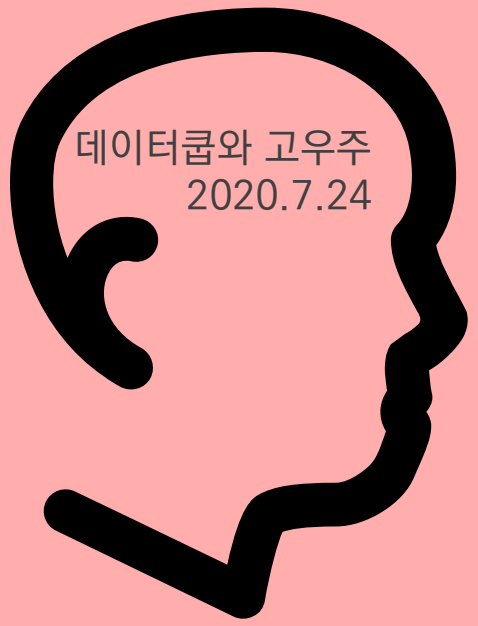
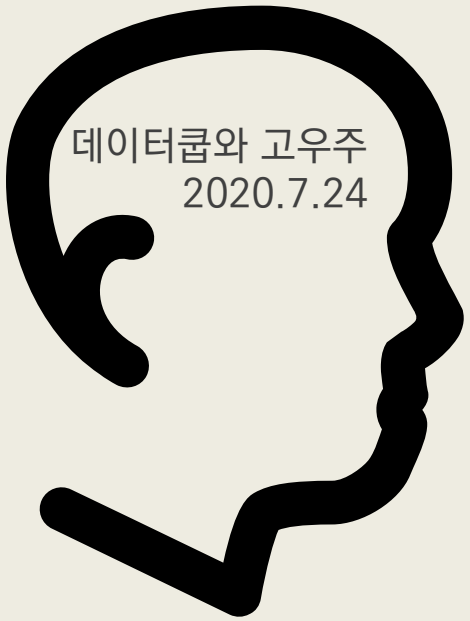
딥러닝 9일차



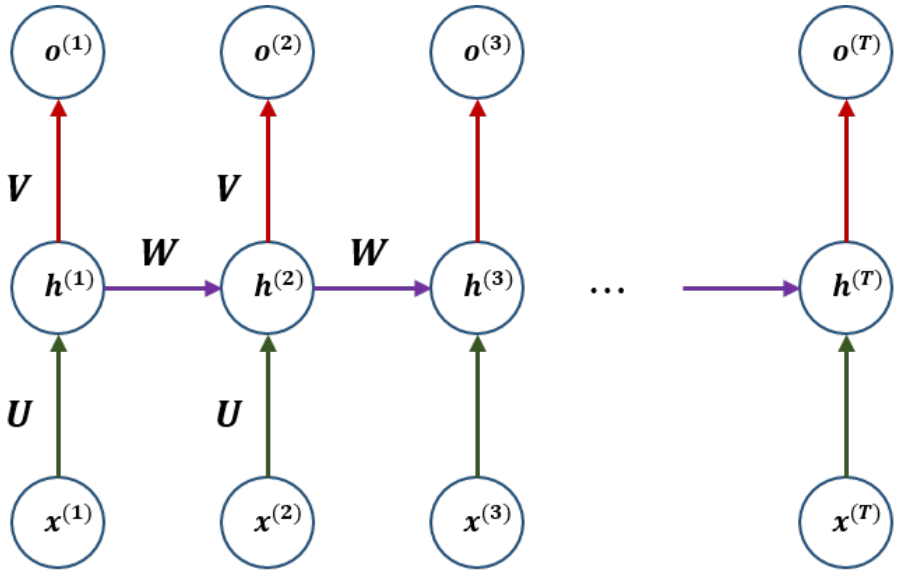
데이터큡와 고우주
2020.7.24

RNN(Recurrent Neural Network

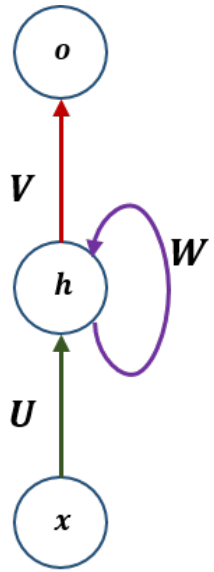


데이터큡와 고우주
2020.7.24

RNN



[Unfolded]



[Folded]

가중치 공유

RNN의 학습 가중치는 시작 층과 도착 층에 따라 크게 세 가지, U, W, V 로 분류

- U: 입력층 → 은닉층
- W: t 시점 은닉층 → t+1 시점 은닉층
- V: 은닉층 → 출력층

가중치 U, W, V 는 모든 시점에서 동일: 가중치 공유

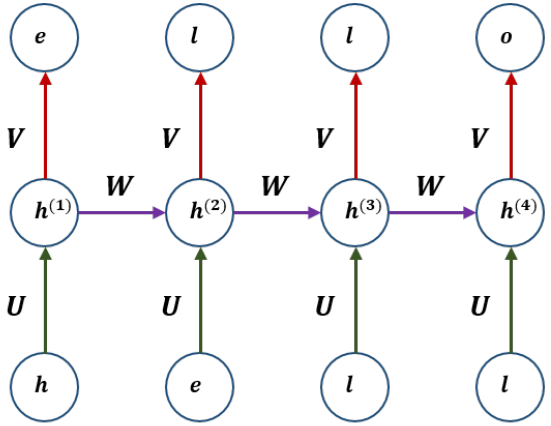
가중치 공유의 이점

- 학습에 필요한 가중치의 수를 줄일 수 있다.
- 데이터별 시간의 길이 T에 유연하다.

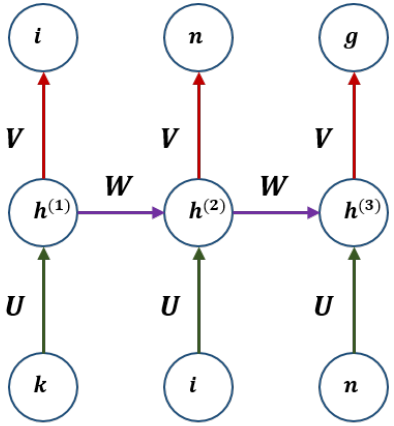
RNN NLP(자연어 처리)

시간의 길이 T에 유연

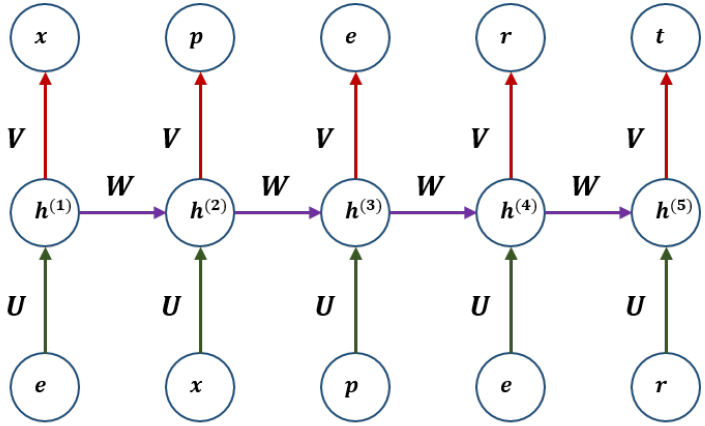
- Ex). 1개의 모형으로 hell의 o를, kin의 g를, exper의 t를 예측 가능
- 같은 가중치를 곱해주고, 은닉층에 단어의 과거 정보가 포함
- 위 예제 단어의 길이 5, 4, 6에 무관하게 같은 모형을 적용



hello



king



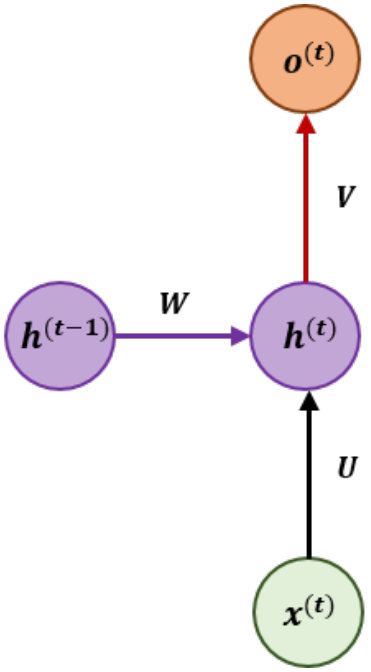
expert

RNN 동작 원리

입력층, 은닉층의 노드의 수가 1개이고, 출력층이 K개 노드

$$x^{(t)} \in \mathbb{R}$$
$$h^{(t)} \in \mathbb{R}$$
$$o^{(t)} \in \mathbb{R}^k$$

