NLP week4

RNN, LSTM, GRU

발제자: 18기 김승하

목차
INDEX

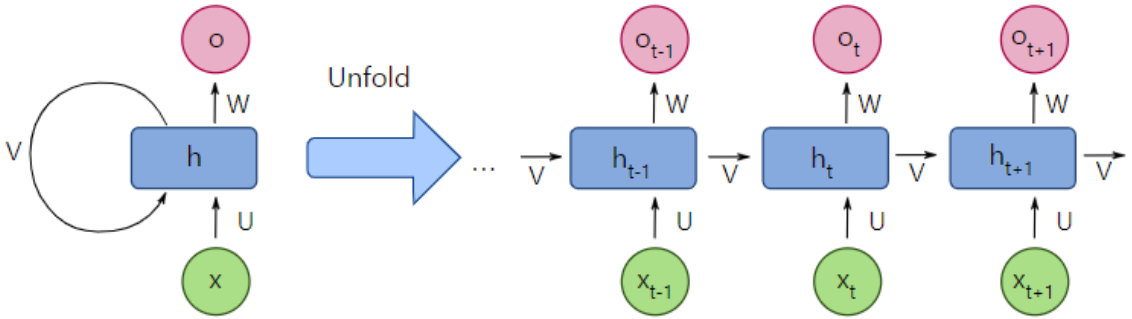
01 Recurrent Neural Network

순환 신경망

RNN의 기본 구조

- 은닉층 노드의 결과값이 은닉층 노드의 다음 입력으로 연결되어 순환을 이루는 인공신경망
- 메모리 셀 (노드) : 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할
- 은닉상태 (hidden state) : 메모리 셀이 다음 시점 t+1의 자기 자신에게 보내는 값

t : 현재 시점
Xt: 현재의 입력값
ht : t 시점의 은닉상태값
Wx: Xt을 위한 가중치
Wh: ht−1을 위한 가중치



[출력벡터]

[은닉층(은닉상태)]

[입력벡터]

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

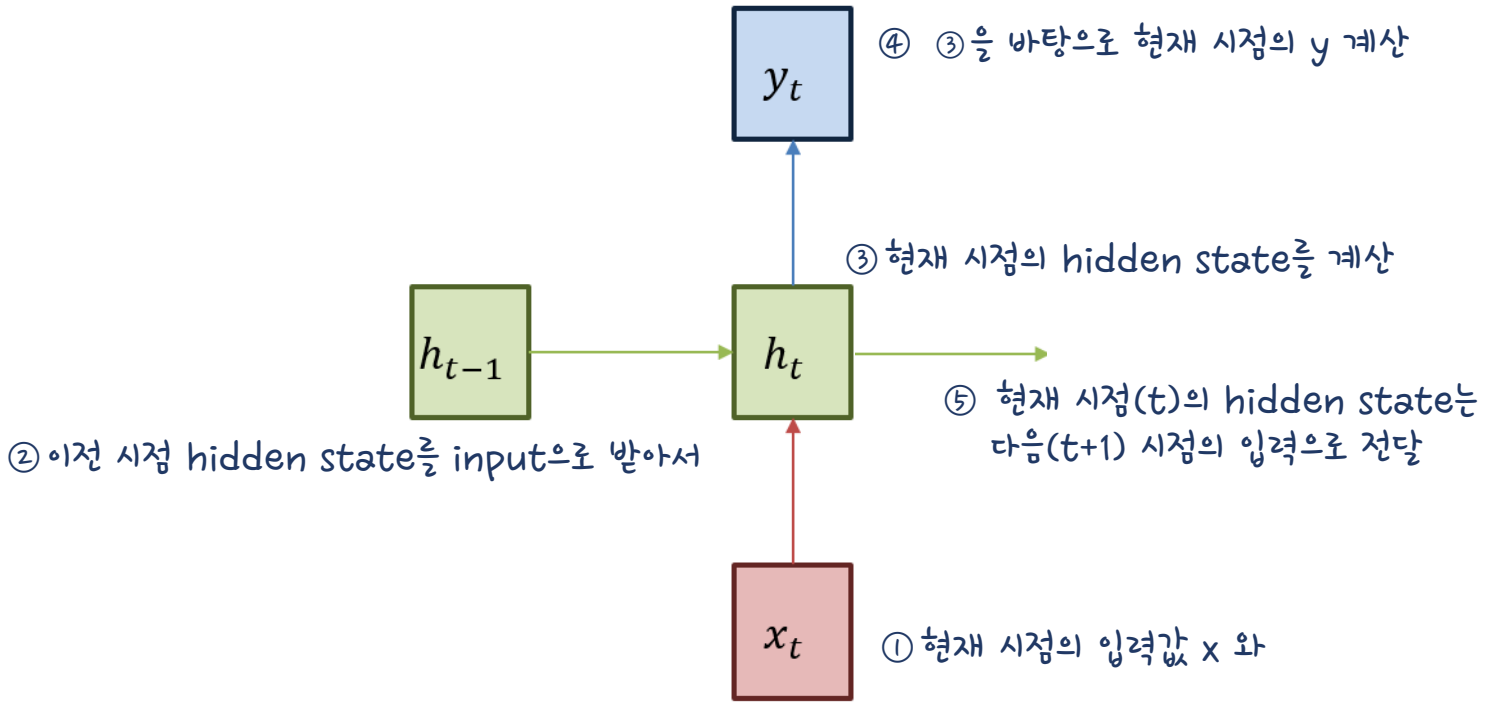
$$y_t = W_{hy}h_t + b_y$$

01 Recurrent Neural Network

순환 신경망

RNN의 기본 구조

- 은닉층 노드의 결과값이 은닉층 노드의 다음 입력으로 연결되어 순환을 이루는 인공신경망
- 메모리 셀 (노드) : 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할
- 은닉상태 (hidden state) : 메모리 셀이 다음 시점 t+1의 자기 자신에게 보내는 값



01 Recurrent Neural Network

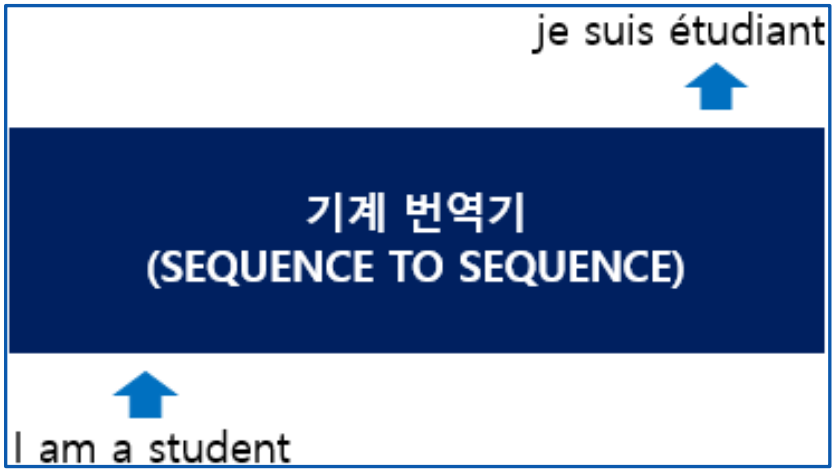
순환 신경망

RNN은 시퀀스 모델이다

- 시퀀스 모델 : 입력과 출력을 시퀀스 단위로 처리하는 모델
- Ex) 번역기 - 입력 : 번역하고자 하는 문장 (I got a bad flu.)

  출력: 번역된 문장 (나 심한 독감에 걸렸어.)

