Review Series of Recent Deep Learning Papers:

Parameter Prediction Paper: HyperNetworks

David Ha, Andrew Dai, Quoc V. Le

ICLR 2017

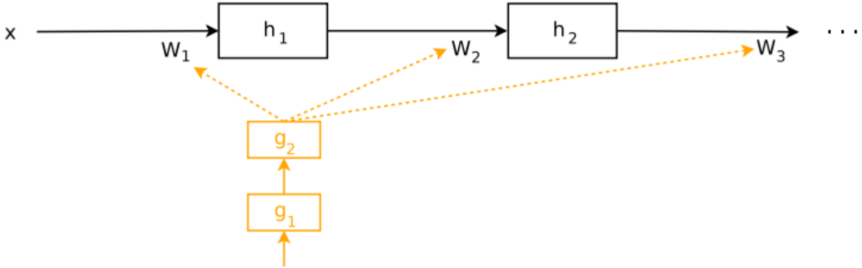
Reviewed by : Arshdeep Sekhon

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

August 26, 2018

What are Hypernetworks?

Use a smaller network to generate weights for a larger network



HyperNetwork

Why Hypernetworks?

1. If the Hypernetwork is smaller than the main network, less number of trainable parameters

Why Hypernetworks?

1. If the Hypernetwork is smaller than the main network, less number of trainable parameters
2. Relaxed form of Weight Sharing

Why Hypernetworks?

1. If the Hypernetwork is smaller than the main network, less number of trainable parameters
2. Relaxed form of Weight Sharing
   1. CNNs: No weight sharing at all between layers
   2. HyperNetworks can be used to encourage weight sharing in CNNs.

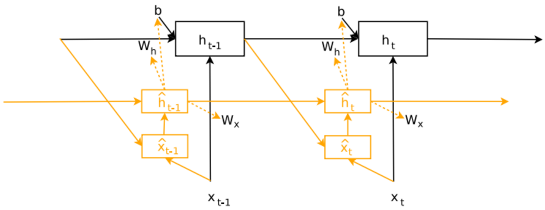
Why Hypernetworks?

1. If the Hypernetwork is smaller than the main network, less number of trainable parameters
2. Relaxed form of Weight Sharing
   1. CNNs: No weight sharing at all between layers
   2. HyperNetworks can be used to encourage weight sharing in CNNs.
   3. RNNs: Weights are shared between timesteps
   4. Standard RNN
   $$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \quad (1)$$
   5. weights W_h and W_x are shared between timesteps $X = (x_1, x_2, \cdots, x_T)$

Why Hypernetworks?

1. If the Hypernetwork is smaller than the main network, less number of trainable parameters
2. Relaxed form of Weight Sharing
   1. CNNs: No weight sharing at all between layers
   2. HyperNetworks can be used to encourage weight sharing in CNNs.
   3. RNNs: Weights are shared between timesteps
   4. Standard RNN
      $$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \tag{1}$$
   5. weights W_h and W_x are shared between timesteps $X = (x_1, x_2, \cdots, x_T)$
   6. Can also be used to make weights different at timesteps in RNNs.

Dynamic Hypernetworks: HyperRNN



HyperRNN

HyperRNN

- **MainRNN**: Standard RNN
- **HyperNetwork**: generates weights W_h and W_x for MainRNN that are different for different timesteps

Dynamic Hypernetworks: MainRNN

Standard RNN

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \tag{2}$$

MainRNN

$$\hat{h}_t = \phi(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)) \tag{3}$$

z_h,z_b and z_x are outputs of HyperNetwork

$$W_h(z_h) =< W_{hz}, z_h > \tag{4}$$

$$W_h(z_x) =< W_{xz}, z_x > \tag{5}$$

$$b(z_b) = W_{bz} z_b + b_0 \tag{6}$$

$W_{hz} \;\epsilon\; \mathbb{R}^{N_h \times N_h \times N_z}$ $\quad W_{xz} \;\epsilon\; \mathbb{R}^{N_h \times N_x \times N_z}$ $\quad W_{bz} \;\epsilon\; \mathbb{R}^{N_h \times N_z}$

$<,>$ denotes a tensor product: A $\epsilon\; \mathbb{R}^{m \times n \times p}$, $B \epsilon\; \mathbb{R}^{p}$, $<A, B> \;\epsilon\; \mathbb{R}^{m \times n}$

Dynamic Hypernetworks: HyperRNN

HyperNetwork

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{7}$$

$$\hat{h}_t = \phi(W_{\hat{h}} \hat{h_{t-1}} + W_{\hat{x}} \hat{x}_t + b) \tag{8}$$

$$z_h = LinearLayer1(\hat{h_{t-1}}) \tag{9}$$

$$z_x = LinearLayer2(\hat{h_{t-1}}) \tag{10}$$

$$z_b = LinearLayer3(\hat{h_{t-1}}) \tag{11}$$

$$W_{hz} \ \epsilon \ \mathbb{R}^{N_h \times N_h \times N_z} \quad W_{xz} \ \epsilon \ \mathbb{R}^{N_h \times N_x \times N_z} \quad W_{bz} \ \epsilon \ \mathbb{R}^{N_h \times N_z}$$

$$\hat{h}_{t-1} \ \epsilon \ \mathbb{R}^{N_{\hat{h}}}$$

Dynamic Hypernetworks: Modification to HyperRNN

MainRNN: More Memory Efficient

Scale each row of W_h linearly by an element in d where $d(z)$ is a linear function of z.

