23 하계 학부연구생 프로그램

ch09. 순환 신경망

12223547 박혜민

순환 신경망으로 리뷰 감정 분류하기

01

RNN 기본 구조

- 순차 데이터
- 입력 (원-핫 인코딩/단어 임베딩)
- 순환층
- 출력

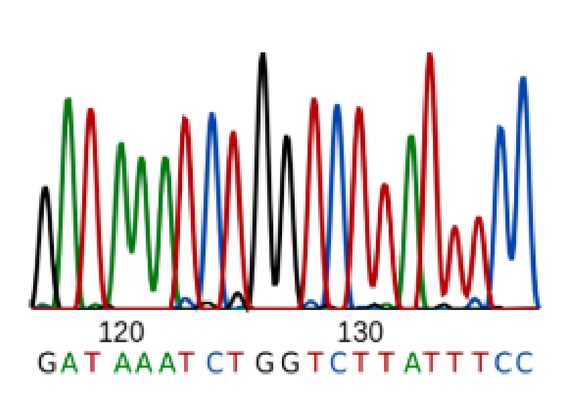
02

LSTM과 GRU셀

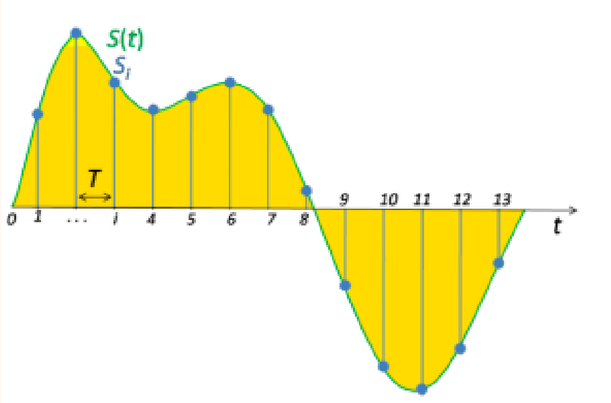
- LSTM 구조, 장점
- GRU 구조, 장점

순차 데이터

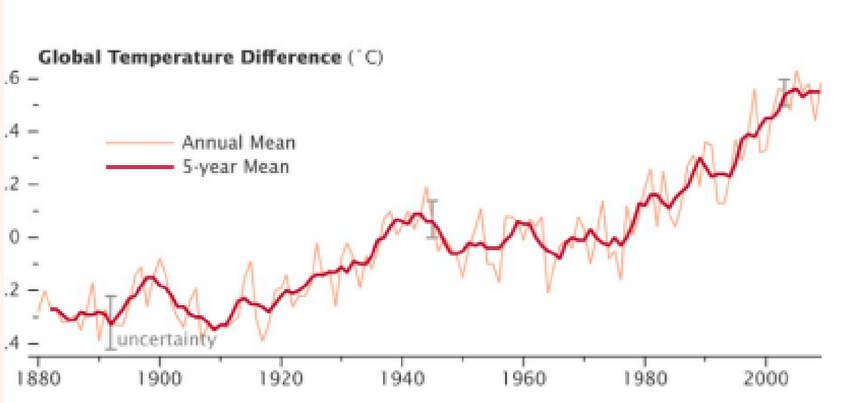
순서가 의미가 있으며, 순서가 달라질 경우 의미가 손상되는 데이터



DNA 염기 서열



세계 기온 변화



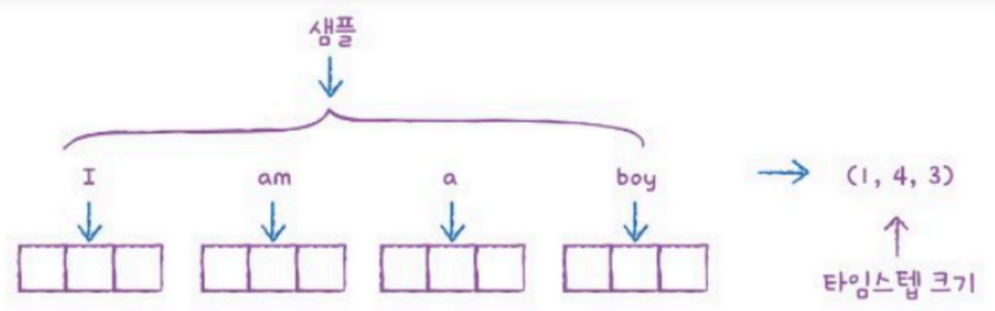
샘플링된 소리 신호

'I am a boy' ≠ 'am I a boy'

RNN 입력

- 컴퓨터는 숫자 데이터만을 처리하므로 텍스트를 숫자로 변환
- 데이터 전달

1. 문장을 단어로 분리(토큰화)



2. 각 단어를 하나의 정수로 매핑(동일한 단어면 동일한 정수)
3. 입력의 2차원 배열에서 빈 공간을 만들지 않기 위해 단어 개수 통일
 - 0으로 패딩/자르기

```
#여섯번째 샘플 출력
print(train_seq[5])
```

```
[  0   0   0   0   1   2 195  19  49   2   2 190   4   2 352   2 183  10
  10  13  82  79   4   2  36  71 269   8   2  25  19  49   7   4   2   2
```