  → *단어 시퀀스 처리*

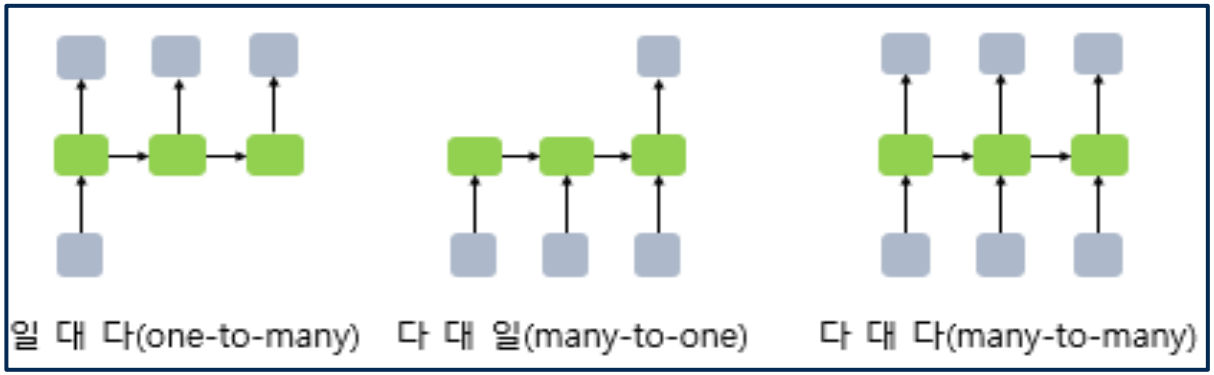


01 Recurrent Neural Network

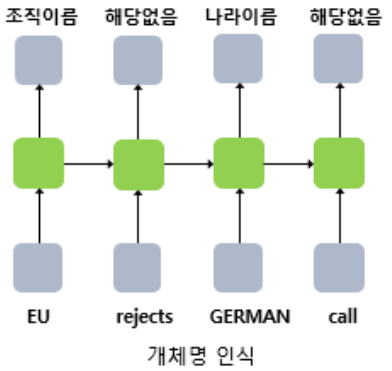
순환 신경망

RNN의 다양한 용도

- 입력과 출력의 길이를 다르게 설계 할 수 있음



- One to one: Vanilla neural network
- One to many: Image captioning
- Many to one: 감성분석, 스팸분류
- Many to many: 기계 번역, 챗봇, 품사 태깅, 개체명 인식



개체명 인식

스팸 메일 분류

01 Recurrent Neural Network

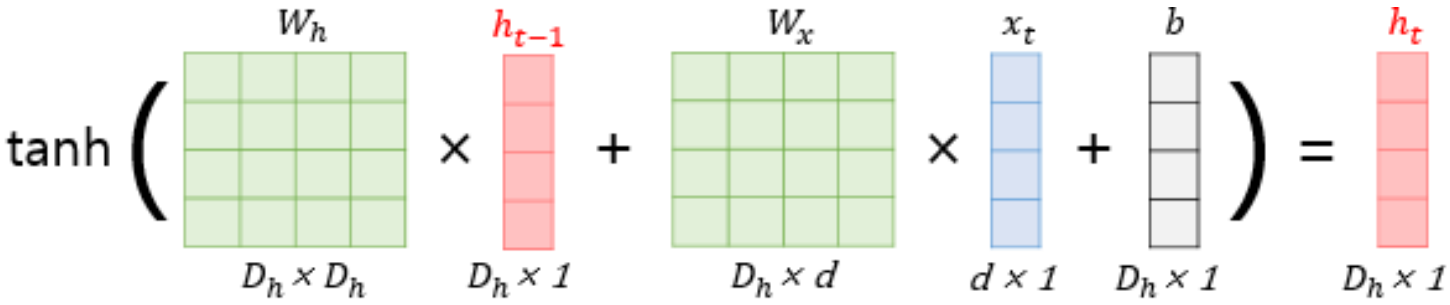
순환 신경망

Computation

- RNN의 은닉층 연산은 벡터와 행렬의 연산으로 이해할 수 있다.

d: 단어 벡터의 차원
Dn: 은닉 상태의 크기

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$



Dh x 1 + Dh x 1 + Dh x 1 =

01 Recurrent Neural Network

순환 신경망

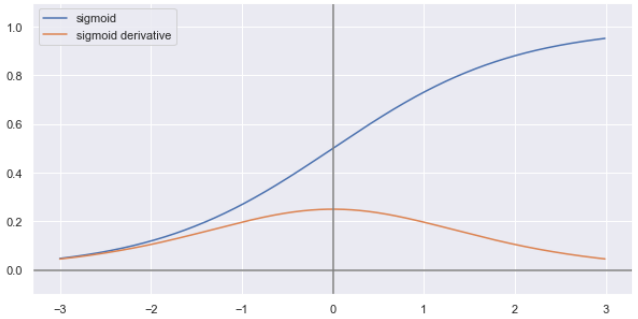
Computation

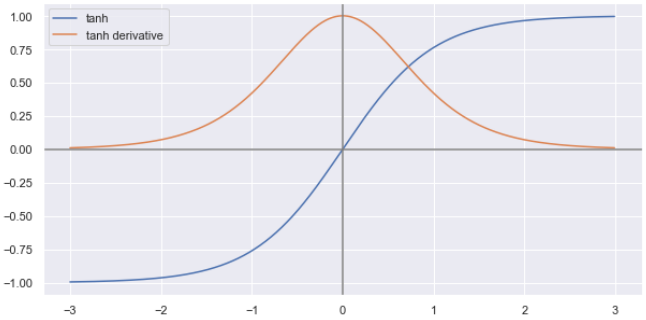
은닉층의 활성화 함수

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = W_{hy}h_t + b_y$$

- Ht의 활성화 함수: 주로 하이퍼볼릭탄젠트 함수(tanh)가 사용됨.
- 출력층 결과값 yt를 계산하기 위한 활성화 함수: 시그모이드, 소프트맥스 함수



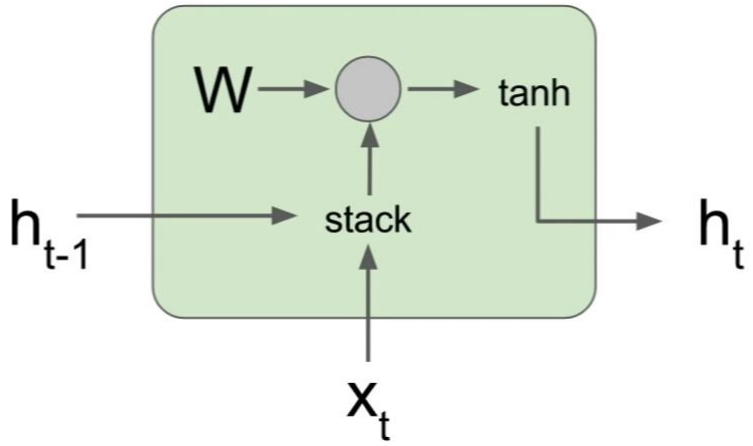


- RNN에서 주로 tanh를 사용하는 이유:
RNN의 Vanishing gradient 문제를 예방(gradient 최대한 오래 유지하기에 유리)
- ReLU 안쓰는 이유: RNN은 같은 레이어를 여러 번 반복하는 성질을 가지고 있기 때문에 1보다 큰 값이 들어오게 되면 반복하면서 값이 너무 커질 수 있음