$$h_t = \phi(d_h(z_h) \odot W_h h_{t-1} + d_x(z_x) \odot W_x x_t + b(z_b)) \tag{12}$$

$$d_h(z_h) = W_{hz} z_h \tag{13}$$

$$d_h(z_x) = W_{xz} z_x \tag{14}$$

$$b(z_b) = W_{bz} z_b + b_0 \tag{15}$$

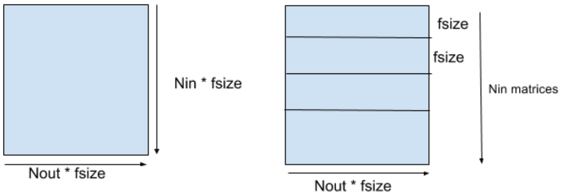
$$d_h(z_h) \odot W_h = \begin{pmatrix} d_0(z) W_0 \\ d_1(z) W_1 \\ \dots \\ d_{N_h}(z) W_{N_h} \end{pmatrix}$$

Static Hypernetworks for CNNs

Consider a CNN layer: N_{in} input channels, N_{out} input channels, and $f_{size} \times f_{size}$ filter size

Total number of parameters $= N_{in} \times N_{out} \times f_{size} \times f_{size}$

Say the weights for each layer j are stored in a matrix K^j of size $N_{in} f_{size} \times f_{size} N_{out}$ for layer j



Static Hypernetworks for CNNs

1. Each layer $j = 1, \cdots, D$ in CNN has a matrix K^j and an embedding z^j
2. The embedding matrix for all the layers $Z \epsilon N_z \times D$.
3. HyperNetwork is a two layer linear network that generates weights for each layer
4. $$K^j = HyperNetwork(z^j) \tag{16}$$

Static Hypernetworks for CNNs

$K^j = HyperNetwork(z^j)$

$$a_i^j = W_i z^j + B_i \qquad \forall i = 1, \cdots, N_{in}, \forall j = 1, \cdots, D \tag{17}$$

$$W_i \,\epsilon\, \mathbb{R}^{d \times N_z} \qquad W_{out} \,\epsilon\, \mathbb{R}^{f_{size} \times N_{out} f_{size} \times d}$$

$$K_i^j = < W_{out}, a_i^j > + B_{out} \qquad \forall i = 1, \cdots, N_{in}, \forall j = 1, \cdots, D \tag{18}$$

$$K^j = \begin{pmatrix} K_1^j & K_2^j & \cdots & K_i^j & \cdots & K_{N_{in}}^j \end{pmatrix} \tag{19}$$

Static Hypernetworks: Weight Sharing for CNNs

1. Weight sharing
2. total number of learnable parameters are now $N_z \times D + (N_z + 1) \times N_i + f_{size} \times N_{out} \times f_{size} \times f_{size} \times (d + 1)$ in comparison to $D \times N_{in} \times f_{size} \times N_{out} \times f_{size}$

Results

Model[1]	Test	Validation	Param Count
ME n-gram (Mikolov et al., 2012)	1.37		
Batch Norm LSTM (Cooijmans et al., 2016)	1.32		
Recurrent Dropout LSTM (Semeniuta et al., 2016)	1.301	1.338	
Zoneout RNN (Krueger et al., 2016)	1.27		
HM-LSTM[3] (Chung et al., 2016)	1.27		
LSTM, 1000 units [2]	1.312	1.347	4.25 M
LSTM, 1250 units[2]	1.306	1.340	6.57 M
2-Layer LSTM, 1000 units[2]	1.281	1.312	12.26 M
Layer Norm LSTM, 1000 units[2]	1.267	1.300	4.26 M
HyperLSTM (ours), 1000 units	1.265	1.296	4.91 M
Layer Norm HyperLSTM, 1000 units (ours)	1.250	1.281	4.92 M
Layer Norm HyperLSTM, 1000 units, Large Embedding (ours)	1.233	1.263	5.06 M
2-Layer Norm HyperLSTM, 1000 units	1.219	1.245	14.41 M

PennTreeBank Language Modeling

1. LSTM has 128 units
2. Embedding size of 4
3. Large Embedding 16

Results

Model[1]	enwik8	Param Count
Stacked LSTM (Graves, 2013)	1.67	27.0 M
MRNN (Sutskever et al., 2011)	1.60	
GF-RNN (Chung et al., 2015)	1.58	20.0 M
Grid-LSTM (Kalchbrenner et al., 2016)	1.47	16.8 M
LSTM (Rocki, 2016b)	1.45	
MI-LSTM (Wu et al., 2016)	1.44	
Recurrent Highway Networks (Zilly et al., 2016)	1.42	8.0 M
Recurrent Memory Array Structures (Rocki, 2016a)	1.40	
HM-LSTM[3] (Chung et al., 2016)	1.40	
Surprisal Feedback LSTM[4] (Rocki, 2016b)	1.37	
LSTM, 1800 units, no recurrent dropout[2]	1.470	14.81 M
LSTM, 2000 units, no recurrent dropout[2]	1.461	18.06 M
Layer Norm LSTM, 1800 units[2]	1.402	14.82 M
HyperLSTM (ours), 1800 units	1.391	18.71 M
Layer Norm HyperLSTM, 1800 units (ours)	1.353	18.78 M
Layer Norm HyperLSTM, 2048 units (ours)	1.340	26.54 M

Hutter Prize Wikipedia Language Modeling

1. Basic HyperLSTM has 256 units
2. Embedding size of 64