가중치 W, U는 스칼라, $V \in \mathbb{R}^{K \times 1}$



1. 은닉층: 은닉층 계산에는 $x^{(t)}$와 $h^{(t-1)}$이 필요,

(τ: hyperbolic tangent)

$$h^{(t)} = \tau\left(a^{(t)}\right)$$
$$a^{(t)} = Wh^{(t-1)} + Ux^{(t)}$$

2. 출력층: 출력층은 ANN 계산과 동일

$$h^{(t)} = \tau\left(a^{(t)}\right)$$
$$o^{(t)} = softmax(Vh^{(t)}$$

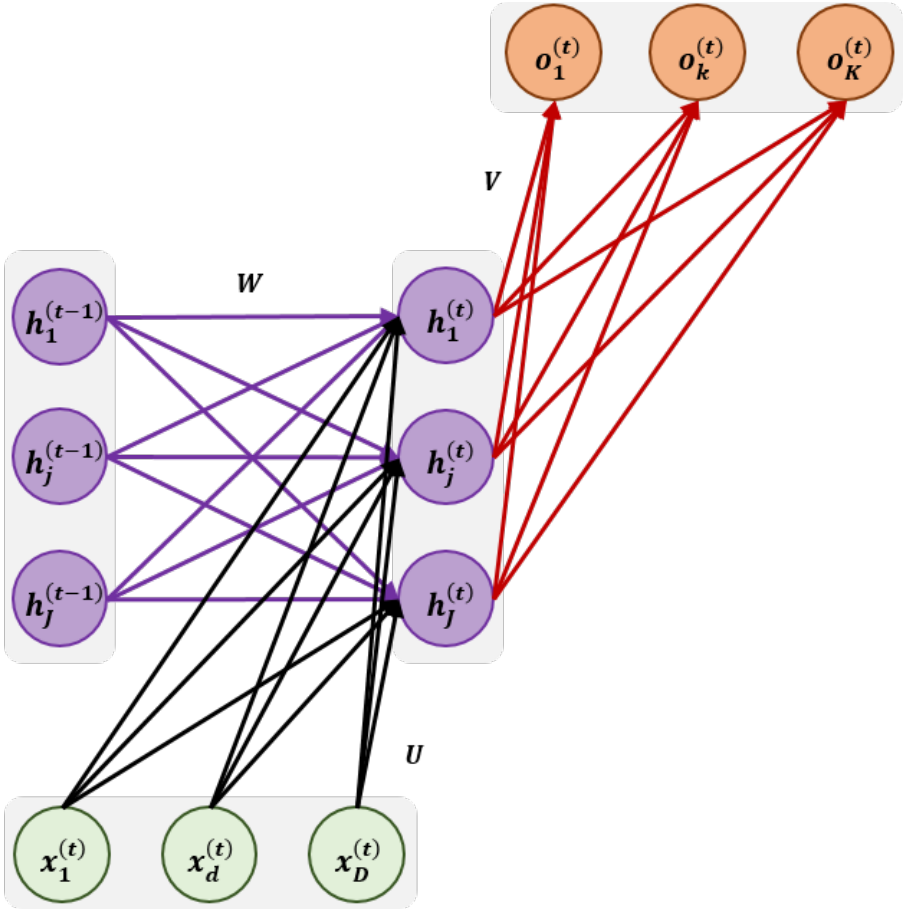
RNN 동작 원리

- x, h, o에서

$$x \in \mathbb{R}^D, h \in \mathbb{R}^J, o \in \mathbb{R}^K$$

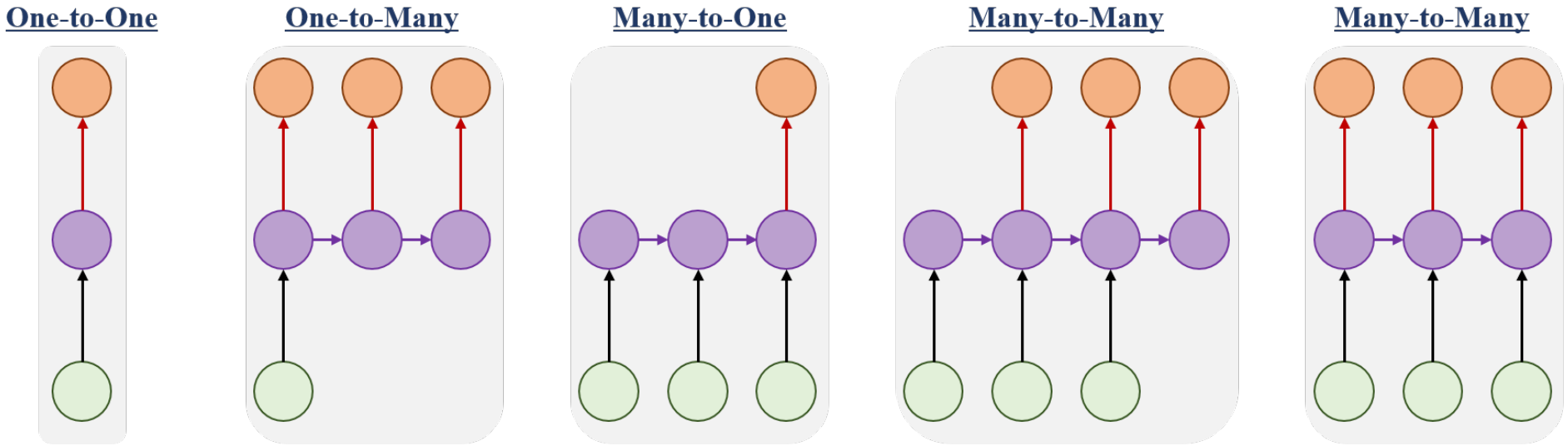
- 각각의 노드 수 D, J, K에서

$$\mathrm{U} \in \mathbb{R}^{J \times D}, h \in \mathbb{R}^{J \times J}, o \in \mathbb{R}^{K \times J}$$



RNN Architectures

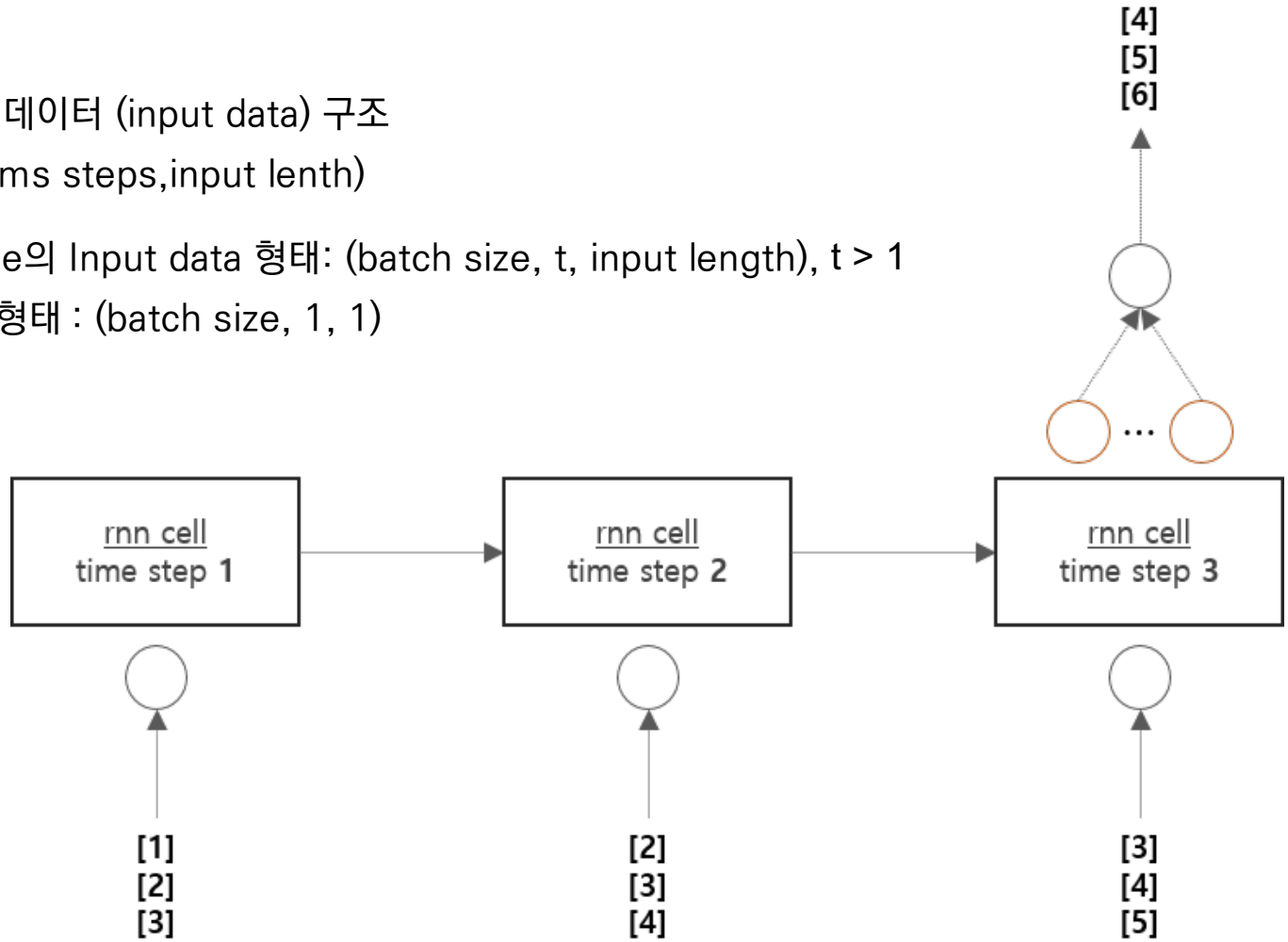
- **One-to-One**: Vanlina Neural Networks, 은닉층이 1층인 신경망 모형
- **One-to-Many**: Image를 입력으로 받아, Image 속 대상에 이름을 붙이는 모형
- **Many-to-One**: word sequence를 입력으로 받아, 감정 분류를 해주는 모형
- **Many-to-Many**: 기계 번역 모형, word sequence를 입력으로 받아, word sequence을 출력
- **Many-to-Many**: 비디오의 frame을 입력으로 받아 frame 속 대상에 이름을 붙이는 모형



