Meta Learning
Seminar on Deep Neural Networks

Tobias Birchler
ETH Zürich

2021
Outline

Motivation

Problem Definition
  Supervised Learning
  Generic Learning

Models
  \( RL^2 \)
  Model Agnostic Meta Learning

Summary

References

Q&A
Motivation

Motivation

Problem Definition
  Supervised Learning
  Generic Learning

Models
  \( RL^2 \)
  Model Agnostic Meta Learning

Summary

References

Q&A
Motivation
Motivation
Motivation

Many RL applications have long tailed state distributions
Motivation

Many RL applications have long tailed state distributions
Motivation

Many RL applications have long tailed state distributions
Motivation

Definition (meta)

*referring to itself or to something of its own type* (Cambridge Dictionary)
Definition (meta)

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

Remark

*Meta Learning is also known as ”Learning to Learn”*
Motivation

training data
Braque
Cezanne

test datapoint

By Braque or Cezanne?
Problem Definition

Motivation

Problem Definition
  Supervised Learning
  Generic Learning

Models
  $RL^2$
  Model Agnostic Meta Learning

Summary

References

Q&A
Problem Definition - Supervised Learning

"Normal" supervised learning:
Problem Definition - Supervised Learning

"Normal" supervised learning:

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

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

We solve this using a learning algorithm \( A \) and the training data.
"Normal" supervised learning:

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

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

\[ \sum_{(x,y) \in D^{test}} L(M_{A(D^{train})}(x), y) \]
Problem Definition - Supervised Learning

"Normal" supervised learning:

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

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

\[ \sum_{(x,y) \in D^{test}} L(M_{A(D^{train})}(x), y) \]
"Normal" supervised learning:

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

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

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

"Meta" supervised learning:
"Normal" supervised learning:

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

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

\[
\sum_{(x,y) \in D^{test}} L(M_{A(D_{train})}(x), y) = L_T(M_{A(D_{train})}, D^{test})
\]

"Meta" supervised learning:

\[
\theta^* = \arg \min_{\theta} \mathbb{E}_{T \sim P(T)}[L_T(M_{A_\theta(D_{T_{train}}^{T})}, D_{T_{test}}^{T})]
\]
"Normal" supervised learning:

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

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

\[
\sum_{(x,y) \in D^{test}} L(M_A(D_{train})(x), y) = L_T(M_A(D_{train}), D^{test})
\]

"Meta" supervised learning:

\[
\theta^* = \arg\min_{\theta} \mathbb{E}_{T \sim P(T)}[L_T(M_{A_\theta}(D_{train}^T), D^{test}_T)] \approx \arg\min_{\theta} \sum_{T \in T^{meta-test}} L_T(M_{A_\theta}(D_{train}^T), D^{test}_T)
\]
Problem Definition - Supervised Learning

"Normal" supervised learning:

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

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

\[
\sum_{(x,y) \in D^{test}} L(M_{A(D_{train})}(x), y) = L_T(M_{A(D_{train})}, D^{test})
\]

"Meta" supervised learning:

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

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

\[
\theta^* \approx f(T^{meta-train})
\]
Problem Definition - Supervised Learning

"Meta" supervised learning:

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

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

\[
\theta^* \approx f(\mathcal{T}^{meta-train})
\]
Problem Definition - Generic Learning

"Normal" supervised learning: $$\phi^* \approx \arg\min_{\phi} \sum_{(x, y) \in D_{test}} L(M(\phi)(x), y)$$

"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.

"Meta" supervised learning: $$\theta^* \approx \arg\min_{\theta} \sum_{T \in T_{meta-test}} L_T(M_{A}(\theta)(D_{train_{T}}), D_{test_{T}})$$

"Meta" generic learning: $$\theta^* \approx \arg\min_{\theta} \sum_{T \in T_{meta-test}} L_T(M_{A}(\theta)(T_{tr}), D_{test_{T}})$$

We solve this using a meta learning algorithm $f$ and the meta-training data $\theta^* \approx f(T_{meta-train})$. 
Problems 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:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in T_{\text{meta-test}}} L(M_A \theta(D_{\text{train}} T), D_{\text{test}} T) \]

