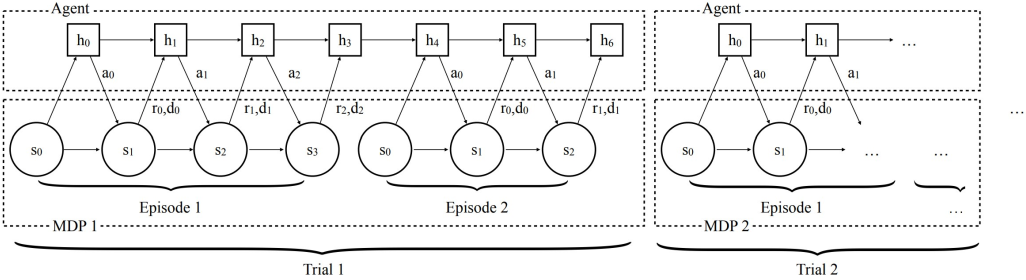
Models - RL^2 - Model Definition

In RL^2 A_θ is a RNN (with GRU cells actually).
The meta parameters θ are the parameters of the RNN.

In RL^2 A_θ is a RNN (with GRU cells actually).
The meta parameters θ are the parameters of the RNN.

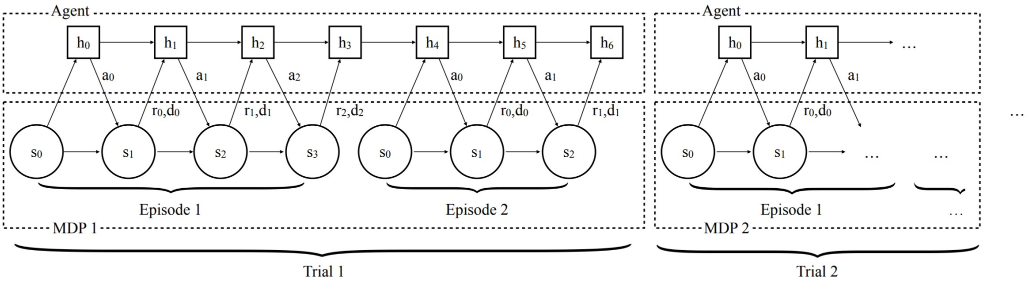


In RL^2 A_θ is a RNN (with GRU cells actually).
The meta parameters θ are the parameters of the RNN.
Hidden state activations h can be seen as internal state of the agent.



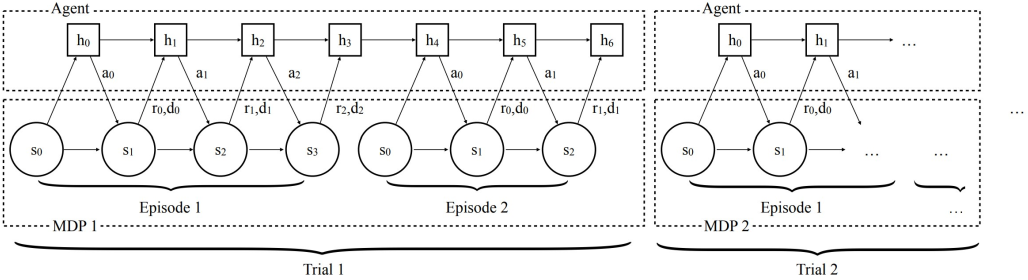
Models - RL^2 - Model Definition

In RL^2 A_θ is a RNN (with GRU cells actually).
The meta parameters θ are the parameters of the RNN.
Hidden state activations h can be seen as internal state of the agent.



The meta problem can be cast as POMDP (more details: link).

In RL^2 A_θ is a RNN (with GRU cells actually).
The meta parameters θ are the parameters of the RNN.
Hidden state activations h can be seen as internal state of the agent.



The meta problem can be cast as POMDP (more details: link).
As meta learning algorithm f the authors use standard TRPO.

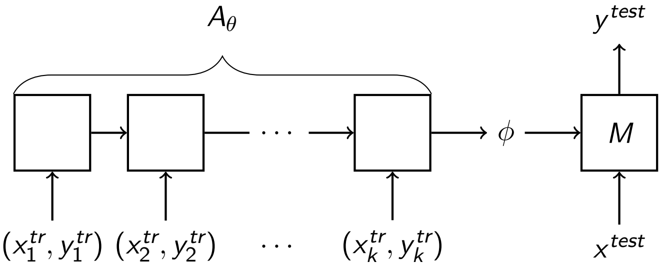
Models - RL^2 - Model Class

Models with entire neural networks as learning algorithm are known as **black-box meta learning** models.

