Long Short-Term Memory 논문 리뷰

(Sepp Hochreiter, Jürgen Schmidhuber, Neural Computation, 1997)

NLP12 초급반 송석리
(greatsong21@gmail.com)

01

논문의 개요

01-1. 논문과 저자 소개

* 출처 : https://brunch.co.kr/@hvnpoet/55 (야만인 블로그)

01-1. 논문과 저자 소개

Google 학술검색

Juergen Schmidhuber

팔로우

The Swiss AI Lab IDSIA / USI & SUPSI
idsia.ch의 이메일 확인됨 - 홈페이지

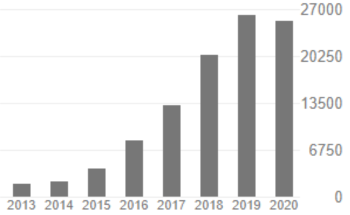
computer science   artificial intelligence   reinforcement learning   neural networks   physics

제목	인용	연도
Long short-term memory S Hochreiter, J Schmidhuber Neural computation 9 (8), 1735-1780	40530	1997
Deep learning in neural networks: An overview J Schmidhuber Neural networks 61, 85-117	10839	2015
Multi-column deep neural networks for image classification D Ciregan, U Meier, J Schmidhuber 2012 IEEE conference on computer vision and pattern recognition, 3642-3649	3609	2012
Learning to forget: Continual prediction with LSTM FA Gers, J Schmidhuber, F Cummins IET Digital Library	3605	1999
LSTM: A search space odyssey K Greff, RK Srivastava, J Koutník, BR Steunebrink, J Schmidhuber IEEE transactions on neural networks and learning systems 28 (10), 2222-2232	2806	2016
Framewise phoneme classification with bidirectional LSTM and other neural network architectures A Graves, J Schmidhuber Neural networks 18 (5-6), 602-610	2739	2005
Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks A Graves, S Fernández, F Gomez, J Schmidhuber Proceedings of the 23rd international conference on Machine learning, 369-376	2643	2006
A novel connectionist system for unconstrained handwriting recognition A Graves, M Liwicki, S Fernández, R Bertolami, H Bunke, J Schmidhuber IEEE transactions on pattern analysis and machine intelligence 31 (5), 855-868	1662	2008
Highway networks RK Srivastava, K Greff, J Schmidhuber arXiv preprint arXiv:1505.00387	1376	2015
Gradient flow in recurrent nets: the difficulty of learning long-term dependencies S Hochreiter, Y Bengio, P Frasconi, J Schmidhuber A field guide to dynamical recurrent neural networks. IEEE Press	1356	2001

내 프로필 만들기

인용 모두 보기

	전체	2015년 이후
서지정보	112341	97973
h-index	101	80
i10-index	358	237



공동 저자 모두 보기

- Sepp Hochreiter — Institute for Machine Learning, J...
- Dan Ciresan — Conndera Research
- Faustino Gomez — CEO NNAISENSE SA. Former S...
- Alex Graves — University of Toronto
- luca maria gambardella — IDSIA Istituto Dalle Molle for Artifi...
- Felix Gers — Professor of Computer Science, ...
- Daan Wierstra — Principal Scientist, DeepMind
- Jonathan Masci — NNAISENSE SA

01-1. 논문과 저자 소개

Google 학술검색

Juergen Schmidhuber

팔로우

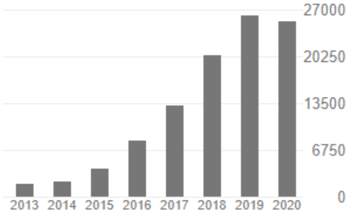
The Swiss AI Lab IDSIA / USI & SUPSI
idsia.ch의 이메일 확인됨 - 홈페이지
computer science artificial intelligence reinforcement learning neural networks physics

제목	인용	연도
Long short-term memory S Hochreiter, J Schmidhuber Neural computation 9 (8), 1735-1780	40530	1997
Deep learning in neural networks: An overview J Schmidhuber Neural networks 61, 85-117	10839	2015
Multi-column deep neural networks for image classification D Ciregan, U Meier, J Schmidhuber 2012 IEEE conference on computer vision and pattern recognition, 3642-3649	3609	2012
Learning to forget: Continual prediction with LSTM FA Gers, J Schmidhuber, F Cummins IET Digital Library	3605	1999
LSTM: A search space odyssey K Greff, RK Srivastava, J Koutník, BR Steunebrink, J Schmidhuber IEEE transactions on neural networks and learning systems 28 (10), 2222-2232	2806	2016
Framewise phoneme classification with bidirectional LSTM and other neural network architectures A Graves, J Schmidhuber Neural networks 18 (5-6), 602-610	2739	2005
Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks A Graves, S Fernández, F Gomez, J Schmidhuber Proceedings of the 23rd international conference on Machine learning, 369-376	2643	2006
A novel connectionist system for unconstrained handwriting recognition A Graves, M Liwicki, S Fernández, R Bertolami, H Bunke, J Schmidhuber IEEE transactions on pattern analysis and machine intelligence 31 (5), 855-868	1662	2008
Highway networks RK Srivastava, K Greff, J Schmidhuber arXiv preprint arXiv:1505.00387	1376	2015
Gradient flow in recurrent nets: the difficulty of learning long-term dependencies S Hochreiter, Y Bengio, P Frasconi, J Schmidhuber A field guide to dynamical recurrent neural networks. IEEE Press	1356	2001

내 프로필 만들기

인용 모두 보기

	전체	2015년 이후
서지정보	112341	97973
h-index	101	80
i10-index	358	237



공동 저자 모두 보기

- Sepp Hochreiter — Institute for Machine Learning, J...
- Dan Ciresan — Conndera Research
- Faustino Gomez — CEO NNAISENSE SA. Former S...
- Alex Graves — University of Toronto
- luca maria gambardella — IDSIA Istituto Dalle Molle for Artifi...
- Felix Gers — Professor of Computer Science, ...
- Daan Wierstra — Principal Scientist, DeepMind
- Jonathan Masci — NNAISENSE SA

01-1. 논문과 저자 소개

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

RNN에서 긴 간격의 정보를 저장하려면 역전파가 시간도 오래 걸리고, 기울기 소실 문제가 발생함.

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

호흐라이터의 1991년 연구를 참고해 LSTM이라고 불리우는 방법을 고안함.

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

LSTM은 특별한 유닛을 통해 1000개 이상의 타임 스텝에서도 안정적으로 오류를 제어할 수 있음.

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

여러 개의 게이트 유닛으로 에러 흐름을 열고 닫도록 학습함. 복잡도는 타임 스텝과 가중치 당 O(1)임.

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

적절히 가공된 데이터로 실험을 했는데, 기존의 방법과 비교했을 때 더 성공적이며 빠름.

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

01-1. 논문과 저자 소개

LSTM은 또한 기존의 RNN으로 해결할 수 없었던 시간 지연 문제를 해결하였음.

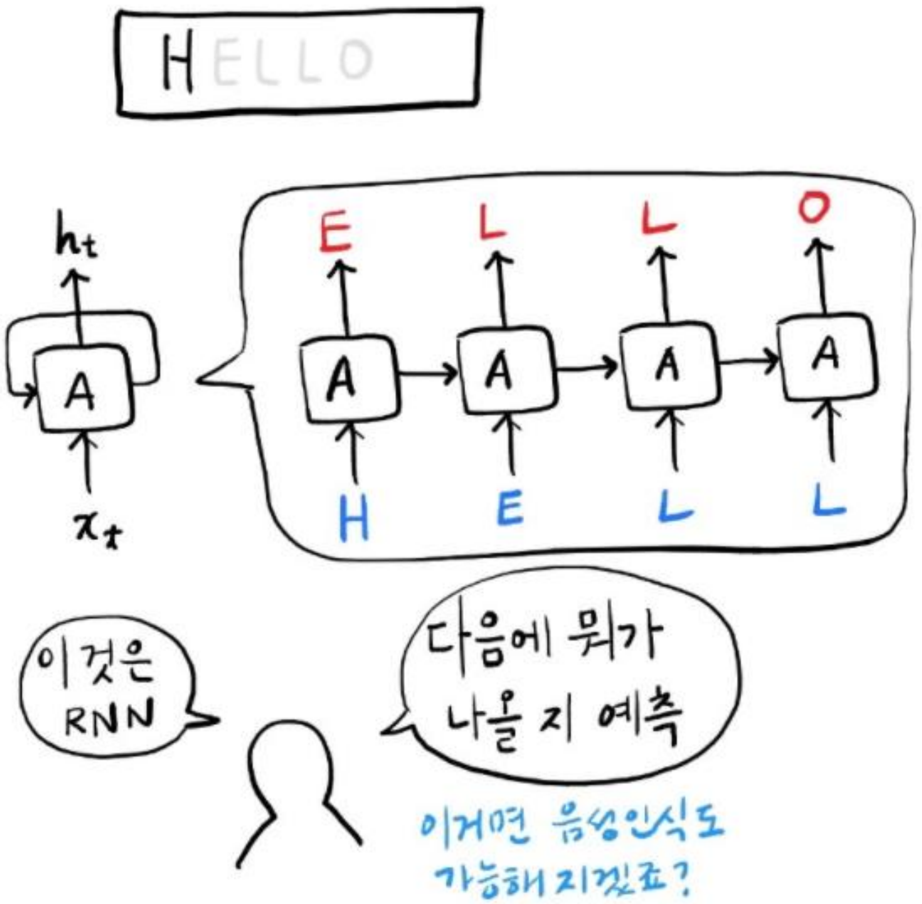
Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

02

Recurrent Neural Network

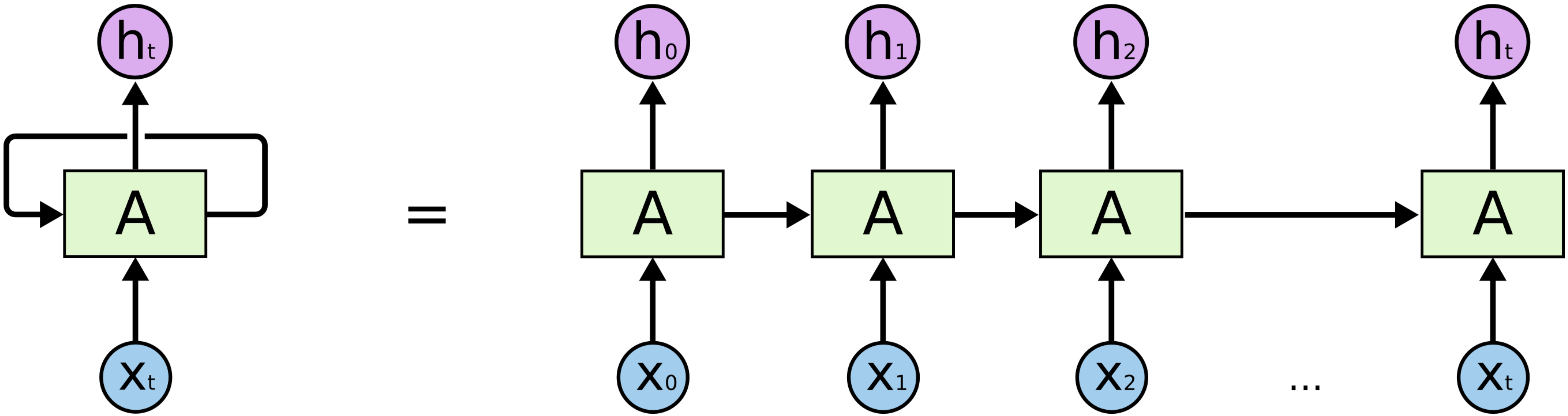
RNN의 개요



- 1980년대 처음 개념 제안됨
- ANN으로 시계열 문제 해결 어려움 해결
- 내부의 메모리를 이용해 시퀀스 형태의 입력 처리 가능해짐
- 필기 인식, 음성 인식 등 시퀀스 데이터 처리에 적용할 수 있음