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

The resulting loss is

\[ \sum_{(x, y) \in D_{\text{test}}} L(M_A(D_{\text{train}} T)(x), y) \]

"Meta" supervised learning: "Meta" generic learning:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in T_{\text{meta-test}}} L(M_A \theta(T_{\text{tr}}), T_{\text{test}}) \]

We solve this using a meta learning algorithm \( f \) and the meta-training data \[ \theta^* \approx f(T_{\text{meta-train}}) \]
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

\[ \sum_{(x,y) \in D_{\text{test}}} L(M_A(D_{\text{train}})(x), y) \]

"Meta" supervised learning: "Meta" generic learning:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in \mathcal{T}_{\text{meta-test}}} L_T(M_A(\theta)(D_{\text{train}}(T)), D_{\text{test}}(T)) \]

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in \mathcal{T}_{\text{meta-test}}} L_T(M_A(\theta(T_{\text{tr}}))) \]

We solve this using a meta learning algorithm \( f \) and the meta-training data \( \theta^* \approx f(T_{\text{meta-train}}) \)
Problem Definition - Generic Learning

"Normal" supervised learning:

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

"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.
"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:

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

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

\[ \sum_{(x,y) \in D_{\text{test}}} L(M_{A(D_{\text{train}})}(x), y) \]

"Normal" generic learning:

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

\[ L_T(M_{A(T_{\text{tr}})}) \]
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:

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

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

\[ \sum_{(x,y) \in D_{\text{test}}} L(M_A(D_{\text{train}})(x), y) \]

"Meta" supervised learning:

\[ \sum_{(x,y) \in D_{\text{test}}} L(M_A(D_{\text{train}})(x), y) \]

"Meta" generic learning:

\[ \sum_{(x,y) \in D_{\text{test}}} L(M_A(D_{\text{train}})(x), y) \]
Problem Definition - Generic Learning

"Normal" supervised learning:

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

"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

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

"Meta" supervised learning:

\[ \theta^* \approx \arg\min_{\theta} \sum_{T \in T^{meta-test}} L_T(M_{A_\theta(D^{train}_T)}, D^{test}_T) \]

"Meta" generic learning:

\[ \theta^* \approx \arg\min_{\theta} L_T(M_{A(T^{tr})}) \]
"Normal" supervised learning:

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

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

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

"Meta" supervised learning:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in T^{\text{meta-test}}} L_T(M_{A_\theta(D_{T^{\text{train}}}), D_{T^{\text{test}}}}) \]

"Normal" generic learning:

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

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

"Meta" generic learning:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in T^{\text{meta-test}}} L_T(M_{A_\theta(T^{\text{train}})}) \]
Problem Definition - Generic Learning

"Normal" supervised learning:

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

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

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

"Meta" supervised learning:

\[ \theta^* \approx \arg \min_{\theta} \sum_{T \in T^{\text{meta-test}}} L_T(M_{A_{\theta}(D^{\text{train}}_T)}, D^{\text{test}}_T) \]

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

\[ \theta^* \approx f(T^{\text{meta-train}}) \]
"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

\[
L_T(M_A(T^{tr}))
\]

"Meta" generic learning:

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

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

\[
\theta^* \approx f(\mathcal{T}^{meta-train})
\]
"Meta" generic learning:

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

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

\[ \theta^* \approx f(T^{\text{meta-train}}) \]
"Meta" generic learning:

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

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

\[
\theta^* \approx f(T^{\text{meta-train}})
\]
Models

Motivation

Problem Definition
  Supervised Learning
  Generic Learning

Models
  \( RL^2 \)
  Model Agnostic Meta Learning

Summary

References

Q&A
RL$^2$: Fast Reinforcement Learning via Slow Reinforcement Learning

Yan Duan$^{\dagger\dagger}$, John Schulman$^{\dagger\dagger}$, Xi Chen$^{\dagger\dagger}$, Peter L. Bartlett$^{\dagger}$, Ilya Sutskever$^{\dagger}$, Pieter Abbeel$^{\dagger\dagger}$
In RL² $A_\theta$ is a RNN (with GRU cells actually).
The meta parameters $\theta$ are the parameters of the RNN.
In $\text{RL}^2$ $A_\theta$ is a RNN (with GRU cells actually). The meta parameters $\theta$ are the parameters of the RNN.
In $RL^2$ $A_\theta$ is a RNN (with GRU cells actually).
The meta parameters $\theta$ are the parameters of the RNN.
Hidden state activations $h$ can be seen as internal state of the agent.
In \(\text{RL}^2\) \(A_\theta\) is a RNN (with GRU cells actually). The meta parameters \(\theta\) 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).
Models - $\text{RL}^2$ - Model Definition

