4° Assignment ISPR

Andrea Lepori

June 3, 2025

Selected paper:

Junyoung Chung, Sungjin Ahn, Yoshua Bengio

Hierarchical Multiscale Recurrent Neural Networks, ICLR 2017

Introduction to the problem

- **Ultimate goal**: learning both **hierarchical and temporal (different timescales) latent representation** in a seq.
- Limitation of previous approaches:
  - **update all units at every time**, not efficient or **handcrafted timescale with fixed update frequencies** don't suit to different segment with different lengths
  - **use explicit hierarchical boundary information** that is difficult to obtain
- What we want for our RNN:
  - **dynamical adaption of timescales** by learning from data (**no fixed update rate**)
  - discover the latent hierarchical structure in temporal data **without explicit boundary information**
  - **the UPDATE operation** should be **executed sparsely** with a finer granularity at the low-level and coarser at the high-level

Model description- Hierarchical Multiscale Recurrent Neural Network (HM-RNN)

- **parametrized binary boundary detector at each layer** turned on at time steps where a segment of the corresponding abstraction level is completely processed and feeds the summarized repre. of the detected info to the upper layer
- Using the boundary states, at each time step, each layer can select among 3 op.:
  - **UPDATE**: similar to LSTM update but executed sparsely according to the detected boundaries
  - **COPY**: retains the whole states (NO leaky integration of LSTM)
  - **FLUSH**: when a boundary is detected, 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 and discovering **automatically** (no boundary info, no fixed update rates) the hierarchical multiscale structure underlying data.
- **straight-through estimator** to train the model: boundary detector introduces **hard choice** and **lose differentiability**

Key catch of the model, represented by a commented equation (I)

HM-LSTM update rule, model of L layers, at each l, generic layer, at each time step t performs the following update:

$$h_t^l, c_t^l, z_t^l = \mathrm{f}_{\text{HM-LSTM}}^l(c_{t-1}^l, h_{t-1}^l, h_t^{l-1}, h_{t-1}^{l+1}, z_{t-1}^l, z_t^{l-1})$$

where **h** and **c** hidden and cell states,respectively. The function $f_{\text{HM-LSTM}}^l$ is implemented as follows. First using the two boundary states z_{t-1}^l and z_t^{l-1}, the cell state c_t^l is updated:

$$c_t^l = \begin{cases} f_t^l \odot c_{t-1}^l + i_t^l \odot g_t^l & \text{if } z_{t-1}^l = 0 \text{ and } z_t^{l-1} = 1 \quad \text{(UPDATE)} \\ c_{t-1}^l & \text{if } z_{t-1}^l = 0 \text{ and } z_t^{l-1} = 0 \quad \text{(COPY)} \\ i_t^l \odot g_t^l & \text{if } z_{t-1}^l = 1 \quad \text{(FLUSH)} \end{cases}$$

Hidden state h_t^l is updated conditionally based on the COPY operation:

$$h_t^l = \begin{cases} h_{t-1}^l & \text{if COPY} \\ o_t^l \odot \tanh(c_t^l) & \text{otherwise} \end{cases}$$

Key catch of the model, represented by a commented equation (II)

Brief explanation (**f**, **i**, **o**, **g**) are forget, input, output gates and cell proposal vector, respectively, as LSTM. But not necessarily these values are computed at every time step (see COPY op.)

COPY operation: $(c_t^l, h_t^l) \leftarrow (c_{t-1}^l, h_{t-1}^l)$, upper layer keeps state unchanged until it receives summarized input from lower layer

UPDATE operation: update summary rep. of layer l if the boundary z_t^{l-1} is detected from below but my boundary z_{t-1}^l was not found at the previous time step.

FLUSH operation: executed if a boundary at my layer is detected at the previous time step, consists of two sub-op.:

- **EJECT** to pass the current state to the upper layer
- **RESET** to reinitialize the state before starting to read a new segment

This operation implicitly forces the upper layer to absorb the summary information of the lower layer segment, because otherwise it will be lost.