Many-to-One

- 기본적인 입력 데이터 (input data) 구조
- (batch size,tims steps,input lenth)
- Many-to-One의 Input data 형태: (batch size, t, input length), t > 1
- Output data 형태 : (batch size, 1, 1)

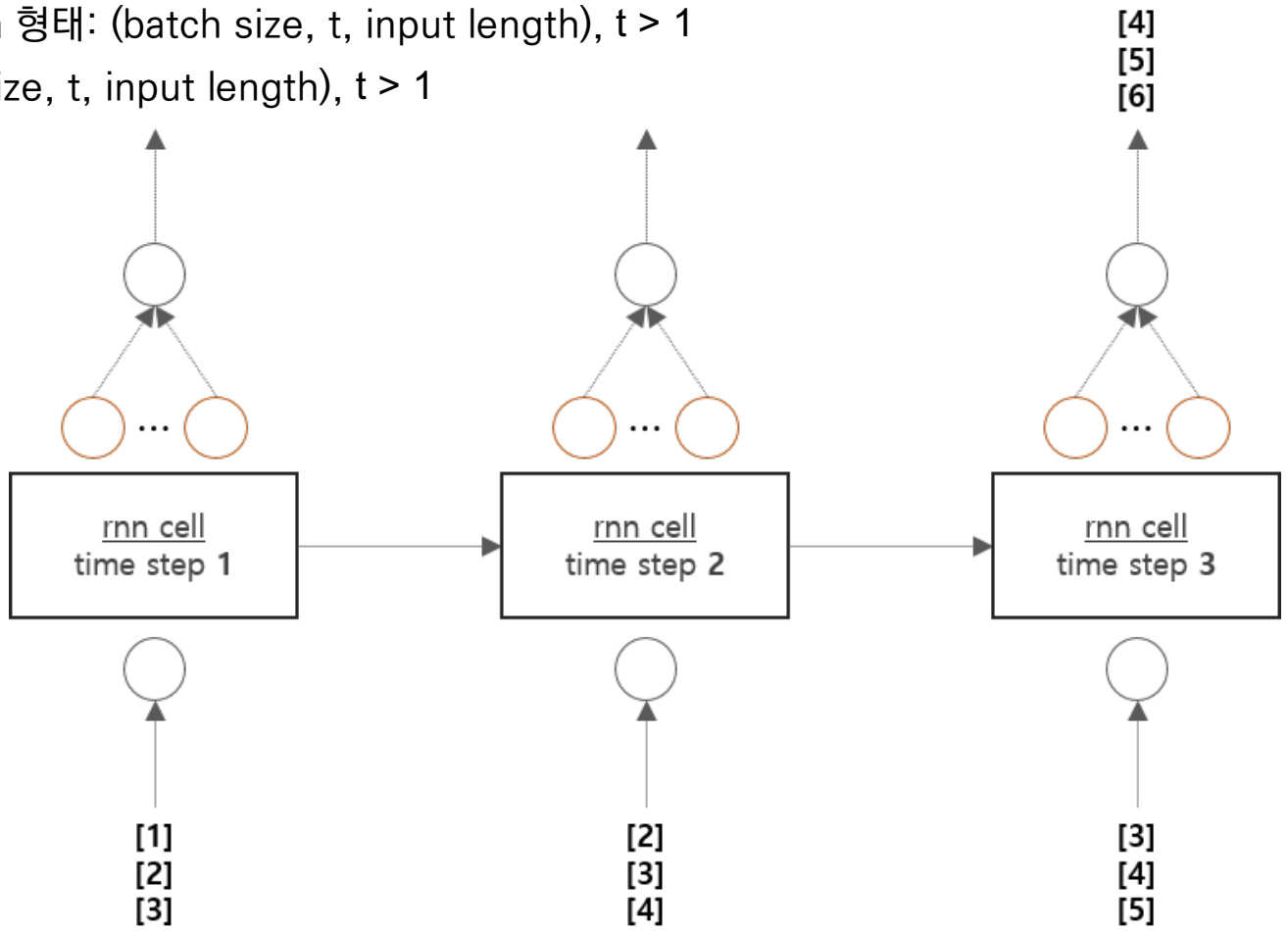
[1,2,3]→[4]
[2,3,4]→[5]
[3,4,5]→[6]



Many-to-Many

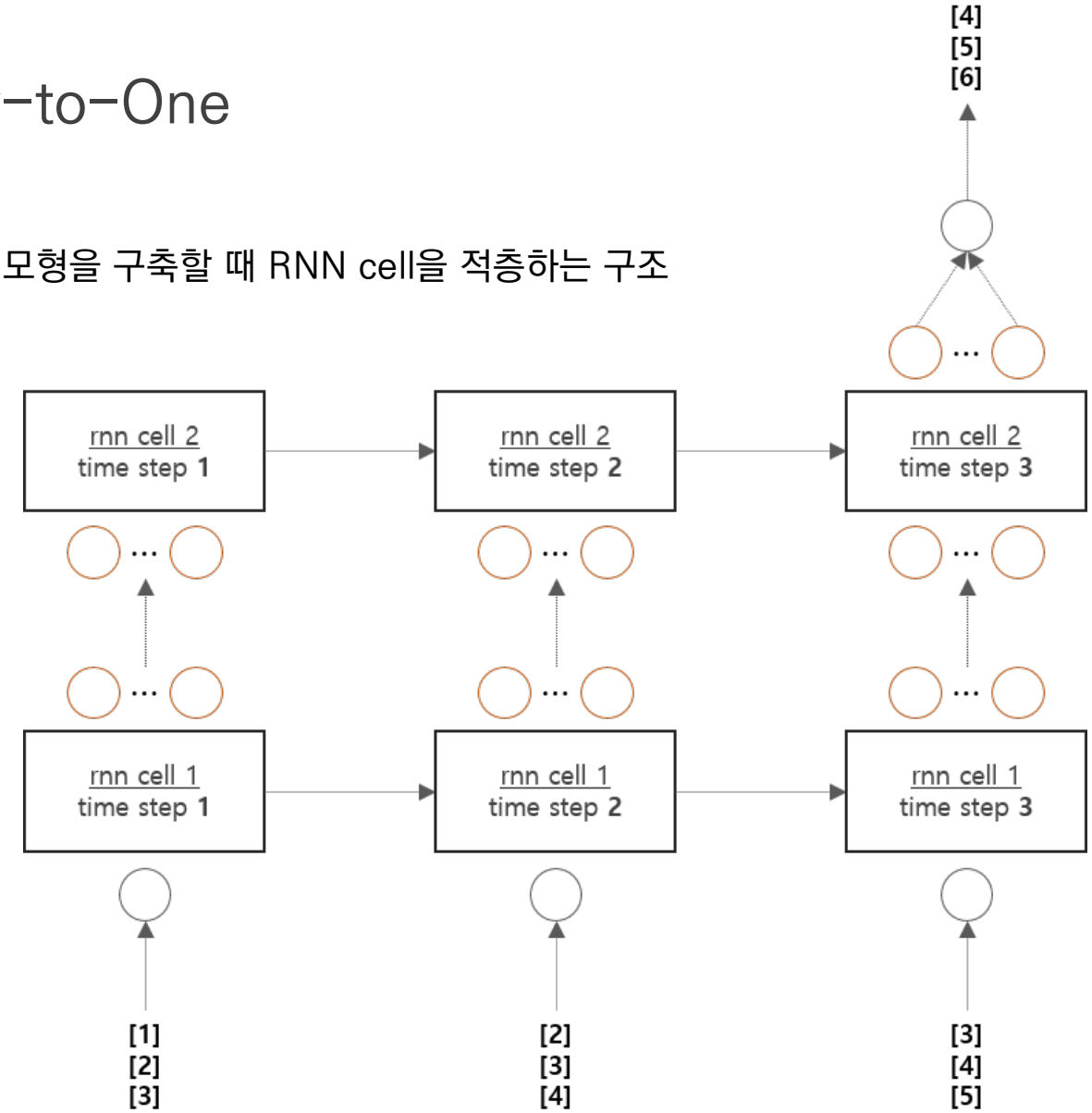
- Many-to-One의 Input data 형태: (batch size, t, input length), t > 1
- Output data 형태 : (batch size, t, input length), t > 1

[1, 2, 3]→[2, 3, 4]
[2, 3, 4]→[3, 4, 5]
[3, 4, 5]→[4, 5, 6]



