HIERARCHICAL MULTISCALE RECURRENT NEURAL NETWORKS

Midterm 4 (Paper n°1), Alberto Dicembre

Introduction

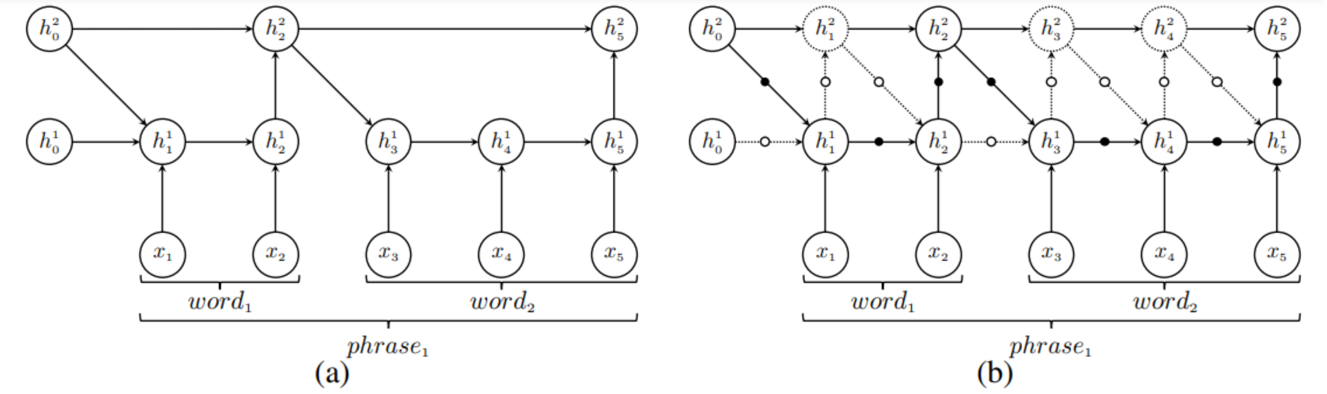
- Many problems require the ability to reason about data in a hierarchical way, with increasing levels of abstraction;
- Multiscale RNNs allow to model both temporal and hierarchical representation, while also tackling the main issues with RNNs: vanishing gradient and unfeasible time complexity;
- On previous works, multiscale RNNs had been implemented, but with timescales treated as hyperparameters (therefore fixed).

Model Description

- **Multiscale Hierarchical Recurrent Neural Networks** introduce a way to learn the structure directly from data: the update rates are adaptive.
- Uses **binary boundary detectors** at each layer to determine when a piece of information at the level of abstraction processed by the current layer ends.
- At each timestep, one of three operations is selected (based on the **boundary** states):
- **UPDATE**: updates the current state based on the previous state and the current input;
- **COPY**: copies the cell and hidden state of the previous timestep (like *Zoneout*, but again, here is used in a learned way and not fixed);
- **FLUSH**: is executed when a boundary is detected. It passes a summarized representation of the current content to the next layer, and reinitializes itself.

- How do we backpropagate? Discrete variables are not differentiable → **Straight-Through Estimator**: the step function of the forward pass is replaced by a differentiable function in the backward pass, in this case a Hard-Sigmoid, defined as
$$\texttt{hard sigm}(x) = \max\left(0, \min\left(1, \frac{ax+1}{2}\right)\right).$$

- α is a *slope* parameter: the idea is to gradually increase it, bringing it closer to the step function and therefore reducing the discrepancy between the forward and the backward pass.



(a) shows a HRNN architecture, *(b)* a HM-RNN. In the first, the hierarchy knowledge is given, in the second is automatically discovered.

Key Catch

A HM-RNN following the update rule of an LSTM:

$$\mathbf{h}_t^\ell, \mathbf{c}_t^\ell, z_t^\ell = f_{\text{HM-LSTM}}^\ell(\mathbf{c}_{t-1}^\ell, \mathbf{h}_{t-1}^\ell, \mathbf{h}_t^{\ell-1}, \mathbf{h}_{t-1}^{\ell+1}, z_{t-1}^\ell, z_t^{\ell-1}). \quad (1)$$

$$\mathbf{c}_t^\ell = \begin{cases} \mathbf{f}_t^\ell \odot \mathbf{c}_{t-1}^\ell + \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 1 \text{ (UPDATE)} \\ \mathbf{c}_{t-1}^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 0 \text{ (COPY)} \\ \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 1 \text{ (FLUSH)}, \end{cases} \quad (2)$$

Another difference with LSTMs/GRU is that here the FLUSH operation executes a *hard reset* after ejecting information towards the upper layer, whereas on those other models information doesn't get completely erased.