- 그림 출처 : https://brunch.co.kr/@hvnpoet/55 (야만인 블로그)
- 내용 출처 : 위키피디아

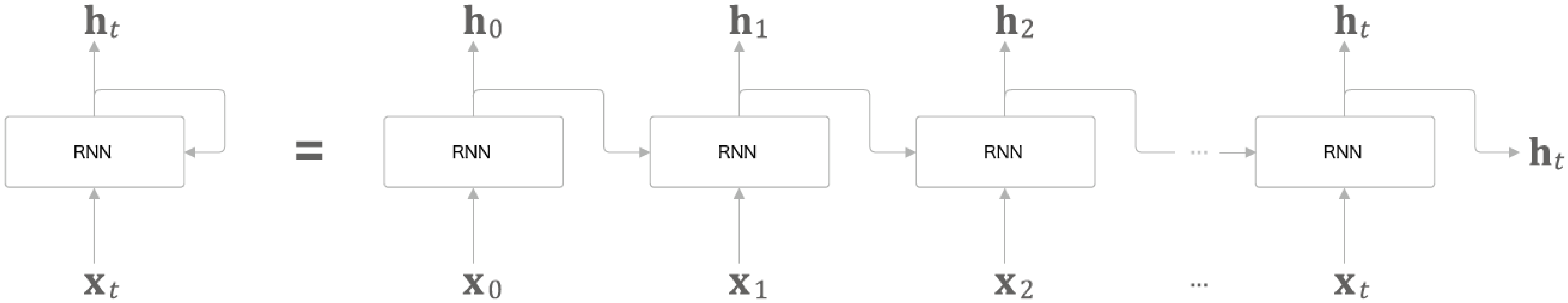
02-2. RNN의 기본 구조(펼친 모습)



- 그림 출처 : https://colah.github.io/posts/2015-08-Understanding-LSTMs/

02-2. RNN의 기본 구조(펼친 모습)

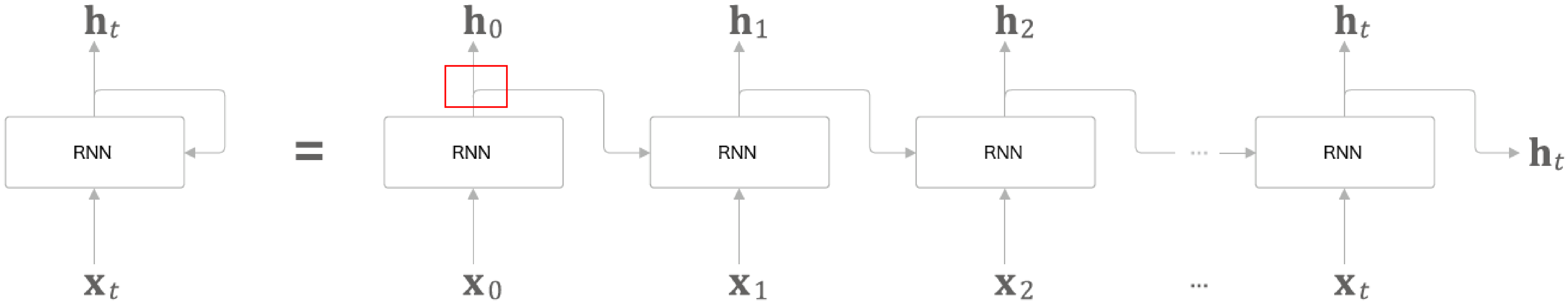
그림 5-8 RNN 계층의 순환 구조 펼치기



- 그림 출처 : 밑바닥부터 시작하는 딥러닝2(사이토 고키 저, 개앞맵시 역, 한빛미디어)

02-2. RNN의 기본 구조(펼친 모습)

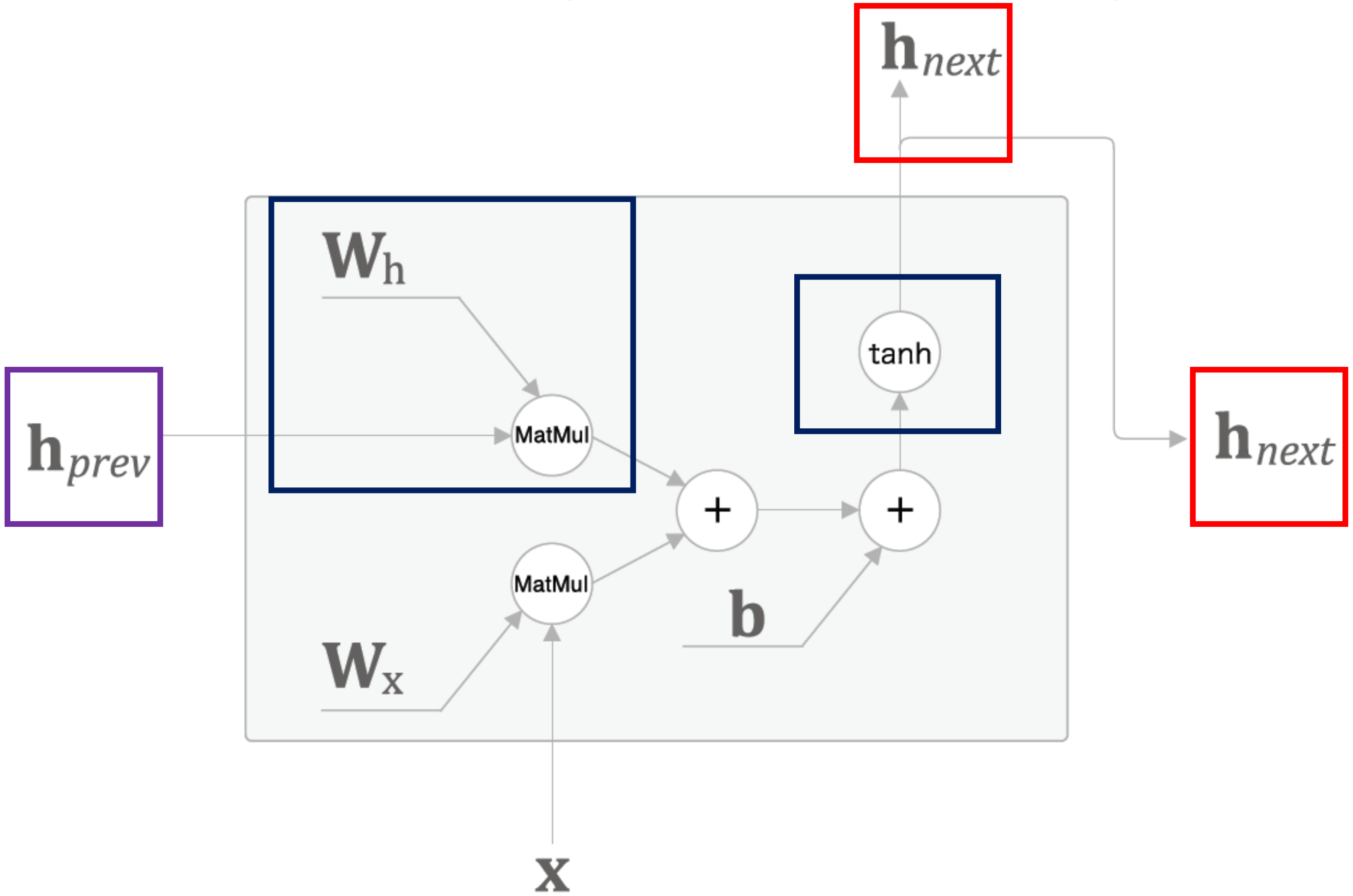
그림 5-8 RNN 계층의 순환 구조 펼치기



- 그림 출처 : 밑바닥부터 시작하는 딥러닝2(사이토 고키 저, 개앞맵시 역, 한빛미디어)

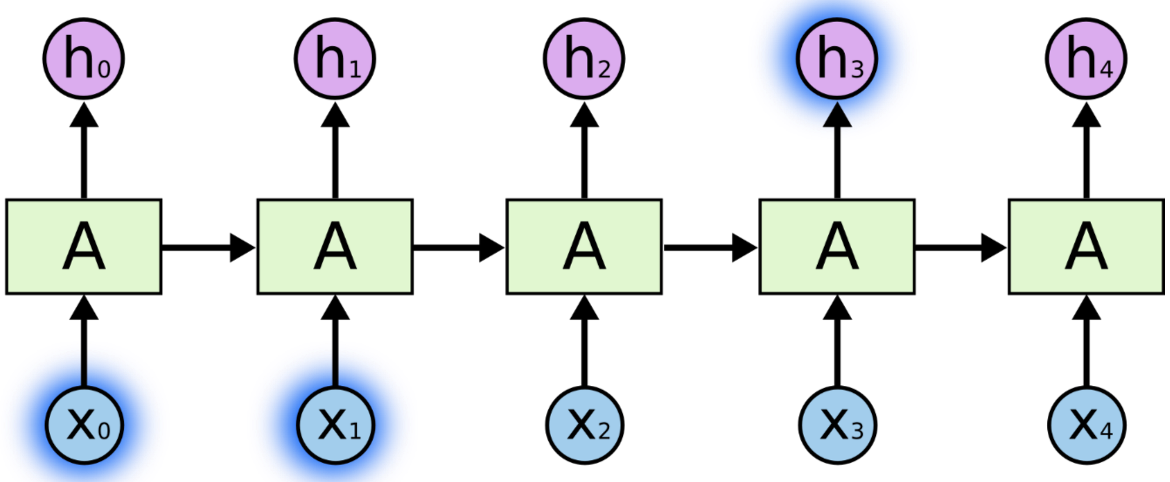
02-3. RNN의 기본 구조

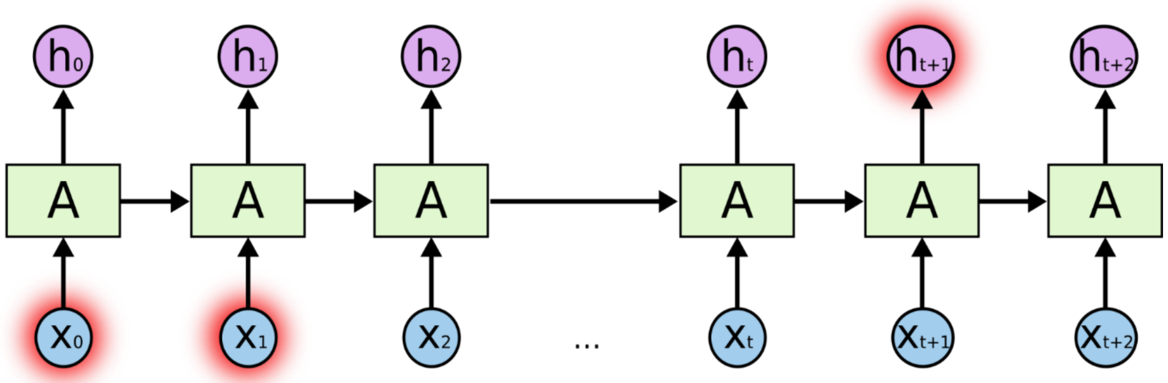
그림 5-19 RNN 계층의 계산 그래프(MatMul 노드는 행렬의 곱셈을 나타냄)



- 그림 출처 : 밑바닥부터 시작하는 딥러닝2(사이토 고키 저, 개앞맵시 역, 한빛미디어)

02-4. RNN의 문제점





- 시계열 데이터의 장기 의존 관계 학습 어려움
  - Long-Term Dependencies
- 기울기 소실 또는 기울기 폭발 때문
- 기울기 폭발 해결책 : 기울기 최대값 고정 등
- 기울기 소실 해결책 : RNN에 게이트 추가
  - 대표적 : LSTM, Gated Recurrent Unit

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

03

1 INTRODUCTION

03-1. 문제와 해결책

문제는 오차 역전파에서 기울기가 사라지는 것. 해결책은 특별한 유닛들로 구성된 LSTM.

The problem. With conventional "Back-Propagation Through Time" (BPTT, e.g., Williams and Zipser 1992, Werbos 1988) or "Real-Time Recurrent Learning" (RTRL, e.g., Robinson and Fallside 1987), error signals "flowing backwards in time" tend to either (1) blow up or (2) vanish: the temporal evolution of the backpropagated error exponentially depends on the size of the weights (Hochreiter 1991). Case (1) may lead to oscillating weights, while in case (2) learning to bridge long time lags takes a prohibitive amount of time, or does not work at all (see section 3).

The remedy. This paper presents *"Long Short-Term Memory"* (LSTM), a novel recurrent network architecture in conjunction with an appropriate gradient-based learning algorithm. LSTM is designed to overcome these error back-flow problems. It can learn to bridge time intervals in excess of 1000 steps even in case of noisy, incompressible input sequences, without loss of short time lag capabilities. This is achieved by an efficient, gradient-based algorithm for an architecture enforcing *constant* (thus neither exploding nor vanishing) error flow through internal states of special units (provided the gradient computation is truncated at certain architecture-specific points — this does not affect long-term error flow though).