FLUSH operation is **hard reset** different from LSTM soft reset

Key catch of the model, represented by a commented equation (III)

The gates, cell proposal and boundary variable are computed as:

$$\begin{pmatrix} f_t^l \\ i_t^l \\ o_t^l \\ g_t^l \\ \tilde{z}_t^l \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \\ \text{hard}\,\sigma \end{pmatrix} \left(\text{fslice} \left(s_t^{\text{recurrent(l)}} + s_t^{\text{top-down(l)}} + s_t^{\text{bottom-up(l)}} + b^l \right) \right) \quad (4)$$

Where:

$$s_t^{\text{recurrent(l)}} = U_l^l h_{t-1}^l \quad (5)$$

$$s_t^{\text{top-down(l)}} = z_{t-1}^l U_{l+1}^l h_{t-1}^{l+1} \quad (6)$$

$$s_t^{\text{bottom-up(l)}} = z_t^{l-1} W_{l-1}^l h_t^{l-1} \quad (7)$$

top-down connection from layer (l+1) to l activated only if a boundary detected at time t-1 at layer l, makes the layer l to be initialized with more long-term info from upper layer after FLUSH op. **bottom-up connection** The input from the lower layer (l-1) becomes effective only when a boundary detected at current t in layer (l-1)

Key catch of the model, represented by a commented equation (III)

The binary boundary state z_t^l is obtained by binarizing the activation $z_t^l = \mathrm{f}_{\mathrm{bound}}(\tilde{z}_t^l)$ with a step function:

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

or sampled from a Bernoulli distribution. Anyway we introduce a *discrete/hard decision* that makes the standard backprop no longer applicable due to the non-differentiability. We use **straight through estimator** to train our model. *The non-differentiable function used in the forward pass (i.e., the step function in our case) is replaced by a differentiable function during the backward pass (i.e., the hard sigmoid function in our case).* **Problem**: the straight-through estimator is a biased estimator. Use **slope annealing trick** to reduce the bias of the straight-through estimator. Gradually increasing the slope a of the hard sigmoid function, we make the hard sigmoid be close to the step function. Start with a smoother slope for stability then increase the slope to reduce bias.

Key (empirical) results

- evaluated on **character-level language modeling** (bits-per-character(BPC) metric) and **handwriting sequence generation**
- On the first task, the HM-LSTM achieved state-of-the-art or comparable results on the Text8, Penn Treebank and Hutter Prize Wikipedia datasets. HM-LSTM using the step function for the hard boundary decision outperforms the others using either sampling or soft boundary decision (i.e., hard sigmoid). BPC is improved with the slope annealing trick, which reduces the bias of the straight-through estimator. However, although this neural models, show remarkable performances, their compression performance is still behind the best models such as PAQ8hp12 and decomp8 (non-neural, classical compression algorithms specialized in lossless data compression)
- On the second task we observe that the HM-LSTM outperforms the standard LSTM. The slope annealing trick further improves the test log-likelihood of the HM-LSTM
- the boundary detector of the first layer, z^1, tends to be turned on when it sees a space or after it sees a space, which is a reasonable breakpoint to separate between words.

Comment on novelties, strong points and weaknesses

- interpretation of the second layer boundaries is not as apparent as the first layer boundaries, it seems to segment at reasonable semantic/syntactic boundaries (end of a word or 2,3-grams)
- The model uses to some extent the concept of surprise to learn the boundary
- The model learns by itself that it is more beneficial to delay the information ejection to some extent, due to the FLUSH operation that 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).
- internal process of the RNN becomes **more interpretable**
- the number of UPDATE operations decreases as the layer level increases. The total number of updates is 335 for the HM-LSTM which is 60% of reduction from the 810 updates of the standard RNN architecture.
- model long term rel. with less update to higher layers $\rightarrow$ computational efficiency & mitigates grad. vanishing, allow flexible resources allocation (more capacity to the higher/abstract levels), very interpretable. Not very compact and to implement all the gates and bound detectors a lot of param!!