Models with entire neural networks as learning algorithm are known as **black-box meta learning** models.

Example

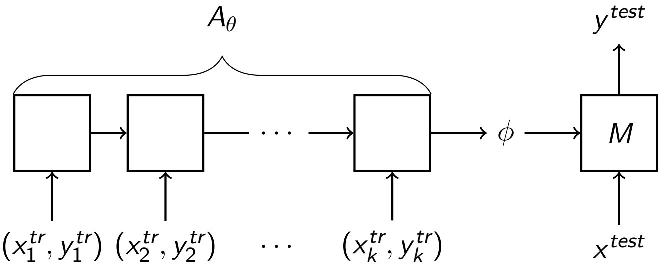
Supervised learning:



Models with entire neural networks as learning algorithm are known as **black-box meta learning** models.

Example

Supervised learning:



The meta learning algorithm f for such models is usually just an off-the-shelf optimization algorithm (e.g. SGD: $\theta \leftarrow \theta - \alpha \nabla_\theta L_T(M_{A_\theta(T^{\text{tr}})})$).

Models - RL^2 - Results

Table 1: MAB Results. Each grid cell records the total reward averaged over 1000 different instances of the bandit problem. We consider $k \in \{5, 10, 50\}$ bandits and $n \in \{10, 100, 500\}$ episodes of interaction. We highlight the best-performing algorithms in each setup according to the computed mean, and we also highlight the other algorithms in that row whose performance is not significantly different from the best one (determined by a one-sided t-test with $p = 0.05$).

Setup	Random	Gittins	TS	OTS	UCB1	ϵ-Greedy	Greedy	RL^2
$n = 10, k = 5$	5.0	**6.6**	5.7	6.5	**6.7**	**6.6**	**6.6**	**6.7**
$n = 10, k = 10$	5.0	**6.6**	5.5	6.2	**6.7**	**6.6**	**6.6**	**6.7**
$n = 10, k = 50$	5.1	6.5	5.2	5.5	**6.6**	6.5	6.5	**6.8**
$n = 100, k = 5$	49.9	**78.3**	74.7	**77.9**	**78.0**	75.4	74.8	**78.7**
$n = 100, k = 10$	49.9	**82.8**	76.7	81.4	82.4	77.4	77.1	**83.5**
$n = 100, k = 50$	49.8	**85.2**	64.5	67.7	84.3	78.3	78.0	**84.9**
$n = 500, k = 5$	249.8	**405.8**	**402.0**	**406.7**	**405.8**	388.2	380.6	**401.6**
$n = 500, k = 10$	249.0	**437.8**	429.5	**438.9**	**437.1**	408.0	395.0	432.5
$n = 500, k = 50$	249.6	**463.7**	427.2	437.6	457.6	413.6	402.8	438.9

Models - RL^2 - Results

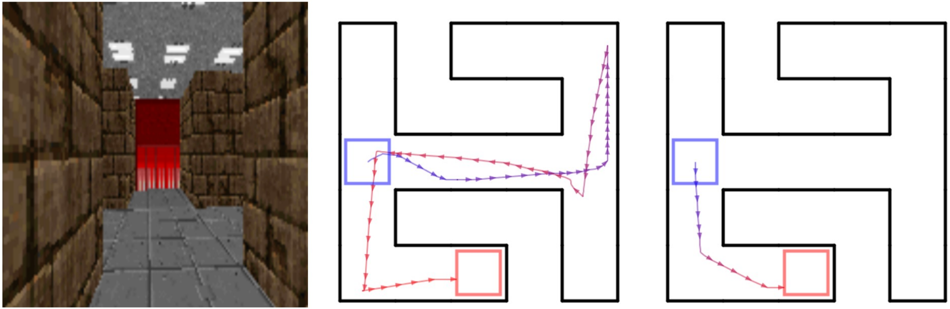
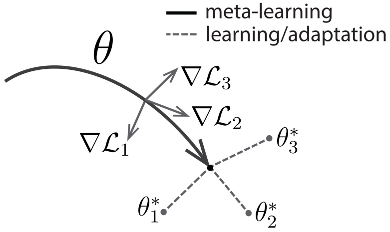
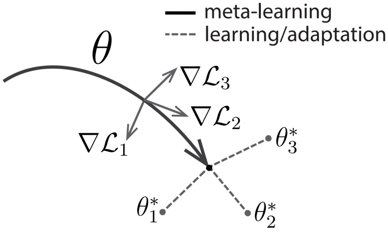