03-2. 논문의 흐름

섹션 2: 이전 연구 리뷰. 섹션 3: 호흐라이터의 연구를 참고하여 오차 소멸 문제 분석.
섹션 4: LSTM 구조에 대한 소개. 섹션 5 : 다양한 실험을 통한 기존 방법과의 비교.
섹션 6: LSTM의 한계와 장점. 부록: 알고리즘에 대한 자세한 소개

Outline of paper. Section 2 will briefly review previous work. Section 3 begins with an outline of the detailed analysis of vanishing errors due to Hochreiter (1991). It will then introduce a naive approach to constant error backprop for didactic purposes, and highlight its problems concerning information storage and retrieval. These problems will lead to the LSTM architecture as described in Section 4. Section 5 will present numerous experiments and comparisons with competing methods. LSTM outperforms them, and also learns to solve complex, artificial tasks no other recurrent net algorithm has solved. Section 6 will discuss LSTM's limitations and advantages. The appendix contains a detailed description of the algorithm (A.1), and explicit error flow formulae (A.2).

04

2 PREVIOUS WORK

04.

기존의 연구들

LSTM이 풀고자하는 문제와 관련된 이전 연구들 소개. 논문 곳곳에서 활용됨.

Gradient-descent variants. The approaches of Elman (1988), Fahlman (1991), Williams (1989), Schmidhuber (1992a), Pearlmutter (1989), and many of the related algorithms in Pearlmutter's comprehensive overview (1995) suffer from the same problems as BPTT and RTRL (see Sections 1 and 3).

Time-delays. Other methods that seem practical for short time lags only are Time-Delay Neural Networks (Lang et al. 1990) and Plate's method (Plate 1993), which updates unit activations based on a weighted sum of old activations (see also de Vries and Principe 1991). Lin et al. (1995) propose variants of time-delay networks called NARX networks.

Time constants. To deal with long time lags, Mozer (1992) uses time constants influencing changes of unit activations (deVries and Principe's above-mentioned approach (1991) may in fact be viewed as a mixture of TDNN and time constants). For long time lags, however, the time constants need external fine tuning (Mozer 1992). Sun et al.'s alternative approach (1993) updates the activation of a recurrent unit by adding the old activation and the (scaled) current net input. The net input, however, tends to perturb the stored information, which makes long-term storage impractical.

Ring's approach. Ring (1993) also proposed a method for bridging long time lags. Whenever a unit in his network receives conflicting error signals, he adds a higher order unit influencing appropriate connections. Although his approach can sometimes be extremely fast, to bridge a time lag involving 100 steps may require the addition of 100 units. Also, Ring's net does not generalize to unseen lag durations.

Bengio et al.'s approaches. Bengio et al. (1994) investigate methods such as simulated annealing, multi-grid random search, time-weighted pseudo-Newton optimization, and discrete error propagation. Their "latch" and "2-sequence" problems are very similar to problem 3a with minimal time lag 100 (see Experiment 3). Bengio and Frasconi (1994) also propose an EM approach for propagating targets. With n so-called "state networks", at a given time, their system can be in one of only n different states. See also beginning of Section 5. But to solve continuous problems such as the "adding problem" (Section 5.4), their system would require an unacceptable number of states (i.e., state networks).

Kalman filters. Puskorius and Feldkamp (1994) use Kalman filter techniques to improve recurrent net performance. Since they use "a derivative discount factor imposed to decay exponentially the effects of past dynamic derivatives," there is no reason to believe that their Kalman Filter Trained Recurrent Networks will be useful for very long minimal time lags.

Second order nets. We will see that LSTM uses multiplicative units (MUs) to protect error flow from unwanted perturbations. It is not the first recurrent net method using MUs though. For instance, Watrous and Kuhn (1992) use MUs in second order nets. Some differences to LSTM are: (1) Watrous and Kuhn's architecture does not enforce constant error flow and is not designed to solve long time lag problems. (2) It has fully connected second-order sigma-pi units, while the LSTM architecture's MUs are used only to gate access to constant error flow. (3) Watrous and Kuhn's algorithm costs $O(W^2)$ operations per time step, ours only $O(W)$, where W is the number of weights. See also Miller and Giles (1993) for additional work on MUs.

Simple weight guessing. To avoid long time lag problems of gradient-based approaches we may simply randomly initialize all network weights until the resulting net happens to classify all training sequences correctly. In fact, recently we discovered (Schmidhuber and Hochreiter 1996, Hochreiter and Schmidhuber 1996, 1997) that simple weight guessing solves many of the problems in (Bengio 1994, Bengio and Frasconi 1994, Miller and Giles 1993, Lin et al. 1995) faster than the algorithms proposed therein. This does not mean that weight guessing is a good algorithm. It just means that the problems are very simple. More realistic tasks require either many free parameters (e.g., input weights) or high weight precision (e.g., for continuous-valued parameters), such that guessing becomes completely infeasible.

Adaptive sequence chunkers. Schmidhuber's hierarchical chunker systems (1992b, 1993) *do* have a capability to bridge arbitrary time lags, but only if there is local predictability across the subsequences causing the time lags (see also Mozer 1992). For instance, in his postdoctoral thesis (1993), Schmidhuber uses hierarchical recurrent nets to rapidly solve certain grammar learning tasks involving minimal time lags in excess of 1000 steps. The performance of chunker systems, however, deteriorates as the noise level increases and the input sequences become less compressible. LSTM does not suffer from this problem.

05

3 CONSTANT ERROR BACKPROP

05. 3 CONSTANT ERROR BACKPROP

RNN에서 사용하는 BPTT(Backpropagation Through Time)에서 기울기 소실 / 폭발 문제가 발생하는 이유

With $l_q = v$ and $l_0 = u$, we obtain:

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \sum_{l_1=1}^{n} \cdots \sum_{l_{q-1}=1}^{n} \prod_{m=1}^{q} f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}} \tag{2}$$

(proof by induction). The sum of the n^{q-1} terms $\prod_{m=1}^{q} f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}}$ determines the total error back flow (note that since the summation terms may have different signs, increasing the number of units n does not necessarily increase error flow).

Intuitive explanation of equation (2). If

$$|f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}}| > 1.0$$

for all m (as can happen, e.g., with linear f_{l_m}) then the largest product increases exponentially with q. That is, the error blows up, and conflicting error signals arriving at unit v can lead to oscillating weights and unstable learning (for error blow-ups or bifurcations see also Pineda 1988, Baldi and Pineda 1991, Doya 1992). On the other hand, if

$$|f'_{l_m}(net_{l_m}(t-m)) w_{l_m l_{m-1}}| < 1.0$$

for all m, then the largest product *decreases* exponentially with q. That is, the error vanishes, and nothing can be learned in acceptable time.

05. 3 CONSTANT ERROR BACKPROP

LSTM의 핵심 아이디어인 CEC(Constant Error Carrousel) 소개

3.2 CONSTANT ERROR FLOW: NAIVE APPROACH

A single unit. To avoid vanishing error signals, how can we achieve constant error flow through a single unit j with a single connection to itself? According to the rules above, at time t, j's local error back flow is $\vartheta_j(t) = f_j'(net_j(t))\vartheta_j(t+1)w_{jj}$. To enforce *constant* error flow through j, we require

$$f_j'(net_j(t))w_{jj} = 1.0.$$

Note the similarity to Mozer's fixed time constant system (1992) — a time constant of 1.0 is appropriate for potentially infinite time lags[1].

The constant error carrousel. Integrating the differential equation above, we obtain $f_j(net_j(t)) = \frac{net_j(t)}{w_{jj}}$ for arbitrary $net_j(t)$. This means: f_j has to be linear, and unit j's activation has to remain constant:

$$y_j(t+1) = f_j(net_j(t+1)) = f_j(w_{jj}y^j(t)) = y^j(t).$$

In the experiments, this will be ensured by using the identity function $f_j : f_j(x) = x, \forall x$, and by setting $w_{jj} = 1.0$. We refer to this as the constant error carrousel (CEC). CEC will be LSTM's central feature (see Section 4).

Of course unit j will not only be connected to itself but also to other units. This invokes two obvious, related problems (also inherent in all other gradient-based approaches):

1. Input weight conflict: for simplicity, let us focus on a single additional input weight w_{ji}. Assume that the total error can be reduced by switching on unit j in response to a certain input, and keeping it active for a long time (until it helps to compute a desired output). Provided i is non-zero, since the same incoming weight has to be used for both storing certain inputs *and* ignoring others, w_{ji} will often receive conflicting weight update signals during this time (recall that j is linear): these signals will attempt to make w_{ji} participate in (1) storing the input (by switching on j) *and* (2) protecting the input (by preventing j from being switched off by irrelevant later inputs). This conflict makes learning difficult, and calls for a more context-sensitive mechanism for controlling "write operations" through input weights.

2. Output weight conflict: assume j is switched on and currently stores some previous input. For simplicity, let us focus on a single additional outgoing weight w_{kj}. The same w_{kj} has to be used for both retrieving j's content at certain times *and* preventing j from disturbing k at other times. As long as unit j is non-zero, w_{kj} will attract conflicting weight update signals generated during sequence processing: these signals will attempt to make w_{kj} participate in (1) accessing the information stored in j *and* — at different times — (2) protecting unit k from being perturbed by j. For instance, with many tasks there are certain "short time lag errors" that can be reduced in early training stages. However, at later training stages j may suddenly start to cause avoidable errors in situations that already seemed under control by attempting to participate in reducing more difficult "long time lag errors". Again, this conflict makes learning difficult, and calls for a more context-sensitive mechanism for controlling "read operations" through output weights.

Of course, input and output weight conflicts are not specific for long time lags, but occur for short time lags as well. Their effects, however, become particularly pronounced in the long time lag case: as the time lag increases, (1) stored information must be protected against perturbation for longer and longer periods, and — especially in advanced stages of learning — (2) more and more already correct outputs also require protection against perturbation.

Due to the problems above the naive approach does not work well except in case of certain simple problems involving local input/output representations and non-repeating input patterns (see Hochreiter 1991 and Silva et al. 1996). The next section shows how to do it right.

06

4 LONG SHORT-TERM MEMORY

06. 4 LONG SHORT-TERM MEMORY

LSTM 구조에 대한 설명 : 메모리 셀과 게이트 유닛

Memory cells and gate units. To construct an architecture that allows for constant error flow through special, self-connected units without the disadvantages of the naive approach, we extend the constant error carrousel CEC embodied by the self-connected, linear unit j from Section 3.2 by introducing additional features. A multiplicative *input gate unit* is introduced to protect the memory contents stored in j from perturbation by irrelevant inputs. Likewise, a multiplicative *output gate unit* is introduced which protects other units from perturbation by currently irrelevant memory contents stored in j.

The resulting, more complex unit is called a *memory cell* (see Figure 1). The j-th memory cell is denoted c_j. Each memory cell is built around a central linear unit with a fixed self-connection (the CEC). In addition to net_{c_j}, c_j gets input from a multiplicative unit out_j (the "output gate"), and from another multiplicative unit in_j (the "input gate"). in_j's activation at time t is denoted by $y^{in_j}(t)$, out_j's by $y^{out_j}(t)$. We have

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)); y^{in_j}(t) = f_{in_j}(net_{in_j}(t));$$

where

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1),$$

and

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1).$$

We also have

$$net_{c_j}(t) = \sum_u w_{c_j u} y^u(t-1).$$

The summation indices u may stand for input units, gate units, memory cells, or even conventional hidden units if there are any (see also paragraph on "network topology" below). All these different types of units may convey useful information about the current state of the net. For instance, an input gate (output gate) may use inputs from other memory cells to decide whether to store (access) certain information in its memory cell. There even may be recurrent self-connections like $w_{c_j c_j}$. It is up to the user to define the network topology. See Figure 2 for an example.

At time t, c_j's output $y^{c_j}(t)$ is computed as

$$y^{c_j}(t) = y^{out_j}(t)h(s_{c_j}(t)),$$

where the "internal state" $s_{c_j}(t)$ is

$$s_{c_j}(0) = 0, s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t)g\left(net_{c_j}(t)\right) \text{ for } t > 0.$$

The differentiable function g squashes net_{c_j}; the differentiable function h scales memory cell outputs computed from the internal state s_{c_j}.