- 크기 : 3차원(샘플 개수, 시퀀스 길이, 단어표현)
- 단어 표현

1. 원-핫 인코딩
2. 단어 임베딩

RNN 입력

원-핫 인코딩

- 입력에서 큰 정수가 큰 활성화 출력을 만드는데, 크기에 의미가 없는 정수들을 주입하면 문제 발생
- 크기 속성을 없애면서 각 정수를 고유하게 표현하는 방식 이용

<방식>

- 정수 x -> x+1번째 원소만 1이고 나머지 0인 배열 생성

  ex) 총 50개의 단어 -> 배열 크기 500

  ex) 10 -> [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ..]

- 케라스의 utils.to-categorical 함수로 편리하게 가능

<단점>

- 단어 표현의 크기가 매우 큼
- 단어를 구분할 뿐, 배열을 통해 추가적인 정보 얻을 수 없음

RNN 입력

단어 임베딩

- 어희 사전의 정수값을 실수배열로 변환
- 결과 벡터의 크기가 원-핫 인코딩보다 매루 작아 메모리 효율적으로 사용

<방식>

- 초기에 랜덤 초기화, 훈련을 통해 학습
- 빈도 수에 따라 같이 언급되는 단어에 비슷한 숫자를 매김

<원-핫 인코딩 vs 단어 임베딩>

Index	Word	Ont-hot vector	Embedding vector
0	영웅은	[1 0 0 0 0]	[0.1 4.2 1.5 1.1 2.8]
1	죽지	[0 1 0 0 0]	[1.0 3.1 2.5 0.7 1.1]
2	않아요	[0 0 1 0 0]	[0.3 2.1 1.5 2.1 0.1]
3	너만	[0 0 0 1 0]	[2.2 1.4 0.5 0.9 1.1]
4	빼고	[0 0 0 0 1]	[0.7 1.7 0.5 0.3 0.2]

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 8)                 4072
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 16)           8000

