DLLAB WEEK 10 Pre-Report : Long Short-Term Memory

Hyunsoo Cha[1], Kwanghoon Sohn[2]
[1]Yonsei University. [2]Yonsei University.

1 Introduction

Issue When attempting to store long-spaced information in an RNN, back-propagation takes a long time[3] and has caused gradient loss problems. It is a paper to improve this. In this paper, the researchers devised a method called LSTM in a 1991 paper by Hochreiter[2]. LSTM shows that special units allow for stable error control for more than a thousand time steps. It learns to open and close error flows through multiple gate units. The complexity is time step and $O(1)$ per weight. They created artificial data to include peripheral features that present noise patterns based on discerning and practicality, and confirmed that it learns faster than conventional methods. LSTM was able to solve long-time tasks that were complex and artificial, which was not solved by conventional RNNs. RNNs are the first neural networks proposed in the 1980s, designed to solve problems that ANNs failed to solve time series problems. RNN presents a method for pre-predicting the following characteristics in chronological order by introducing time step. Internal memory exists, enabling sequence-type input processing. This network is useful for sequence data processing such as handwriting recognition and speech recognition.

2 Constant Error Backprop

2.1 Recurrent Neural Network

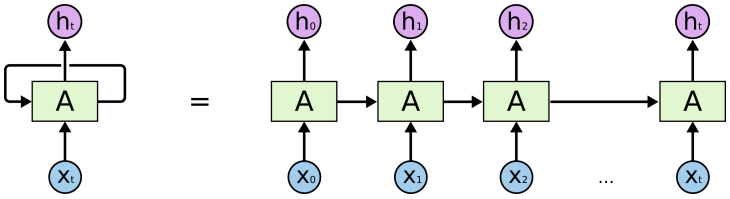

Figure 1: Basic diagram of RNN

Brief summary of RNN[1] The Recurrent Neural Network can be displayed as shown in the figure above. The network is solved through a time series structure. The feature point is that the output appears as the input of the next step. First, x_0 input enters the RNN network, and then h_0 enters the output. They see that this value and x_1 are re-entered into the RNN and iterated in such a way that h_1 is output to the new output. More specifically, x input and weight w_x will result in values through matmul operations, and h_{prev} and weight w_h will result in values through matmul operations. After adding two values, bias b, a value of h_{next} is derived through tanh activation function. It can be seen that this value and the input value of the next time sequence become the new x, h_{prev}, which results in a continuous computation.

Long-Term Dependency Difficulties exist in learning long-term dependency relationships of time series data, which is called long-term dependencies. The reason for this difficulty is the loss or divergence of gradients, and for the loss, research has been conducted by adding gates to RNNs through subsequent studies. The upper bound of gradient is also introduced to solve the divergence. To summarize this problem, gradient is lost in Back-Propagation Through Time. This solution becomes LSTM, which consists of special units. The reason for the slope problem can be seen from the following formula[2].

$$\left| f'_{l_m}(net_{l_m}(t-m))\omega_{l_m l_{m-1}} \right| \tag{1}$$

If above function is greater than 1.0, learning becomes unstable and weight vibrates. In other words, the slope emits. If this value is less than 1.0, gradient is lost. In other words, errors are lost and nothing is learned at acceptable time.

2.2 Constant Error Carrousel

About CEC With only a single connection, they can discuss how to implement constant error flow without error signal. The following formula is established for the f_j linear function.

$$y_j(t+1) = f_j(net_j(t+1)) = f_j(\omega_{jj} y^j(t)) = y^j(t) \tag{2}$$

CEC is a key principle of LSTM. Conflict can be caused by input weight and output weight. This conflict is often not occurring in long-time lag, but in short-time lag.