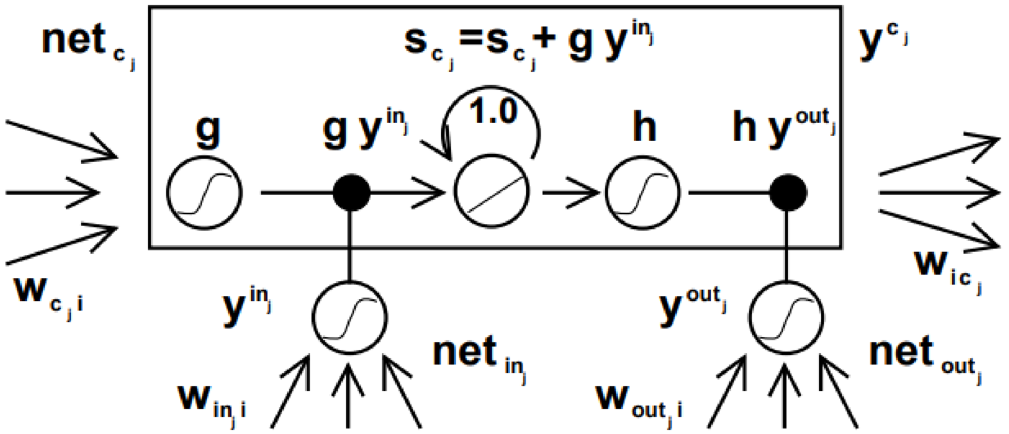
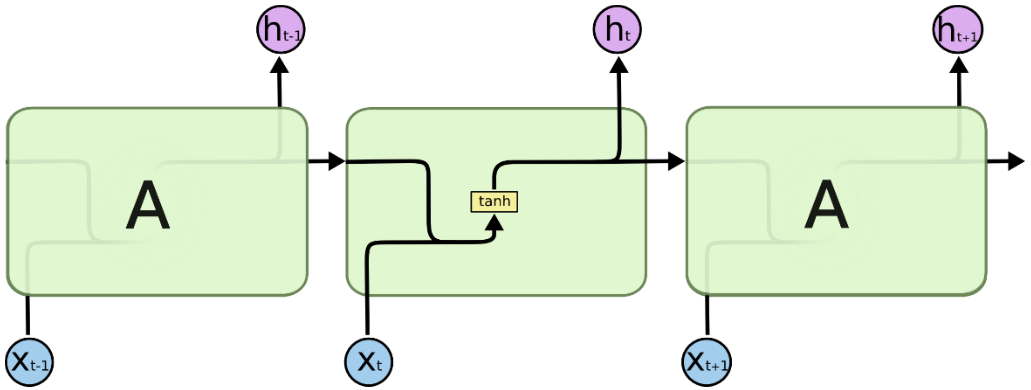
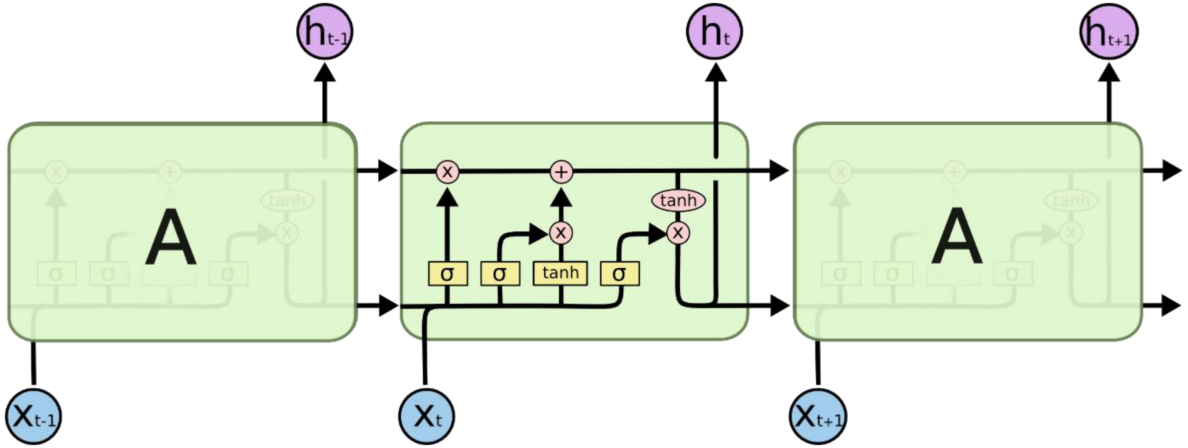


Figure 1: *Architecture of memory cell c_j (the box) and its gate units in_j, out_j. The self-recurrent connection (with weight 1.0) indicates feedback with a delay of 1 time step. It builds the basis of the "constant error carrousel" CEC. The gate units open and close access to CEC. See text and appendix A.1 for details.*

06. 4 LONG SHORT-TERM MEMORY



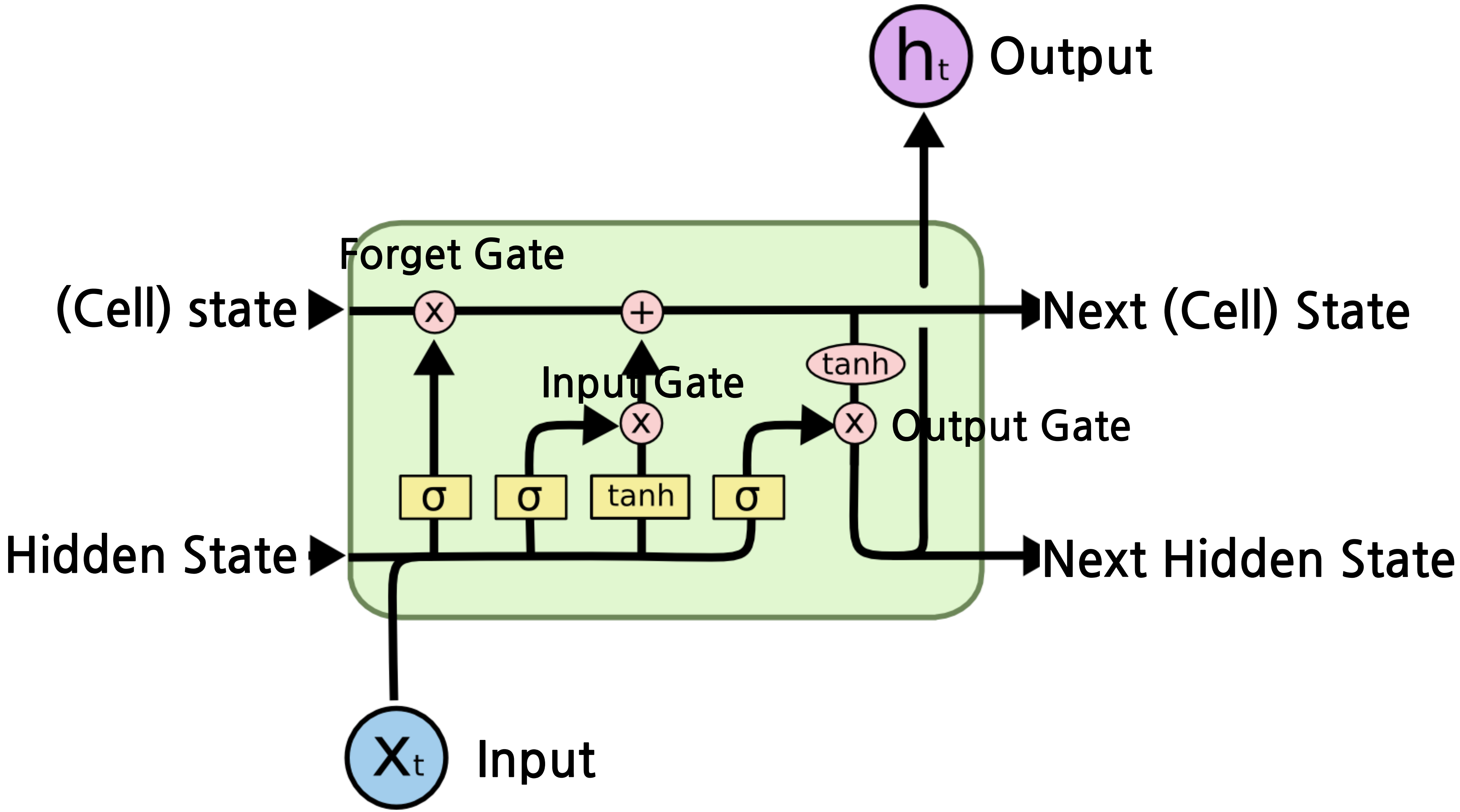
RNN



LSTM

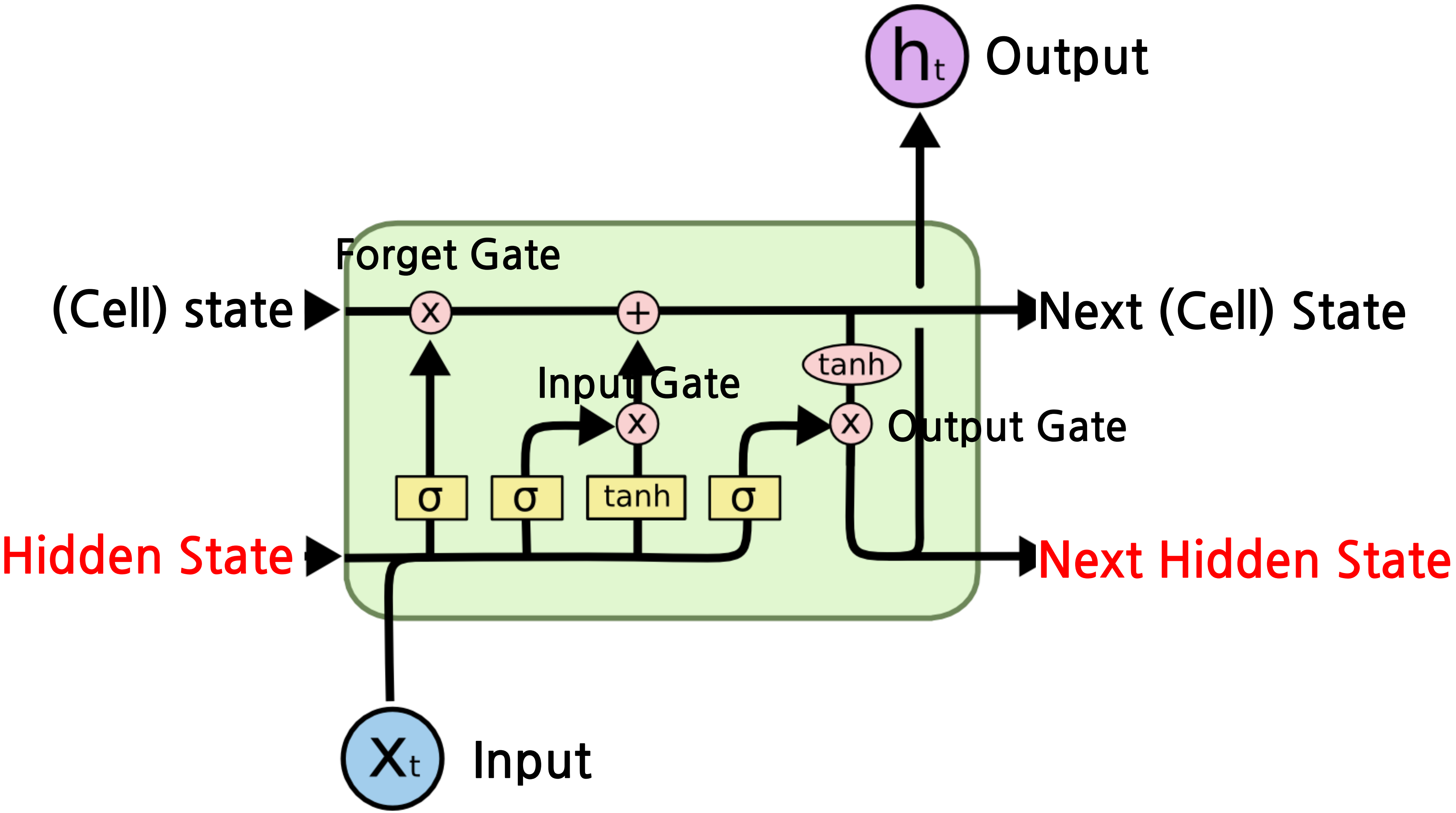
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