simple_rnn (SimpleRNN)       (None, 8)                 200
```

순환층

<타임 스텝>

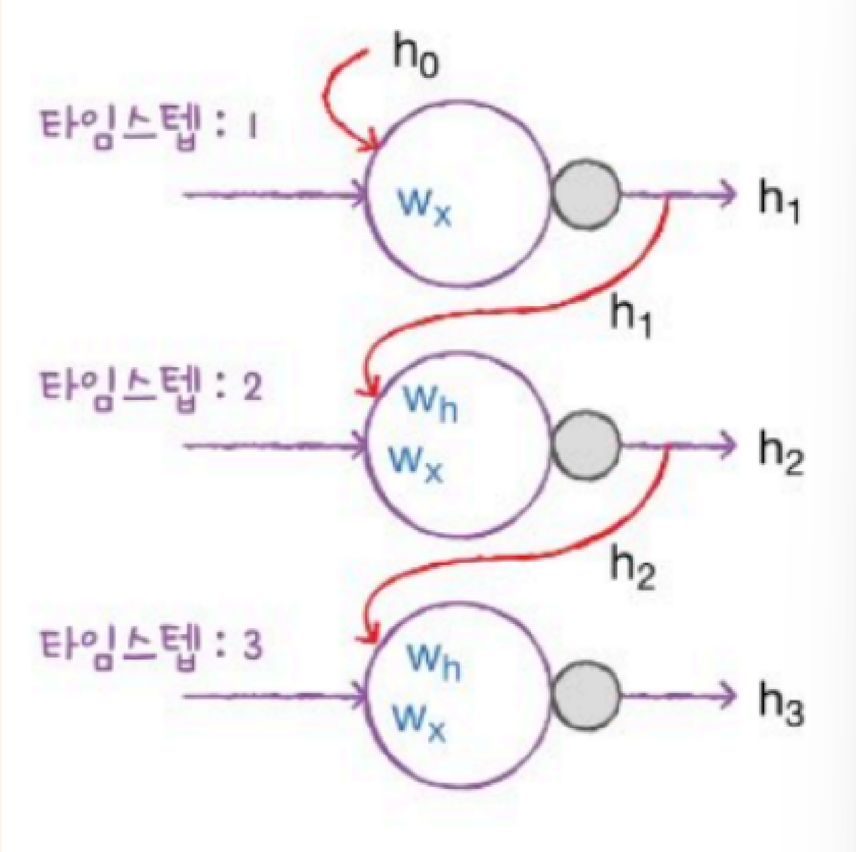
- 순환 신경망에서 샘플 처리하는 한 단계

<기본 과정>
입력이 A, B, C 일 때

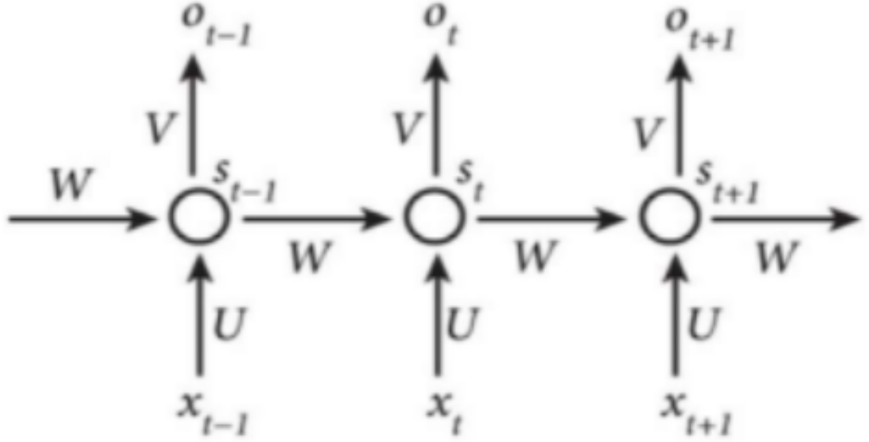
1. A 출력 , 출력 결과가 다시 뉴런으로 들어감
2. A출력 결과와 B를 사용하여 출력(A에 대한 정보 포함)
3. B출력 결과와 C를 사용하여 출력(A, B에 대한 정보 포함)

- 두 개의 가중치 존재

1. 이전 타임스텝의 은닉 상태(출력)에 곱해지는 가중치 (Wh)
2. 현재 입력에 곱해지는 가중치(Wx)

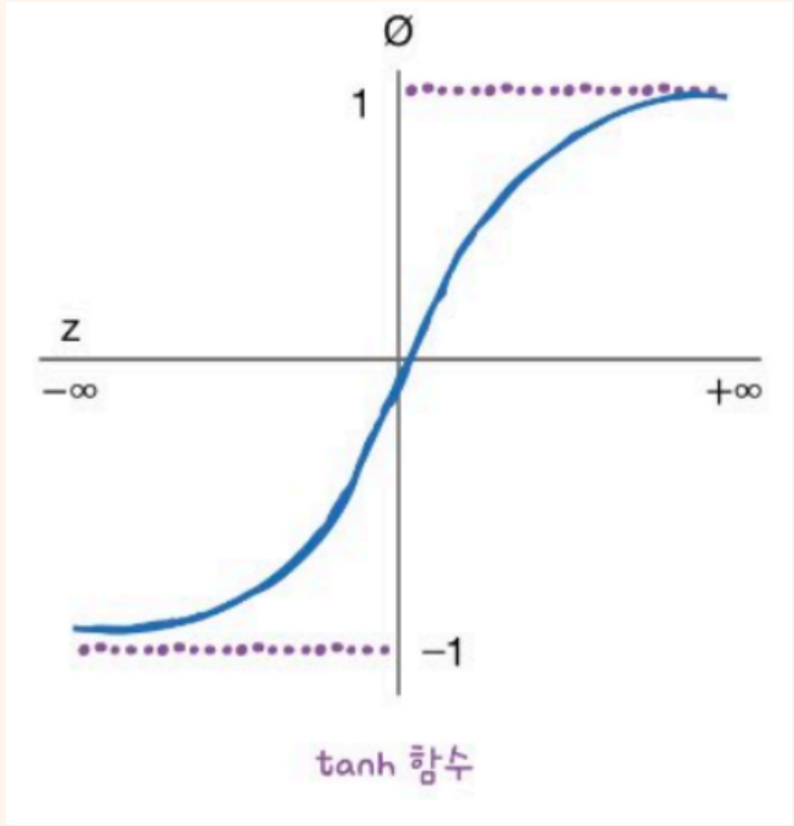


순환층



