2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

Layer specificity of acquired memory duration in multilayer LSTM networks

Kazuki Hatanaka†, Jun-nosuke Teramae‡ and Naoki Wakamiya§

†‡§Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
‡§Center for Information and Neural Networks
1–4 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
†Email: k-hatank@ist.oasaka-u.ac.jp
‡Email: teramae@ist.oasaka-u.ac.jp
§Email: wakamiya@ist.osaka-u.ac.jp

Abstract—The Long Short-Term Memory (LSTM) network is a recurrent neural network achieving impressive performance recently on machine learning tasks of sequential data such as speech recognition and machine translation. The network consists of LSTM units that are able to store past inputs into their internal memory variables, which allows the network to suitably respond to current input with reflecting a past history of a given input sequence. If the number of LSTM units of the network is fixed, achieved performance of the network generally increases as the numbers of layers of the network increases. It remains unclear, however, why deeper LSTM networks can achieve higher performance. Functional roles of the layered structure of the LSTM network also remain largely elusive. As a first step to answer these questions, here, we analyze layer-wise difference of temporal dynamics of the memory variable of each unit. We train layered LSTM networks to solve the simplest task of natural language processing called Character Level Language Model (CLLM) that requires networks to predict the next character in given texts. We found that, after learning, LSTM units in a deeper layer averagely keep longer memory duration than these in a shallower layer. Memory duration is broadly distributed among units in a deeper layer than a shallower layer. To understand underlying mechanisms of the difference of memory durations, we performed additional experiments to artificially modulate these durations. Results implies that the sequential task for LSTM networks require units in different layers to share different roles of memorization.

1. Introduction

Recurrent neural networks are a type of neural networks in which outputs of the network feedback to the network again to be used as parts of inputs at the next time step. Input at a time, therefore, includes past history of input sequences. This type of networks has been applied to tasks where temporal structure of input sequences is crucial to solve them. These include natural language processing such as speech recognition and machine translation.

One of the most successful recurrent networks is the LSTM network [1]. In place of usual neural units, the network consists of many LSTM units with a memory variable and multiple gate variables that are used to control the value of the memory variable. Owing to the memory variable, the LSTM network is able to store long history of input, which allows the network to properly respond to current input with reflecting past history of input sequences. This architecture of the LSTM unit with a memory and gate variables are designed to stabilize the backpropagation learning that is generally employed to train LSTM networks.

In most of studies, LSTM units are arranged on a multilayered structure. Achieved performance of a LSTM network is generally high for the network having more numbers of layers if the number of units is fixed [2, 3, 4]. Actually, while variants of the LSTM network have been proposed so far, most of them have been based on layered network architectures [5, 6, 7, 8].

However, questions remain unclear; why multilayered structure is required for LSTM networks to achieve higher performance, why deeper LSTM networks generally achieve better performance than shallower networks, and what are difference of functional roles among LSTM units in different layers. Lack of these fundamental knowledges is one of major obstacles of us to develop recurrent neural networks that are able to solve sequential tasks with higher performance.

As a first step to reveal functional roles of multilayer structure of the LSTM networks, here, we study layer-wise difference among temporal behavior of memory variables of LSTM units. We train multilayered LSTM networks with sequential tasks with different characteristics and measure temporal duration of memory variables of LSTM units in different layers. We found that memory durations vary from layer to layer regardless of training data sequences, which implies that LSTM units in different layers are intrinsically responsible for sequential memories with different length.

Rest of the paper is organized as follows. In section 2, we introduce details of the LSTM networks. Experimental settings and results of analysis are given in section 3 and 4 respectively. The last section gives discussion, remaining future subjects, and conclusion.

2. Long Short-Term Memory (LSTM)

The LSTM [9] unit has a memory variable and four gates variables that control accesses to the memory variable. For network with n LSTM units, temporal development of values of a memory ($c_t \in \mathbb{R}^n$) and gate variables ($g, i, f, o \in \mathbb{R}^n$) are given as,

$$g = \tanh(W_{gx}x_t + W_{gy}y_{t-1} + b_g) \quad (1)$$
$$i = \sigma(W_{ix}x_t + W_{iy}y_{t-1} + b_i) \quad (2)$$
$$f = \sigma(W_{fx}x_t + W_{fy}y_{t-1} + b_f) \quad (3)$$
$$o = \sigma(W_{ox}x_t + W_{oy}y_{t-1} + b_o) \quad (4)$$
$$c_t = f \cdot c_{t-1} + i \cdot g \quad (5)$$
$$y_t = o \cdot \tanh(c_t). \quad (6)$$

Here, variables $x_t, y_t \in \mathbb{R}^n$ represent input and output of memory variables at the time step t respectively. $W \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ are network parameters that should be trained by a supervised learning. σ is a logistic sigmoid function and *tanh* is a hyperbolic tangent function. Both of σ and *tanh* are applied in element-wise. The midpoint represents vector product. Owing to the memory variables, the network is able to story memory in relatively long-term, which allows the LSTM networks to learn tasks in which the network is required to represent long and complex temporal correlation embedded in input sequences.