01 Recurrent Neural Network

순환 신경망

RNN의 순전파



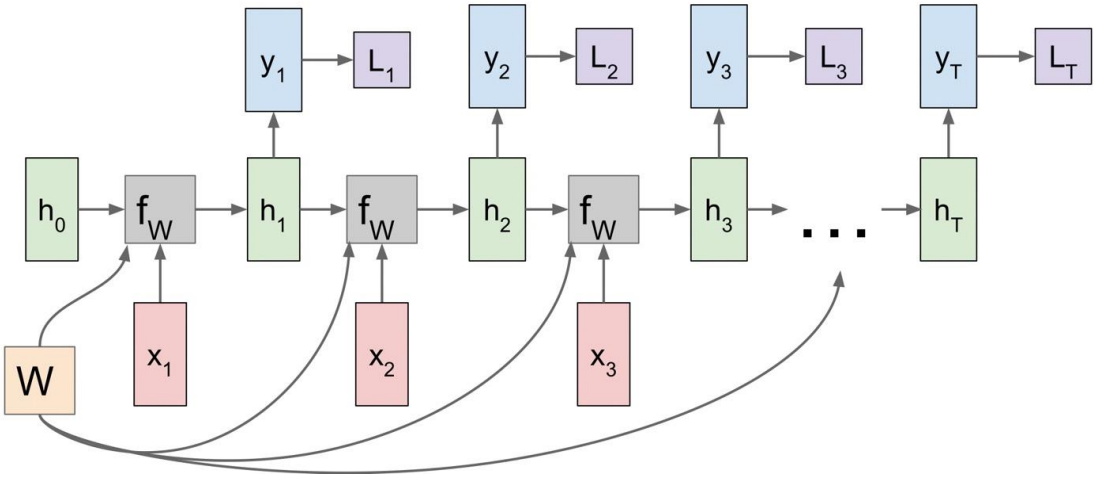
$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

01 Recurrent Neural Network

순환 신경망

RNN의 역전파

- Back propagation through time



- forward pass로 각 타임 스텝 별 출력시퀀스(y)를 구한다.
- 출력 시퀀스 y와 손실(비용)함수를 사용하여 각 타임 스텝 별 Loss를 구한다.