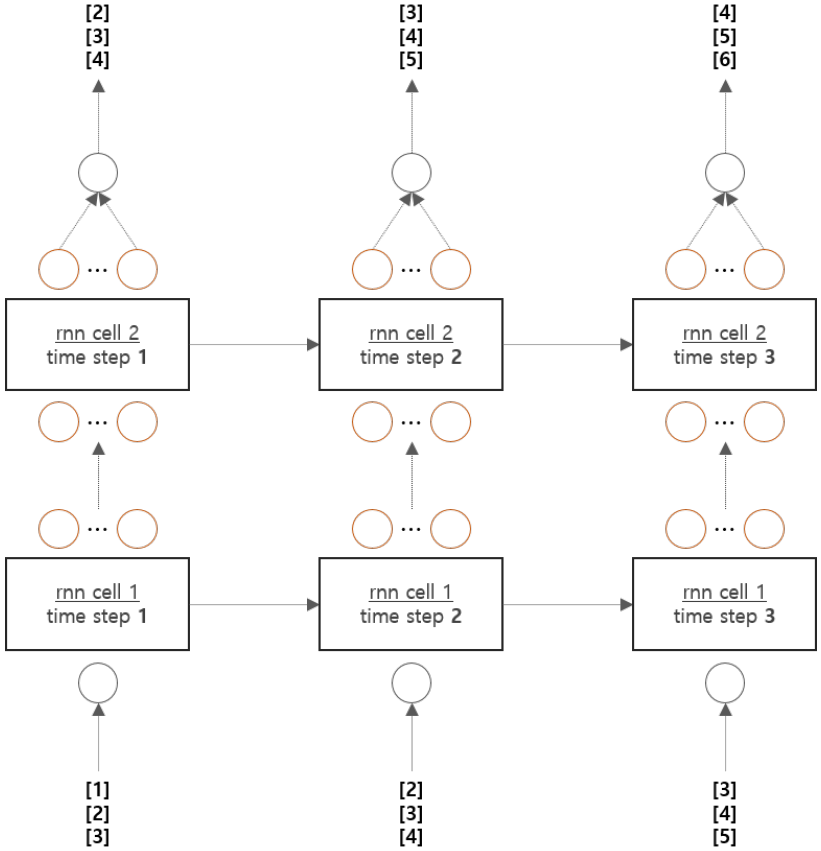
Stacked Many-to-One

Stacked RNNs은 모형을 구축할 때 RNN cell을 적층하는 구조



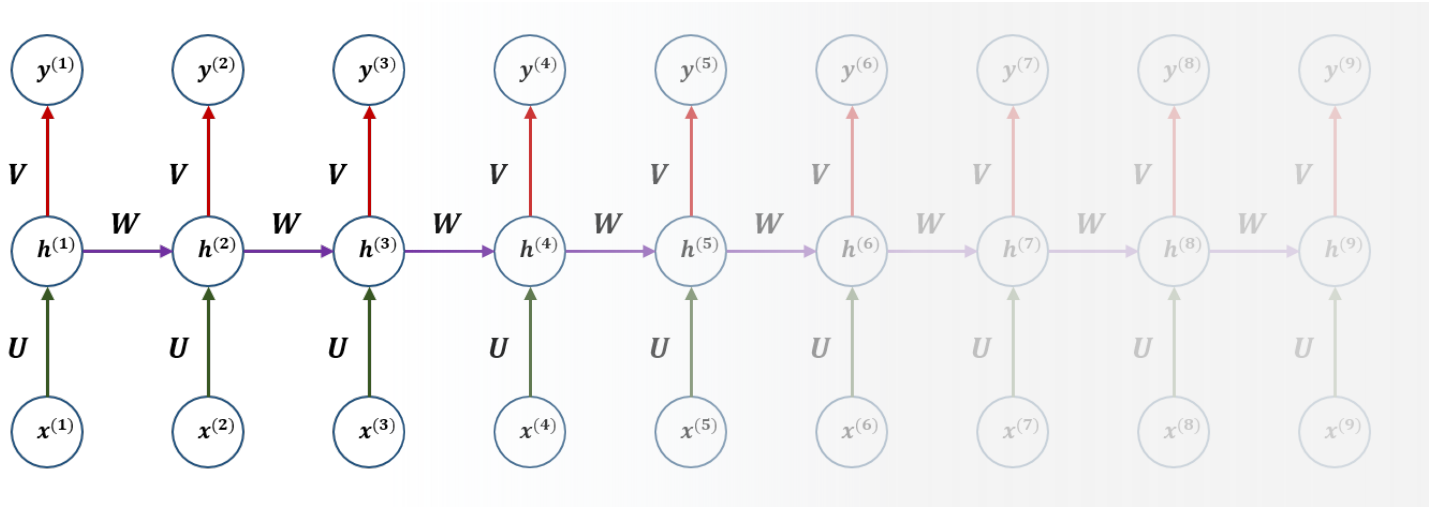
Stacked Many-to-Many

Stacked RNNs은 모형을 구축할 때 RNN cell을 적층하는 구조



return_sequence=True

RNN의 Long-term Dependencies



$$h^{(t)} = \tau\left(Ux^{(t)} + Wh^{(t-1)}\right)$$

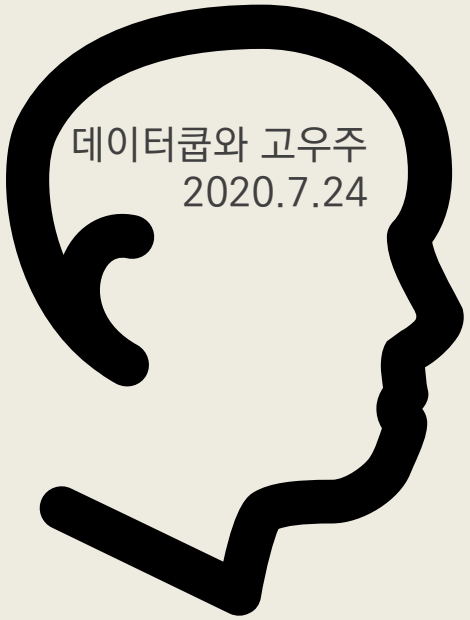
t-2 시점까지 진행 시 h(t)가 tangent hyperbolic(tanh) τ에서 반복적으로 곱해짐

$$h^{(t)} = \tau\left[Ux^{(t)} + W\tau(Ux^{(t-1)} + Wh^{(t-2)})\right]$$

1보다 작은 값이 반복적으로 곱해지기 때문에,

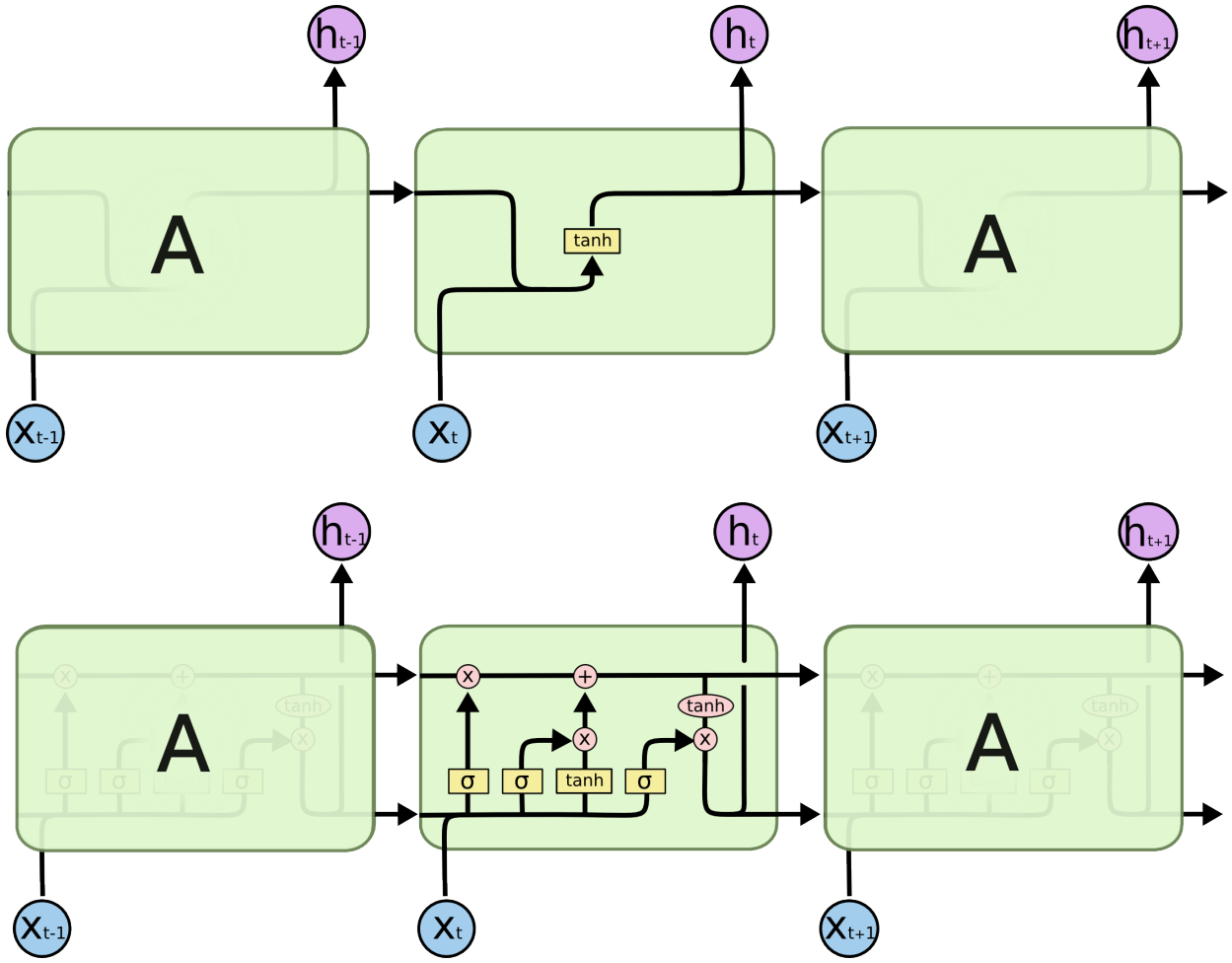
- feed-forward 시 데이터가 뒤로 갈수록 전달 미비
- back-propagation 시 tanh의 함수 기울기가 0으로 수렴하여 경사하강 손실(Vashing Gradients)

