Parameter Prediction Paper: Learning to Learn by gradient descent by gradient descent

Marcin Andrychowicz, Misha Denil, Sergio Gmez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, Nando de Freitas

NIPS 2016

Reviewed by : Arshdeep Sekhon

[1]Department of Computer Science, University of Virginia
https://qdata.github.io/deep2Read/

August 25, 2018

Optimizer

A standard machine learning algorithm

$$\theta^* = arg \min_{\theta} f(\theta) \tag{1}$$

Optimizer

A standard machine learning algorithm

$$\theta^* = arg \min_{\theta} f(\theta) \tag{1}$$

A standard optimizer

$$\theta t + 1 = \theta_t - \alpha_t \nabla f(\theta_t) \tag{2}$$

Motivation

Optimization strategies tailored to different classes of tasks:

- Deep Learning: High Dimensional, non convex optimization problems Adagrad,RMSprop, Rprop, etc.
- Combinatorial Optimization: Relaxations

Motivation

Optimization strategies tailored to different classes of tasks:

- Deep Learning: High Dimensional, non convex optimization problems Adagrad,RMSprop, Rprop, etc.
- Combinatorial Optimization: Relaxations

Generally, hand designed update rules.

Learning to Learn

Meta-learning / Learning to Learn

Replace hand designed update rules with learned update rule

$$\theta_{t+1} = \theta_t + g_t(\nabla(f(\theta_t)), \phi) \tag{3}$$

g_t is the optimizer with its own parameters

Learning to Learn

Meta-learning / Learning to Learn

Replace hand designed update rules with learned update rule

$$\theta_{t+1} = \theta_t + g_t(\nabla(f(\theta_t)), \phi) \tag{3}$$

g_t is the optimizer with its own parameters

g_t is a recurrent neural network that predicts update at each timestep, parameterised by ϕ

Generalization for optimizers

Goal

Find an optimizer with learned updates (instead of hand designed updates) that performs well on a class of optimization problems.

Generalization for optimizers

Goal

Find an optimizer with learned updates (instead of hand designed updates) that performs well on a class of optimization problems.

Generalization in machine learning: Capacity to make predictions about the target at novel unseen points

Generalization for optimizers

Goal

Find an optimizer with learned updates (instead of hand designed updates) that performs well on a class of optimization problems.

Generalization in machine learning: Capacity to make predictions about the target at novel unseen points

Generalization in 'Learning to Learn' context: Optimizer should perform well in unseen problems of the same 'type':

Generalization for optimizers

Goal

Find an optimizer with learned updates (instead of hand designed updates) that performs well on a class of optimization problems.

Generalization in machine learning: Capacity to make predictions about the target at novel unseen points

Generalization in 'Learning to Learn' context: Optimizer should perform well in unseen problems of the same 'type':

transfer learning

Learning to Learn with Recurrent Neural Networks

1. When can you say an optimizer is good?
2. θ^* (optimal parameters) is a function of the class of functions f being optimized and the optimizer parameters

$$\theta^*(f, \phi) \tag{4}$$

Learning to Learn with Recurrent Neural Networks

1. When can you say an optimizer is good?
2. θ^* (optimal parameters) is a function of the class of functions f being optimized and the optimizer parameters

$$\theta^*(f, \phi) \tag{4}$$

3. Expected Loss

$$\mathbb{L}(\phi) = \mathbb{E}_f\Big[f(\theta^*(f, \phi))\Big] \tag{5}$$

Learning to Learn with Recurrent Neural Networks

1. When can you say an optimizer is good?
2. θ^* (optimal parameters) is a function of the class of functions f being optimized and the optimizer parameters

$$\theta^*(f, \phi) \tag{4}$$

3. Expected Loss

$$\mathbb{L}(\phi) = \mathbb{E}_f \Big[f(\theta^*(f, \phi)) \Big] \tag{5}$$

4. Expected Loss with RNN optimizer

$$\mathbb{L}(\phi) = \mathbb{E}_f \Big[\sum_{t=1}^{T} w_t f(\theta_t) \Big] \tag{6}$$

$$\theta_{t+1} = \theta_t + g_t \tag{7}$$

$$\begin{pmatrix} g_t \\ h_{t+1} \end{pmatrix} = m(\nabla_t, h_t, \phi)$$

Learning to Learn with Recurrent Neural Networks

1. When can you say an optimizer is good?
2. θ^* (optimal parameters) is a function of the class of functions f being optimized and the optimizer parameters

$$\theta^*(f, \phi) \tag{4}$$

3. Expected Loss

$$\mathbb{L}(\phi) = \mathbb{E}_f \Big[f(\theta^*(f, \phi)) \Big] \tag{5}$$

4. Expected Loss with RNN optimizer

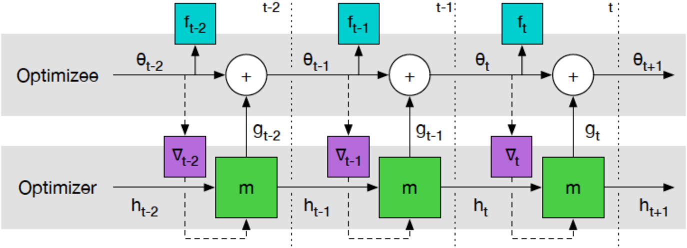
$$\mathbb{L}(\phi) = \mathbb{E}_f \Big[\sum_{t=1}^{T} w_t f(\theta_t) \Big] \tag{6}$$