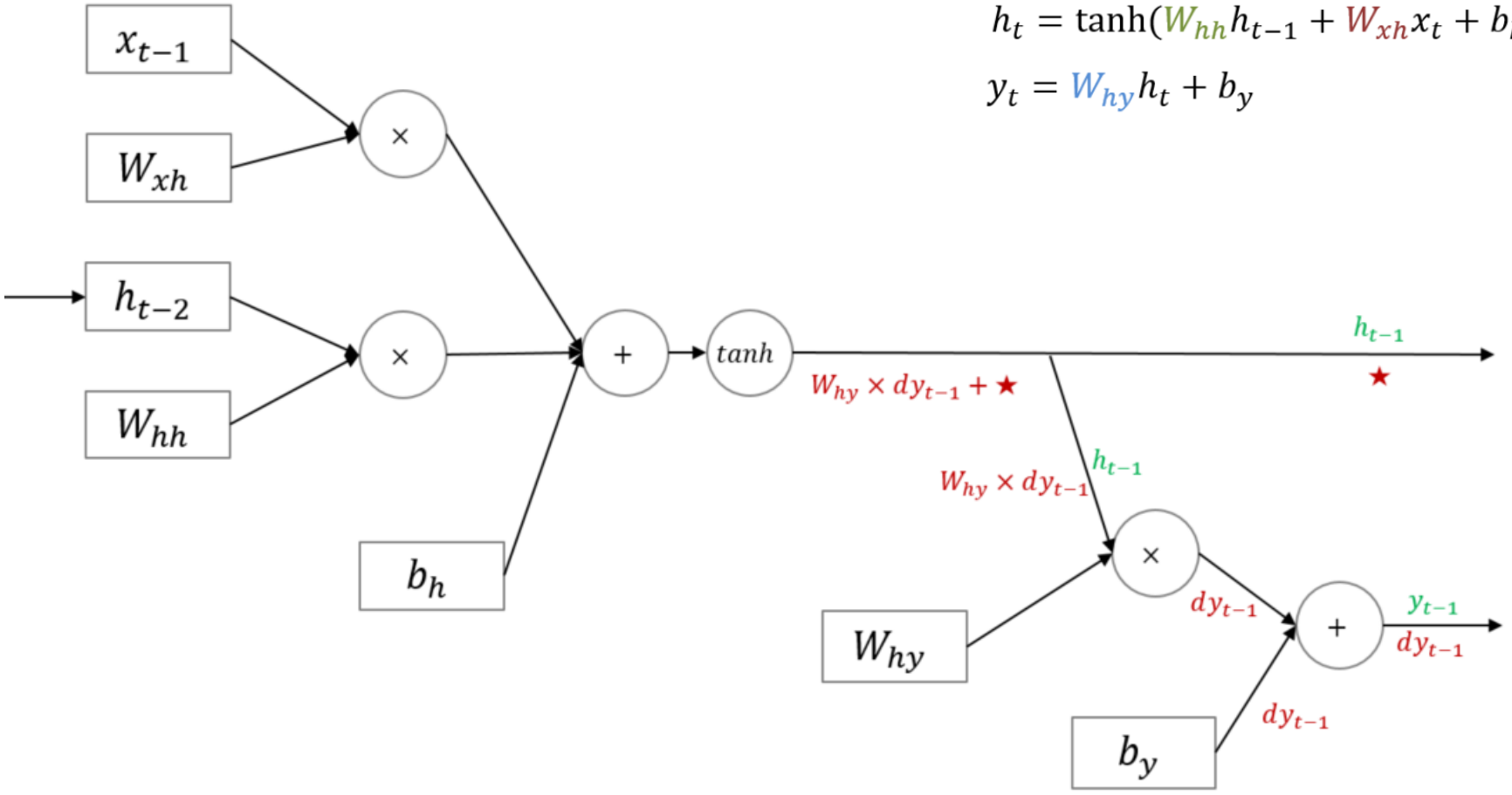
01 Recurrent Neural Network

순환 신경망

RNN의 역전파

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = W_{hy}h_t + b_y$$



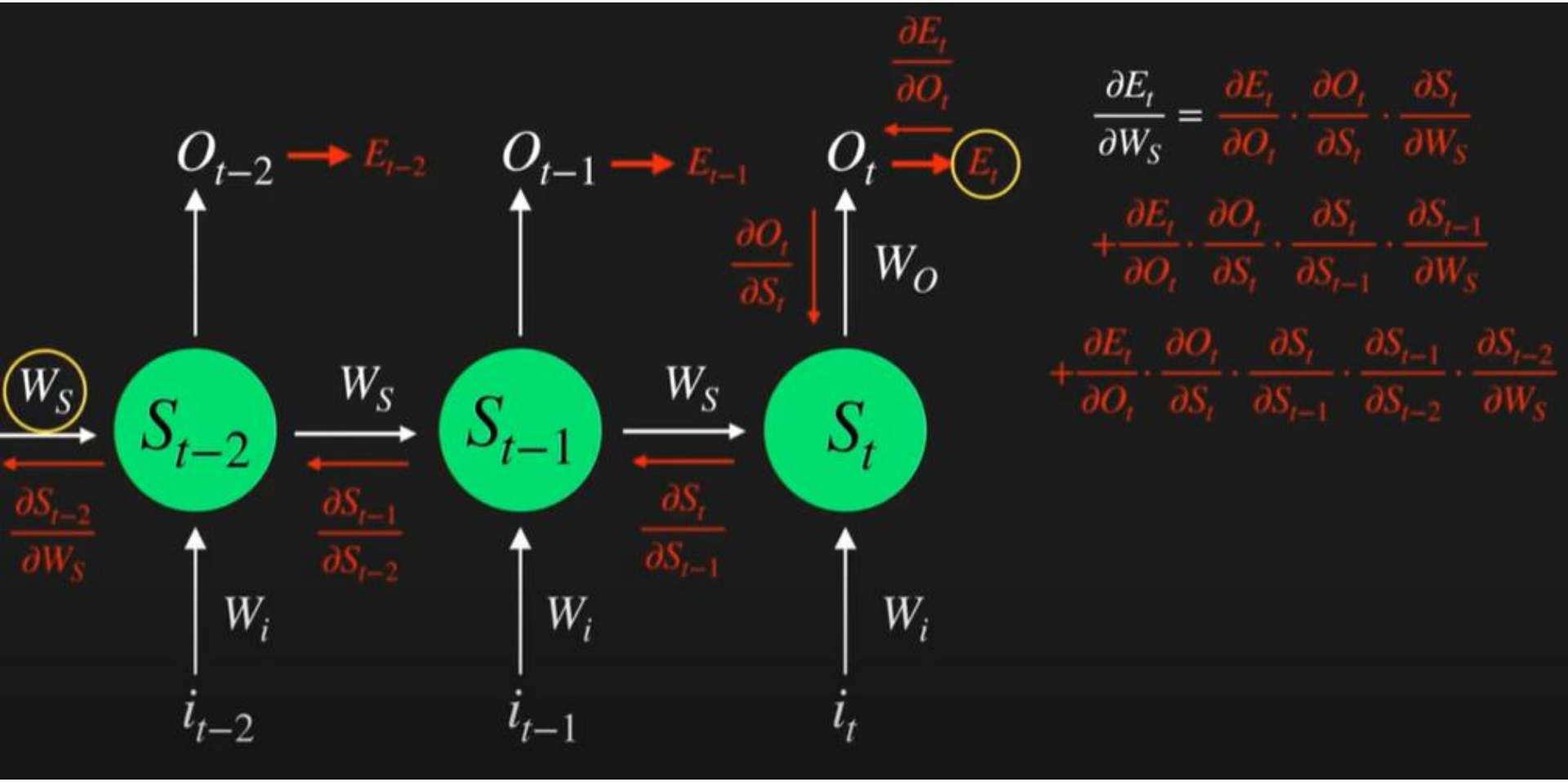
01 Recurrent Neural Network

순환 신경망

https://mmuratarat.github.io/2019-02-07/bptt-of-rnn

https://m.blog.naver.com/infoefficien/221210061511

RNN의 역전파



01 Recurrent Neural Network

순환 신경망

Exploding/Vanishing Gradient Problem

- Vanishing Gradient Problem: RNN은 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우, 역전파 시 gradient가 점차 줄어 학습능력이 크게 저하되는 것으로 알려져 있다.
- 입력층에 가까운 층들에서 가중치들이 업데이트가 제대로 되지 않으면 제대로 모델 학습 X
- Gradient Exploding: 기울기가 점차 커져 가중치들이 비정상적으로 큰 값이 되면서 결국 발산하는 현상

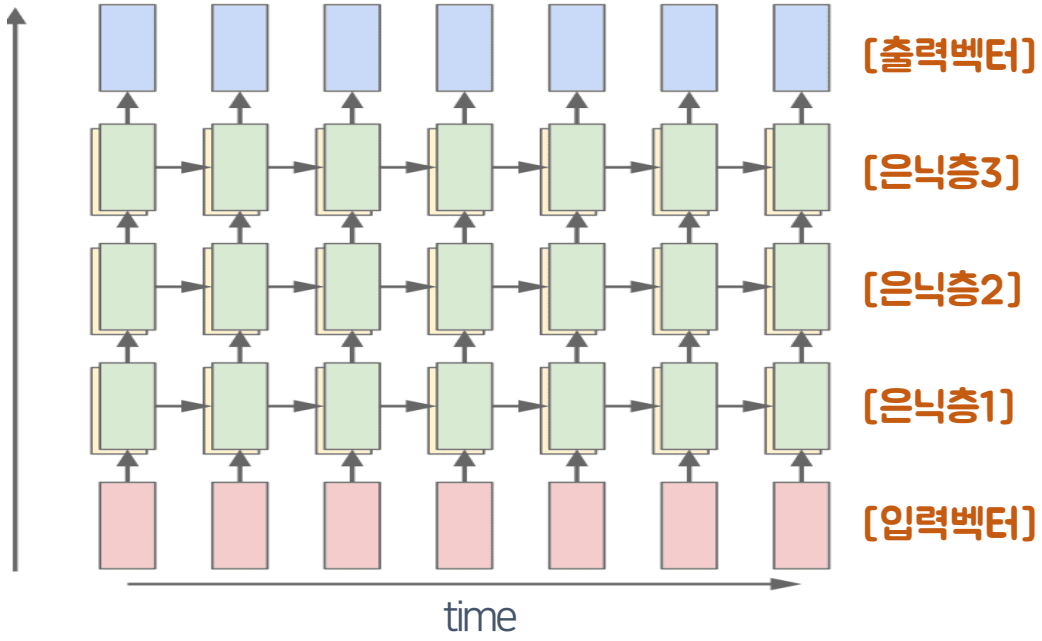
→ One solution to vanishing gradient problem = LSTM!

01 Recurrent Neural Network

순환 신경망

Multilayer RNNs

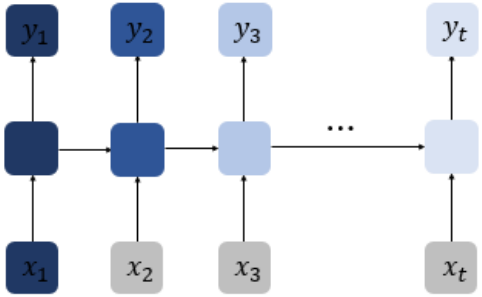
- 깊은 순환 신경망(Deep Recurrent Neural Network)
- Input goes into a layer and produces a sequence of hidden states
- These hidden states are passed onto the next recurrent layer as inputs
- These layers stacked on top of each other is called Multilayer RNN (2~4 layers in general)



02 LSTM

Long-Short Term Memory

RNN의 한계점



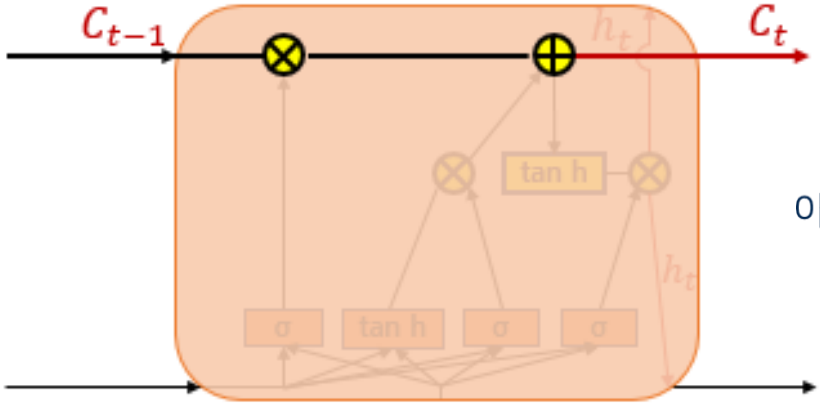
- RNN은 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 역전파시 그래디언트가 점차 줄어 학습능력이 크게 저하: vanishing gradient problem
- 장기 의존성 문제(the problem of Long-Term Dependencies): 비교적 짧은 시퀀스에 대해서만 효과를 보이는 문제. (time step이 길어질수록 정보량이 소실됨)
- ex) I grew up in France, I love watching movies……I speak fluent *French*"

02 LSTM

Long-Short Term Memory

LSTM 기본구조

- LSTM에는 Hidden state와 더불어 Cell state가 추가됨
- Hidden state : 단기 메모리 (ht)
- Cell state : 장기 메모리 (Ct)



이전 시점의 셀 상태가 다음 시점의 셀 상태를 구하기 위한 입력으로 사용됨.