06. 4 LONG SHORT-TERM MEMORY



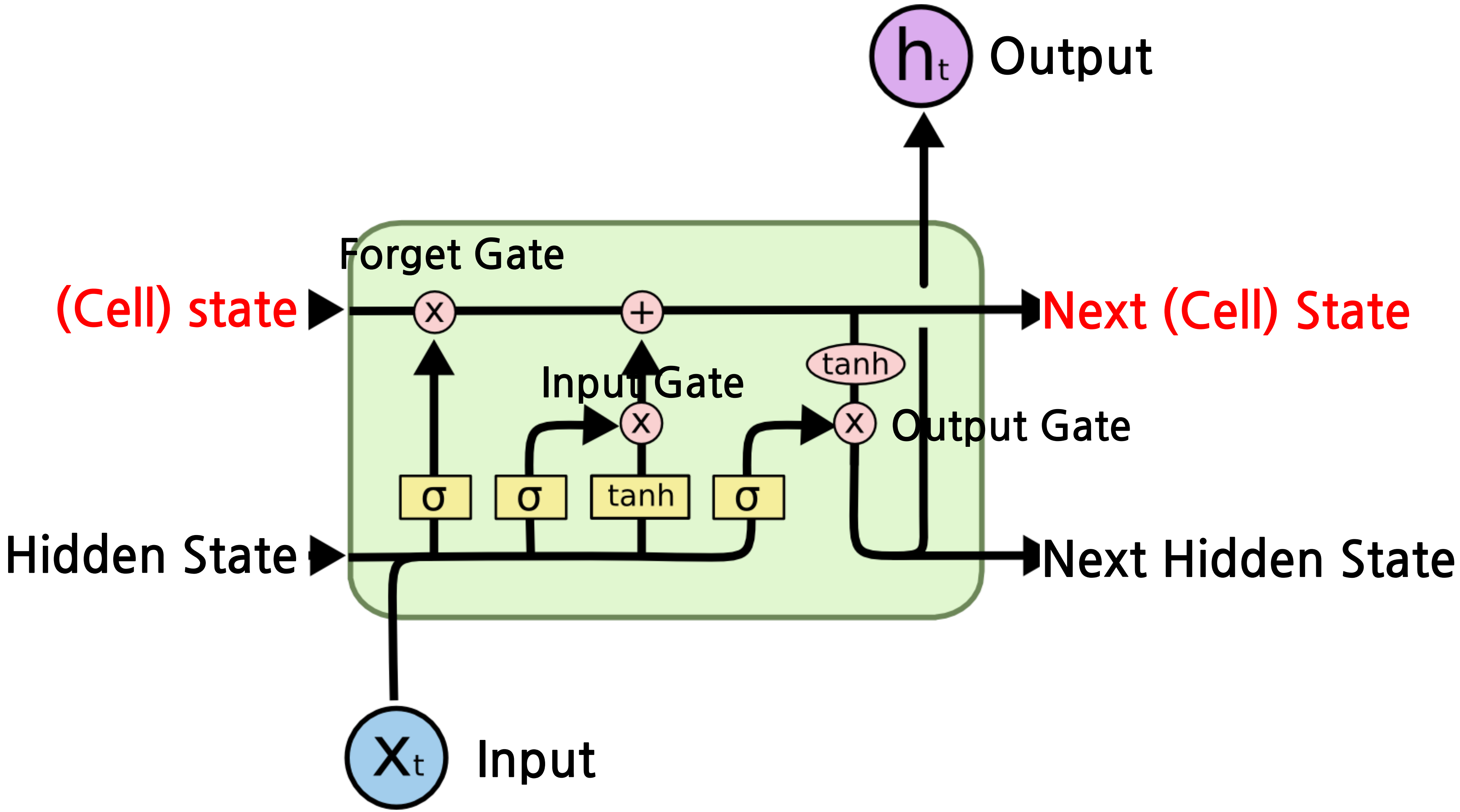
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

06. 4 LONG SHORT-TERM MEMORY



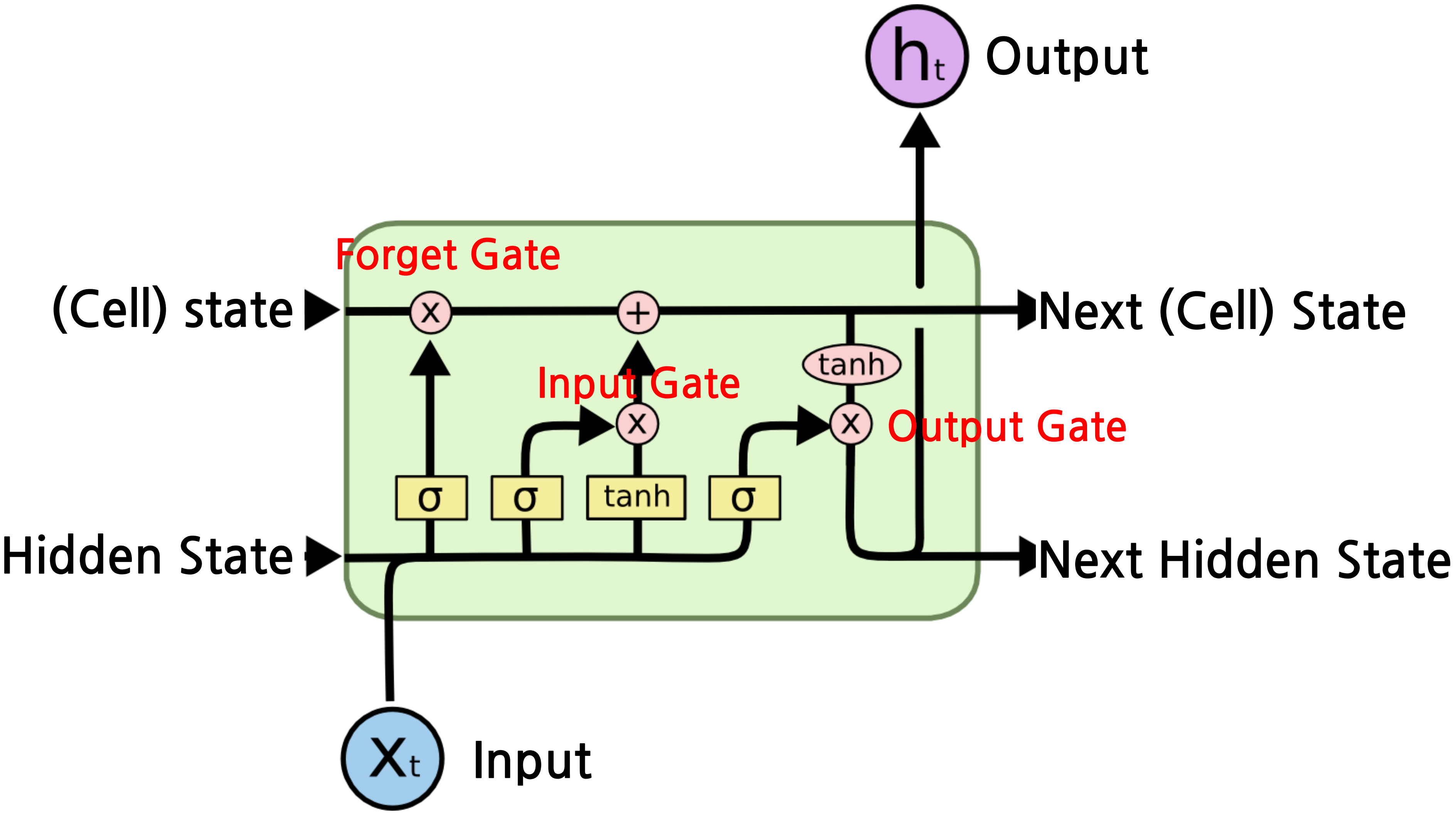
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

06. 4 LONG SHORT-TERM MEMORY



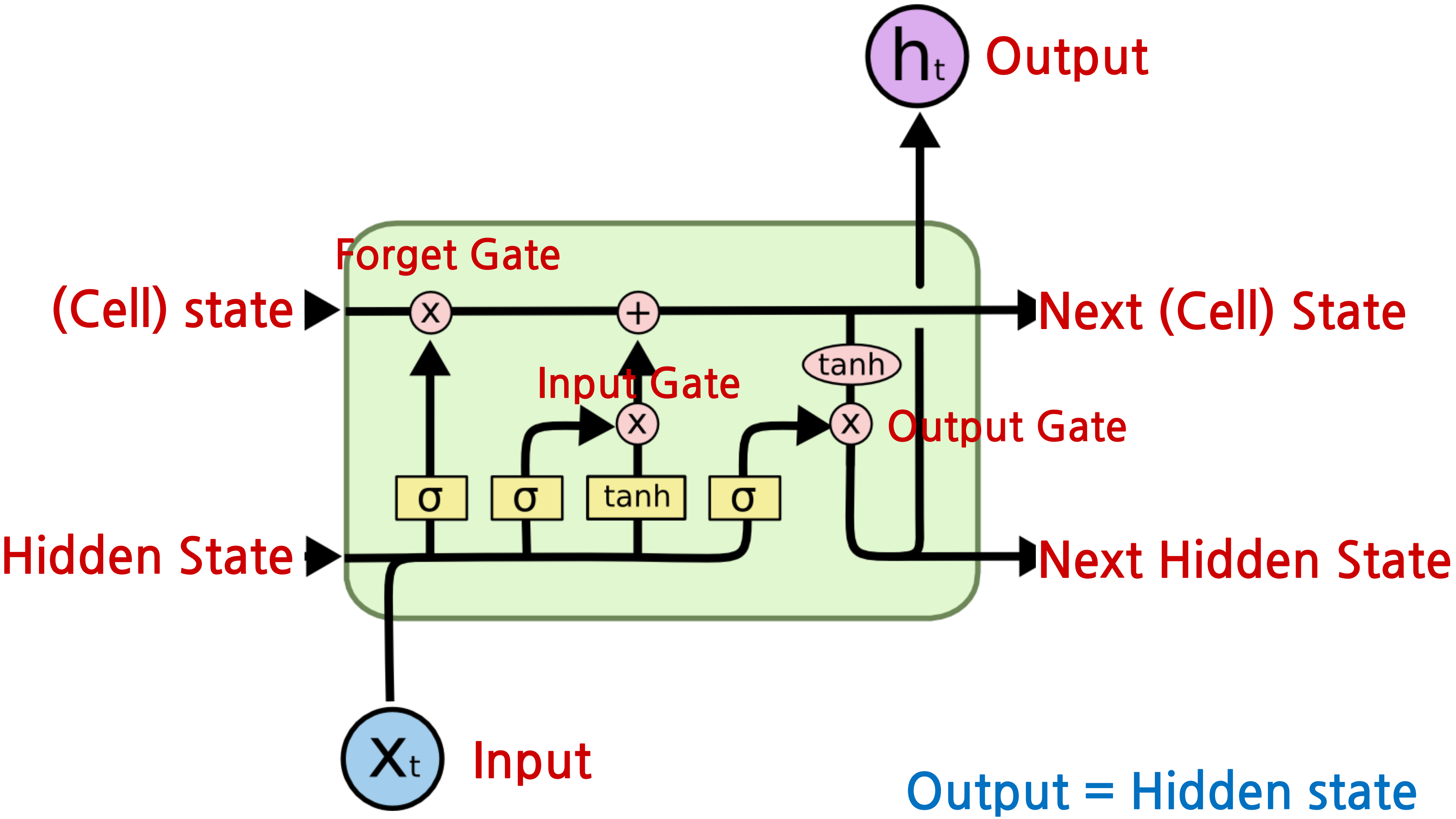
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