$$s_t = \tanh(Ux_t + Ws_{t-1})$$

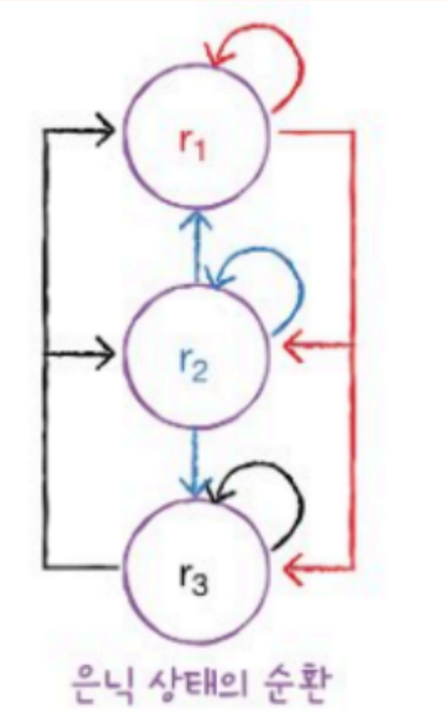
$$\hat{y}_t = softmax(Vs_t)$$



- 활성화 함수로 tanh 사용
  - U,V,W : 가중치, x: 현재 시점의 입력 값, s: 은닉층의 메모리 셀의 출력 값, y : 해당 시점의 출력 값.

순환층

- 한 뉴런의 은닉상태가 자신 뿐만 아니라 다른 뉴런들에도 모두 전달



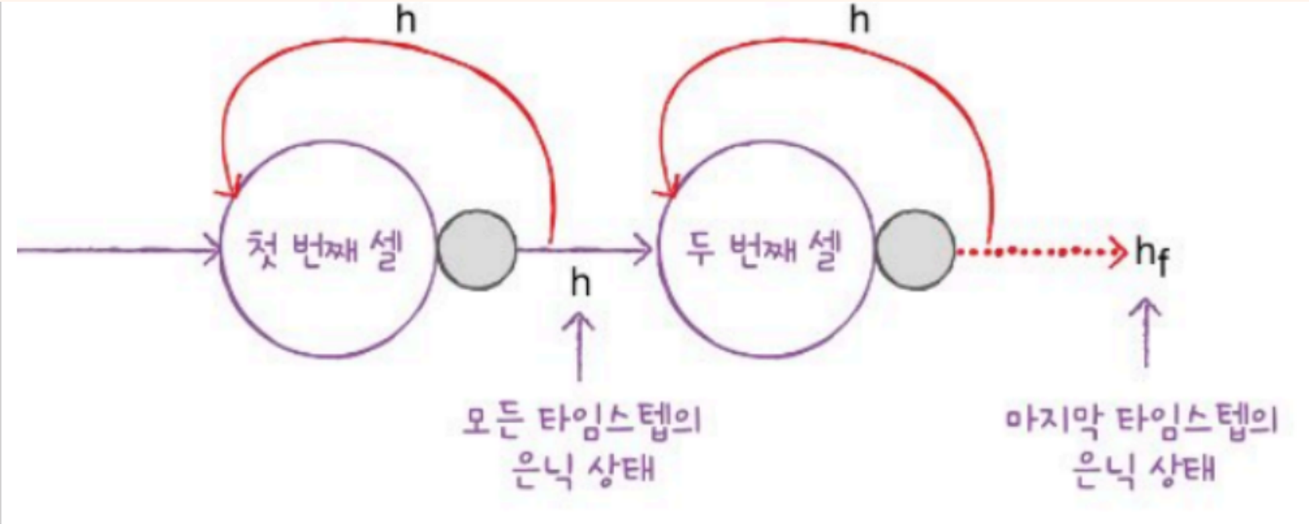
-> 가중치 크기 : (특성 수) X (셀의 뉴런 수) + (셀의 뉴런 수) X (셀의 뉴런 수)(

순환층

출력

심층 RNN

- 마지막 셀을 제외한 다른 모든 셀은 모든 타임스텝의 은닉상태 출력
- 마지막 셀만 마지막 타임스템의 은닉 상태만 출력(정보 기억)