(1): state at time t gets information not only from time t-1, but also from the previous layer l-1 (also at time t), and from layer $l+1$ as well.
(2): in particular, we UPDATE when a boundary has been detected ($z = 1$) in layer l-1 at time t, but not in layer l at time t-1. We are englobing the (finished) information from the previous layer;
We COPY when no boundaries are detected: the piece of information is still in progress so we just keep our state untouched;
We FLUSH when we just had a boundary at current level: we finished our piece of information and are passing its representation to the layer above (which will UPDATE).
Unlike a standard RNN, UPDATE is not executed at every time step, **improving computational efficiency**.

Key Catch

(1) When is information from layer l + 1 used in layer l? In the top-down connection (6). Layer l is reinitialized with long term information after a FLUSH (except if l is the last layer) → allows the lower layer to be guided by the broader context of the higher layer.

$$\mathbf{h}_t^\ell, \mathbf{c}_t^\ell, z_t^\ell = f^\ell_{\text{HM-LSTM}}(\mathbf{c}^\ell_{t-1}, \mathbf{h}^\ell_{t-1}, \mathbf{h}^{\ell-1}_t, \mathbf{h}^{\ell+1}_{t-1}, z^\ell_{t-1}, z^{\ell-1}_t). \quad (1)$$

$$\begin{pmatrix} \mathbf{f}_t^\ell \\ \mathbf{i}_t^\ell \\ \mathbf{o}_t^\ell \\ \mathbf{g}_t^\ell \\ \tilde{z}_t^\ell \end{pmatrix} = \begin{pmatrix} \texttt{sigm} \\ \texttt{sigm} \\ \texttt{sigm} \\ \texttt{tanh} \\ \texttt{hard sigm} \end{pmatrix} f_{\texttt{slice}}\left(\mathbf{s}_t^{\text{recurrent}(\ell)} + \mathbf{s}_t^{\text{top-down}(\ell)} + \mathbf{s}_t^{\text{bottom-up}(\ell)} + \mathbf{b}^{(\ell)}\right), \quad (4)$$

$$\mathbf{s}_t^{\text{recurrent}(\ell)} = U_\ell^\ell \mathbf{h}_{t-1}^\ell, \quad (5)$$

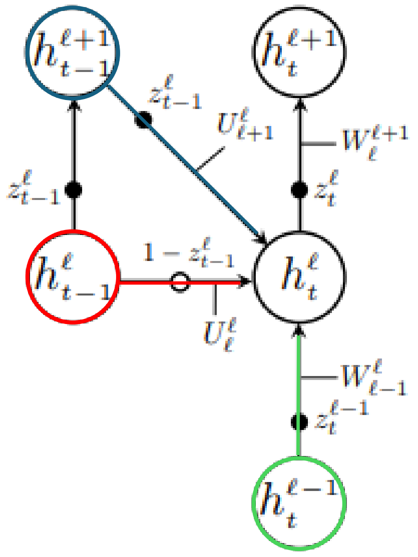
$$\mathbf{s}_t^{\text{top-down}(\ell)} = z_{t-1}^\ell U_{\ell+1}^\ell \mathbf{h}_{t-1}^{\ell+1}, \quad (6)$$

$$\mathbf{s}_t^{\text{bottom-up}(\ell)} = z_t^{\ell-1} W_{\ell-1}^\ell \mathbf{h}_t^{\ell-1}. \quad (7)$$

$$z_t^\ell = \begin{cases} 1 & \text{if } \tilde{z}_t^\ell > 0.5 \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

Binary boundary state update

Recurrent connection (5), maintaining and updating the hidden state.



We have a **bidirectional** exchange of information, not in time but in abstraction.

The bottom-up connection (7), instead, fires when we need to perform an UPDATE, englobing information from the lower-level layer. This allows the l layer to progressively build its high level representation.