Figure: left: sample input; middle: first episode; right: second episode

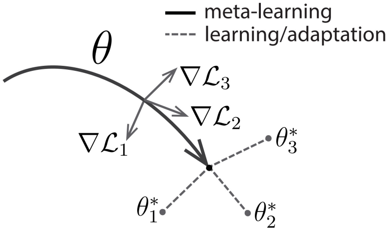
Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

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


meta-learning
learning/adaptation
θ
$\nabla\mathcal{L}_3$
$\nabla\mathcal{L}_2$
$\nabla\mathcal{L}_1$
θ_3^*
θ_1^*
θ_2^*



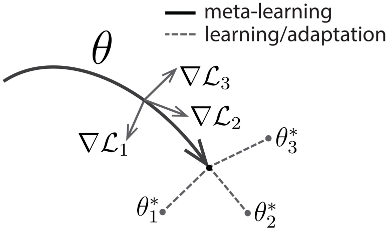
In MAML A_θ is one (or a fixed number of) gradient descent steps.



In MAML A_θ is one (or a fixed number of) gradient descent steps.

$$A_\theta(T^{\text{tr}}) = \theta - \alpha\nabla_\theta L_T(M_\theta)$$

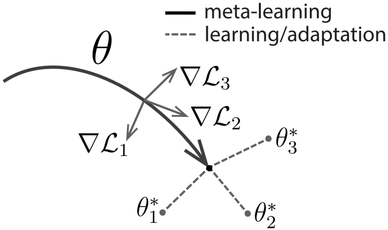
Models - Model Agnostic Meta Learning - Model Definition



In MAML A_θ is one (or a fixed number of) gradient descent steps.

$$A_\theta(T^{\mathrm{tr}}) = \theta - \alpha \nabla_\theta L_T(M_\theta)$$

The meta parameters θ are the initialization.



In MAML A_θ is one (or a fixed number of) gradient descent steps.

$$A_\theta(T^{\text{tr}}) = \theta - \alpha \nabla_\theta L_T(M_\theta)$$

The meta parameters θ are the initialization.
The meta learning algorithm f can be standard gradient descent with the following update rule

$$\theta \leftarrow \theta - \beta \sum_{T \in \mathcal{T}^{\text{meta-train}}} \nabla_\theta L_T(M_{\theta - \alpha \nabla_\theta L_T(M_\theta)})$$

Models - Model Agnostic Meta Learning - Model Class

MAML is an **optimization-based meta learning** model.

MAML is an **optimization-based meta learning** model.
The idea of such models is to start with an existing learning algorithm like SGD and learn parts of it.

Models - Model Agnostic Meta Learning - Model Class

MAML is an **optimization-based meta learning** model.
The idea of such models is to start with an existing learning algorithm like SGD and learn parts of it.

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} L_T(M_{\phi})$$

Possibile meta parameters are initialisation, learning rate, the entire update and more.

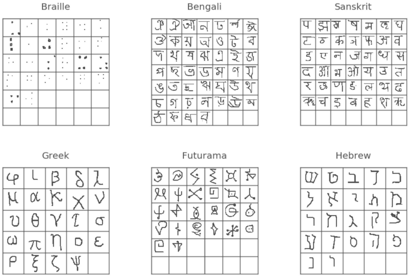
Definition (n-way k-shot classification)

We get k different samples for each of n different unseen classes and evaluate the model's ability to classify new instances within the n classes.

Definition (n-way k-shot classification)

We get k different samples for each of n different unseen classes and evaluate the model's ability to classify new instances within the n classes.

Omniglot data set: 1623 handwritten characters from 50 alphabets, 20 samples per character

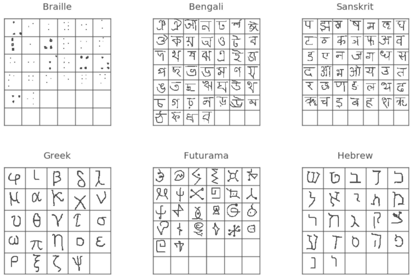


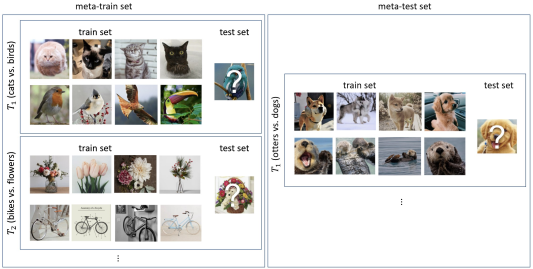
Definition (n-way k-shot classification)

We get k different samples for each of n different unseen classes and evaluate the model's ability to classify new instances within the n classes.

