Hierarchical Multiscale Recurrent Neural Networks

Junyoung Chung, Sungjin Ahn and Yoshua Bengio

Intelligent Systems for Pattern Recognition
Midterm 4
Academic year 2023/24

Irene Testa

Introduction to the problem

Background

- Temporal data is often structured **hierarchically** (e.g., characters → words → phrases)
- Representations at different levels of abstraction change at **different timescales**: high-level abstractions change slowly with temporal coherency (i.e., are robust to small local changes in the timing of events), whereas low-level abstractions have quickly changing features, sensitive to the precise timing of events [1]
- **Want to learn both hierarchical and temporal representations and make efficient use of their hierarchical structure**

Solutions Proposed in Related Works

- **LSTM** [2]: employs the multiscale update concept, where hidden units have different forget and update rates and thus can operate with different timescales
  - 👎 timescales are not organized hierarchically
  - 👎 in practice, gradient propagation is still limited to a few hundred of steps
  - 👎 computationally expensive because it has to update units at every time step
- **Multiscale RNN**: stack multiple layers of RNNs in a decreasing order of update frequency
  - 👎 non-adaptive update rate, either
    - the hierarchical boundary structure is known (often expensive to obtain and limited to the number of boundary levels explicitly observed in the data) [3], [4]
    - updates are performed at a fixed rate (i.e., controlled by a hyperparameter, constraining the ability to learn variable-length representations) [1], [5]
  - 👍 computationally efficient as units are not updated at each time step

Model description – Intuition

Idea

Hierarchical Multiscale Recurrent Neural Network (**HM-RNN**) is an innovative model that adaptively learns the hierarchical multiscale structure from temporal data without explicit boundary information.

It uses:

- **Binary boundary detectors**: learned binary variables (one for each layer) turned on only at the time steps where a segment of the corresponding abstraction level (e.g., word or phrase) should end in order to optimize the overall target objective
- A novel **update mechanism**

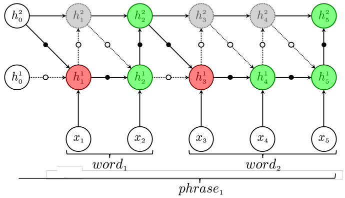

Figure 1: The HM-RNN architecture that discovers the hierarchical multiscale structure in the data without explicit boundary information. Units are color-coded based on the operation performed at each corresponding time step: green, gray or red for an UPDATE, a COPY or a FLUSH operation respectively.

Update Mechanism

At each time step, based on the state of boundary detectors, one of the following operations is executed:

- **UPDATE**: similar to the update rule of LSTMs only that it is executed sparsely according to the boundary detectors (performed if the layer below detected a boundary at the current time step and no boundary was detected by the layer at the previous time step)
- **COPY**: simply a copy of the cell and hidden states of the previous time step (executed when no boundaries are detected)
- **FLUSH**: involves ejecting the summarized representation of the current segment to the upper layer and erasing the state to start processing the next segment (executed if the layer detected a boundary at the previous time step)

Model description – Details

Hierarchical Multiscale LSTM (HM-LSTM)

In a HM-LSTM model of L layers ($\ell = 1, \ldots, L$), at time step t, each layer ℓ performs the following update

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

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

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

where $\mathbf{h}$ ($\mathbf{h}_t^0 = \mathbf{x}_t$), $\mathbf{c}$ and z denote the hidden, the cell and the boundary detector state, respectively, while $(\mathbf{f}_t^\ell, \mathbf{i}_t^\ell, \mathbf{o}_t^\ell)$ are forget, input and output gates and $\mathbf{g}_t^\ell$ denotes a cell proposal vector, whose values are given by

$$\begin{pmatrix} \mathbf{f}_t^\ell \\ \mathbf{i}_t^\ell \\ \mathbf{o}_t^\ell \\ \mathbf{g}_t^\ell \\ \tilde{z}_t^\ell \end{pmatrix} = \begin{pmatrix} \mathtt{sigm} \\ \mathtt{sigm} \\ \mathtt{sigm} \\ \mathtt{tanh} \\ \mathtt{hard\text{-}sigm} \end{pmatrix} f_{\mathtt{slice}}\Big(\underbrace{(1 - z_{t-1}^\ell) U_\ell^\ell \mathbf{h}_{t-1}^\ell}_{\mathbf{s}_t^{\text{recurrent}(\ell)}} + \underbrace{z_{t-1}^\ell U_{\ell+1}^\ell \mathbf{h}_{t-1}^{\ell+1}}_{\mathbf{s}_t^{\text{top-down}(\ell)}} + \underbrace{z_t^{\ell-1} W_{\ell-1}^\ell \mathbf{h}_t^{\ell-1}}_{\mathbf{s}_t^{\text{bottom-up}(\ell)}} + \mathbf{b}^\ell\Big)$$

$U_i^j \in \mathbb{R}^{(4dim(\mathbf{h}^j)+1) \times dim(\mathbf{h}^i)}$, $W_i^j \in \mathbb{R}^{(4dim(\mathbf{h}^j)+1) \times dim(\mathbf{h}^i)}$ denote state transition parameters from layer i to layer j, $\mathbf{b}^\ell \in \mathbb{R}^{4dim(\mathbf{h}^\ell)+1}$ is a bias term and $\mathtt{hard\text{-}sigm}(x) = \max(0, \min(1, \frac{ax+1}{2}))$ with a being a slope variable.
The binary boundary state z_t^ℓ could be obtained by

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

or sampling from a Bernoulli distribution ($z_t^\ell \sim \text{Bernoulli}(\tilde{z}_t^\ell)$). Since the input should not be omitted, $z_t^0 = 1$ for all t.

