Meta Learning (Part 2):  
Gradient Descent as LSTM  
Hung-yi Lee
Can we learn more than initialization parameters?

The learning algorithm looks like RNN.

Optimization as a Model for Few-Shot Learning

Sachin Ravi* and Hugo Larochelle
Twitter, Cambridge, USA
{sachinr,hugo}@twitter.com

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz¹, Misha Denil¹, Sergio Gómez Colmenarejo¹, Matthew W. Hoffman¹,
David Pfau¹, Tom Schaul¹, Brendan Shillingford¹,², Nando de Freitas¹,²,³

¹Google DeepMind  ²University of Oxford  ³Canadian Institute for Advanced Research
Recurrent Neural Network

• Given function $f$: $h', y = f(h, x)$

No matter how long the input/output sequence is, we only need one function $f$
c change slowly \[\rightarrow\] \(c^t\) is \(c^{t-1}\) added by something

h change faster \[\rightarrow\] \(h^t\) and \(h^{t-1}\) can be very different
Review: LSTM

\[
\begin{align*}
    z & = \tanh(\begin{pmatrix} \mathbf{W} \\ \mathbf{h}^{t-1} \end{pmatrix}) \\
    z^i & = \sigma(\begin{pmatrix} \mathbf{W}^i \\ \mathbf{h}^{t-1} \end{pmatrix}) \\
    z^f & = \sigma(\begin{pmatrix} \mathbf{W}^f \\ \mathbf{h}^{t-1} \end{pmatrix}) \\
    z^o & = \sigma(\begin{pmatrix} \mathbf{W}^o \\ \mathbf{h}^{t-1} \end{pmatrix})
\end{align*}
\]
Review: LSTM

\[ c^t = z^f \odot c^{t-1} + z^i \odot z \]

\[ h^t = z^o \odot \tanh(c^t) \]

\[ y^t = \sigma(W'h^t) \]
Review: LSTM
Similar to gradient descent based algorithm

\[
\theta^t = \theta^{t-1} - \eta \nabla_\theta l
\]

\[
\theta^t = [1]_{z^f, \theta^{t-1}} + [\eta]_{z^i, -\nabla_\theta l}
\]

\[
h^t = z^o \odot \tanh(c^t)
\]

\[
y^t = \sigma(W'h^t)
\]
Similar to gradient descent based algorithm

\[ \theta^t = \theta^{t-1} - \eta \nabla_\theta l \]

Something like regularization

\[ \theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_\theta l \]

Dynamic learning rate

How about machine learn to determine \( z^f \) and \( z^i \) from \( -\nabla_\theta l \) and other information?
Typical LSTM

LSTM for Gradient Descent

\[ \theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l \]

Learn to minimize

3 training steps

Learnable

Independence assumption

Batch from train

Batch from train

Batch from train

Testing Data

\[ l(\theta^3) \]
Real Implementation

The LSTM used only has one cell. Share across all parameters

➢ Reasonable model size
➢ In typical gradient descent, all the parameters use the same update rule
➢ Training and testing model architectures can be different.
Experimental Results

\[ \theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l \]

(a) Forget gate values for 1-shot meta-learner

(b) Input gate values for 1-shot meta-learner
Parameter update depends on not only current gradient, but *previous gradients*.

RMSProp

\[
\begin{align*}
    w^1 &\leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \\
    w^2 &\leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \\
    w^3 &\leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \\
    &\vdots \\
    w^{t+1} &\leftarrow w^t - \frac{\eta}{\sigma^t} g^t
\end{align*}
\]

Momentum

Still not guarantee reaching global minima, but give some hope ......

Movement = Negative of $\partial L/\partial w$ + Momentum

- Red: Negative of $\partial L / \partial w$
- Green dots: Momentum
- Blue: Real Movement

$\partial L/\partial w = 0$
LSTM for Gradient Descent (v2)

3 training steps

m can store previous gradients

\[ \nabla \theta l \]

Batch from train

Batch from train

Batch from train

\[ l(\theta^3) \]
Experimental Results

https://arxiv.org/abs/1606.04474
Meta Learning (Part 3)
Hung-yi Lee
Even more crazy idea ...

- **Input:**
  - Training data and their labels
  - Testing data
- **Output:**
  - Predicted label of testing data
Face Verification

In each task:

- **Training**
  - Few-shot Learning
- **Registration (Collecting Training data)**
- **Testing**
  - Unlock your phone by Face

https://support.apple.com/zh-tw/HT208109
Meta Learning

Training Tasks

<table>
<thead>
<tr>
<th>Train</th>
<th>Test</th>
<th>Yes or No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Test</td>
<td>Yes</td>
</tr>
<tr>
<td>Train</td>
<td>Test</td>
<td>No</td>
</tr>
<tr>
<td>Train</td>
<td>Test</td>
<td>No</td>
</tr>
</tbody>
</table>

Testing Tasks

<table>
<thead>
<tr>
<th>Train</th>
<th>Test</th>
<th>Yes or No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Test</td>
<td>Yes</td>
</tr>
<tr>
<td>Train</td>
<td>Test</td>
<td>No</td>
</tr>
</tbody>
</table>

Same approach for Speaker Verification

Network

Yes

No

?
Siamese Network

Train

Same

Test

CNN

CNN

embedding

embedding

share

Similarity

score

Large score → Yes

Small score → No

Network

No
Siamese Network

Train

Different

Test

No

Network

Large score ➞ Yes

Small score ➞ No

score

CNN

embedding

share

Similarity

embedding

CNN

embedding
Siamese Network
- Intuitive Explanation

- Binary classification problem: “Are they the same?”

Training Set

Testing Set

Same or Different

Network

Face 1

Face 2

same
different
different

?
Siamese Network - Intuitive Explanation

Learning embedding for faces

e.g. learn to ignore the background

Train

Test

Far away

As close as possible

score

Large score ➞ Yes

Small score ➞ No
To learn more ...

• What kind of distance should we use?
  • SphereFace: Deep Hypersphere Embedding for Face Recognition
  • Additive Margin Softmax for Face Verification
  • ArcFace: Additive Angular Margin Loss for Deep Face Recognition

• Triplet loss
  • Deep Metric Learning using Triplet Network
  • FaceNet: A Unified Embedding for Face Recognition and Clustering
N-way Few/One-shot Learning

• Example: 5-ways 1-shot

Network (Learning + Prediction)

一花 二乃 三玖 四葉 五月

Training Data (Each image represents a class)

Testing Data

三玖
Prototypical Network

\[ s_1, s_2, s_3, s_4, s_5 \]

\[ \text{CNN} \]

Testing Data

https://arxiv.org/abs/1703.05175

= similarity
Matching Network

Considering the relationship among the training examples

\[ s_1, s_2, s_3, s_4, s_5 = \text{similarity} \]

https://arxiv.org/abs/1606.04080
Relation Network

https://arxiv.org/abs/1711.06025
Few-shot learning for imaginary data

https://arxiv.org/abs/1801.05401
Few-shot learning for imaginary data

https://arxiv.org/abs/1801.05401