LSTM(Long Short-Term Memory)



데이터큡와 고우주
2020.7.24

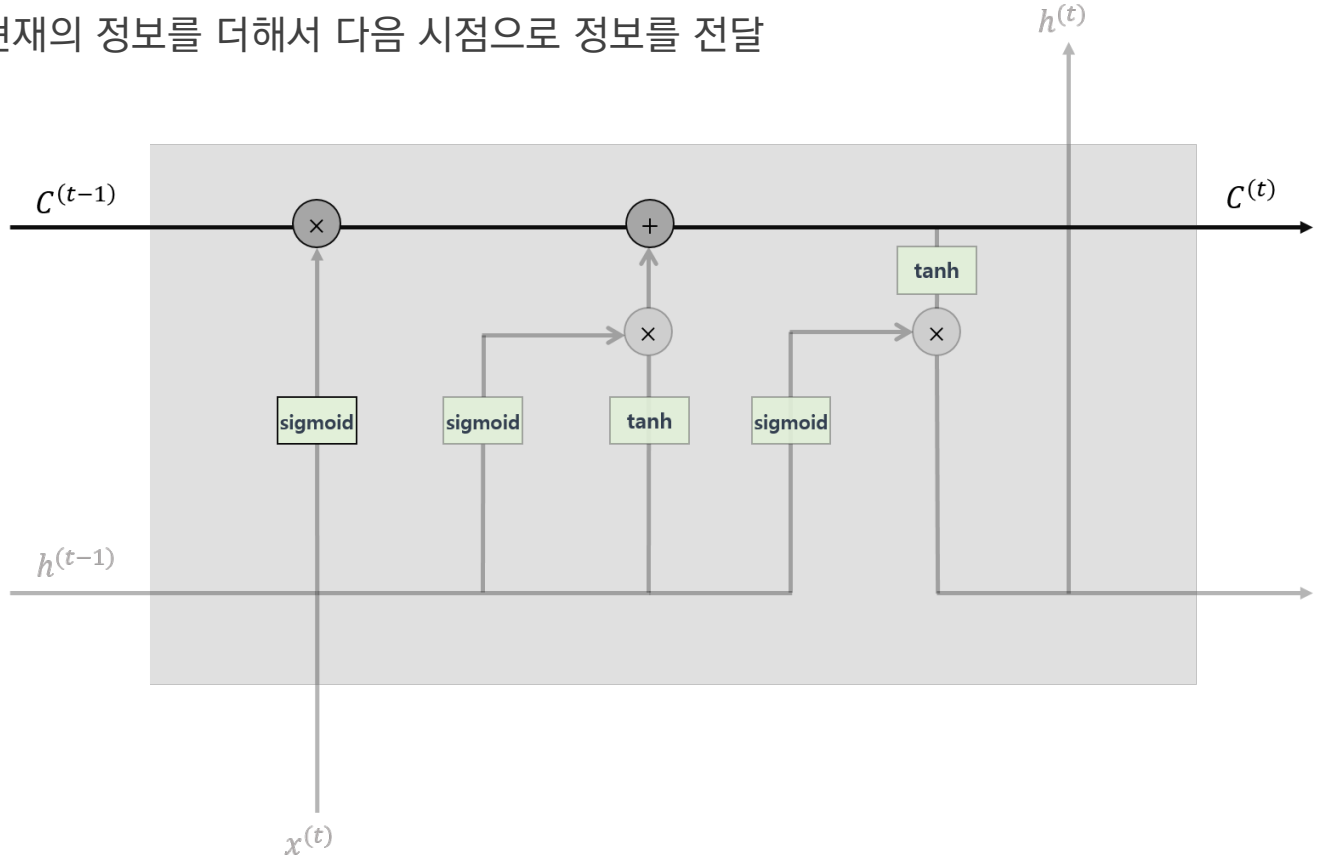
LSTM, RNN 비교


h_{t-1}
h_t
h_{t+1}
A
tanh
A
x_{t-1}
x_t
x_{t+1}
h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
x_{t-1}
x_t
x_{t+1}

LSTM Concept

LSTM Key Concept: 이전 단계의 정보를 memory cell에 저장하여 흘려보내서,

- 현재 시점의 정보를 바탕으로 과거의 내용을 얼마나 잊을지 곱해주고
- 그 결과에 현재의 정보를 더해서 다음 시점으로 정보를 전달



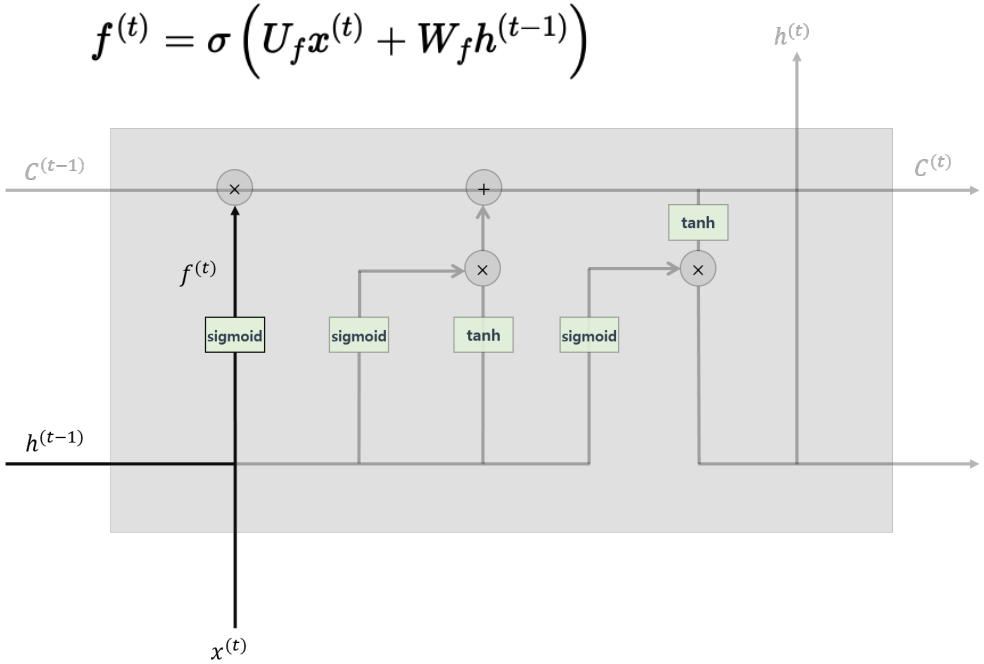
LSTM 기본 원리

1단계. Forget Gate (망각 게이트)

과거의 정보를 얼마나 잊을지 결정하는 게이트

- 현시점의 정보와 과거의 은닉층의 값에 각각 가중치를 곱하여 더한 후 sigmoid(0, 1) 함수를 적용
- 그 출력 값을 직전 시점의 cell에 곱하기 연산.
- 1에 가까울수록 과거 정보를 많이 활용, 0에 가까울수록 과거 정보를 많이 망각

$$f^{(t)} = \sigma\left(U_f x^{(t)} + W_f h^{(t-1)}\right)$$



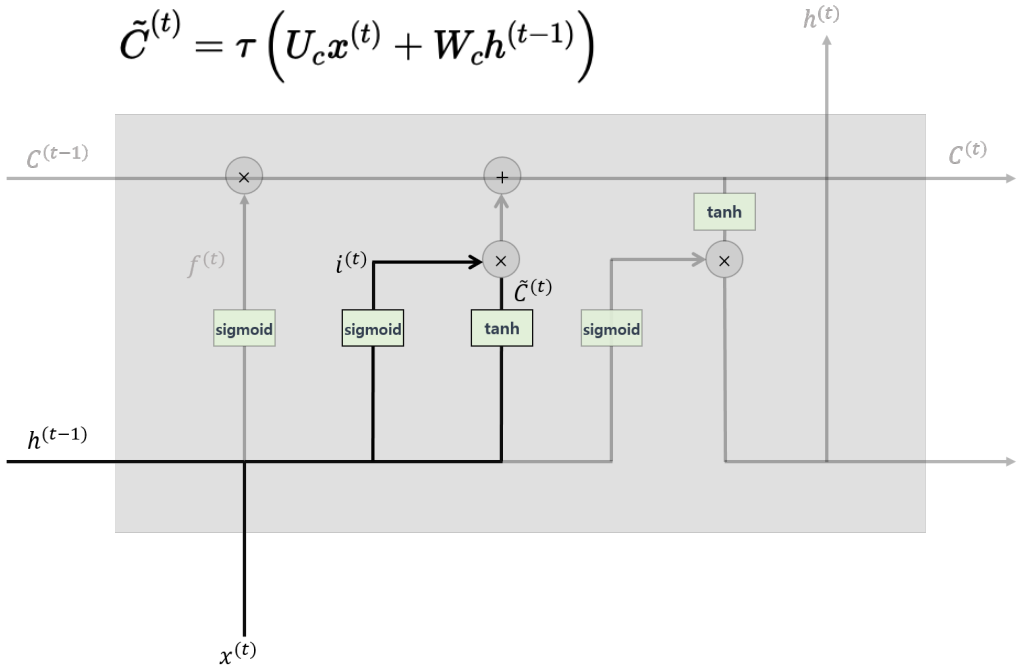
LSTM 기본 원리

2단계.