$$\theta_{t+1} = \theta_t + g_t \tag{7}$$

$$\begin{pmatrix} g_t \\ h_{t+1} \end{pmatrix} = m(\nabla_t, h_t, \phi)$$

5. Generate updates at each timestep using a recurrent neural network

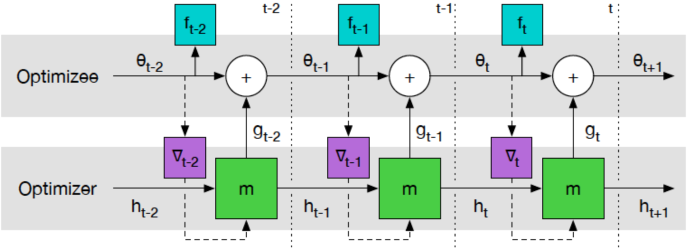
Optimizing the optimizer



Computational Graph used for computing the gradient

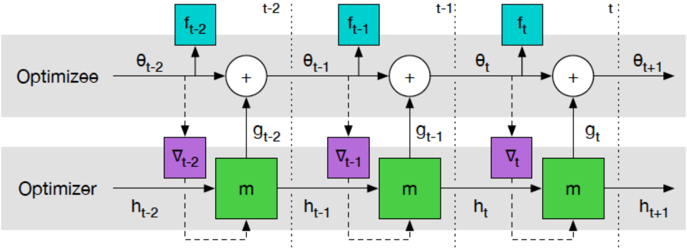
1. Minimize loss $\mathbb{L}(\phi)$w.r.t the parameters of the optimizer (ϕ)
2. calculate $\frac{\delta \mathbb{L}}{\delta \phi}$ and backpropagate through time

Optimizing the optimizer

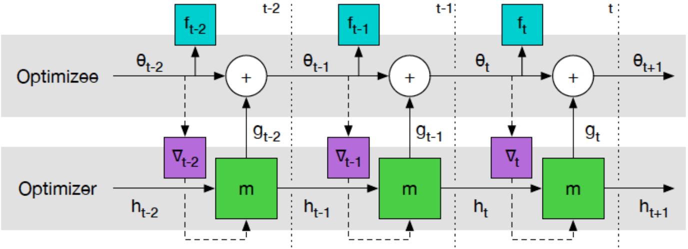


1. backpropagation through the computational graph

Optimizing the optimizer



1. backpropagation through the computational graph
2. Assumption: Optimizee gradients do not depend on optimizer parameters
3. $\nabla_t = \nabla_\theta f(\theta_t)$; $\dfrac{\nabla_t}{\delta \phi} = 0$: Drop gradients along dotted edges



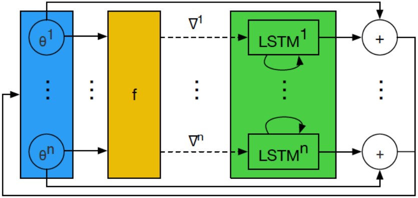
1. backpropagation through the computational graph
2. Assumption: Optimizee gradients do not depend on optimizer parameters
3. $\nabla_t = \nabla_\theta f(\theta_t)$; $\frac{\nabla_t}{\delta\phi} = 0$: Drop gradients along dotted edges

1. Issue: Huge hidden state RNN if we want an update for each parameter

$$\theta_{t+1} = \theta_t + g_t \tag{8}$$

$$\begin{pmatrix} g_t \\ h_{t+1} \end{pmatrix} = m(\nabla_t, h_t, \phi)$$

2. Solution: use an RNN to get an update coordinate wise
3. Coordinate wise LSTM shares the weights across all parameters

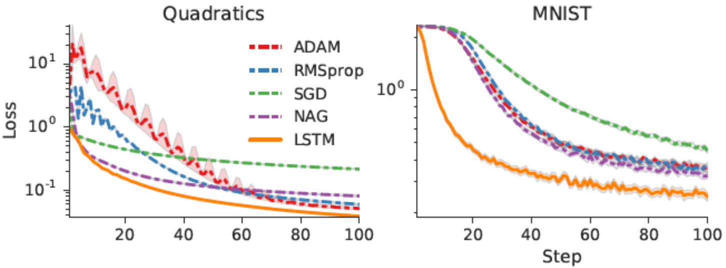


Separate hidden states, but shared parameters

Experiments and Results: Quadratic Functions

1. Quadratic Functions:

$$f(\theta) = ||W\theta - y||_2^2 \tag{9}$$

2. Optimzer trained on random functions from this family
3. Test on a random function sampled from this family distribution
4. 

Learning Curves