In $\text{RL}^2$ $A_\theta$ is a RNN (with GRU cells actually). The meta parameters $\theta$ 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 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:*

![Diagram](image-url)
Models with entire neural networks as learning algorithm are known as **black-box meta learning** models.

**Example**

*Supervised learning:*

\[
(x_{1tr}, y_{1tr}) (x_{2tr}, y_{2tr}) \ldots (x_{ktr}, y_{ktr}) \rightarrow M \rightarrow \phi \rightarrow y_{test}
\]

The meta learning algorithm \( f \) for such models is usually just an off-the-shelf optimization algorithm (e.g. SGD: \( \theta \leftarrow \theta - \alpha \nabla_{\theta} L_T(M_{A_{\theta}(T_{tr})}) \)).
Models - $\text{RL}^2$ - 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 $p = 0.05$).

<table>
<thead>
<tr>
<th>Setup</th>
<th>Random</th>
<th>Gittins</th>
<th>TS</th>
<th>OTS</th>
<th>UCB1</th>
<th>$\epsilon$-Greedy</th>
<th>Greedy</th>
<th>$\text{RL}^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 10, k = 5$</td>
<td>5.0</td>
<td>6.6</td>
<td>5.7</td>
<td>6.5</td>
<td>6.7</td>
<td>6.6</td>
<td>6.6</td>
<td>6.7</td>
</tr>
<tr>
<td>$n = 10, k = 10$</td>
<td>5.0</td>
<td>6.6</td>
<td>5.5</td>
<td>6.2</td>
<td>6.7</td>
<td>6.6</td>
<td>6.6</td>
<td>6.7</td>
</tr>
<tr>
<td>$n = 10, k = 50$</td>
<td>5.1</td>
<td>6.5</td>
<td>5.2</td>
<td>5.5</td>
<td>6.6</td>
<td>6.6</td>
<td>6.5</td>
<td>6.8</td>
</tr>
<tr>
<td>$n = 100, k = 5$</td>
<td>49.9</td>
<td>78.3</td>
<td>74.7</td>
<td>77.9</td>
<td>78.0</td>
<td>75.4</td>
<td>74.8</td>
<td>78.7</td>
</tr>
<tr>
<td>$n = 100, k = 10$</td>
<td>49.9</td>
<td>82.8</td>
<td>76.7</td>
<td>81.4</td>
<td>82.4</td>
<td>77.4</td>
<td>77.1</td>
<td>83.5</td>
</tr>
<tr>
<td>$n = 100, k = 50$</td>
<td>49.8</td>
<td>85.2</td>
<td>64.5</td>
<td>67.7</td>
<td>84.3</td>
<td>78.3</td>
<td>78.0</td>
<td>84.9</td>
</tr>
<tr>
<td>$n = 500, k = 5$</td>
<td>249.8</td>
<td>405.8</td>
<td>402.0</td>
<td>406.7</td>
<td>405.8</td>
<td>388.2</td>
<td>380.6</td>
<td>401.6</td>
</tr>
<tr>
<td>$n = 500, k = 10$</td>
<td>249.0</td>
<td>437.8</td>
<td>429.5</td>
<td>438.9</td>
<td>437.1</td>
<td>408.0</td>
<td>395.0</td>
<td>432.5</td>
</tr>
<tr>
<td>$n = 500, k = 50$</td>
<td>249.6</td>
<td>463.7</td>
<td>427.2</td>
<td>437.6</td>
<td>457.6</td>
<td>413.6</td>
<td>402.8</td>
<td>438.9</td>
</tr>
</tbody>
</table>
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\textsuperscript{1} Pieter Abbeel\textsuperscript{1,2} Sergey Levine\textsuperscript{1}
In MAML

\( A^{\theta} \) is one (or a fixed number of) gradient descent steps.

\[ A^{\theta}(T) = \theta - \alpha \nabla_{\theta} L^T(M^{\theta}) \]

The meta parameters \( \theta \) are the initialization.