3 Long Short-Term Memory

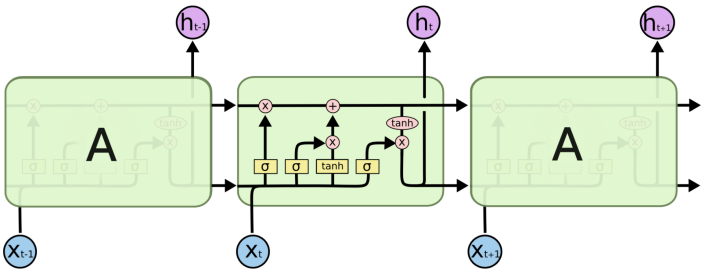

Figure 2: Long Short-Term Memory, abbreviated as LSTM. Brief structure[1] with functions.

Memory Cell The LSTM structure adds one more memory cell. The structure of the detailed memory cell is shown in the figure above. They want to summarize through this illustration because it is a different or more understandable illustration from the actual paper. A self-recurrent connection has a weight of 1.0 and a delay time step of 1. This is the basic structure of constant error carousel, CEC. Cell state is a mechanism that continues to flow. The algorithm proceeds in such a way that it is added to flow through the gate mechanism. First, let's find out about forget gate. The input to the forget gate is as follows.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

That is, they multiply the previous input and output by weight and add bias to obtain f_t through the activation function. This value refers to information that determines whether it is discarded or added to the cell state. Activation function plays a role in lowering dimensions. Input gate is represented by the following formula:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5}$$

The input gate, like the forget gate, determines whether the value is discarded or not in the cell state. These operations are possible through element-wise product. $\tilde{C}_t$ is a candidate value for the cell state, which determines how much value is added to the cell state. The cell state then undergoes an update process. Define a new cell state by multiplying the input gate and current gate by the value of the forget gate and the prior gate. Expressing this in a formula is as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6}$$

Output gate is a hidden state, a function that determines how many values the newly created cell state will be extracted. Expressing this in a formula is as follows:

$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0) \tag{7}$$
$$h_t = o_t * \tanh(C_t) \tag{8}$$

All processes so far can be summarized as determining how to input and extract values into the cell state using the current input and previous output.

4 Experiments

First Problem The first problem is the embedded Reber grammar problem. This is not a long-time lag issue, and in step nine, short-time lag is resolved through training sequences. In the transition diagram of the embedded Reber grammar, the symbol string passes through the Reber grammar, adding 11 new symbols. If a choice is made, it is made through a random choice with half the probability. In a time sequence, it is a network that predicts which symbols are the following. To accurately predict the next symbol, they must remember the second symbol.

method	hidden units	# weights	learning rate	% of success	success after
RTRL	3	≈ 170	0.05	"some fraction"	173,000
RTRL	12	≈ 494	0.1	"some fraction"	25,000
ELM	15	≈ 435		0	>200,000
RCC	7-9	≈ 119-198		50	182,000
LSTM	4 blocks, size 1	264	0.1	100	39,740
LSTM	3 blocks, size 2	276	0.1	100	21,730
LSTM	3 blocks, size 2	276	0.2	97	14,060
LSTM	4 blocks, size 1	264	0.5	97	9,500
LSTM	3 blocks, size 2	276	0.5	100	8,440

Figure 3: Result table of first experiment.

As can be seen from the table above, they can see that LSTM has significant accuracy close to 100 compared to existing models, RTRL, ELM, and RCC.

Second Problem The second problem is noise-free and noise sequences. This problem predicts the next symbol via input symbolic sequences for $a_1, \cdots, a_{p-1}, a_p = x, \quad a_{p+1} = y$. It consists of two sequences, a training set, to emphasize that it is a long-time lag problem. Expressing the second problem in the formula is as follows. $(y, a_1, a_2, \cdots, a_{p-1}, y)$, $(x, a_1, a_2, \cdots, a_{p-1}, x)$. The probability of choosing this is 0.5. To get the last element, they learn to store the first element representation for p time step. The resulting table is as follows:

Method	Delay p	Learning rate	# weights	% Successful trials	Success after
RTRL	4	1.0	36	78	1,043,000
RTRL	4	4.0	36	56	892,000
RTRL	4	10.0	36	22	254,000
RTRL	10	1.0-10.0	144	0	> 5,000,000
RTRL	100	1.0-10.0	10404	0	> 5,000,000
BPTT	100	1.0-10.0	10404	0	> 5,000,000
CH	100	1.0	10506	33	32,400
LSTM	100	1.0	10504	100	5,040

Figure 4: Result table of second experiment.