- U,V,W : 가중치, x: 현재 시점의 입력 값, s: 은닉층의 메모리 셀의 출력 값, y : 해당 시점의 출력 값.

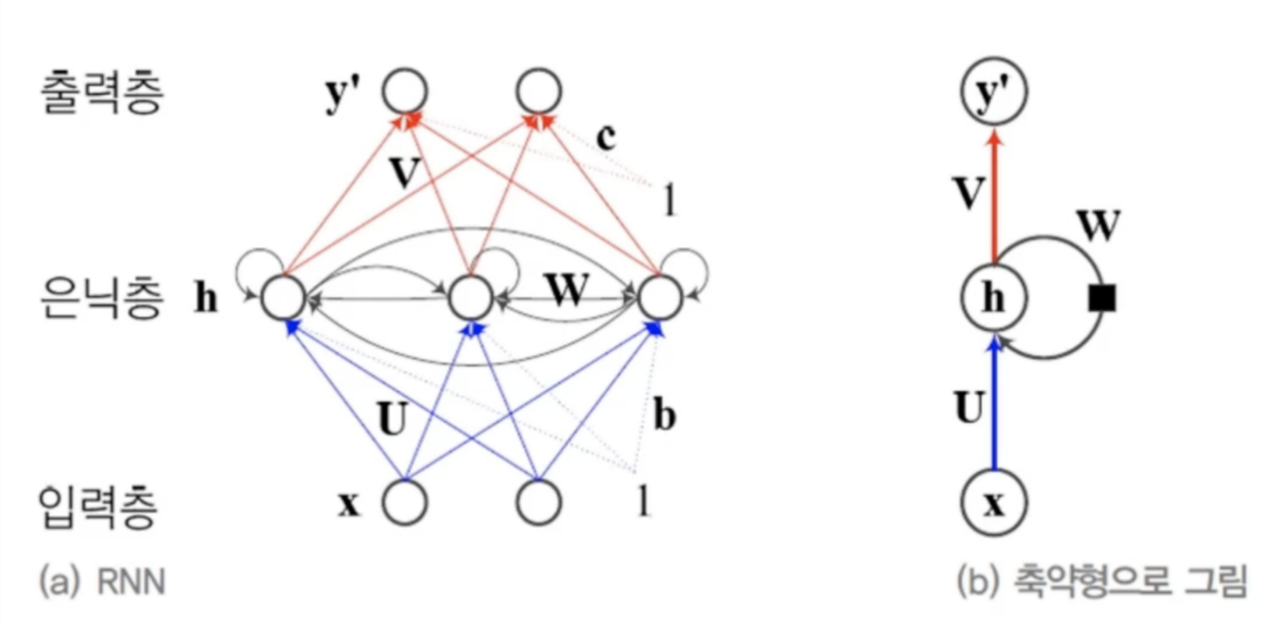
출력층

- 마지막 셀의 출력이 1차원이기 때문에 셀의 출력 그대로 밀집층에 사용
- 다중분류

-> 마지막에 클래스 개수만큼의 뉴런을 가진 밀집층을 두고 소프트맥스적용

- 이진 분류

-> 하나의 뉴런을 두고 시그모이드 적용 (이진)



LSTM과 GRU

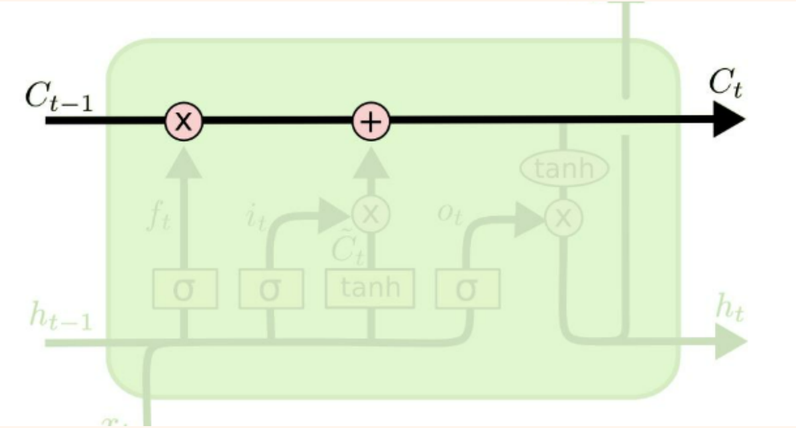
-기본 RNN의 장기 의존성 문제 극복

LSTM(Long Short-Term Memory) 구조

- 단기기억을 오래하기 위해 고안
- 하나의 셀 안에 가중치 곱을 활성화 함수에 통과시키는 셀이 4개 존재

셀 상태(c)

- 은닉 상태와 달리 다음 층으로 전달되지 않고 셀에서 순환되는 값