- **입력게이트 (Input Gate):** 현시점의 정보를 셀에 입력할 크기를 지정
- **입력후보 (Candidate):** 현지점의 정보를 계산
- 현시점에서 실제로 갖고 있는 정보(입력후보)가 얼마나 중요한지(입력게이트)를 반영하여 셀에 기록

$$i^{(t)} = \sigma\left(U_{in}x^{(t)} + W_{in}h^{(t-1)}\right)$$

$$\tilde{C}^{(t)} = \tau\left(U_{c}x^{(t)} + W_{c}h^{(t-1)}\right)$$

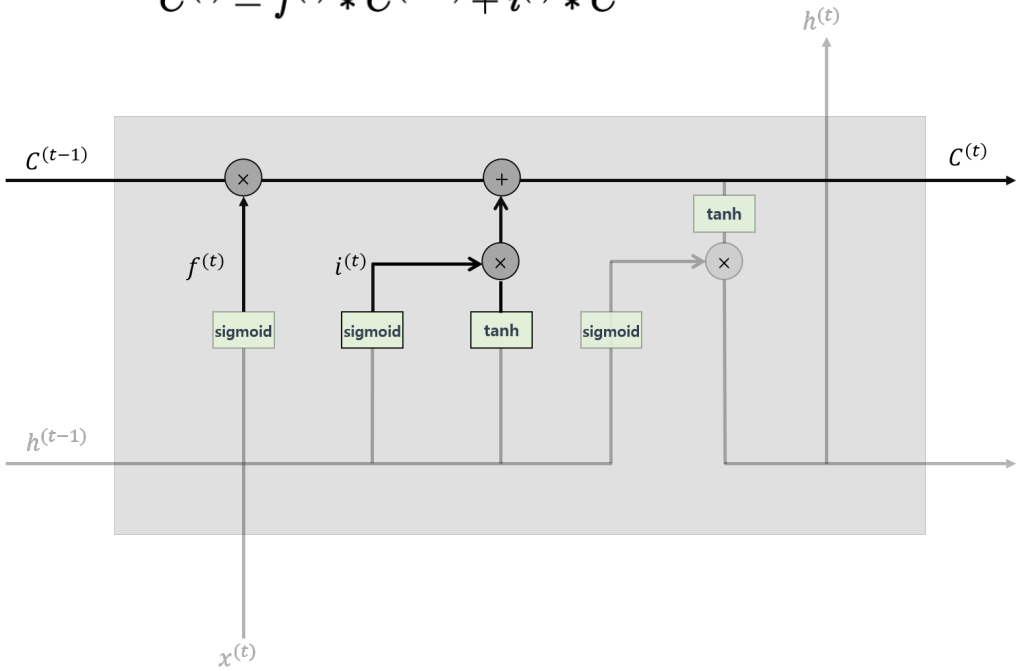


LSTM 기본 원리

3단계. Memory Cell의 계산
계산한 망각게이트, 입력게이트, 입력후보를 이용하여 memory cell에 저장

- 과거의 정보를 망각게이트에서 계산 된 만큼 잊고,
- 현시점의 정보 후보에 입력게이트의 중요도를 곱해준 것을 더하여 현시점 기준 memory cell을 계산.

$$C^{(t)} = f^{(t)} * C^{(t-1)} + i^{(t)} * \tilde{C}^{(t)}$$



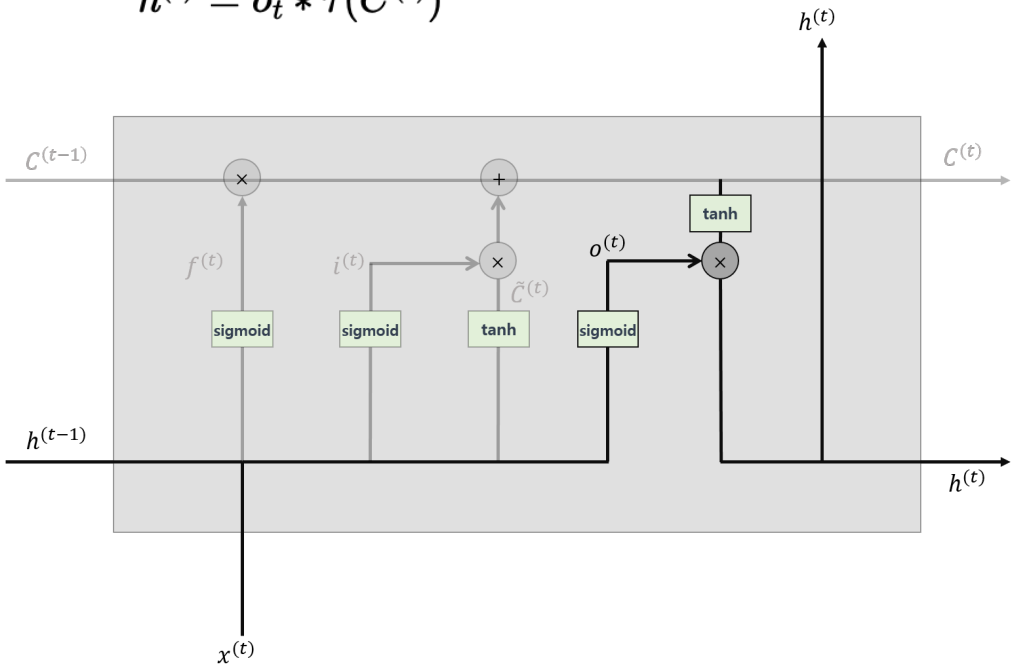
LSTM 기본 원리

4단계. 출력게이트 (Output Gate)