The Slope Annealing Trick

- To backpropagate z_t^ℓ, use the **Straight-Through Estimator** [6]: replace the step function used in the forward pass (non-differentiable) with the `hard-sigmoid` (differentiable).
- To reduce the discrepancy between the two functions used during the forward pass and the backward pass, gradually increase the slope a of the `hard-sigmoid` function.

Standard LSTM

For comparison, in a LSTM model with a single layer, at time step t, the following operations are performed

$$\mathbf{h}_t, \mathbf{c}_t = f_{LSTM}(\mathbf{c}_{t-1}, \mathbf{h}_{t-1})$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathtt{tanh}(\mathbf{c}_t)$$

$$\begin{pmatrix} \mathbf{f}_t \\ \mathbf{i}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \mathtt{sigm} \\ \mathtt{sigm} \\ \mathtt{sigm} \\ \mathtt{tanh} \end{pmatrix} f_{\mathtt{slice}}(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b})$$

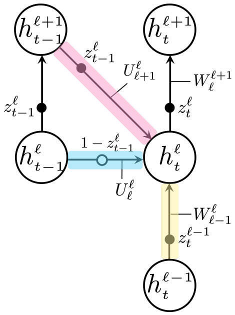

Figure 2: Gating mechanism of the HM-LSTM.

In the paper $\mathbf{s}_t^{\text{recurrent}(\ell)}$ does not include the factor $(1 - z_{t-1}^\ell)$, however this contrasts with the definition of the FLUSH operation (that should perform a 'hard' reset) and with the diagram in Figure 2. Furthermore, the dimensions of W_i^j and U_i^j were wrongly defined.

Model description – Key points

Key Features of the Model

- The model **discovers** underlying hierarchical structure in sequences **without using explicit boundary information**
- Unlike the LSTM, $\mathbf{f}, \mathbf{i}, \mathbf{o}$ and $\mathbf{g}$ do not need to be computed at every time step, **reducing computational costs** (e.g., in case of COPY none of them needs to be computed)
- The top down connection from layer $(\ell + 1)$ to (ℓ) makes the layer (ℓ) to be **initialized** with more **long-term information** (i.e., a broader context) after a boundary is detected
- The model employs the **multiscale update** concept:
  - an UPDATE at a layer can occur only after at least one UPDATE is performed at its previous layer, resulting in a lower frequency of updates at higher layers compared to lower ones
- The COPY operation retains the whole state without any loss of information, **improving gradient propagation** (similar to Zoneout [7], but instead of being randomly applied, is performed based on the input)
- The FLUSH operation **passes a summary information** to the upper layer (allowing it to build its higher-level representation) and, differently from the forget operation in LSTM, it **completely erases the previous state** of the layer. Furthermore, it **incorporates both a reward** (feeding fresh information to upper layers) and a **penalty** (erasing accumulated information).
- To compute the target, the model **considers representations at every level of abstraction**, not solely those at the higher levels (see Figure 3)

Experiments

Character-level Language Modeling

- **Task**: predict the next character, i.e., discrete sequence modeling
- **Datasets**: Penn Treebank [8], Text8 [9], Hutter Prize Wikipedia [10]
- **Evaluation metric**: Bits-per-character, $BPC = \mathbb{E}[-\log_2 p(x_{t+1}|x_{\leq t})]$

Handwriting Sequence Generation

- **Task**: predict the next pen coordinates and pen state (up or down), i.e., real-valued sequence modeling
- **Dataset**: IAM-OnDB [11]
- **Evaluation metric**: Average Log-Likelihood

Model Architecture

- **Input Embedding Layer**: maps each input symbol into a 128-dimensional continuous vector without using any non-linearity
- **3 HM-LSTM Layers** (512 units on Penn Treebank, 1024 units on Text8 and Hutter Prize Wikipedia, 400 units on IAM-OnD)
- **Output Embedding Layer**: a feedforward neural network (512 units on Penn Treebank, 2048 units on Text8 and Hutter Prize Wikipedia, 400 units on IAM-OnD) receiving at each time step the hidden states of the three HM-LSTM layers, adaptively weighted by additional scalar gating units (Figure 3)
- **Output Layer**: softmax layer for character-level language modeling, Mixture Density Network [12] for handwriting sequence generation

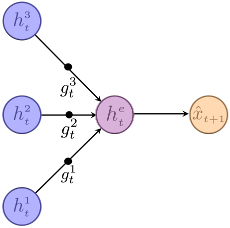

Figure 3: Output Embedding Layer.

