

- [Hierarchical Multiscale Recurrent Neural Networks](#)
 - [Takeaways](#)
 - [Problem proposed in this paper](#)
 - [Approach](#)
 - [Model](#)
 - [HM-RNN based on LSTM update rule](#)
 - [Some claims made in this paper](#)
- [References](#)

Hierarchical Multiscale Recurrent Neural Networks

Takeaways

Problem proposed in this paper

Learn the hierarchical multiscale structure from temporal data *without explicit boundary information*.

Approach

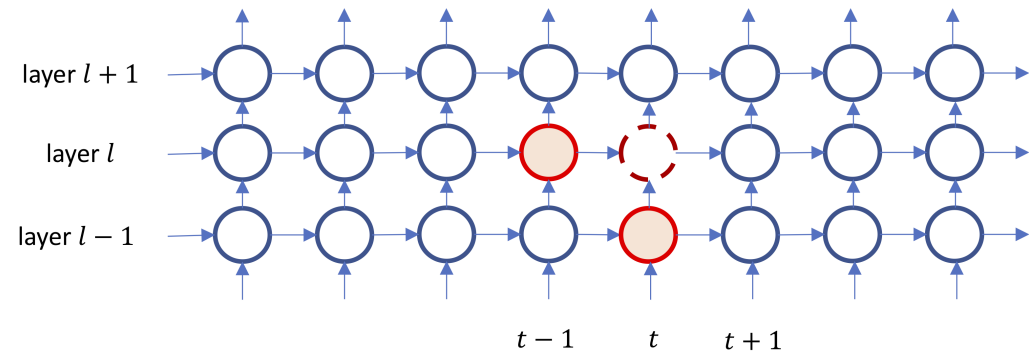
1. [key] [Introduce *a parametrized binary boundary detector*] at each layer.
 - turned on only at the time steps where a segment of the corresponding abstraction level is completely processed.
2. Implement three operations: **UPDATE**, **COPY**, **FLUSH**.
 - **UPDATE**: similar to update rule of the LSTM.
 - **COPY**: *simply copies* cell and hidden states of the previous time step which is unlike the *leaky integration* in LSTM/GRU.
 - **FLUSH**: executed when a boundary is detected, where it first ejects the summarized representation of the current segment to the upper layer and then reinitializes the states to start processing the next segment.
3. Use *straight-through estimator* to train this model to learn how to select a proper operation at each time step and to detect the boundaries.
 - “selecting one of the three operations” is a discrete variable.

4. Straight-through estimator is an very easy-to-implement solution to incorporate *a binary stochastic variable* into the neural network. It seems that some research work generalize this method into binary vector to quantize NN.

Model

1. A parametrized boundary detector in each layer of a stacked RNN which outputs a boundary state z_t^l .
 - Uses parameterized binary boundary detectors at each layer. Avoids “soft” gating which leads to “curse of updating every timestep”.
 - Whenever the boundary detector is turned on at a time step of layer l , the model considers this to be the end of a segment corresponding to the latent abstraction level of layer l .
2. Each layer selects **one** of **UPDATE**, **COPY**, **FLUSH** using the boundary states at each time step:

The computation of RNN is a grid. The selection of boundary state of the current timestep t at layer l is determined by the boundary state at left and below.



z_{t-1}^l left	z_t^{l-1} bottom	The Selected Operation	
0	0	COPY	bottom and left states both do not reach to a boundary.

z_{t-1}^l left	z_t^{l-1} buttom	The Selected Operation	
0	1	UPDATE	left state does not reaches to a boundary but buttom does. UPDATE is executed sparsely
1	0	FLUSH	left state reaches to a boundary.

z_{t-1}^l left	z_t^{l-1} bottom	The Selected Operation	
1	1	FLUSH	left state reaches to a boundary.