02 LSTM

Long-Short Term Memory

LSTM 기본구조

$$
\begin{aligned}
f_t &= \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f}) \\
i_t &= \sigma(W_{xh\_i}x_t + W_{hh\_i}h_{t-1} + b_{h\_i}) \\
o_t &= \sigma(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o}) \\
g_t &= \tanh(W_{xh\_g}x_t + W_{hh\_g}h_{t-1} + b_{h\_g}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

$$
\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}
$$

$$
\begin{aligned}
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot \tanh(c_t)
\end{aligned}
$$

- Input gate, Forget gate, Output Gate : ht와 Ct를 구하기 위해 추가된 3개의 게이트
- Forget gate - Ct-1 중 유지/삭제할 부분 선택
- Input gate - Gt(candidate values) 중 Ct값에 더할 부분 선택
- Output gate – ht에 반영될 Ct 값을 선택
- Gt – Ct에 더해질 가능성이 있는 값들 (candidate values)

02 LSTM

Long-Short Term Memory

LSTM 기본구조

이전시점 cell state 중 잊거나 기억할 부분 선택

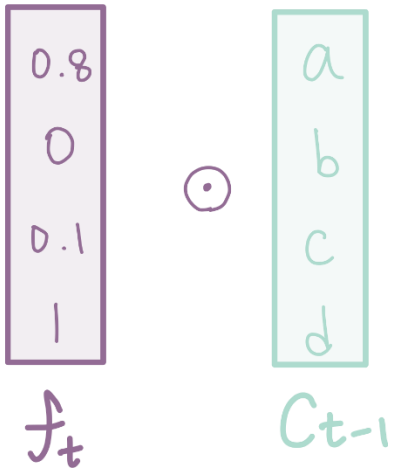
Candidate value 인 g 중 Ct 에
추가할 부분을 input gate로 선택

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Cell state를 -1~1 사이의 값으로 표현한 값과 output gate를 곱하여
hidden state에 반영할 부분 선택

ex)



02 LSTM

Long-Short Term Memory

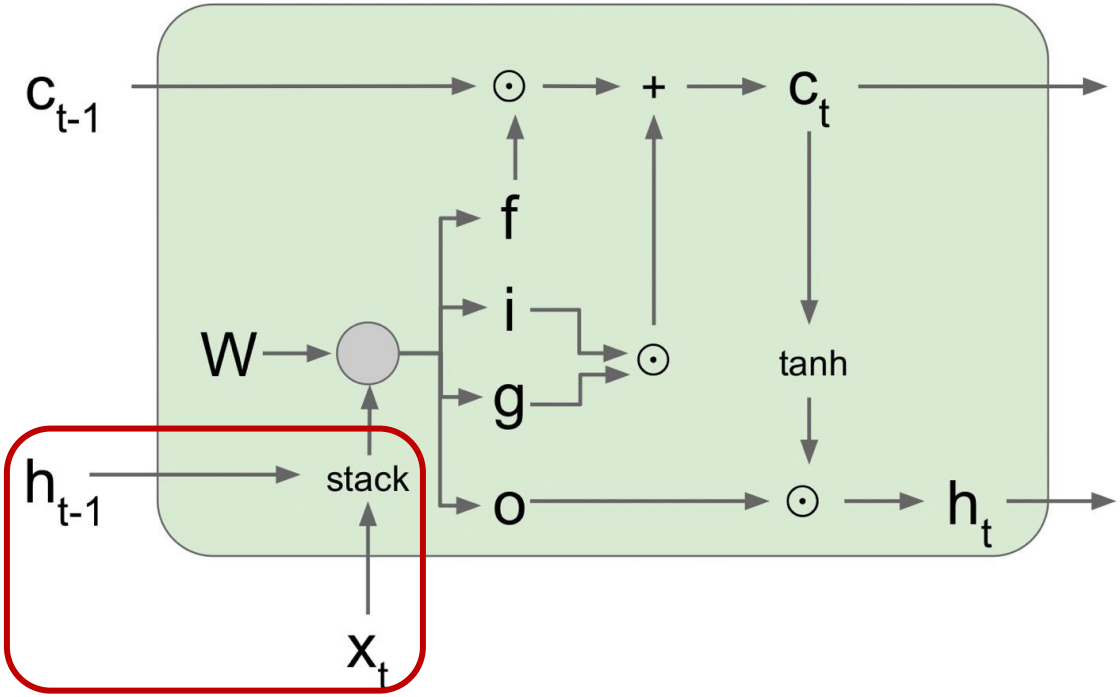
LSTM 의 순전파

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

1. 입력: ht-1, Xt → concatenate(stack)