- Gate에 의해 정보가 추가/제거



LSTM(Long Short-Term Memory) 구조

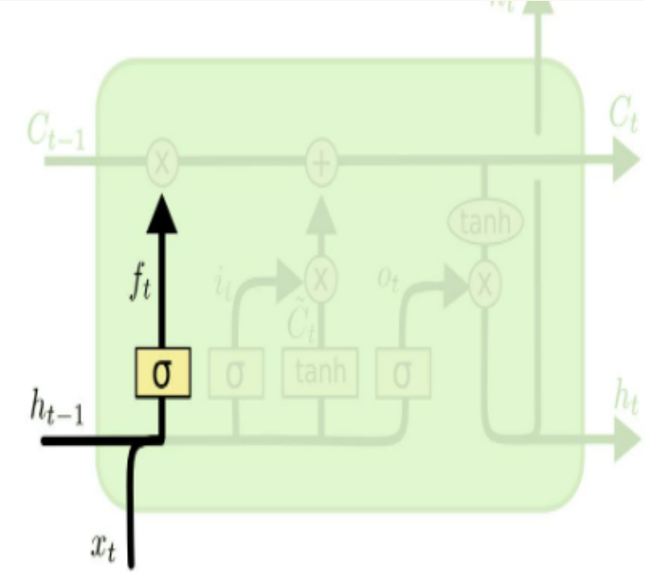
GATE

1. Forget Gate

- 과거 정보 보존 여부를 결정하는 과정
- 입력과 이전 은닉상태의 가중치 곱을 시그모이드 통과시킨 값)

-> 1이면 모든 정보 보존, 0이면 모든 정보 삭제

- 계산 결과를 X 이전 셀 상태에 곱



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

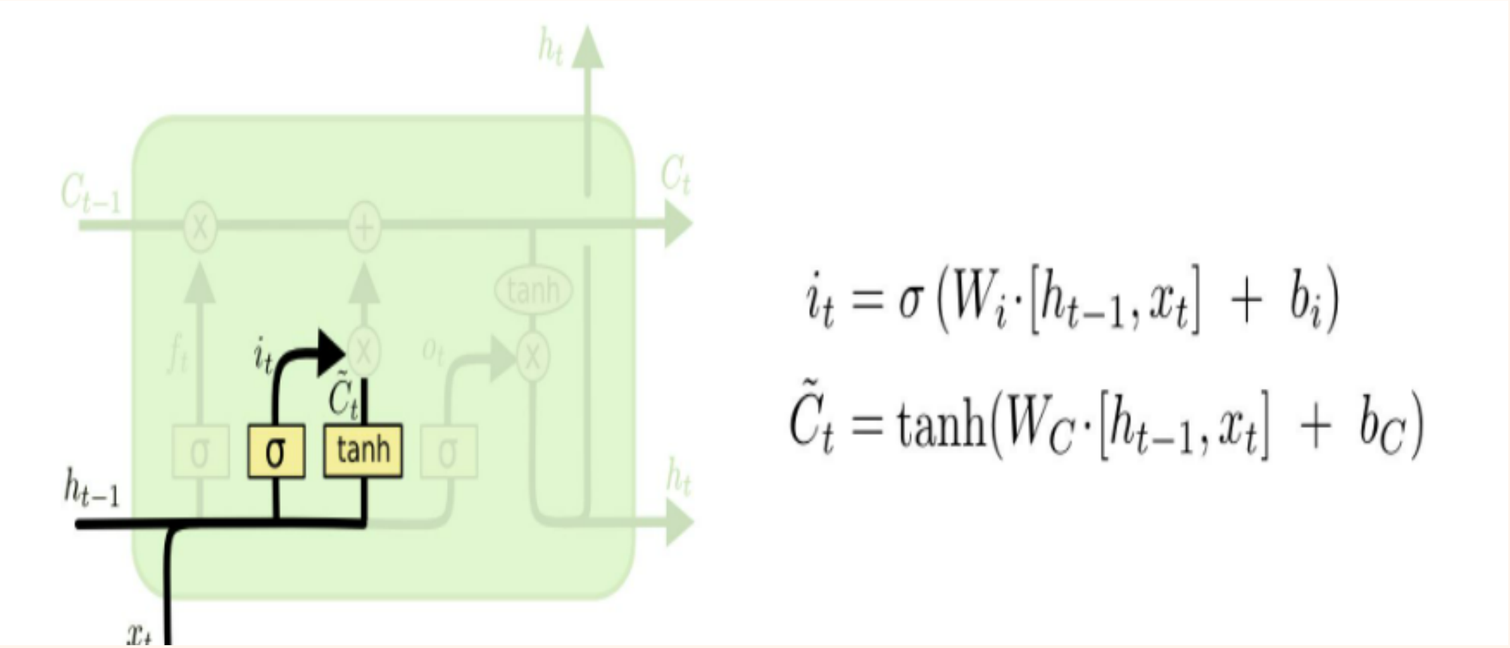
LSTM(Long Short-Term Memory) 구조

GATE

2. Input Gate

- 현재 정보를 기억하기 위한 게이트
- 입력과 이전 은닉상태를 각기 다른 가중치에 곱하여 하나는 시그모이드, 하나는 tanh 함수 통과시킨 결과들의 곱