HM-RNN based on LSTM update rule

Recap LSTM's equation first:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{u}_t \\ \mathbf{o}_t \end{bmatrix} = W\mathbf{x}_t + U\mathbf{h}_{t-1} + \mathbf{b} \quad \text{(gates and candidate)}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) \quad \text{(cell state)}$$

$$\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot \tanh(\mathbf{c}_t) \quad \text{(hidden state)}$$

HM-RNN based on LSTM cell.

1. compute pre-activation

$$\begin{pmatrix} \mathbf{f}_t^\ell \\ \mathbf{i}_t^\ell \\ \mathbf{o}_t^\ell \\ \mathbf{g}_t^\ell \\ \tilde{z}_t^\ell \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \\ \text{hard sigm} \end{pmatrix} f_{\text{slice}} \left(\mathbf{s}_t^{\text{recurrent}(\ell)} + \mathbf{s}_t^{\text{top-down}(\ell)} + \mathbf{s}_t^{\text{bottom-up}(\ell)} + \mathbf{b}^{(\ell)} \right),$$

where

$$\begin{aligned} \mathbf{s}_t^{\text{recurrent}(\ell)} &= U_\ell^\ell \mathbf{h}_{t-1}^\ell, \\ \mathbf{s}_t^{\text{top-down}(\ell)} &= z_{t-1}^\ell U_{\ell+1}^\ell \mathbf{h}_{t-1}^{\ell+1}, \\ \mathbf{s}_t^{\text{bottom-up}(\ell)} &= z_t^{\ell-1} W_{\ell-1}^\ell \mathbf{h}_t^{\ell-1}. \end{aligned}$$

2. cell update

$$\mathbf{c}_t^l = \begin{cases} f_t^l \odot \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \odot \mathbf{g}_t^l, & \text{if UPDATE} \\ \mathbf{c}_{t-1}^l, & \text{if COPY} \\ \mathbf{i}_t^i \odot \mathbf{g}_t^l, & \text{if FLUSH} \end{cases}$$

3. hidden update

$$\mathbf{h}_t^l = \begin{cases} h_{t-1}^l, & \text{if COPY} \\ \mathbf{o}_t^l \odot \tanh(\mathbf{c}_t^l), & \text{otherwise} \end{cases}$$

- \mathbf{g} is a cell proposal vector.
- $\mathbf{i}, \mathbf{f}, \mathbf{o}$ are the input/forget/output gate.

Calculate the gradient for boundary detector

1. Straight-through estimator

- forward pass uses the step function to activate z_t^l
- backward pass uses [hardsigmoid](#) function as the biased estimator of the outgoing gradient.

$$\sigma(x) = \max(0, \min(1, (\alpha x + 1)/2))$$

2. Slope annealing

- start from slope $\alpha = 1$.
- slowly increase the slope until it reaches a threshold. In the paper, the annealing function task-specific.

Some claims made (and learned) from this paper

1. However, because *non-stationarity is prevalent in temporal data*, and that many entities of abstraction such as words and sentences are in variable length, we claim that *it is important for an RNN to dynamically adapt its timescales* to the particulars of the input entities of various length.
2. It has been a challenge for an RNN to discover the latent hierarchical structure in temporal data without explicit boundary information.
3. Although the LSTM has a *self-loop for the gradients that helps to capture the long-term dependencies* by mitigating the vanishing gradient problem, in practice, it is still limited to a few hundred-time steps due to the leaky integration by which the contents to memorize for a long-term is gradually diluted at every time step.

References

1. [Hierarchical Multiscale Recurrent Neural Networks](#)

- [its slides](#)
- 2. [Revisiting the Hierarchical Multiscale LSTM](#)
- 3. This paper uses the [*straight-through estimator*] to train neural networks with discrete variables. This paper points to two references using this methods:
 - [Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or−1](#)
 - [Strategic attentive writer for learning macro-actions](#)
 - [Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation](#)
- 4. [Notes on Hierarchical Multiscale Recurrent Neural Networks](#)