계산된 현시점의 memory cell을 현시점의 은닉층 값으로 출력할 양을 결정

$$o^{(t)} = \sigma\left(U_o x^{(t)} + W_o h^{(t-1)}\right)$$

$$h^{(t)} = o_t * \tau(C^{(t)})$$



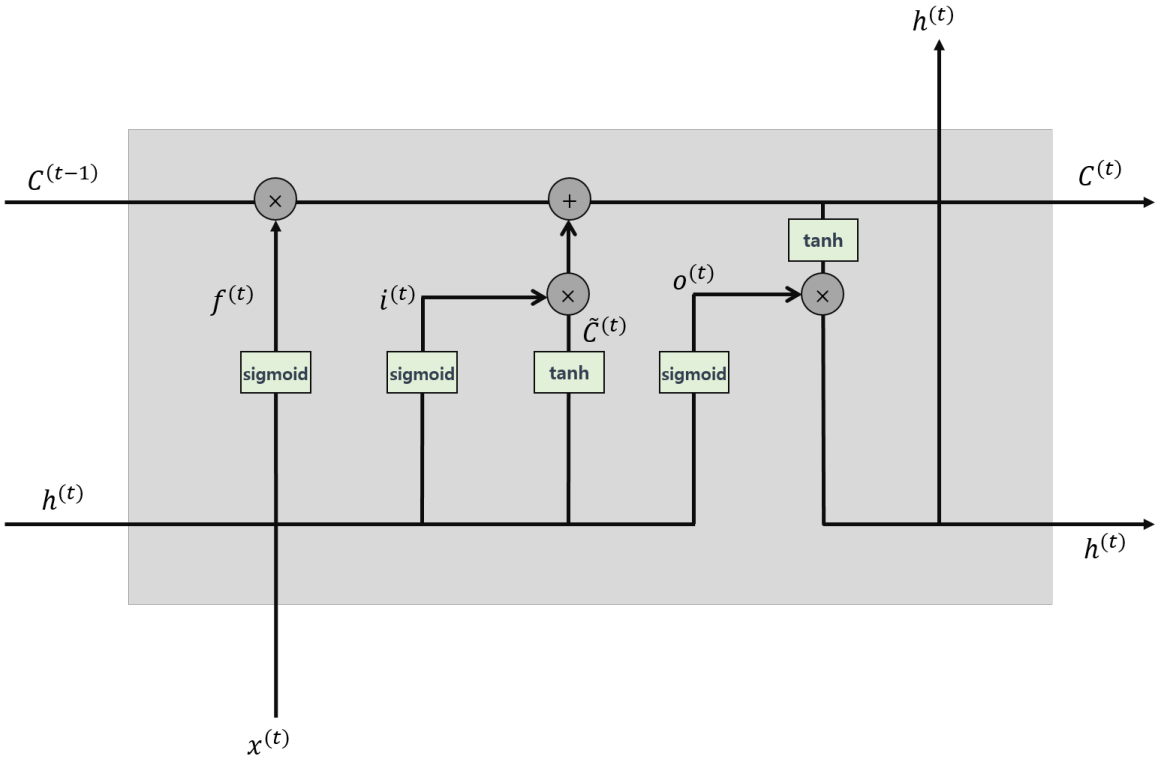
LSTM 기본 원리

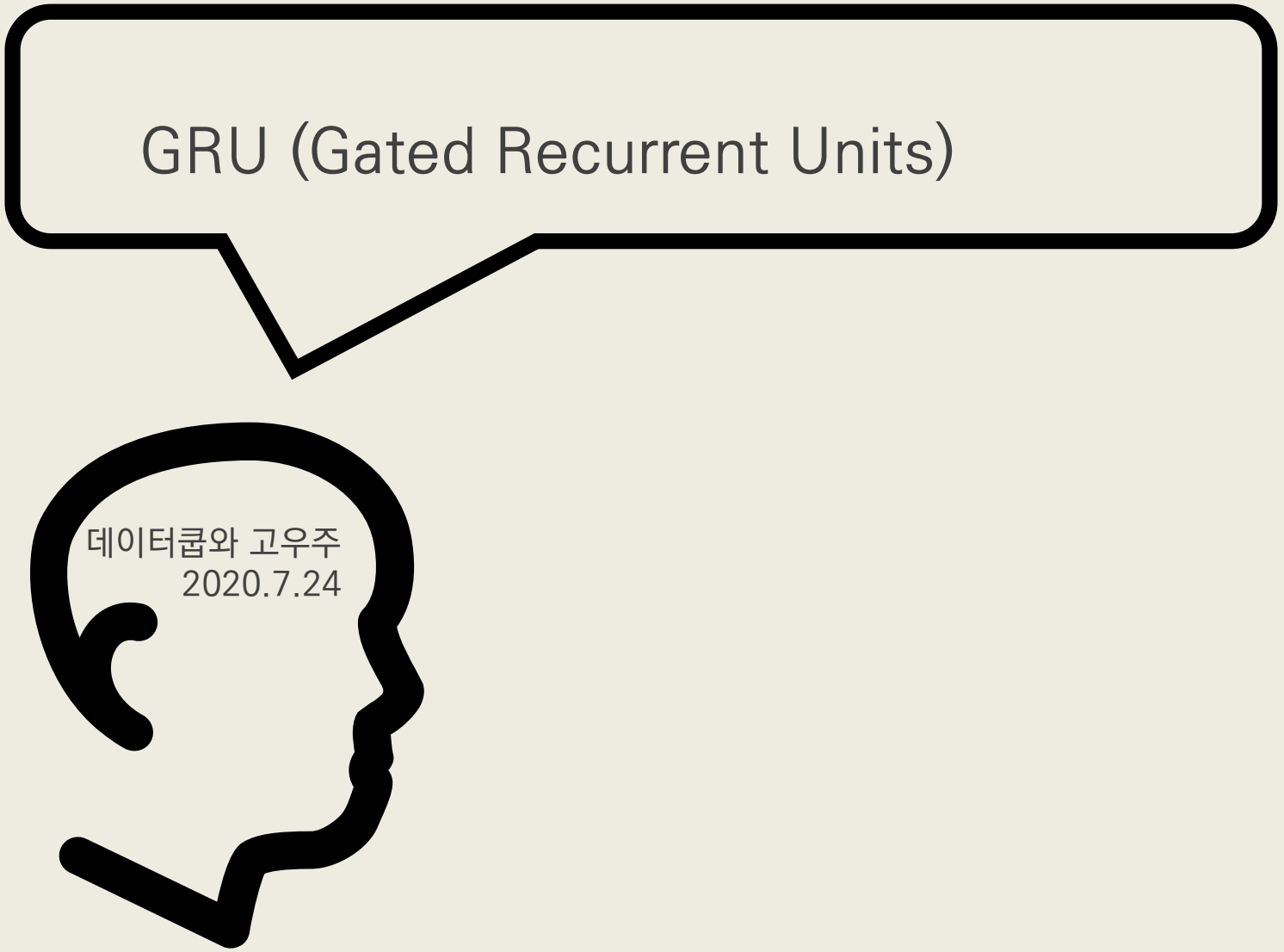
5단계. 출력층

RNN과 동일 softmax 함수 $\hat{y}^{(t)} = softmax\left(Vh^{(t)}\right)$

전체 과정 요약

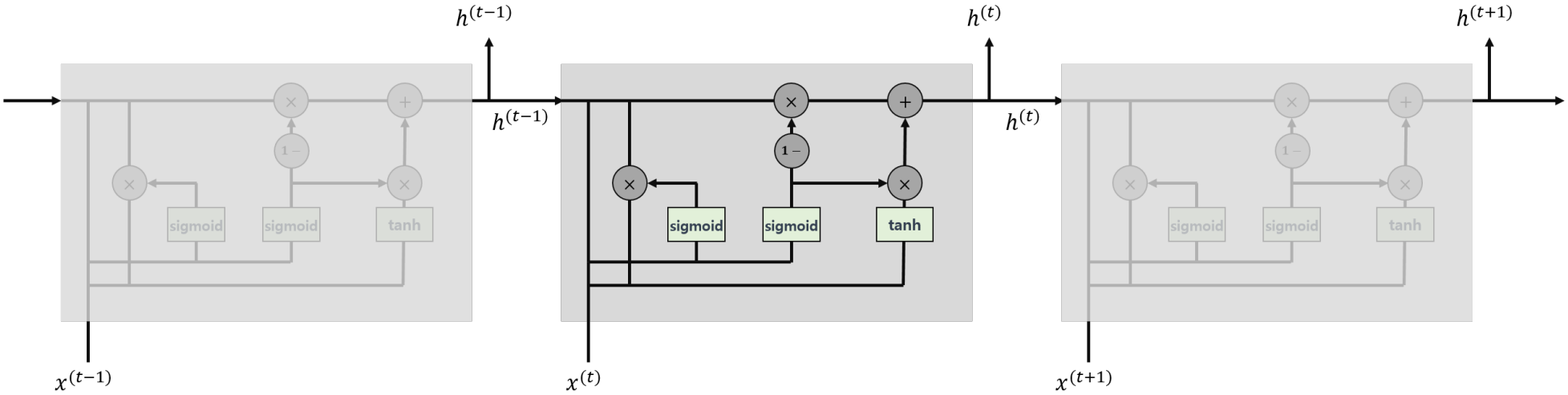
$$f^{(t)} = \sigma\left(U_f x^{(t)} + W_f h^{(t-1)}\right)$$
$$i^{(t)} = \sigma\left(U_{in} x^{(t)} + W_{in} h^{(t-1)}\right)$$
$$\tilde{C}^{(t)} = \tau\left(U_c x^{(t)} + W_c h^{(t-1)}\right)$$
$$C^{(t)} = f^{(t)} * C^{(t-1)} + i^{(t)} * \tilde{C}^{(t)}$$
$$o^{(t)} = \sigma\left(U_o x^{(t)} + W_o h^{(t-1)}\right)$$
$$h^{(t)} = o^{(t)} * \tau(C^{(t)})$$
$$\hat{y}^{(t)} = softmax(Vh^{(t)})$$




GRU (Gated Recurrent Units)
데이터큡와 고우주
2020.7.24

GRU(Gated Reccurent Units) 개요

- GRU는 게이트 메커니즘이 적용된 RNN 프레임워크의 일종
- LSTM과 유사하지만 더 간략한 구조
- 한국인 조경현 박사가 연구논문에서 제안 (Cho et al., 2014)

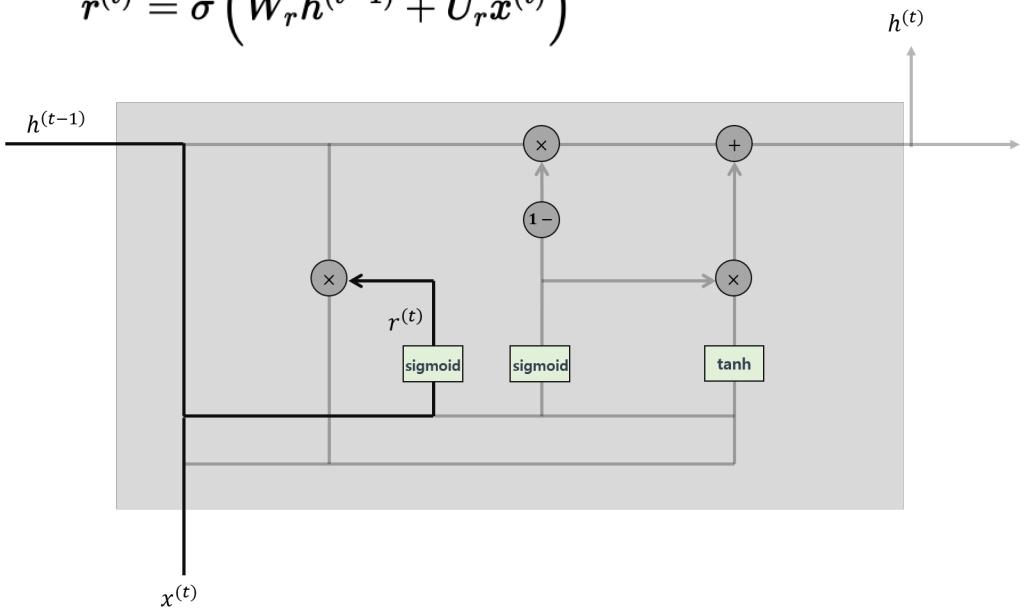


GRU 기본 원리

1단계. Reset Gate

- 과거의 정보를 적당히 리셋시키는게 목적
- sigmoid 함수를 출력으로 이용해 (0,1) 값을 이전 은닉층에 곱 연산
- 직전 시점의 은닉층의 값과 현시점의 정보에 가중치를 곱하여 진행