-> 이전 셀 상태에 더해서 새로운 셀 상태 계산



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

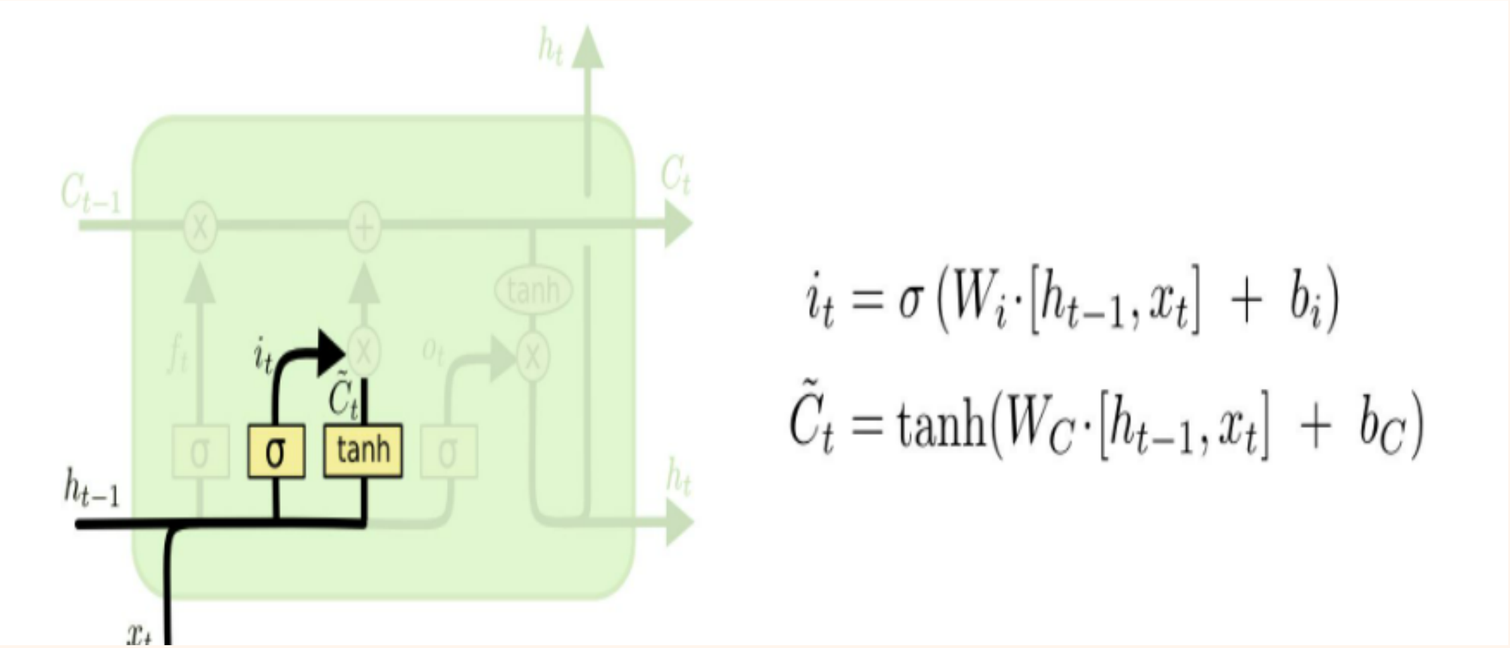
$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] \; + \; b_C\right)$$

LSTM(Long Short-Term Memory) 구조

GATE

3. Output Gate

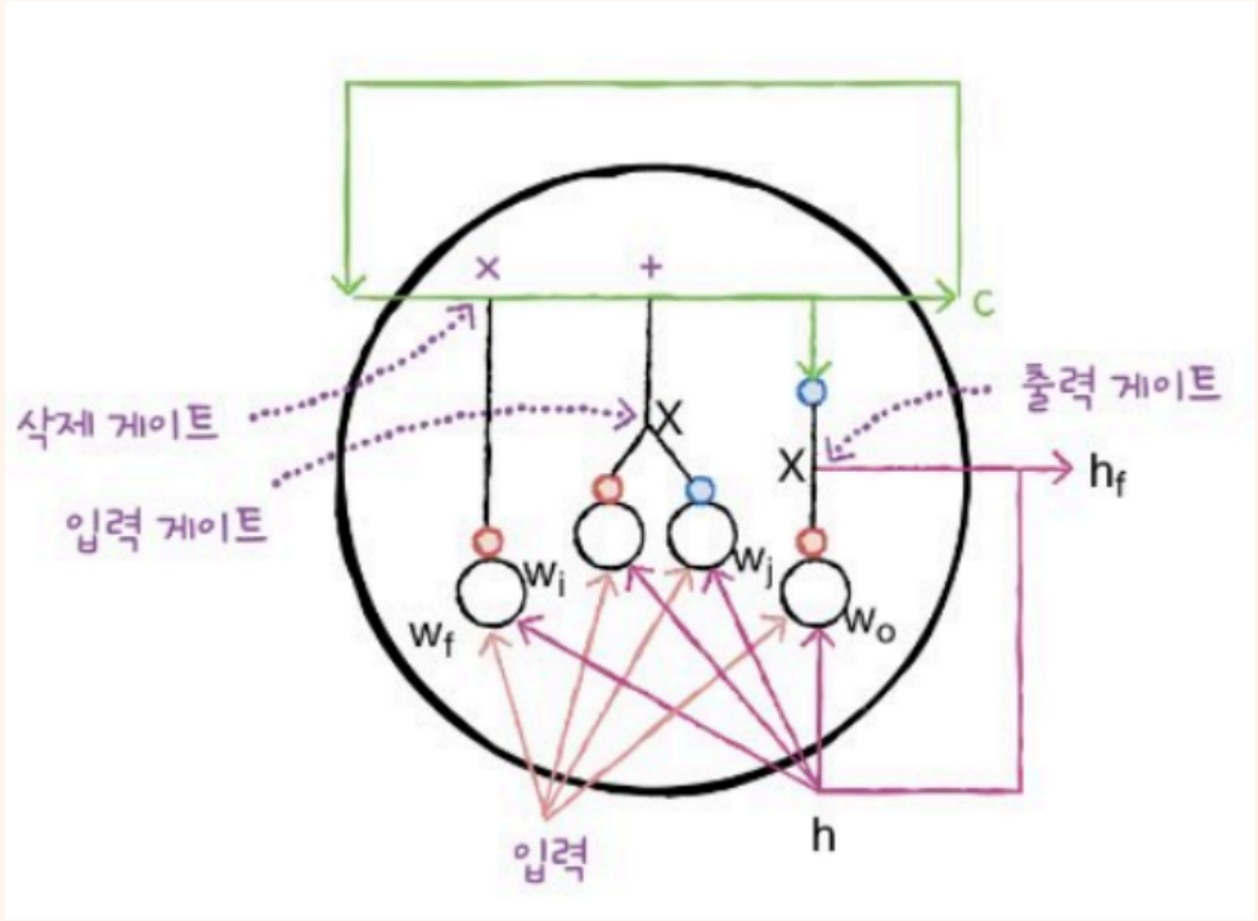
- 최종적으로 얻어진 셀 상태 값을 얼마나 사용할지 결정
- (입력과 이전 은닉상태의 가중치 곱을 시그모이드 활성화 함수 통과시킨 값) X (tanh 활성화 함수를 통과한 셀 상태)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