The meta learning algorithm can be standard gradient descent with the following update rule

\[ \theta \leftarrow \theta - \beta \sum_{T \in T_{\text{meta-train}}} \nabla_{\theta} L^T(M^{\theta} - \alpha \nabla_{\theta} L^T(M^{\theta})) \]
In MAML $A_\theta$ is one (or a fixed number of) gradient descent steps.
In MAML $A_\theta$ is one (or a fixed number of) gradient descent steps.

$$A_\theta(T^{tr}) = \theta - \alpha \nabla_\theta L_T(M_\theta)$$
In MAML $A_\theta$ is one (or a fixed number of) gradient descent steps.

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

The meta parameters $\theta$ are the initialization.
In MAML $A_\theta$ 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 $\theta$ are the initialization.

The meta learning algorithm $f$ can be standard gradient descent with the following update rule

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

\[
\phi \leftarrow \phi - \alpha \nabla_\phi L_T(M_\phi)
\]

Possible 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

<table>
<thead>
<tr>
<th>Models</th>
<th>5-way Accuracy</th>
<th>20-way Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>Omniglot (Lake et al., 2011)</td>
<td>82.8%</td>
<td>94.9%</td>
</tr>
<tr>
<td>MANN, no conv (Santoro et al., 2016)</td>
<td>89.7% ± 1.1%</td>
<td>97.5% ± 0.6%</td>
</tr>
<tr>
<td>MAML, no conv (ours)</td>
<td>90.7% ± 0.4%</td>
<td>99.9% ± 0.1%</td>
</tr>
<tr>
<td>Siamese nets (Koch, 2015)</td>
<td>97.3%</td>
<td>98.4%</td>
</tr>
<tr>
<td>matching nets (Vinyals et al., 2016)</td>
<td>98.1%</td>
<td>98.9%</td>
</tr>
<tr>
<td>neural statistician (Edwards &amp; Storkey, 2017)</td>
<td>98.1%</td>
<td>99.5%</td>
</tr>
<tr>
<td>memory mod. (Kaiser et al., 2017)</td>
<td>98.4%</td>
<td>99.6%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>5-way Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td>MiniImagenet (Ravi &amp; Larochelle, 2017)</td>
<td></td>
</tr>
<tr>
<td>fine-tuning baseline</td>
<td>28.86 ± 0.54%</td>
</tr>
<tr>
<td>nearest neighbor baseline</td>
<td>41.08 ± 0.70%</td>
</tr>
<tr>
<td>matching nets (Vinyals et al., 2016)</td>
<td>43.56 ± 0.84%</td>
</tr>
<tr>
<td>meta-learner LSTM (Ravi &amp; Larochelle, 2017)</td>
<td>43.44 ± 0.77%</td>
</tr>
<tr>
<td>MAML, first order approx. (ours)</td>
<td>48.07 ± 1.75%</td>
</tr>
<tr>
<td>MAML (ours)</td>
<td>48.70 ± 1.84%</td>
</tr>
</tbody>
</table>
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

Motivation

Problem Definition
  Supervised Learning
  Generic Learning

Models
  RL²
  Model Agnostic Meta Learning

Summary

References

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:

\[
\nabla_\theta A_\theta(T^{tr}) = \nabla_\theta (\theta' - \alpha \nabla_{\theta'} L_T(M_{\theta'})) \\
= \nabla_\theta (\theta - \alpha \nabla_\theta L_T(M_\theta) - \alpha \nabla_{\theta'} L_T(M_{\theta'})) \\
= I - \alpha \nabla^2_\theta L_T(M_\theta) - \alpha \nabla^2_{\theta'} L_T(M_{\theta'}) \frac{\partial \theta'}{\partial \theta}
\]