$$r^{(t)} = \sigma\left(W_r h^{(t-1)} + U_r x^{(t)}\right)$$

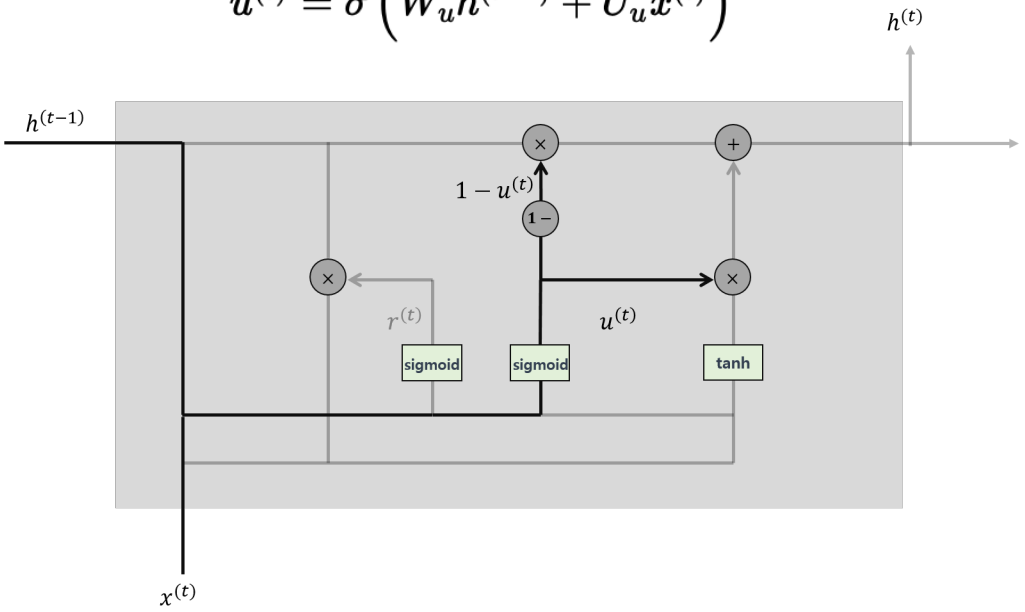


GRU 기본 원리

2단계. Update Gate

- LSTM의 forget gate와 input gate를 합쳐놓은 것과 유사, 과거와 현재의 정보의 최신화 비율을 결정
- Update gate에서는 sigmoid로 출력된 결과(u(t))는 현시점의 정보의 양을 결정
- 1에서 뺀 값(1−u(t))는 직전 시점의 은닉층의 정보에 곱해 줌
- 각각이 LSTM의 input gate와 forget gate

$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)}\right)$$

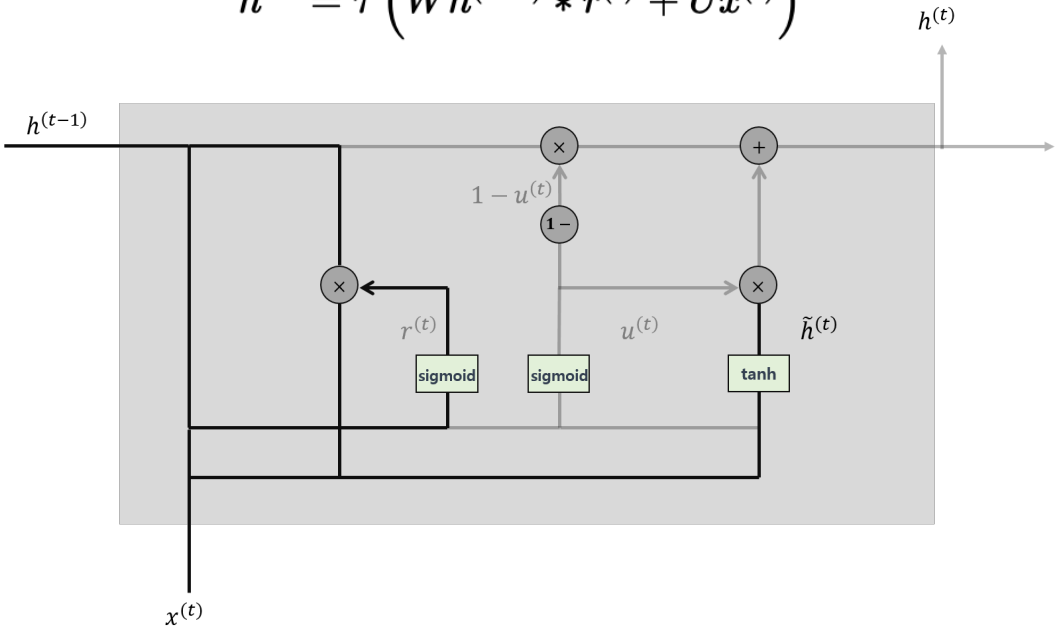


GRU 기본 원리

3단계. Candidate

- 현 시점의 정보 후보군을 계산하는 단계
- 과거 은닉층의 정보를 그대로 이용하지 않고 리셋 게이트의 결과를 곱하여 사용
  (τ는 tangent hyperbolic, $*$은 pointwise operation)

$$\tilde{h}^{(t)} = \tau\left(Wh^{(t-1)} * r^{(t)} + Ux^{(t)}\right)$$

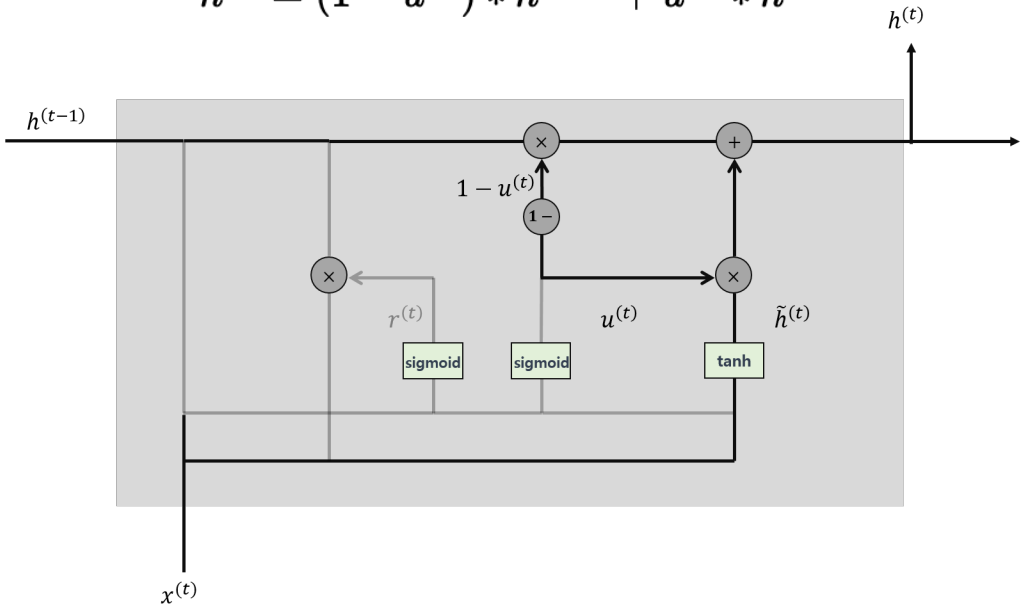


GRU 기본 원리

4. 은닉층 계산

- 마지막 단계로 update gate 결과와 candidate 결과를 결합하여 현시점의 은닉층을 계산
- sigmoid 함수의 결과는 현시점 결과의 정보의 양을 결정
- 1-sigmoid 함수의 결과는 과거 시점의 정보 양을 결정

$$h^{(t)} = (1 - u^{(t)}) * h^{(t-1)} + u^{(t)} * \tilde{h}^{(t)}$$



Summary

LSTM & GRU 비교 시

- 기본 구조와 성능 유사
- 학습 가중치가 LSTM보다 적다

전체 과정 요약

$$r^{(t)} = \sigma\left(W_r h^{(t-1)} + U_r x^{(t)}\right)$$

$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)}\right)$$

$$\tilde{h}^{(t)} = \tau\left(W h^{(t-1)} * r^{(t)} + U x^{(t)}\right)$$

$$h^{(t)} = (1 - u^{(t)}) * h^{(t-1)} + u^{(t)} * \tilde{h}^{(t)}$$