Results

Table 1: Results for character-level language modeling.

Penn Treebank		Text8		Hutter Prize Wikipedia	
Model	**BPC**	**Model**	**BPC**	**Model**	**BPC**
LayerNormHyperNetworks [13]	**1.23**	BatchNorm LSTM [14]	1.36	Recurrent Highway Networks [15]	1.32
LayerNorm HM-LSTM Sampling	1.27	HM-LSTM	1.32	decomp8 [9] **	**1.28**
LayerNorm HM-LSTM Soft *	1.27	LayerNorm HM-LSTM	**1.29**	HM-LSTM	1.34
LayerNorm HM-LSTM Step	1.25			LayerNorm HM-LSTM	1.32
LayerNorm HM-LSTM Step & SA	1.24				

* a variant of HM-LSTM that does not discretize the boundary detector states; ** compression-based model (non-neural)

Findings

- **PennTreebank**: comparable results to SotA; best score achieved with the Slope Annealing (SA) trick
- **Text8**: SotA results
- **Hutter Prize Wikipedia**: tie with SotA results among neural models
- **IAM-OnDB**: better than standard LSTM; best score achieved with the SA trick
- The **discovered hierarchical structure is very similar to the intrinsic structure** observed in the data: z^1 tends to be turned on when it sees a space or after it sees a space; z^2 tends to fire when it sees either the end of a word or 2, 3-grams (Figure 4)

Table 2: Results for handwriting sequence generation.

IAM-OnDB	
Model	**Average Log-Likelihood**
Standard LSTM [2]	1081
HM-LSTM	1137
HM-LSTM Step & SA	**1167**

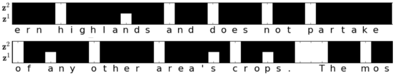

Figure 4: Discovered hierarchical multiscale structure in the first line of the Hutter Prize Wikipedia dataset ($z^l = 1$ in white, $z^l = 0$ in black).

Comments

Pros

- **Novelties**:
  - The model **learns latent hierarchical structure** of sequences without using explicit boundary information
  - The **Slope Annealing Trick**, proven effective and potentially applicable in other contexts
- **Computational efficiency**: upper layers are updated less frequently
- Efficacy in capturing **long term dependencies**: less frequent updates reduce the gradient vanishing problem
- Flexible **resource allocation**: may allocate more units to higher layers, which model long-term dependencies, without significantly increasing computational costs
- **Interpretability**: discrete variables allow to inspect the discovered hierarchical structure
- **State-of-the-art results** (or comparable performance) on several datasets

Cons

- The use of discrete variables (z variables) makes the model **no longer differentiable** and the Slope Annealing Trick requires finding a good schedule, which may not be feasible with large-scale datasets
- Results on the claimed computational savings were not provided because, at the time of publication, it was **not entirely clear how to implement the state conditional computation in a mini-batch setting** [16]
- **Experiments were limited** to two tasks, and hierarchies beyond three levels were not investigated
- Current SotA results in the studied tasks are achieved by transfomer-based models [17]

References

[1] S. Hihi and Y. Bengio, "Hierarchical Recurrent Neural Networks for long-term dependencies," *Advances in neural information processing systems*, vol. 8, 1995.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] W. Ling, I. Trancoso, C. Dyer, and A. W. Black, "Character-based neural machine translation," *arXiv preprint arXiv:1511.04586*, 2015.

[4] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 553–562.

[5] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," in *International conference on machine learning*, PMLR, 2014, pp. 1863–1871.

[6] G. Hinton, "Neural networks for machine learning," *Coursera, video lectures*, 2012.

[7] D. Krueger, T. Maharaj, J. Kramár, *et al.*, "Zoneout: Regularizing RNNs by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, 2016.

[8] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The Penn Treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[9] M. Mahoney, "Large text compression benchmark,", 2011.

[10] M. Hutter, "The human knowledge compression contest,", 2012.

[11] M. Liwicki and H. Bunke, "IAM-OnDB-an on-line english sentence database acquired from handwritten text on a whiteboard," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, IEEE, 2005, pp. 956–961.

[12] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[13] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[14] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[15] J. G. Zilly, R. K. Srivastava, J. Koutnık, and J. Schmidhuber, "Recurrent highway networks," in *International conference on machine learning*, PMLR, 2017, pp. 4189–4198.

[16] J. Chung, S. Ahn, and Y. Bengio, "Responses to reviews of 'Hierarchical multiscale recurrent neural networks',", 2016. [Online]. Available: `https://openreview.net/forum?id=S1di0sfgl`.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[18] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical Multiscale Recurrent Neural Networks," *arXiv preprint arXiv:1609.01704*, 2016.