06. 4 LONG SHORT-TERM MEMORY



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

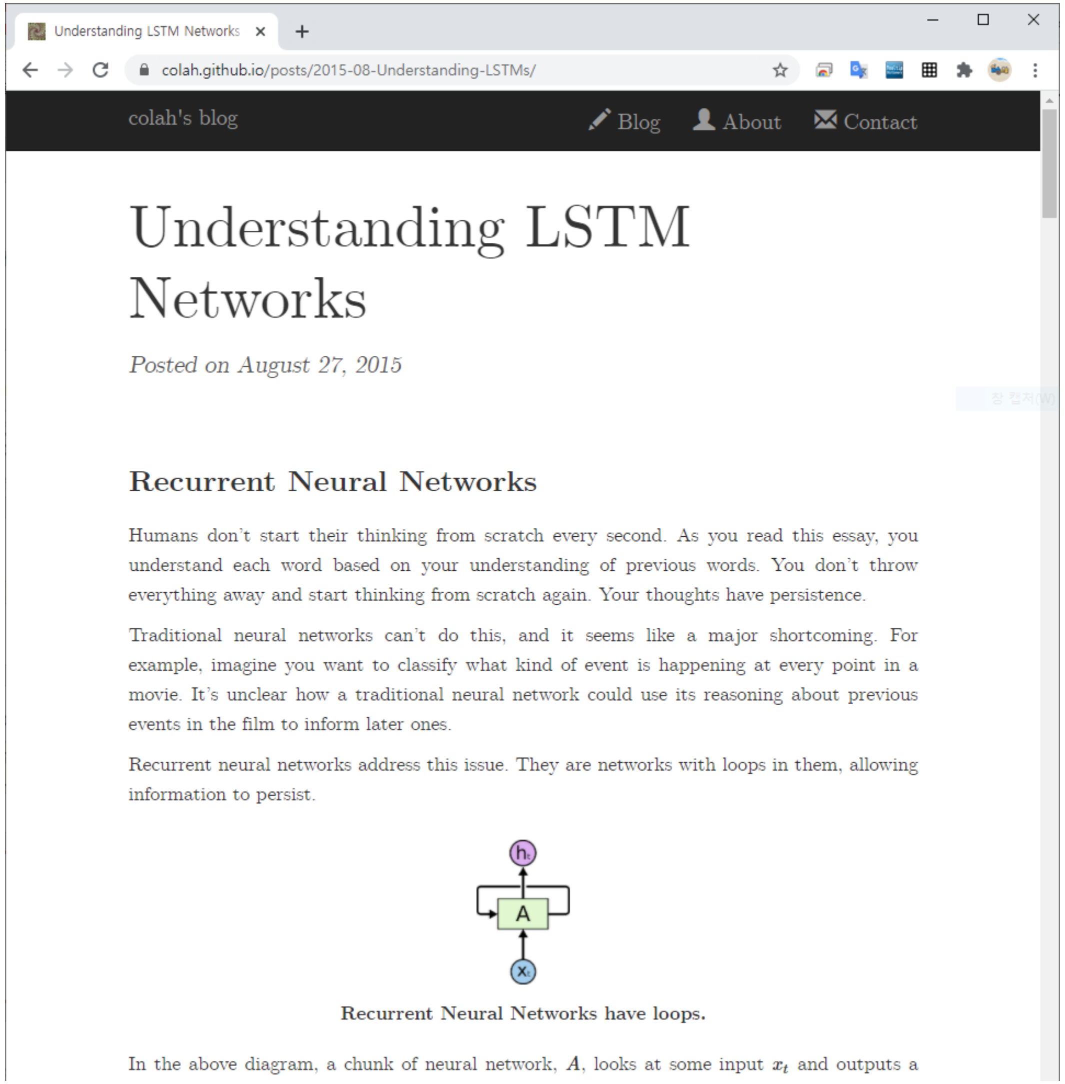
06. 4 LONG SHORT-TERM MEMORY



Output = Hidden state

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

06. LSTM 관련 참고 자료(1)



Understanding LSTM Networks

colah.github.io/posts/2015-08-Understanding-LSTMs/

colah's blog   Blog   About   Contact

Understanding LSTM Networks

Posted on August 27, 2015

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

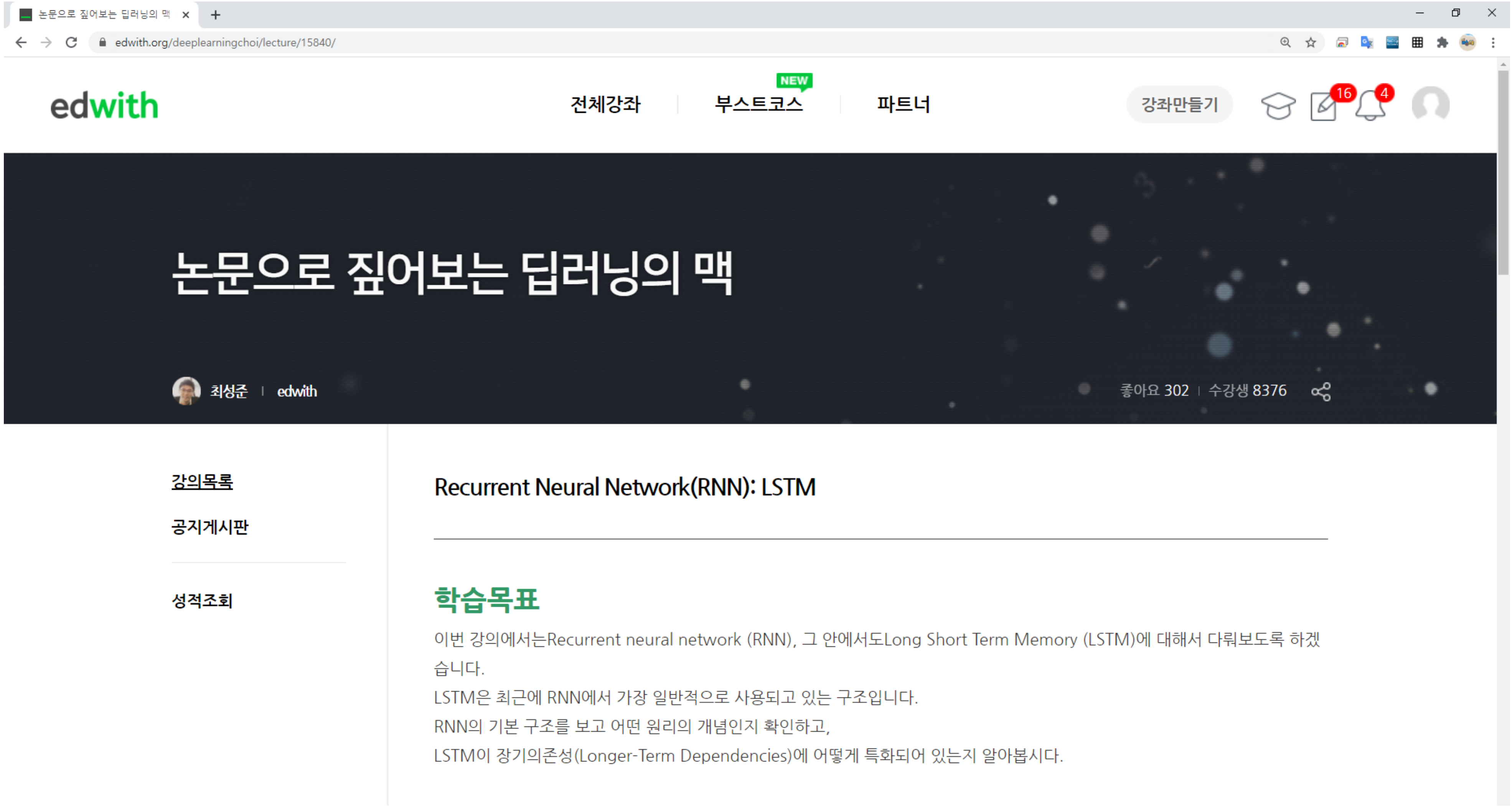
Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Recurrent Neural Networks have loops.

In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a

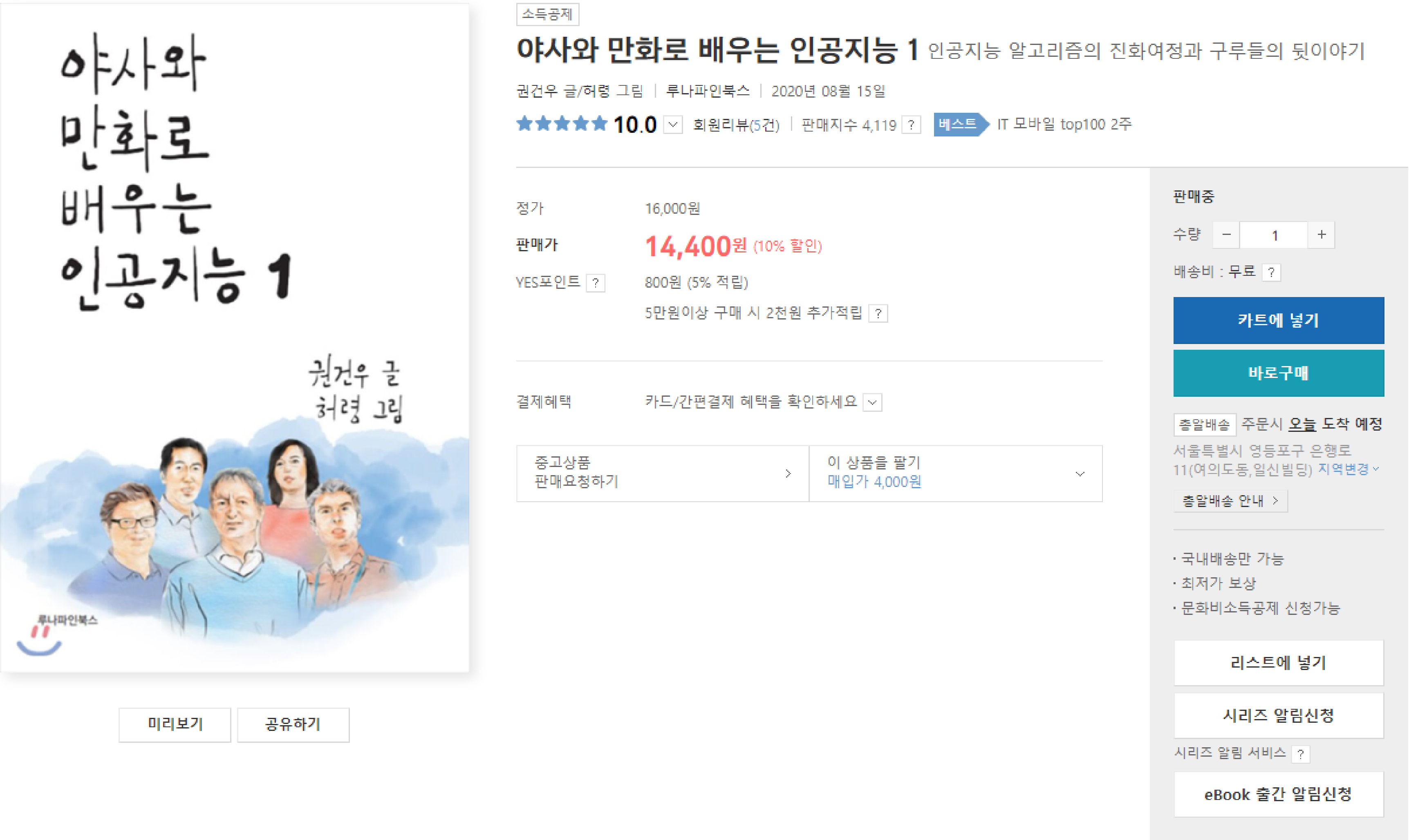
Colah's Blog

06. LSTM 관련 참고 자료(2)



Edwith 최성준 교수님 강의

06. LSTM 관련 참고 자료(3)



야사와 만화로 배우는 인공지능(브런치 / 도서)

07

5 EXPERIMENTS

07. 5 EXPERIMENTS

실험1: Embedded Reber grammar 문제(not a long time lag problem)

5.1 EXPERIMENT 1: EMBEDDED REBER GRAMMAR

Task. Our first task is to learn the "embedded Reber grammar", e.g. Smith and Zipser (1989), Cleeremans et al. (1989), and Fahlman (1991). Since it allows for training sequences with short time lags (of as few as 9 steps), it is *not* a long time lag problem. We include it for two reasons: (1) it is a popular recurrent net benchmark used by many authors — we wanted to have at least one experiment where RTRL and BPTT do not fail completely, and (2) it shows nicely how output gates can be beneficial.

