Meta Learning, part 2

Seminar in Deep Neural Networks - 2021

Davide Plozza
Meta learning = learning to learn
Training Loop

Optimizer (ADAM, SGD)
\[ \theta_{t+1} = \theta_t + \alpha_t \nabla L(\theta_t) \]

Model (Optimizee)

Parameters \( \theta_t \)

Loss, Objective function
\[ L(\theta_t, X, Y) \]

Gradient
\[ \Delta_\theta L(\theta_t) \]

Error signal

Training data

Input X

Target Y

Prediction
Model (Optimizee)

Training data

Optimizer (ADAM, SGD)

\[ \theta_{t+1} = \theta_t + \alpha_t \nabla L(\theta_t) \]

Parameters \( \theta_t \)

Input X

Target Y

Prediction

Loss, Objective function

\[ L(\theta_t, X, Y) \]

Gradient

\[ \Delta_\theta L(\theta_t) \]

Error signal

Optimizer (ADAM, SGD)

\[ \theta_{t+1}^1 \]

\[ \theta_{t+1}^2 \]

\[ \theta_{t+1}^n \]
\[ \theta_{t+1} = \theta_t + \alpha_t \nabla L(\theta_t) \]

Optimizer (ADAM, SGD)

Training data

Input X

Target Y

Model (Optimizee)

Parameters \( \theta_t \)

Prediction

Loss, Objective function

\[ L(\theta_t, X, Y) \]

Gradient

\[ \Delta \theta L(\theta_t) \]

Optimizer

Error signal

Optimizer (ADAM, SGD)
Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz\textsuperscript{1}, Misha Denil\textsuperscript{1}, Sergio Gómez Colmenarejo\textsuperscript{1}, Matthew W. Hoffman\textsuperscript{1}, David Pfau\textsuperscript{1}, Tom Schaul\textsuperscript{1}, Brendan Shillingford\textsuperscript{1,2}, Nando de Freitas\textsuperscript{1,2,3}

\textsuperscript{1}Google DeepMind \hspace{1cm} \textsuperscript{2}University of Oxford \hspace{1cm} \textsuperscript{3}Canadian Institute for Advanced Research
Revisiting gradient descent

\[ \theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) \]
Can we learn gradient descent?

Model (Optimizee) with parameters $\theta_t$.

Training data $X$ and target $Y$.

Objective function $f(\theta_t)$.

Gradient $\Delta_f f(\theta_t)$.

Parameters updated iteratively: $\theta^{t+1}_1, \theta^{t+1}_2, \ldots, \theta^{t+1}_n$.
Two network architecture

Optimizer Network

Optimizee network

Parameters $\theta_t$

Input $X$

Target $Y$

Training data

Loss, Objective function $f(\theta_t)$

Gradient $\Delta_{\theta} f(\theta_t)$

Prediction

Error signal

$\theta^1_{t+1}$

$\theta^2_{t+1}$

$\vdots$

$\theta^n_{t+1}$
Recurrent NN (LSTM) as optimizer

Optimizer Network

LSTM

Optimizee network

Parameters $\theta_t$

Training data

Input $X$

Target $Y$

Error signal

Prediction

Loss, Objective function $f(\theta_t)$

Gradient $\Delta_{\theta} f(\theta_t)$
Recurrent NN (LSTM) as optimizer

\[ \theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) \]

\[ \theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi) \]
Recurrent NN (LSTM) as optimizer

\[ \theta_{t+1} = \theta_t + g_t , \]

\[
\begin{bmatrix}
  g_t \\
  h_{t+1}
\end{bmatrix}
= m(\nabla_t, h_t, \phi)
\]

\[ \nabla_t = \nabla_{\theta} f(\theta_t) \]
Recurrent NN (LSTM) as optimizer

\[ \theta_{t+1} = \theta_t + g_t , \]

\[
\begin{bmatrix}
g_t \\
h_{t+1}
\end{bmatrix} = m(\nabla_t, h_t, \phi)
\]

\[ \nabla_t = \nabla_\theta f(\theta_t) \]
Outer optimization loss

\[ \mathcal{L}(\phi) = \mathbb{E}_f \left[ \sum_{t=1}^{T} w_t f(\theta_t) \right] \]

where \( \theta_{t+1} = \theta_t + g_t \)
Outer optimization computational graph
Coordinatewise LSTM optimizer
Results

Quadratics

Loss

$10^1$

$10^0$

$10^{-1}$

Step

20 40 60 80 100

MNIST

Step

20 40 60 80 100

ADAM

RMSprop

SGD

NAG

LSTM
Results

[Graphs showing loss over steps for CIFAR-10, CIFAR-5, and CIFAR-2 datasets with different optimization algorithms: ADAM, RMSprop, SGD, NAG, LSTM, LSTM-sub]
Conclusion

• Better performance than state-of-the-art optimizers
• High degree of transfer between different tasks and different architectures

• LSTM is costlier to run than SGD
• Need meta training
Meta-Gradient Reinforcement Learning with an Objective Discovered Online

Zhongwen Xu, Hado van Hasselt, Matteo Hessel
Junhyuk Oh, Satinder Singh, David Silver
DeepMind
Model (Optimizee)\n
Parameters $\theta_t$\n
Training Loop

Optimizer (ADAM, SGD)\n
Training data\n
Input X\n
Target Y

Loss, Objective function $L(\theta_t)$\n
$L(\theta_t, X, Y)$

Gradient $\Delta_\theta L(\theta_t)$

Error signal

Prediction

Optimizer (ADAM, SGD): Parameters $\theta_{t+1}$ are updated from $\theta_t$. Training data is used to predict $Y$ and calculate the error signal for optimization.