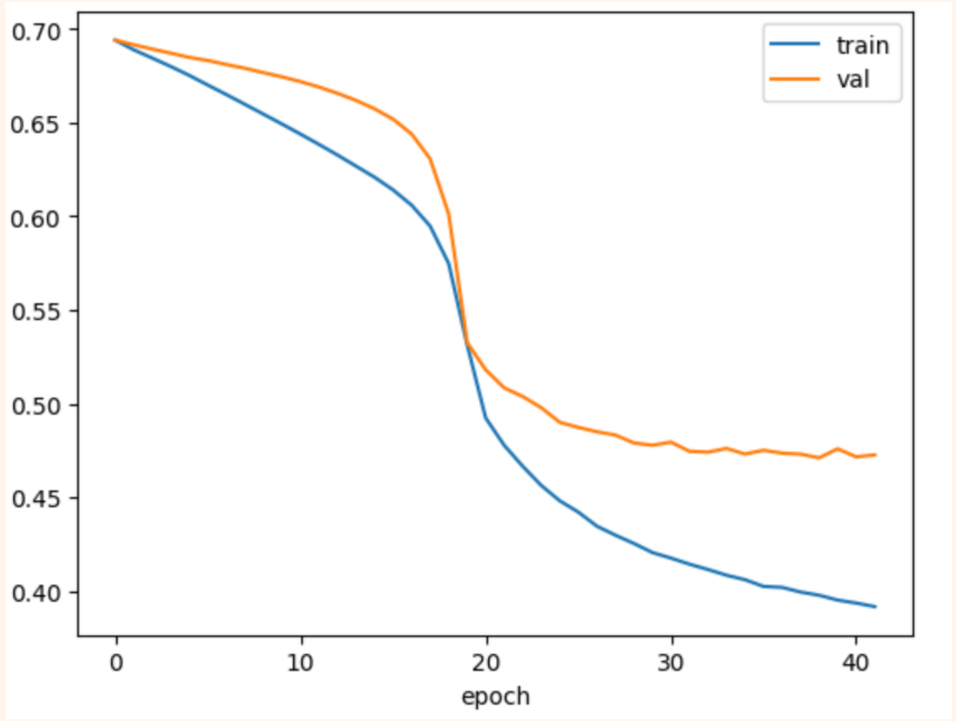
$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] \; + \; b_C\right)$$

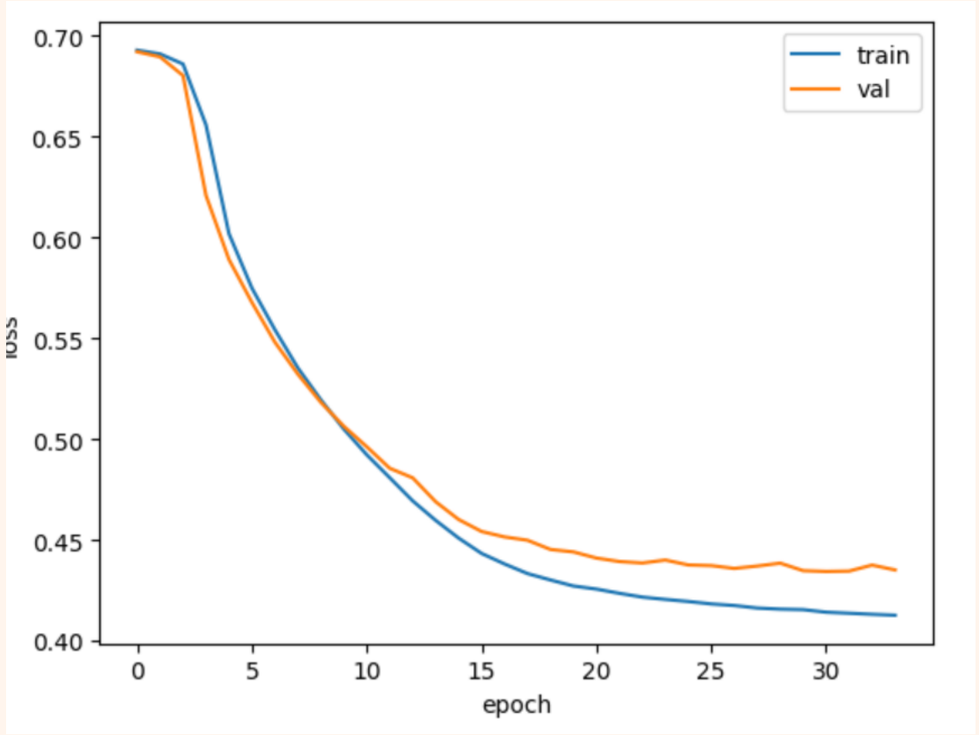
LSTM(Long Short-Term Memory) 구조



LSTM(Long Short-Term Memory) 구조

- 기본 RNN 보다 과대적합 억제





GRU(Gated Recurrent Unit)구조

- LSTM의 간소화 버전(계산량 감소)
- Wz : 삭제 게이트 역할
- Wz 가중치 갖는 셀의 출력을 1에서 뺀 값을 Wg 셀의 출력에 곱 -> 입력되는 정보 제어
- Wr 셀의 출력은 Wg 셀이 사용할 은닉 상태의 정보 제어