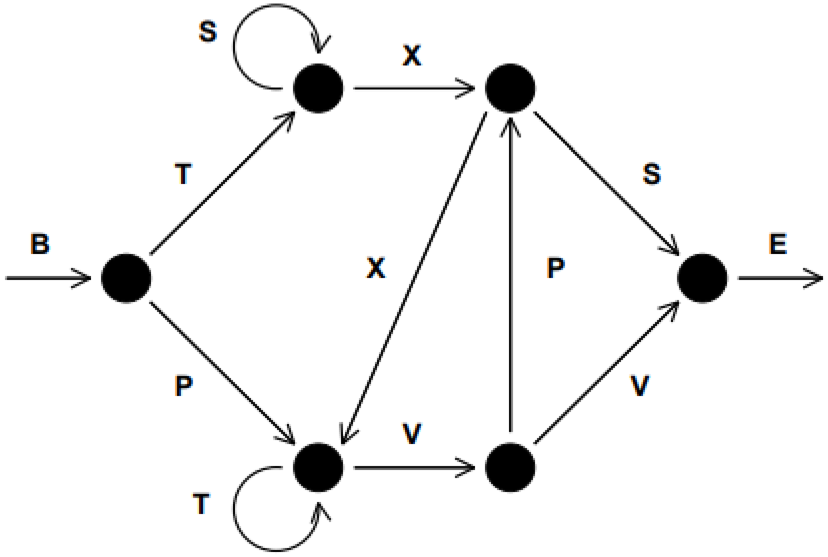


Figure 3: *Transition diagram for the Reber grammar.*

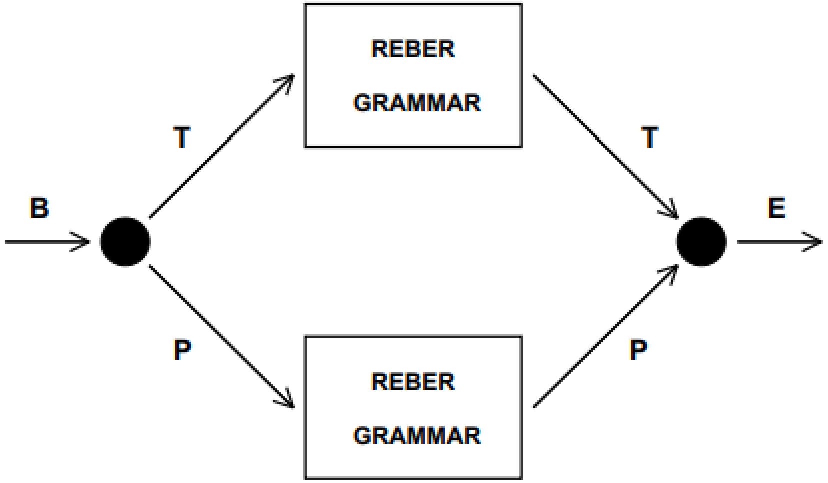


Figure 4: *Transition diagram for the embedded Reber grammar. Each box represents a copy of the Reber grammar (see Figure 3).*

5 EXPERIMENTS

실험1: Embedded Reber grammar 문제(not a long time lag problem)
결과: 다른 방법에 비해 빠르고 정확한 결과를 보여줌

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

Table 1: *EXPERIMENT 1: Embedded Reber grammar: percentage of successful trials and number of sequence presentations until success for RTRL (results taken from Smith and Zipser 1989), "Elman net trained by Elman's procedure" (results taken from Cleeremans et al. 1989), "Recurrent Cascade-Correlation" (results taken from Fahlman 1991) and our new approach (LSTM). Weight numbers in the first 4 rows are estimates — the corresponding papers do not provide all the technical details. Only LSTM almost always learns to solve the task (only two failures out of 150 trials). Even when we ignore the unsuccessful trials of the other approaches, LSTM learns much faster (the number of required training examples in the bottom row varies between 3,800 and 24,100).*

5 EXPERIMENTS

5.2 EXPERIMENT 2: NOISE-FREE AND NOISY SEQUENCES
Task 2a: noise-free sequences with long time lags
LSTM이 학습도 빠르고 정확한 결과를 보여줌

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

Table 2: *Task 2a: Percentage of successful trials and number of training sequences until success, for "Real-Time Recurrent Learning" (RTRL), "Back-Propagation Through Time" (BPTT), neural sequence chunking (CH), and the new method (LSTM). Table entries refer to means of 18 trials. With 100 time step delays, only CH and LSTM achieve successful trials. Even when we ignore the unsuccessful trials of the other approaches, LSTM learns much faster.*

07. 5 EXPERIMENTS

Task 2c: noise-free sequences with long time lags.(LSTM만으로 실험)

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

Table 3: *Task 2c: LSTM with very long minimal time lags $q+1$ and a lot of noise. p is the number of available distractor symbols ($p+4$ is the number of input units). $\frac{q}{p}$ is the expected number of occurrences of a given distractor symbol in a sequence. The rightmost column lists the number of training sequences required by LSTM (BPTT, RTRL and the other competitors have no chance of solving this task). If we let the number of distractor symbols (and weights) increase in proportion to the time lag, learning time increases very slowly. The lower block illustrates the expected slow-down due to increased frequency of distractor symbols.*

5 EXPERIMENTS

실험3: NOISE AND SIGNAL ON SAME CHANNEL 문제
Task 3a. 문제가 너무 간단해서 random weight guessing 으로 더 빨리 풀림

T	N	stop: ST1	stop: ST2	# weights	ST2: fraction misclassified
100	3	27,380	39,850	102	0.000195
100	1	58,370	64,330	102	0.000117
1000	3	446,850	452,460	102	0.000078

Table 4: *Task 3a: Bengio et al.'s 2-sequence problem. T is minimal sequence length. N is the number of information-conveying elements at sequence begin. The column headed by ST1 (ST2) gives the number of sequence presentations required to achieve stopping criterion ST1 (ST2). The rightmost column lists the fraction of misclassified post-training sequences (with absolute error $>$ 0.2) from a test set consisting of 2560 sequences (tested after ST2 was achieved). All values are means of 10 trials. We discovered, however, that this problem is so simple that random weight guessing solves it faster than LSTM and any other method for which there are published results.*

5 EXPERIMENTS

실험3: NOISE AND SIGNAL ON SAME CHANNEL 문제
Task 3b. 노이즈를 추가했지만 노이즈에 의해 불안한 결과를 보임

T	N	stop: ST1	stop: ST2	# weights	ST2: fraction misclassified
100	3	41,740	43,250	102	0.00828
100	1	74,950	78,430	102	0.01500
1000	1	481,060	485,080	102	0.01207

Table 5: *Task 3b: modified 2-sequence problem. Same as in Table 4, but now the information-conveying elements are also perturbed by noise.*

5 EXPERIMENTS

실험3: NOISE AND SIGNAL ON SAME CHANNEL 문제

Task 3c. Task 3a에 노이즈 추가된 문제. Random weight guessing으로 풀리지 않지만 LSTM으로 잘 풀림.

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

Table 6: *Task 3c: modified, more challenging 2-sequence problem. Same as in Table 4, but with noisy real-valued targets. The system has to learn the conditional expectations of the targets given the inputs. The rightmost column provides the average difference between network output and expected target. Unlike 3a and 3b, this task cannot be solved quickly by random weight guessing.*

07. 5 EXPERIMENTS

실험4: ADDING PROBLEM

기존에 RNN 계열의 알고리즘으로 해결할 수 없었던 문제.
LSTM이 장기 의존성 문제를 해결할 수 있음을 보여줌.

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

Table 7: *EXPERIMENT 4: Results for the Adding Problem. T is the minimal sequence length, $T/2$ the minimal time lag. "# wrong predictions" is the number of incorrectly processed sequences (error > 0.04) from a test set containing 2560 sequences. The rightmost column gives the number of training sequences required to achieve the stopping criterion. All values are means of 10 trials. For $T = 1000$ the number of required training examples varies between 370,000 and 2,020,000, exceeding 700,000 in only 3 cases.*

5 EXPERIMENTS

실험5: MULTIPLICATION PROBLEM

실험4와 다른 non-integrative solutions으로 과제를 해결할 수 있는지 확인하기 위한 실험.
이 실험을 통해 LSTM이 non-integrative information processing도 가능하다는 것을 보여줌.

T	minimal lag	# weights	n_{seq}	# wrong predictions	MSE	Success after
100	50	93	140	139 out of 2560	0.0223	482,000
100	50	93	13	14 out of 2560	0.0139	1,273,000

Table 8: *EXPERIMENT 5: Results for the Multiplication Problem. T is the minimal sequence length, $T/2$ the minimal time lag. We test on a test set containing 2560 sequences as soon as less than n_{seq} of the 2000 most recent training sequences lead to error > 0.04. "# wrong predictions" is the number of test sequences with error > 0.04. MSE is the mean squared error on the test set. The rightmost column lists numbers of training sequences required to achieve the stopping criterion. All values are means of 10 trials.*

5 EXPERIMENTS

실험6: TEMPORAL ORDER

LSTM이 광범위하게 분리된 입력의 시간적 순서에 의해 전달된 정보를 추출할 수 있다는 것을 보여줌.

The experiment shows that LSTM is able to extract information conveyed by the temporal order of widely separated inputs. In Task 6a, for instance, the delays between first and second relevant input and between second relevant input and sequence end are at least 30 time steps.

task	# weights	# wrong predictions	Success after
Task 6a	156	1 out of 2560	31,390
Task 6b	308	2 out of 2560	571,100

Table 9: *EXPERIMENT 6: Results for the Temporal Order Problem. "# wrong predictions" is the number of incorrectly classified sequences (error > 0.3 for at least one output unit) from a test set containing 2560 sequences. The rightmost column gives the number of training sequences required to achieve the stopping criterion. The results for Task 6a are means of 20 trials; those for Task 6b of 10 trials.*

08

6 DISCUSSION

08-1. Limitations of LSTM

Limitations of LSTM.

- The particularly efficient truncated backprop version of the LSTM algorithm will not easily solve problems similar to "strongly delayed XOR problems", where the goal is to compute the XOR of two widely separated inputs that previously occurred somewhere in a noisy sequence. The reason is that storing only one of the inputs will not help to reduce the expected error — the task is non-decomposable in the sense that it is impossible to incrementally reduce the error by first solving an easier subgoal.

  In theory, this limitation can be circumvented by using the full gradient (perhaps with additional conventional hidden units receiving input from the memory cells). But we do not recommend computing the full gradient for the following reasons: (1) It increases computational complexity. (2) Constant error flow through CECs can be shown only for truncated LSTM. (3) We actually did conduct a few experiments with non-truncated LSTM. There was no significant difference to truncated LSTM, exactly because outside the CECs error flow tends to vanish quickly. For the same reason full BPTT does not outperform truncated BPTT.

- Each memory cell block needs two additional units (input and output gate). In comparison to standard recurrent nets, however, this does not increase the number of weights by more than a factor of 9: each conventional hidden unit is replaced by at most 3 units in the LSTM architecture, increasing the number of weights by a factor of 3^2 in the fully connected case. Note, however, that our experiments use quite comparable weight numbers for the architectures of LSTM and competing approaches.

Advantages of LSTM

Advantages of LSTM.

- The constant error backpropagation within memory cells results in LSTM's ability to bridge very long time lags in case of problems similar to those discussed above.
- For long time lag problems such as those discussed in this paper, LSTM can handle noise, distributed representations, and continuous values. In contrast to finite state automata or hidden Markov models LSTM does not require an *a priori* choice of a finite number of states. In principle it can deal with unlimited state numbers.
- For problems discussed in this paper LSTM generalizes well — even if the positions of widely separated, relevant inputs in the input sequence do not matter. Unlike previous approaches, ours quickly learns to distinguish between two or more widely separated occurrences of a particular element in an input sequence, without depending on appropriate short time lag training exemplars.
- There appears to be no need for parameter fine tuning. LSTM works well over a broad range of parameters such as learning rate, input gate bias and output gate bias. For instance, to some readers the learning rates used in our experiments may seem large. However, a large learning rate pushes the output gates towards zero, thus automatically countermanding its own negative effects.
- The LSTM algorithm's update complexity per weight and time step is essentially that of BPTT, namely $O(1)$. This is excellent in comparison to other approaches such as RTRL. Unlike full BPTT, however, LSTM is *local in both space and time*.

09

7 CONCLUSION

09. Future work(1997)

LSTM의 실제적인 한계를 찾기 위해 우리는 이를 real world 데이터에 적용하고자 한다.
적용의 영역에는 (1) 시계열 예측, (2) 음악 작곡, (3) 음성 처리 등이 포함된다.

7 CONCLUSION

Each memory cell's internal architecture guarantees constant error flow within its constant error carrousel CEC, provided that truncated backprop cuts off error flow trying to leak out of memory cells. This represents the basis for bridging very long time lags. Two gate units learn to open and close access to error flow within each memory cell's CEC. The multiplicative input gate affords protection of the CEC from perturbation by irrelevant inputs. Likewise, the multiplicative output gate protects other units from perturbation by currently irrelevant memory contents.

Future work. To find out about LSTM's practical limitations we intend to apply it to real world data. Application areas will include (1) time series prediction, (2) music composition, and (3) speech processing. It will also be interesting to augment sequence chunkers (Schmidhuber 1992b, 1993) by LSTM to combine the advantages of both.