3. Experiments

In this section, we explain the simplest language task that is called Character-Level Language Model (CLLM). We also provide details of network structure of multilayered LSTM networks we employ in the paper.

3.1. Character-Level Language Model (CLLM)

CLLM is a language task that requires a network to predict the next character of a text input sequences. The network receives a character in the text sequentially as an input at a time. Output of the network at the time is recognized as predicted probability of appearance of the next character in the text. Model parameters, Ws and bs, are trained by a supervised learning as the network predict next characters of a training input sequence as precisely as possible. To realize prediction, therefore, the network should properly learn temporal structure of a given sequential text.

3.2. Datasets

To reduce special features of training data set, in this study, we use three datasets whose features are largely different among each other (Table1).

The "King Lear" is a script written by Shakespeare. Most of the sentences of the script are composed of the form as (name of the characters): (speech). The data "Linux" is the source code of the Linux program. It is characterized by lots of nested structures and frequent appearance of reserved words like "if" and "return". The data named "Wikipedia" is a XML code of Wikipedia. Similar to the "Linux", it contains nested structures and frequently appeared keywords while many ordinal natural language sentences are also contained simultaneously.

Dateset	# of characters	# of vocabulary
King Lear	99993	62
Linux	999589	96
Wikipedia	352158	194

Table 1: Datasets

For all of the datasets, we use the first 90% of them as training data and the rest 10% as test data.

3.3. Multilayerd model and learnig protocol

A multilayered LSTM network we used has 3 intermediate layers. Each layer consists of 400 LSTM units. We use full-connected networks to connect units between subsequent layers. During training phase, we introduce dropout [10] with 50% of connections. Objective of the learning phase is to minimize the cross-entropy loss between target characters and softmax of outputs units. As an adaptive gradient method, we use the Adam [11]. To realize efficient learning, we use a minibatch of 100 characters of length 10 each in time for backpropagation. Gradient clipping [12] with threshold 5 is also employed to prevent values of gradient from getting extremely large.

4. Results

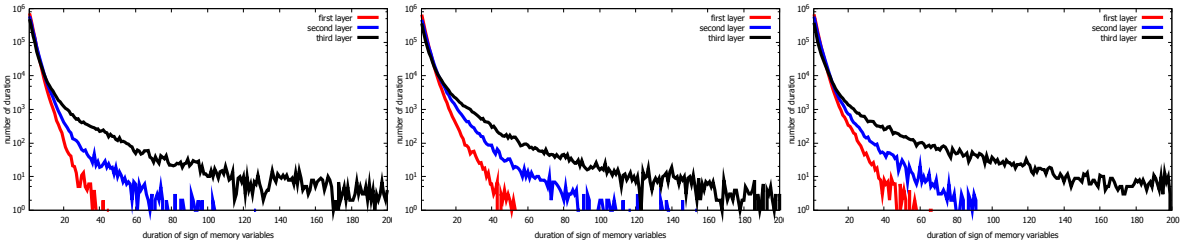
After we train the network, we feed test sequences to the network as input sequences and measure duration during which memory variables keep their sign unchanged.

4.1. Duration of the sign of memories

As input characters are given sequentially to the network, values of the memory variables are continuously changing in time with reflecting input sequences in past and current. To characterize temporal structure of these dynamics, we focus on duration in which the sign of a memory variables changes.

Figure 1 shows histograms of the duration of the memory variables for LSTM units of different layers and different datasets. For all of three datasets, histograms are more broadly distributed for deeper layers. Units in a deeper layer, therefore, averagely show longer duration than these in a shallower layer. The mean duration of memories monotonically increases as the position of a layer becomes deep.

These results suggest that LSTM units in different layers responsible for memories with different length. While



(a) King Lear (b) Wikipedia (c) Linux

Figure 1: Histograms of duration in which the sign of memory variables for LSTM units unchanged. Red, blue and black lines represent histograms of memory duration of LSTM units of the first, second and third layers respectively. (a), (b), and (c) show results obtained from the network trained for different datasets, KingLear, Wikipedia, and Linux respectively.

memory variables of units in shallower layers frequently change their value to represent recent history of inputs, these in deeper layers tend to keep longer memories to represent further long dependency of characters in given texts. In other words, results imply that multilayer LSTM networks can install different aspects of temporal structure of given texts into units of different layers. In this sense, units in different layers share different functional roles to store information of given text with different time scales. It is, thus, suggested that multilayer LSTM networks have ability to solve tasks with complex input correlation because they are able to separately store different temporal correlation into layers.