02 LSTM

Long-Short Term Memory

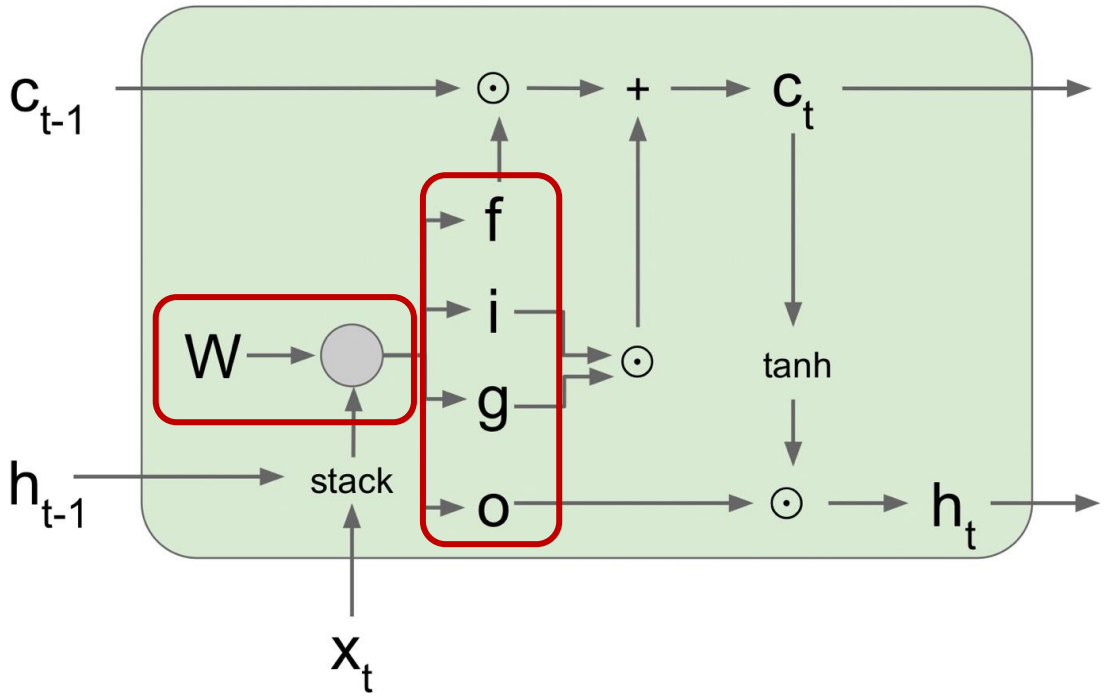
LSTM 의 순전파

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

2. 가중치 행렬 W와 multiply --> i, f, o, g 생성

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



02 LSTM

Long-Short Term Memory

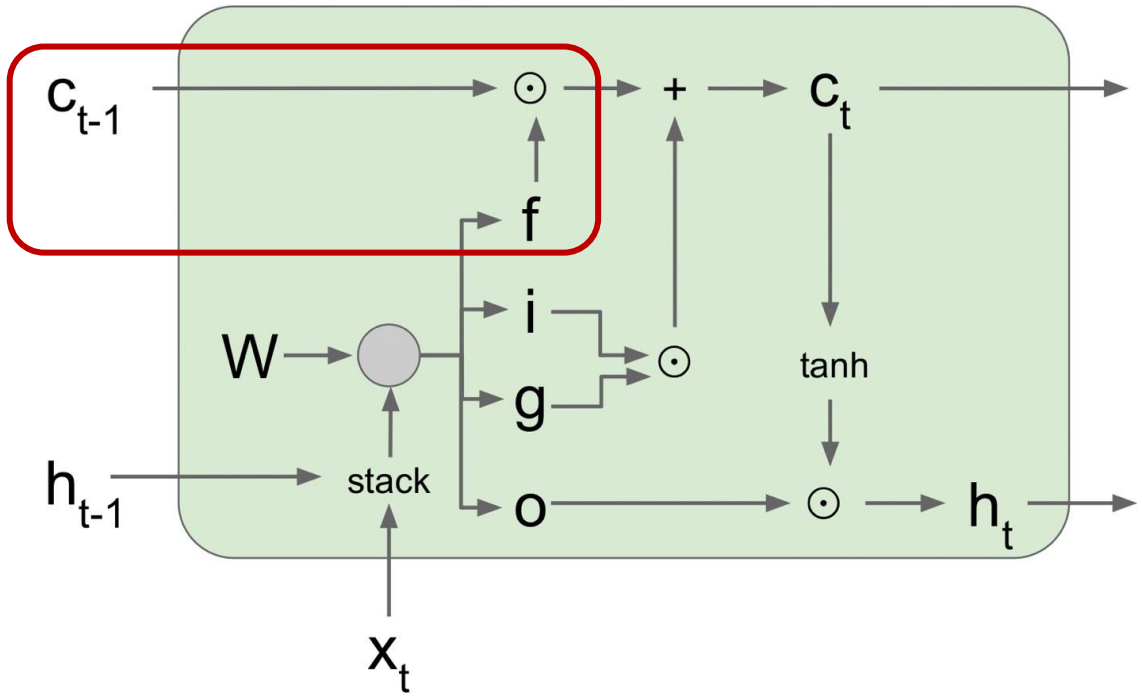
LSTM 의 순전파

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

3. Forget gate 와 이전 시점의 Cell state 요소별 곱셈

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



02 LSTM

Long-Short Term Memory

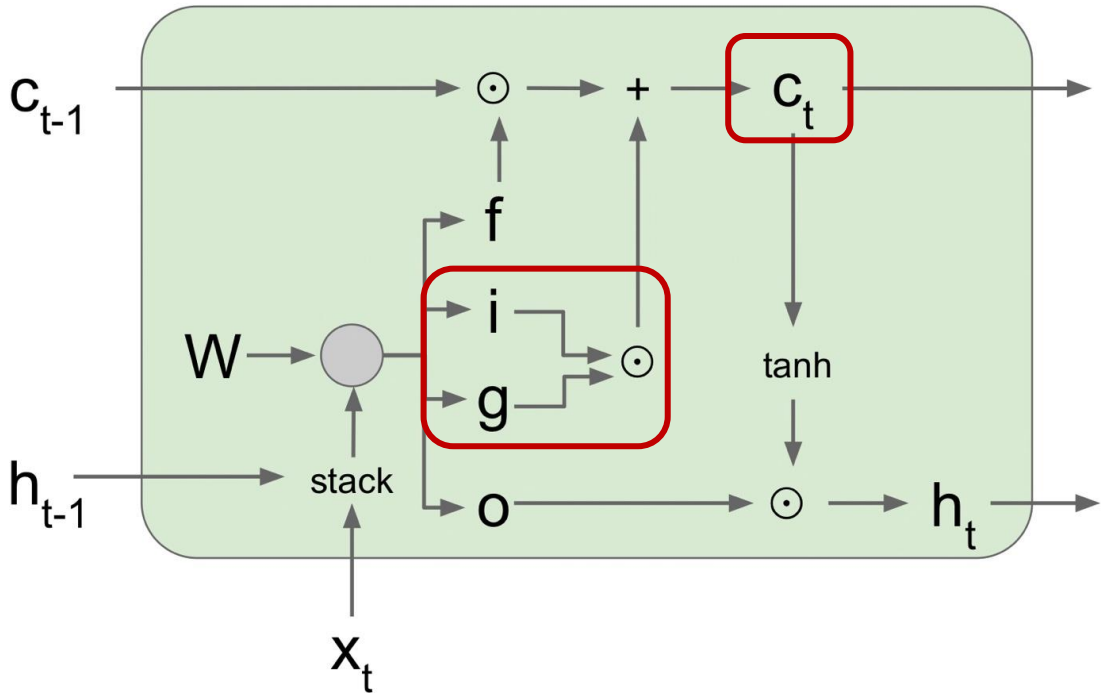
LSTM 의 순전파

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

4. Input gate 와 gt 요소별 곱셈
5. 2,3으로부터 Ct 도출

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



02 LSTM

Long-Short Term Memory

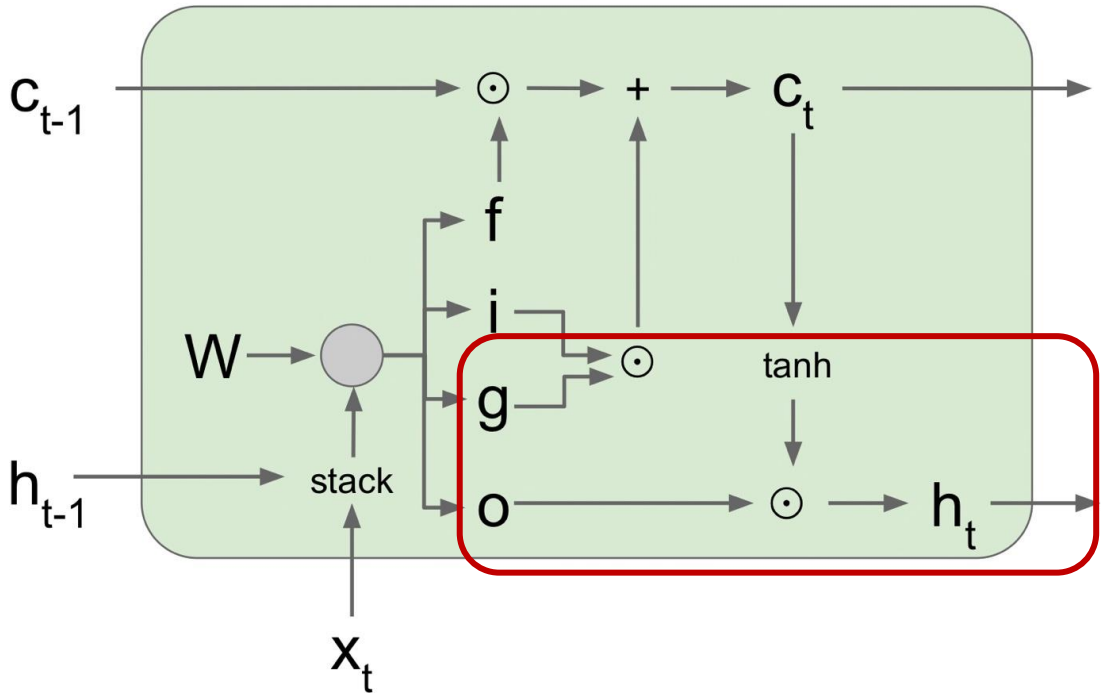
LSTM 의 순전파

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

6. Ct에 tanh 활성화함수를 씌운 후 output gate와 요소별 곱셈을 통해 ht 도출



03 GRU

Gated Recurrent Unit

GRU

- 기존 LSTM의 구조를 보다 간단하게 개선한 모델
- LSTM: 3개의 gate (forget, input, output) → GRU: reset gate, update gate 2개의 gate
- LSTM: cell state, hidden state → GRU: hidden state 하나로 통합
- 데이터 양이 적을 때는 매개 변수의 양이 적은 GRU, 데이터 양이 많으면 LSTM이 더 낫다고 알려져 있음

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad \text{---(1)}$$ Update Gate: 과거와 현재의 정보를 각각 얼마나 반영할지에 대한 비율 결정

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \quad \text{---(2)}$$ Reset Gate: 이전 hidden state 값을 얼마나 활용할 것인가? 를 결정

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \quad \text{---(3)}$$ Candidate: 현 시점의 정보 후보군을 계산하는 단계.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad \text{---(4)}$$

03 GRU

Gated Recurrent Unit

GRU

- 기존 LSTM의 구조를 보다 간단하게 개선한 모델
- LSTM: 3개의 gate (forget, input, output) → GRU: reset gate, update gate 2개의 gate
- LSTM: cell state, hidden state → GRU: hidden state 하나로 통합
- 데이터 양이 적을 때는 매개 변수의 양이 적은 GRU, 데이터 양이 많으면 LSTM이 더 낫다고 알려져 있음

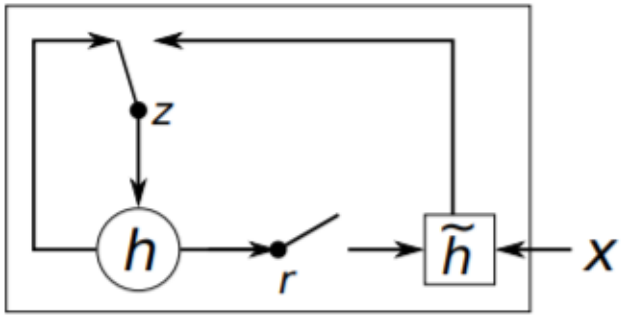


Figure 2: An illustration of the proposed hidden activation function. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$. The reset gate r decides whether the previous hidden state is ignored. See Eqs. (5)–(8) for the detailed equations of r, z, h and $\tilde{h}$.