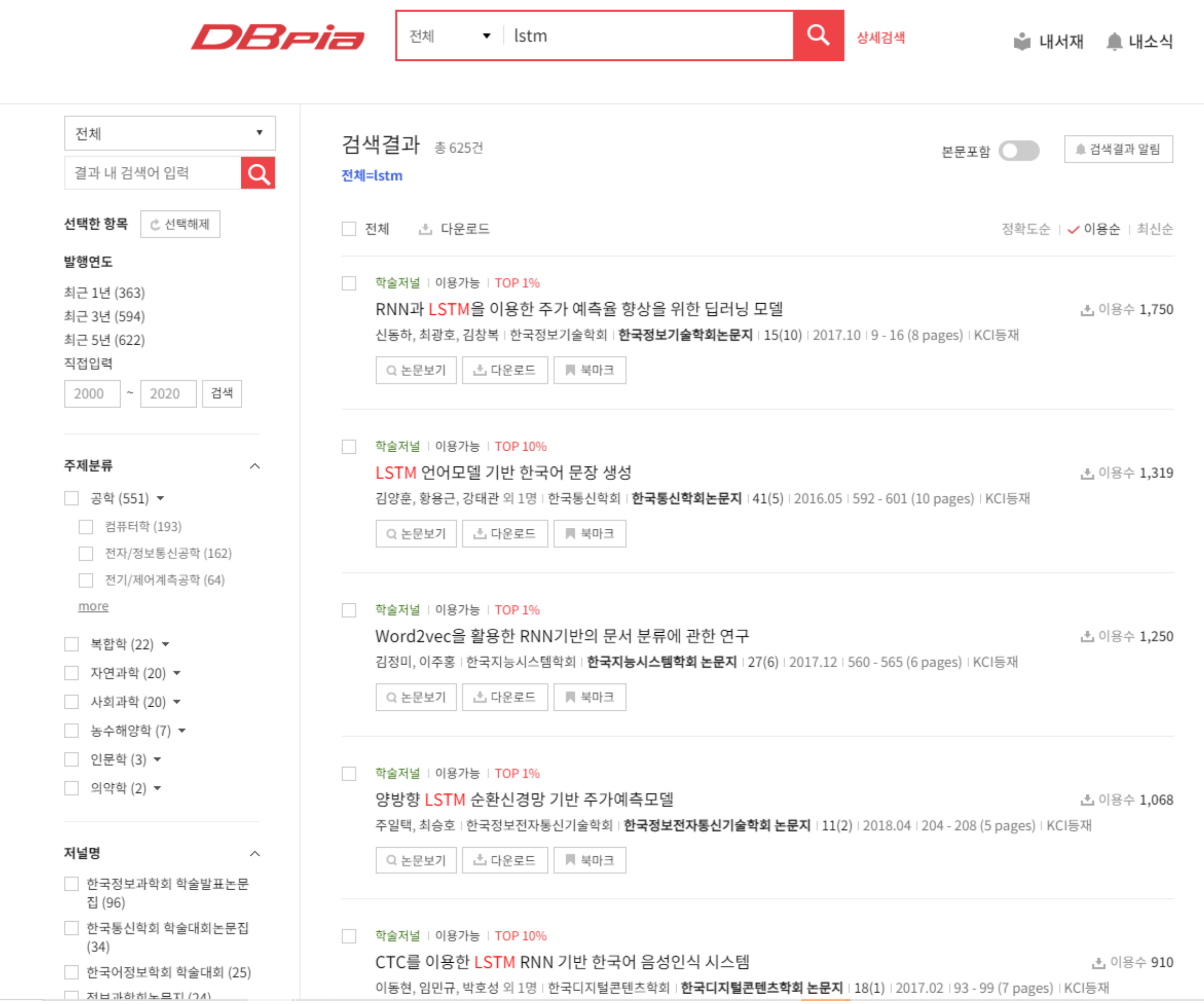
8 ACKNOWLEDGMENTS

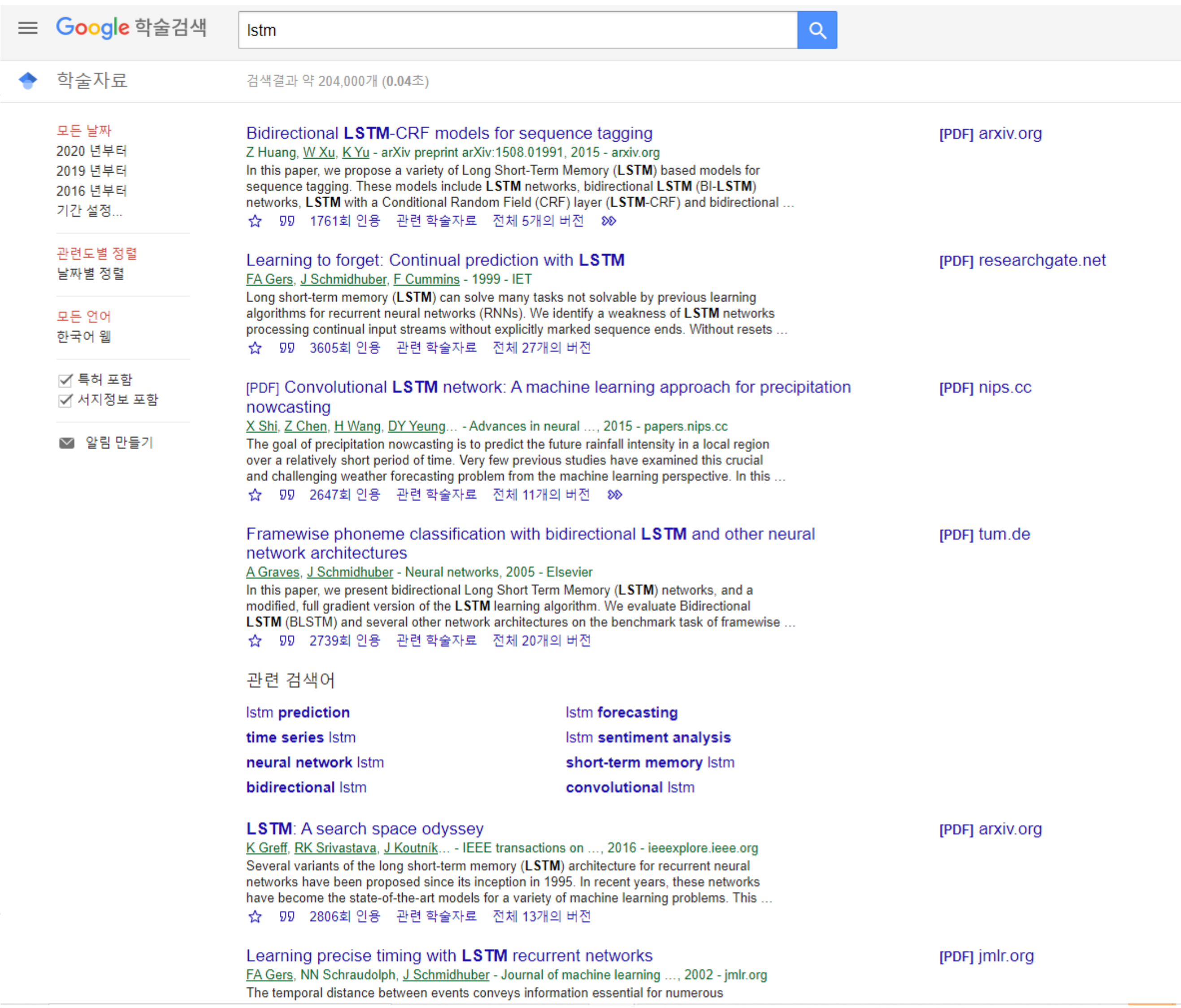
Thanks to Mike Mozer, Wilfried Brauer, Nic Schraudolph, and several anonymous referees for valuable comments and suggestions that helped to improve a previous version of this paper (Hochreiter and Schmidhuber 1995). This work was supported by *DFG grant SCHM 942/3-1* from "Deutsche Forschungsgemeinschaft".

10

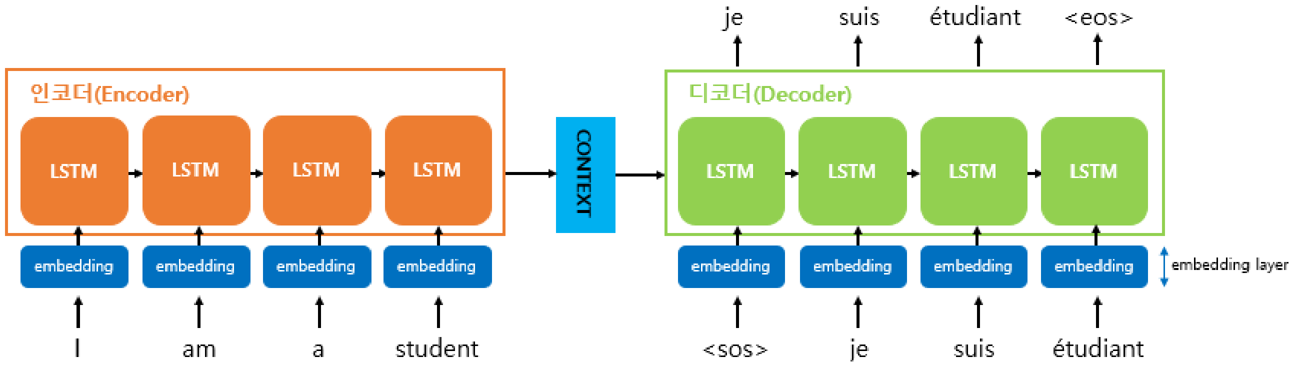
1997, LSTM의 미래

10. 1997, LSTM의 미래

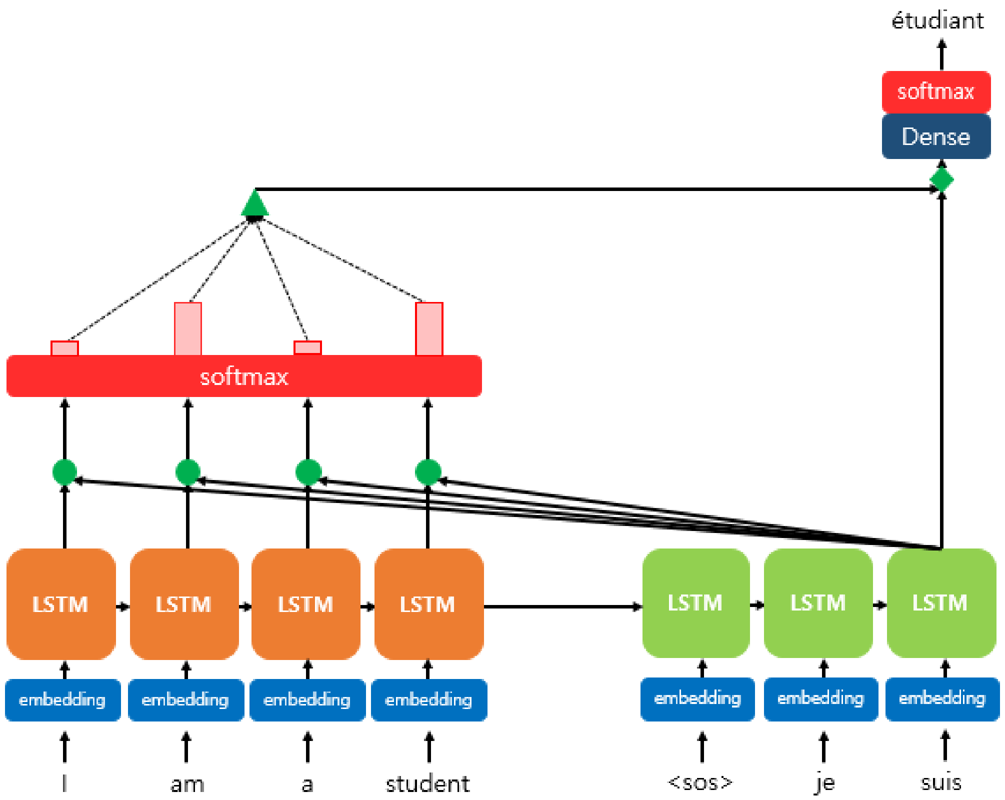




10. 1997, LSTM의 미래



seq2seq



BiLSTM with Attention

* 출처 : 위키독스, 딥러닝을 이용한 자연어처리 입문, https://wikidocs.net/22893

Long Short-Term Memory 논문 리뷰

(Sepp Hochreiter, Jürgen Schmidhuber, Neural Computation, 1997)

감사합니다 :)

NLP12 초급반 송석리
(greatsong21@gmail.com)