While delay p shows around 50% accuracy in RTRL and shows significant learning results, they show zero learning results when delay p is greater than 10. For the CH model, they show an accuracy of 33%, but they can see that LSTM shows an accuracy of 100. The results of experiments with LSTM only for more complex problems are shown in the following table.

q (time lag -1)	p (# random inputs)	$\frac{q}{p}$	# weights	Success after
50	50	1	364	30,000
100	100	1	664	31,000
200	200	1	1264	33,000
500	500	1	3064	38,000
1,000	1,000	1	6064	49,000
1,000	500	2	3064	49,000
1,000	200	5	1264	75,000
1,000	100	10	664	135,000
1,000	50	20	364	203,000

Figure 5: Table of results of experiments with LSTM only.

Applying LSTM for long minimal time lag $q+1$ yields results as shown in the table above. The table shows that success increases proportionally as the value of q/p increases.

Third Problem The third experiment is the noise and signal on the same channel problem. The table below shows more specific results. It tells the algorithm of how LSTM solves a two-sequence problem if noise and signals are properly mixed and entered the input. In this problem, there are two classes, and the probability is 0.5.

T	N	stop	# weights	fraction misclassified	av. difference to mean
100	3	269,650	102	0.00558	0.014
100	1	565,640	102	0.00441	0.012

Figure 6: Table of results of 2-sequence problem.

The third experiment consisted of three stages. T is the minimum sequence length, and N defines an information-submissive element as the number at which the sequence begins. In this case, it was a simple task, which could be solved faster through random weight guidance. For the above table, the results are made with noisy real-valued targets. In the case of this task, it can be seen that random weight guidance did not solve it easily.

Fourth Problem The fourth experiment is the adding problem. Adding problem is an experiment that shows that problems that cannot be solved by existing RNN family of algorithms have been solved through long-term dependency problems.

T	minimal lag	# weights	# wrong predictions	Success after
100	50	93	1 out of 2560	74,000
500	250	93	0 out of 2560	209,000
1000	500	93	1 out of 2560	853,000

Figure 7: Table of results of adding problem.

The researchers used three layers through two input units for the fourth experiment. It also implements one output unit and two cell blocks in size 2. The output layer can only be connected to memory cells. The input layer consists of forward connection and non-input units consist of bias weight. Parameters of this structure can be easily stored as a two-input signal. For a test set containing 2560 sequences, the number of false predictions can be determined. All values were obtained as an average for 10 attempts.

Fifth Problem The fifth experiment was to test whether the problem could be solved with a multiplication problem and, unlike the fourth, with a non-integrative solution. The fifth experiment confirmed that non-integrative information processing was possible.

Sixth Problem The sixth experiment is a temporary order, which looks into whether LSTM can extract the information passed about the order of the time sequence that is widely separated. Existing RNN algorithms were able to solve problems that could not be solved.

5 Limitation of LSTM

There are four limitations to LSTM. The first is that it does not respond effectively to strongly delayed XOR problems. The second problem is that each block of memory cells requires two units, which represent the input and output gates. However, compared to the existing recurrent net, the number of weights did not increase by nine times. The third problem is that LSTM fails to run a network that feeds the entire input column at a time because CEC has a constant error flow within the memory cell. The fourth problem is that LSTM does not solve problems properly, including the concept of recency.

6 Conclusion

Backprop prevents leakage of memory cells within the internal structure so that error flow can continue through CEC. This led to solving the problem of long delays. Proposed as future research suggested time sequences prediction, music composition, and speech processing. In fact, LSTM is used in a variety of fields, such as being used for seq2seq and also for stock price forecasting. Especially in natural language processing, LSTM is used as the basis for classical literature.

7 Reference

[1] colah. Understanding lstm networks, Aug 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.