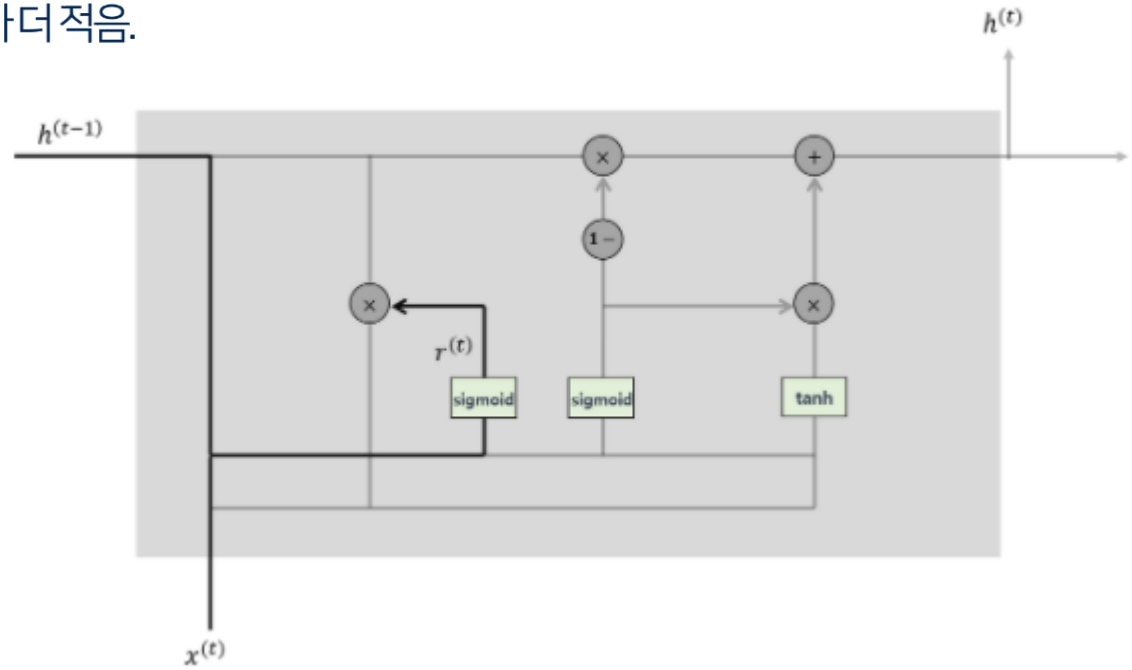
Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua(2014)

03 GRU

Gated Recurrent Unit

GRU의 장점

- LSTM에 비해 더 간단한 구조
- 마지막 출력값에 활성화함수 적용하지 않음.
- 학습할 파라미터가 더 적음.



04 RNN Language Model

RNN 언어 모델

기존 언어 모델의 한계

n-gram

- ✓ Lack of Generalization(Sparsity problem): 한번도 본 적 없는 단어(or n-gram)에 대해서는 처리 불가
- ✓ 단어 간 유사도 고려 X

⇩

NNLM

- ✓ Fixed length input: 다음 단어를 예측하기 위해 정해진 n개의 이전 단어만 참고

⇩

RNN LM

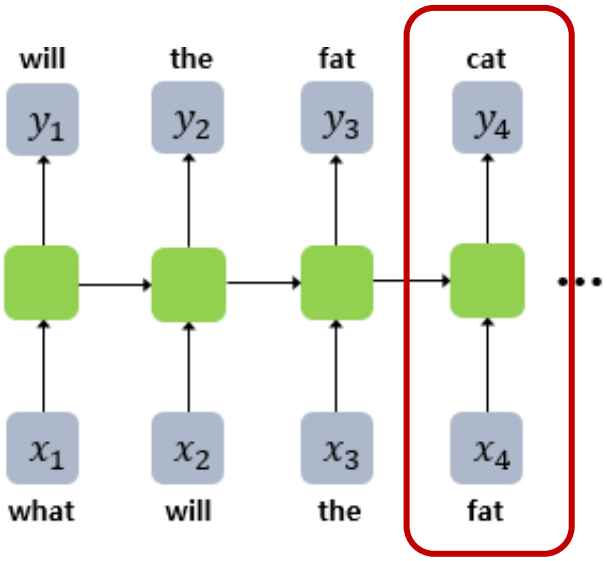
- ✓ Time step 개념의 도입으로 입력 길이를 유동적으로 정할 수 있음

04 RNN Language Model

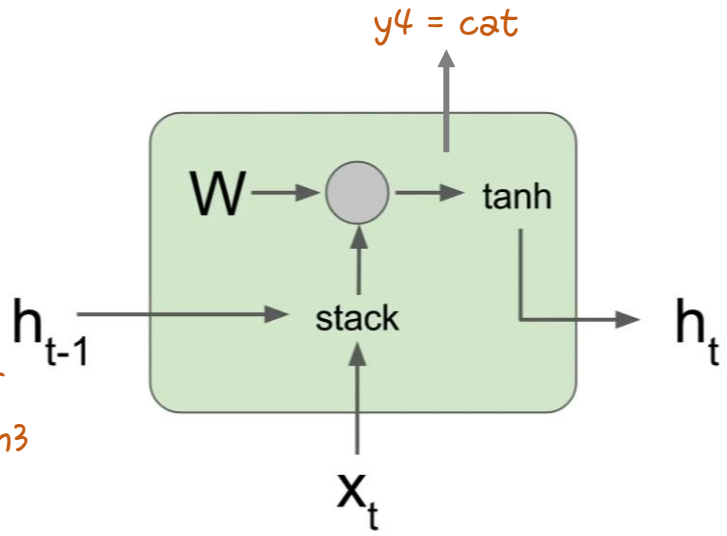
RNN 언어 모델

RNN LM

- 이전 시점 단어들 + 현재 시점의 단어 => 다음 단어 예측



(what, will, the)를 요약한
이전 시점의 hidden state h3

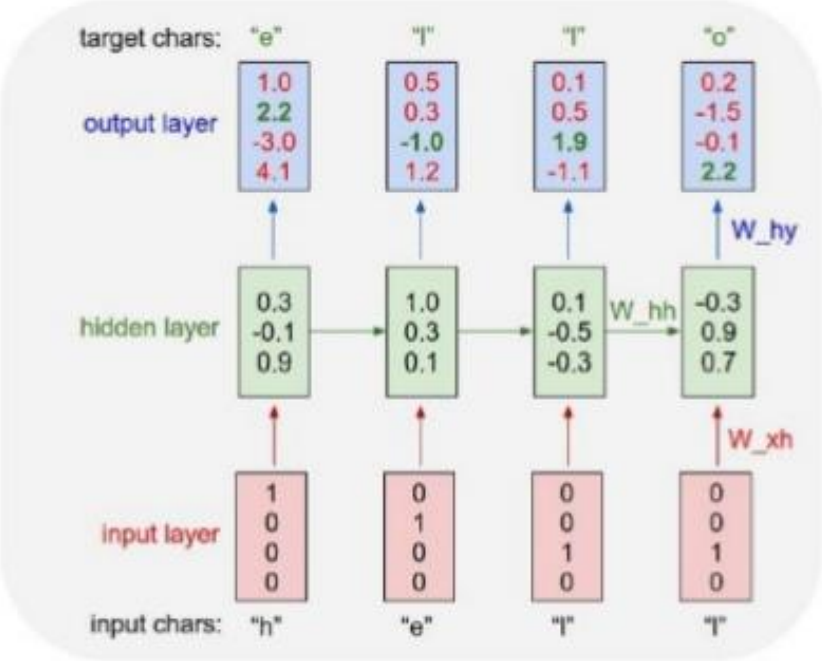


04 RNN Language Model

RNN 언어 모델

교사 강요(teacher forcing)