Omniglot data set: 1623 handwritten characters from 50 alphabets, 20 samples per character

MiniImagenet data set: 64 training classes, 12 validation classes, 24 test classes



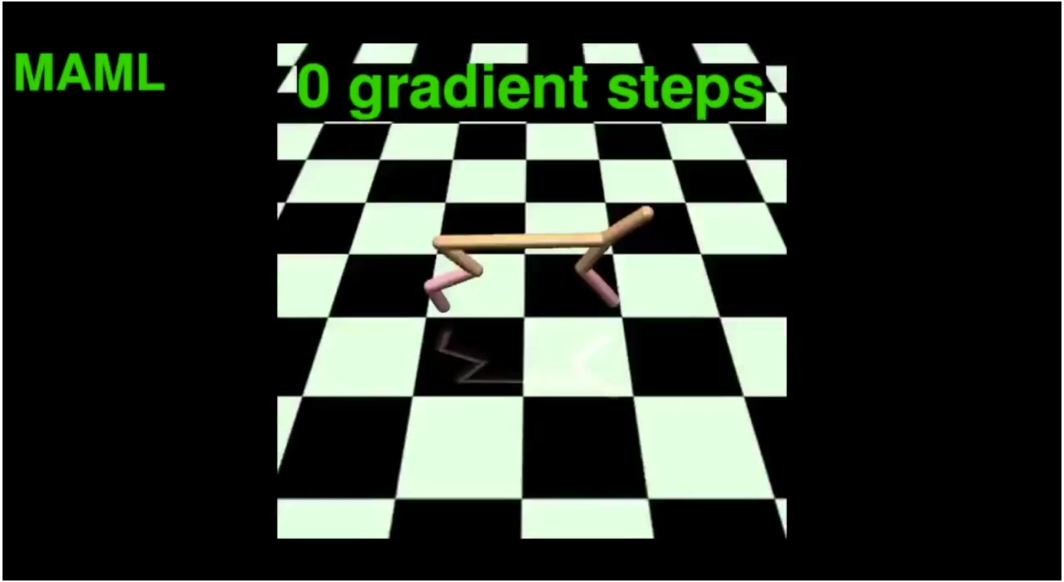


Models - Model Agnostic Meta Learning - Results

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%**

Models - Model Agnostic Meta Learning - Results



Summary

- The idea of Meta Learning is to optimize the parameterised learning algorithm for a class of tasks.
- RL^2 solves the problem by applying a RL algorithm to learn a RNN which represents the RL algorithm (applies RL to RL).
- MAML searches for a good initialisation of gradient based models.
- MAML does scale very well and is broadly applied.

References I

- Mathew Botvinick et al. "Reinforcement Learning, Fast and Slow". In: *Trends in Cognitive Sciences* 23 (Apr. 2019). DOI: 10.1016/j.tics.2019.02.006.
- Yan Duan et al. "RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning". In: (2016). arXiv: 1611.02779 [cs.AI].
- Chelsea Finn. *Learning to Learn*. 2017. URL: https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: (2017). arXiv: 1703.03400 [cs.LG].
- Chelsea Finn and Sergey Levine. *Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning*. 2019. URL: https://sites.google.com/view/icml19metalearning.
- D. Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (2017), pp. 354–359.

Lilian Weng. *Meta Reinforcement Learning*. 2019. URL: https://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html.

Lilian Weng. *Meta-Learning: Learning to Learn Fast*. 2018. URL: http://lilianweng.github.io/lil-log/2018/11/29/meta-learning.html.

Q&A

Some interesting questions:

- What is the meta learning algorithm and meta parameters of animals/nature?
- Have we formulated the problem we might want to solve with meta learning?

Why there are no higher order terms in multi-step MAML:

$$
\begin{aligned}
\nabla_\theta A_\theta(T^{\text{tr}}) &= \nabla_\theta(\theta' - \alpha \nabla_{\theta'} L_T(M_{\theta'})) \\
&= \nabla_\theta(\theta - \alpha \nabla_\theta L_T(M_\theta) - \alpha \nabla_{\theta'} L_T(M_{\theta'})) \\
&= I - \alpha \nabla_\theta^2 L_T(M_\theta) - \alpha \nabla_{\theta'}^2 L_T(M_{\theta'}) \frac{\partial \theta'}{\partial \theta}
\end{aligned}
$$