Model (Optimizee): The model with parameters $\theta_t$ is used to make predictions.

Loss, Objective function: The loss function $L(\theta_t)$ is calculated from the prediction and target $Y$.

Gradient: The gradient $\Delta_\theta L(\theta_t)$ is used to update the parameters $\theta_{t+1}$ using the optimizer (ADAM, SGD).

Error signal: The error between the predicted and target $Y$ is used to update the model parameters.

Training Loop Diagram:
- Input data $X$ is fed into the model.
- The model makes predictions.
- The error between predictions and target $Y$ is calculated.
- The optimizer updates the model parameters $\theta_{t+1}$ using the calculated gradient.
TD learning

Agent

Value Function

Parameters $\theta_t$

Optimizer (ADAM, SGD)

Loss, Objective function $L(\theta_t)$

Gradient $\Delta_\theta L(\theta_t)$

Enviroment

Action

Observation

Reward

Error signal

$\theta_{t+1}^1$

$\theta_{t+1}^2$

$\vdots$

$\theta_{t+1}^n$
Previous work

• $\text{RL}^2$: solves the problem by applying a RL algorithm to learn a RNN which represents the RL algorithm

• MAML: searches for a good initialisation of gradient based models.
Proposed solution

• Black box method
• Parameterizes target (loss) of RL algorithm
• Meta-learned online
• No «meta training»
Update Targets in RL algorithms

\[ \tau_t = \{S_t, A_t, R_{t+1}, \ldots \} \]

\[ G_t = R_{t+1} + \gamma v_\theta(S_{t+1}) \]

\[ \theta \leftarrow \theta + \alpha (G_t - v_\theta(S_t)) \nabla_\theta v_\theta(S_t) \]
Update Targets in RL algorithms

\[ \tau_t = \{ S_t, A_t, R_{t+1}, \ldots \} \]

\[ G_t = R_{t+1} + \gamma \max_a q_\theta(S_{t+1}, a) \]

\[ \theta \leftarrow \theta + \alpha (G_t - q_\theta(S_t, A_t)) \nabla_\theta q_\theta(S_t, A_t) \]
Idea: parameterize update target

\[ G_t = g_\eta(\tau_t) \]

\[ g_\eta : \tau_t \rightarrow \mathbb{R} \]
Meta gradients

\[ \Delta \theta_i \propto \nabla_{\theta_i} L_{\eta}^{\text{inner}}(\tau_i, \theta_i) \quad \theta_{i+1} = \theta_i + \Delta \theta_i \]

\[ \theta_i \xrightarrow{\eta} \theta_{i+1} \xrightarrow{\eta} \ldots \xrightarrow{\eta} \theta_{i+M-1} \xrightarrow{\eta} \theta_{i+M} \]

\[ \Delta \eta \propto \nabla_{\eta} L_{\eta}^{\text{outer}}(\tau_{i+M+1}, \theta_{i+M}) \quad \eta \leftarrow \eta + \Delta \eta \]
Value based control

\[ \Delta \theta \propto (g_\eta(\tau) - v_\theta(S)) \nabla_\theta v_\theta(S). \]

\[ \nabla_\theta', L^{\text{outer}} = (G(\tau') - v_{\theta'}(S')) \nabla_{\theta'} v_{\theta'}(S') \]
Consistent update targets heuristic

\[ G^m_t = R_{t+1} + \gamma G^m_{t+1} \]

\[ L^{\text{outer}} \leftarrow L^{\text{outer}} + c || \perp (R_{t+1} + \gamma G^m_{t+1}) - G^m_t ||^2 \]
Results
Results
Conclusion

• FRODO can outperform a strong actor critic baseline
• Solves boostrapping
• Solves non-stationarity
• Learns online, during single agent lifetime
• Can adapt to changes

• Needs heuristic
• Still needs hand tuning
Summary

• First paper proposes learned optimizer in form of recurrent NN, for supervised learning tasks

• FRODO proposes to learn the update target of RL algorithms
Alchemy: A structured task distribution for meta-reinforcement learning

Jane X. Wang* †1, Michael King* †1, Nicolas Porcell1, Zeb Kurth-Nelson1,2, Tina Zhu1, Charlie Deck1, Peter Choy1, Mary Cassin1, Malcolm Reynolds1, Francis Song1, Gavin Buttimore1, David P. Reichert1, Neil Rabinowitz1, Loic Matthey1, Demis Hassabis1, Alexander Lerchner1, Matthew Botvinick†,2

1DeepMind, London, UK
2University College London, London, UK

February 8, 2021
References

• Marcin Andrychowicz et al. (2016). Learning to learn by gradient descent by gradient descent. [cs.AI]
• Zhongwen Xu et al. (2020). Meta-Gradient Reinforcement Learning with an Objective Discovered Online. [cs.LG]
• Jane X. Wang et al. (2021). Alchemy: A structured task distribution for meta-reinforcement learning. [cs.LG]

Additional references
• https://sites.google.com/view/icml19metalearning
• https://medium.com/dataseries/learning-to-learn-gradient-descent-by-gradient-descent-a-paper-review-44292f2fb1ff
• https://becominghuman.ai/paper-repro-learning-to-learn-by-gradient-descent-descent-6e504cc1c0de