- 이전 시점 단어들 + 현재 시점의 단어 => 다음 단어 예측



학습 단계

- 출력: 각 글자별 '확신' 정도
- 녹색 값이 높고, 빨간 색 값이 낮도록 매개변수 W_*를 조정(학습)

테스트(추론) 단계

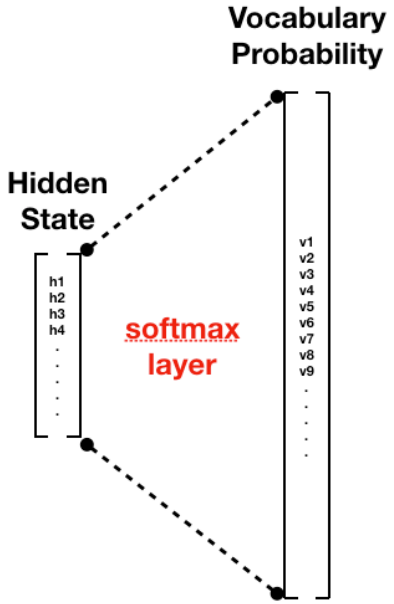
- 한 글자를 입력
- 글자별 확률값이 출력됨
- 이 확률분포에서 글자를 하나 샘플링
- 선택한 글자를 다음 입력으로 사용

04 RNN Language Model

RNN 언어 모델

RNN LM 학습시키기

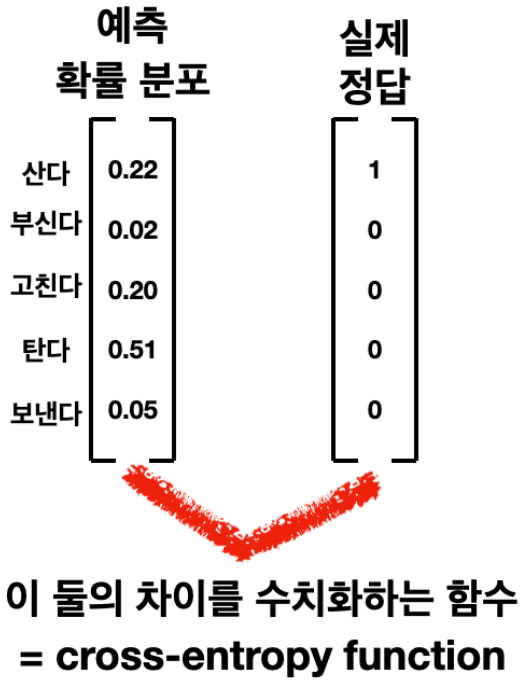
- 학습 데이터에 존재하는 단어들 중 빈도수가 일정 이상인 단어들을 가지고 vocabulary 리스트
- Softmax layer로 vocab 각각의 점수 계산
- cross entropy function으로 비용 함수 계산



나는 자전거를 _____

빈칸에 들어갈 다음 단어는?

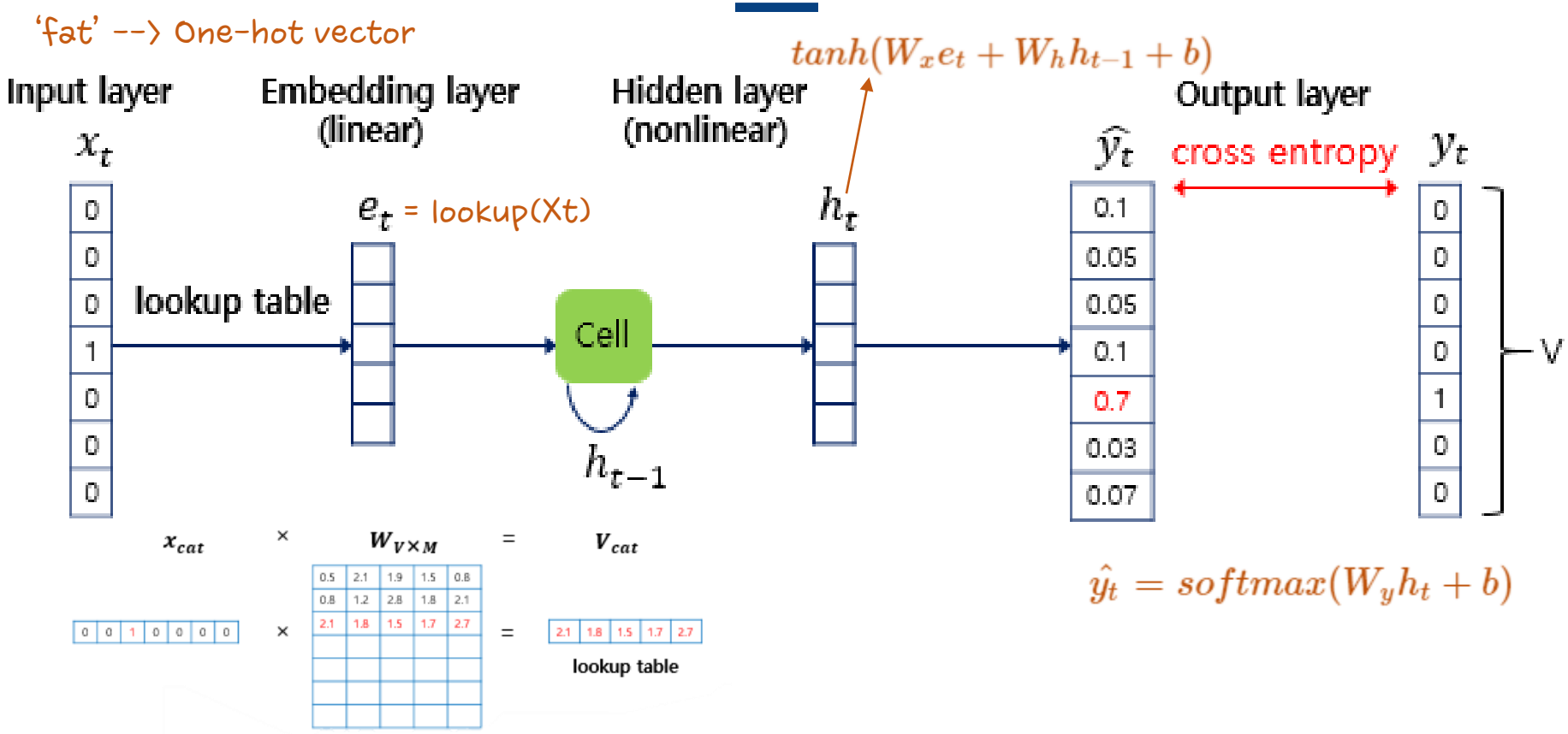
산다	0.22
부신다	0.02
고친다	0.20
탄다	0.51
보낸다	0.05



04 RNN Language Model

RNN 언어 모델

RNN LM 정리



- 역전파 과정에서 가중치 행렬들(W), 임베딩 벡터 (E) 학습됨

실습 (●'◡'●)

Text generation using RNN

NLP-Week4

과제

과제 제출

Text generation using LSTM

18기_김승하_week4