Results: Char-level Language Modeling

- Minimize BPC (Bits-Per-Character) metric, defined as: $\mathbb{E}[-\log_2 p(x_{t+1} \mid x_{\leq t})]$
- Three datasets

Penn Treebank		
Model		BPC
Norm-stabilized RNN	(Krueger & Memisevic, 2015)	1.48
CW-RNN	(Koutník et al., 2014)	1.46
HF-MRNN	(Mikolov et al., 2012)	1.41
MI-RNN	(Wu et al., 2016)	1.39
ME n-gram	(Mikolov et al., 2012)	1.37
BatchNorm LSTM	(Cooijmans et al., 2016)	1.32
Zoneout RNN	(Krueger et al., 2016)	1.27
HyperNetworks	(Ha et al., 2016)	1.27
LayerNorm HyperNetworks	(Ha et al., 2016)	**1.23**
LayerNorm CW-RNN†		1.40
LayerNorm LSTM†		1.29
LayerNorm HM-LSTM	Sampling	1.27
LayerNorm HM-LSTM	Soft*	1.27
LayerNorm HM-LSTM	Step Fn.	1.25
LayerNorm HM-LSTM	Step Fn. & Slope Annealing	1.24

Layer Normalization and different approaches in *step function* handling: **Slope Annealing** outperformed the others. It wasn't implemented in the other two due to the difficulty in finding a good annealing schedule, since they are larger-scale datasets.

Hutter Prize Wikipedia	
Model	BPC
Stacked LSTM (Graves, 2013)	1.67
MRNN (Sutskever et al., 2011)	1.60
GF-LSTM (Chung et al., 2015)	1.58
Grid-LSTM (Kalchbrenner et al., 2015)	1.47
MI-LSTM (Wu et al., 2016)	1.44
Recurrent Memory Array Structures (Rocki, 2016a)	1.40
SF-LSTM (Rocki, 2016b)‡	1.37
HyperNetworks (Ha et al., 2016)	1.35
LayerNorm HyperNetworks (Ha et al., 2016)	1.34
Recurrent Highway Networks (Zilly et al., 2016)	1.32
LayerNorm LSTM†	1.39
HM-LSTM	1.34
LayerNorm HM-LSTM	1.32
PAQ8hp12 (Mahoney, 2005)	1.32
decomp8 (Mahoney, 2009)	**1.28**

~100M characters from Wikipedia, including XML and special characters (205 symbols). Got **1.32 test BPC**, tying neural models state-of-the-art.

Text8	
Model	BPC
td-LSTM (Zhang et al., 2016)	1.63
HF-MRNN (Mikolov et al., 2012)	1.54
MI-RNN (Wu et al., 2016)	1.52
Skipping-RNN (Pachitariu & Sahani, 2013)	1.48
MI-LSTM (Wu et al., 2016)	1.44
BatchNorm LSTM (Cooijmans et al., 2016)	1.36
HM-LSTM	1.32
LayerNorm HM-LSTM	**1.29**

Dataset consists in 100M characters from Wikipedia corpus, containing only alphabet and spaces (27 symbols in total).

Got **state-of-the-art** results.

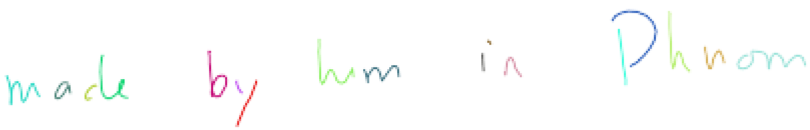
Results: Handwriting Sequence Generation

Real-valued handwriting sequence modelling task: sequences of (x_t, y_t, p_t) (coordinates + binary pen-tip location) → the goal is to predict $(x_{t+1}, y_{t+1}, p_{t+1})$.

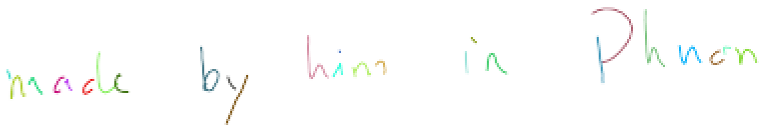
- pt $= 0$ indicates that the pen is still stroking; pt$=1$ if it's raised from the whiteboard. Usually a big shift in coordinates happens after the pen is raised.

IAM-OnDB	
Model	**Average Log-Likelihood**
Standard LSTM	1081
HM-LSTM	1137
HM-LSTM & Slope Annealing	**1167**

HM-LST outperforms Standard LSTM; moreover, results are further improved with the use of **Slope Annealing**.



Visualization by segments using the ground truth of pen-tip location



Visualization by segments using the states of z^2

Letter shapes are pretty much equal, though the model seems to be «raising the pen» more often (most noticeable on «m» letters).

Conclusions

- HM-LSTM proved to be a robust refining of Standard LSTMs. The sparsification of the updates helps in reducing computational demands, and the hierarchical design improves the ability to better retain information for long-term dependencies;

- The architecture showcased good results on the proposed tasks, obtaining either state-of-the-art results or very comparable ones. When applicable, the Slope Annealing technique proved worthy;

- Unfortunately, experiments were limited to two tasks only. The paper shows no information regarding performances on other domains.

Thanks for your attention!