4.2. Artificial modulation of memory duration

Since networks are trained to reduce cross-entropy loss function between output and coming character in sequential texts. Above results of role sharing of memory duration among layers suggest that effective constraints of the loss function to units are different among layers.

To confirm the hypothesis, here, we put an additional term into the loss function to artificially modulate memory duration of each layer. The modulated loss function is given as,

$$loss = -\sum_{k=1}^{62} t_k log_e \frac{y_k}{\sum_{j=1}^{62} y_j} + \frac{r_i}{400} \sum_{k=1}^{400} (c_t^{ik} - c_{t-1}^{ik})^2. \qquad (7)$$

Here, the first term is the original cross entropy loss. t_k and y_k are the kth training label in 62 character of given text data and value of the kth node in output layer respectively. c_t^{ik} is the value of the memory variable of kth unit in ith layer at time t, $1 \leq i \leq 3$. For positive coefficient r_i, therefore, the second term tries to decrease difference between temporally subsequent values of memory variables of units in ith layrer. In other words, this term forces memory variables of ith layer do not fluctuate largely in time. Thus, it is expected that memory duration of ith layer will increase as we increase r_i.

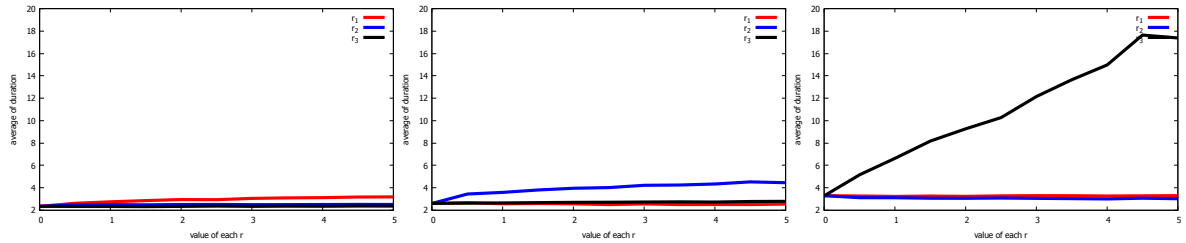
Figure 2 shows the mean duration of memory variables of different layers when we increase r_1, r_2, and r_3 respectively. As exected, whereas ranges of values of r_i are the same for all of them, effects of them are largely different for different r_i. While r_1 hardly to change memory duration of the first layer, r_2 slightly and r_3 largely change that of the second and the third layer respectively. This means that the first term of equation (7), i.e. cross entropy loss of sequence prediction, strongly regulates dynamics of memory variables of the first layer but weakly do so these of the third layer.

5. Conclusion

In this study, we numerically analyzed temporal dynamics of memory variables over different layers in order to reveal functional roles of layered structure of the LSTM networks. We measure duration in which the sign of memory variables is fixed for multilayered LSTM networks that are trained to solve the CLLM tasks with three datasets with different features.

For all of datasets we used, memory duration is broadly distributed with longer mean for units in deeper layers. While memory durations are short and almost homogeneous among units in shallower layers, they are long and divers widely in deep layers. This difference is understood as a result of different contribution of the cross-entropy loss for different layers. While this gives stronger constraints for units in shallower layers. We can then conclude that multilayered architecture allows the LSTM network to store temporal correlations of given text inputs separately into units in different layers.

As suggested by above results, if the separate representation of temporal structure is the key to achieve high performance to solve sequential tasks, it may be possible to improve the LSTM networks by enhancing the temporal separation of layer structures. It will be a fascinating future subject, therefore, to propose a novel network structure and a learning algorithm for LSTM networks where acquired memory duration is suitably controlled by a learning algorithm itself to achieve higher performance for sequential tasks. It is also an interesting future work to study tem-



(a) First layer (b) Second layer (c) Third Layer

Figure 2: mean duration of memory variables of LSTM units in different layers. Red, blue and black lines represent duration of the first, second and third layer as functions of r_1, r_2, and r_3 (from the left to the right) respectively.

poral dynamics of gate variables in addition to the memory variables since temporal dynamics of memory variables are directly controlled by gate variables.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP16H01719, JP17K00338.

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[3] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[4] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

[5] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[6] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[7] Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075, 2015.

[8] Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in neural information processing systems*, pages 190–198, 2013.

[9] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[10] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.