Learning Latent Multiscale Structure Using Recurrent Neural Networks

Junyoung Chung[1], **Sungjin Ahn**[1], **Yoshua Bengio**[1,2]
[1] Université de Montréal
[2] CIFAR Senior Fellow
junyoung.chung@umontreal.ca

Abstract

In this paper, we introduce a hierarchical recurrent neural network architecture that enables the model to adpatively capture the underlying temporal dependencies in sequences with different timescales while not using explicit boundary information. In experiments on character-level language modelling, we demonstrate that our proposed model performs significantly better than previously proposed models, achieving the state-of-the-art.

1 Introduction

Learning both hierarchical and temporal representation has been among the long-standing challenges of RNNs in spite of the fact that hierarchical multiscale structures naturally exist in many temporal data (Schmidhuber, 1991; Mozer, 1993; El Hihi and Bengio, 1995; Kleinberg, 2003; Koutník *et al.*, 2014; Chung *et al.*, 2016). A promising approach to model such hierarchical and temporal representation is the multiscale RNNs (Schmidhuber, 1992; El Hihi and Bengio, 1995; Koutník *et al.*, 2014). Based on the observation that high-level abstraction changes slowly with temporal coherency while low-level abstraction has quickly changing features sensitive to the precise local timing (El Hihi and Bengio, 1995), the multiscale RNNs group hidden units into multiple modules of different timescales. The multiscale approach provides the following advantages that resolve some inherent problems of standard RNNs: (a) computational efficiency obtained by updating the high-level layers less frequently, (b) efficiently delivering long-term dependencies with fewer updates at the high-level layers, which mitigates the vanishing gradient problem, (c) efficient resource allocation. In addition, the learned latent hierarchical structure can provide useful information to other downstream tasks such as module structures in computer program learning, sub-task structures in hierarchical reinforcement learning, and story segments in video understanding.

2 The Proposed Model

We propose a novel framework to implement the multiscale RNN, which does not require explicit boundary information. This model, called a *hierarchical multiscale recurrent neural network* (HM-RNN), does not assign fixed update rates, but adaptively determines proper update times corresponding to different abstraction levels of the layers. We find that this model tends to learn fine timescales for low-level layers and coarse timescales for high-level layers. To do this, we introduce a binary boundary detector at each layer. The boundary detector is turned on only at the time steps where a segment of the corresponding abstraction level is completely processed. Otherwise, i.e., during the within segment processing, it stays turned off. Using the hierarchical boundary states, we implement three operations, UPDATE, COPY and FLUSH, and choose one of them at each time step. The UPDATE operation is similar to the usual update rule of the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), except that it is executed sparsely according to the

1st Workshop on Neural Abstract Machines & Program Induction (NAMPI), @NIPS 2016, Barcelona, Spain.

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 Highway Networks (Zilly *et al.*, 2016)	1.42
Recurrent Memory Array Structures (Rocki, 2016)	1.40
HyperNetworks (Ha *et al.*, 2016)	1.39
Layer-normalized HyperNetworks (Ha *et al.*, 2016)	1.38
Layer-normalized LSTM†	1.39
HM-LSTM	1.34
Layer-normalized HM-LSTM	**1.32**
PAQ8hp12 (Mahoney, 2005)	**1.32**
decomp8 (Mahoney, 2009)	**1.28**

Table 1: Bits-per-character on the Hutter Prize Wikipedia test set. (†) This model is implemented by the authors as the standard LSTM architecture using layer normalization (Ba *et al.*, 2016).

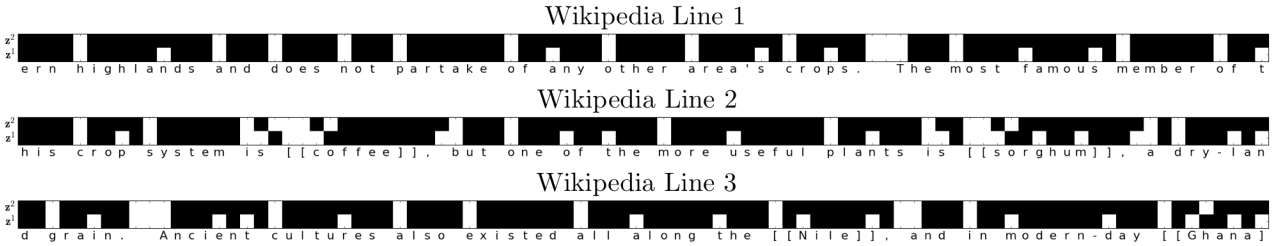

Figure 1: Latent hierarchical structure in the `enwik8` validation set captured by the HM-LSTM

detected boundaries. The COPY operation simply copies the cell and hidden states of the previous time step. Unlike the leaky integration of the LSTM or the Gated Recurrent Unit (GRU) (Cho *et al.*, 2014), the COPY retains the whole states without any loss of information. The FLUSH operation is 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. Learning to select a proper operation at each time step and to detect the boundaries, the HM-RNN discovers the latent hierarchical structure of the sequences. We find that the straight-through estimator (Bengio *et al.*, 2013; Courbariaux *et al.*, 2015) is efficient for training this model containing discrete variables.

3 Experimental Results

Table 1 shows the character-level language modelling results of the HM-LSTM. The HM-LSTM achieves the state-of-the-art BPC of 1.33. Although the HM-LSTM performs at the state-of-the-art for the neural models, its compression performance is still behind the best models such as PAQ8hp12 (Mahoney, 2005) and decomp8 (Mahoney, 2009). Figure 1 shows the states of the boundary detectors of the HM-LSTM when a validation sequence from the Hutter Prize Wikipedia dataset is fed to the model. The boundary detector of the first layer tends to detect the boundaries of the words, where the boundary detector of the second layer tends to fire when it detects either a word or $2, 3$-grams.

A remarkable point is the fact that we do not pose any constraint on the number of boundaries that the model can fire up. The model, however, learns that it is more beneficial to delay the information ejection to some extent. We conjecture the reason that the model works in this way is due to the FLUSH operation. That is, the FLUSH poses an implicit constraint on the frequency of boundary detection because it contains both a reward (feeding fresh information to upper layers) and a penalty (erasing accumulated information). The model finds an optimal balance between the reward and the penalty.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2015). Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. *Association for Computational Linguistics (ACL)*.

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131.

El Hihi, S. and Bengio, Y. (1995). Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, pages 493–499. Citeseer.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Kleinberg, J. (2003). Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, **7**(4), 373–397.

Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork rnn. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*.

Mahoney, M. V. (2005). Adaptive weighing of context models for lossless data compression.

Mahoney, M. V. (2009). Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html*.

Mozer, M. C. (1993). Induction of multiscale temporal structure. *Advances in neural information processing systems*, pages 275–275.

Rocki, K. M. (2016). Recurrent memory array structures. *arXiv preprint arXiv:1607.03085*.

Schmidhuber, J. (1991). Neural sequence chunkers.

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, **4**(2), 234–242.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 1017–1024.

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., and Salakhutdinov, R. (2016). On multiplicative integration with recurrent neural networks. *arXiv preprint arXiv:1606.06630*.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2016). Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.