13기 정규세션

ToBig's 12기 신윤종

자연어 처리 심화

Contents

Unit 01 | 임베딩 깊이 살펴보기

오늘 강의 목표

우리가 해왔던 것들을 되짚어가며
자연어처리를 더 깊이 이해해보자
&
이번에 새로 배울 Recurrent Neural Network와
LSTM을 알아보자

Unit 1.

임베딩 깊이 살펴보기

Unit 01 | 임베딩 깊이 살펴보기

1주차 EDA와 전처리 중에서...

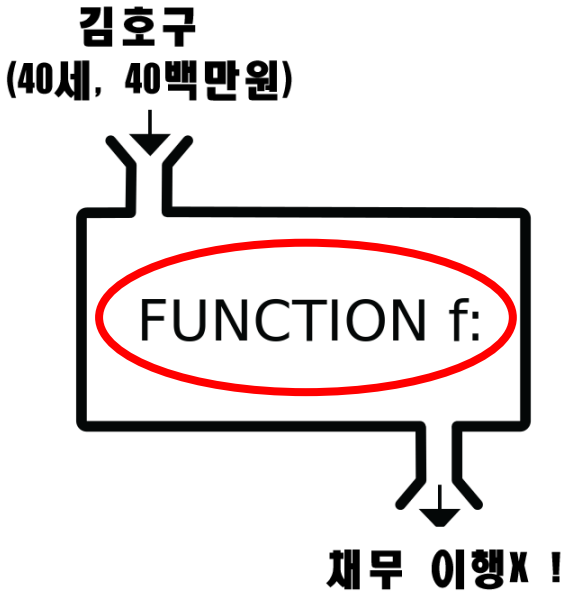
전통적인 머신러닝 과정 :

1) 데이터를 면밀히 탐구해 특징을 파악한다.
   - EDA

2-1) 데이터를 다듬고, 적절한 피쳐(특징)를 만든다
   - 전처리

2-2) 적절한 모델을 선택한다.
   - ~5주차

3) 일반화



"머신러닝은 Feature Engineering이 시작이자 끝이다." **김주호** (1998 ~)

Unit 01 | 임베딩 깊이 살펴보기

그래서 우리는 이런 과제를 했습니다!

'Auction_master_train.csv ' 와 함께하는 **“EDA & 전처리 & 인코딩”**

그러다 마주친....

3) addr_do

```
]:  data.groupby(data['addr_do']).mean()['Hammer_price']

t[12]: addr_do
       부산    2.455039e+08
       서울    5.989656e+08
       Name: Hammer_price, dtype: float64
```

‘범주형 변수는 더미변수로 만들라고 했는데?’

Unit 01 | 임베딩 깊이 살펴보기

그래서 우리는 이런 과제를 했습니다!

'Auction_master_train.csv ' 와 함께하는 **“EDA & 전처리 & 인코딩”**

```
s_dummy = pd.get_dummies(data.addr_do, columns=['부산', '서울'])
s_dummy.columns=['부산', '서울']
s_dummy.head(-5)
```

[11]:

	부산	서울
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
1923	0	1
1924	0	1
1925	0	1
1926	0	1
1927	0	1

1928 rows × 2 columns

‘범주형 변수는 더미변수로 만들라고 했는데?’

그래서 범주형 변수를 Feature로 표현했어!

Unit 01 | 임베딩 깊이 살펴보기

좀 더 깊이 살펴보겠습니다. 이해를 위해 연속형 경매가를 '높음'과 '낮음 ' 분류로 접근해보죠

서울	부산	경매가
1	0	**높음**
1	0	**높음**
1	0	**낮음**
0	1	**낮음**
0	1	**낮음**
1	0	**?**

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

*Naïve Bayes

Q. 서울이면서 부산이 아닌 경우 경매가는?

$$P(Y = \text{높음}|X = \text{서울}) = 2/3$$

$$P(Y = \text{낮음}|X = \text{서울}) = 1/3$$

$$P(Y = \text{높음}|X = \text{서울}) = P(X = \text{서울}|Y = \text{높음})P(Y = \text{높음}) \div P(Y = \text{서울})$$

Unit 01 | 임베딩 깊이 살펴보기

알고 계셨나요? 우리나라는 17개의 시가 있습니다!

서울	부산	인천	…	대전	대구	경매가
1	0	0	…	0	0	**높음**
0	1	0	…	0	0	**높음**
0	0	1	…	0	0	**낮음**
0	0	0	…	1	0	**낮음**
0	0	0	…	0	1	**낮음**
1	0	0	…	0	0	**?**

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

*Naïve Bayes

Q. 어..? $P(Y = ? \,|\, X = \text{서울})$

Unit 01 | 임베딩 깊이 살펴보기

알고 계셨나요? 우리나라는 17개의 시가 있습니다!

서울	부산	인천	…	대전	대구	경매가
1	0	0	…	0	0	**높음**
0	1	0	…	0	0	**높음**
0	0	1	…	0		
0	0	0	…	1		
0	0	0	…	0	1	**낮음**
1	0	0	…	0	0	**?**

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

*Naïve Bayes

Q. 어..? $P(Y = ?\,|X = \text{서울})$

$$P(x1 = 1, x2 = 0, \ldots, x16 = 0, x17 = 0, |Y = \text{높음})$$

-> 서울이면서 부산이 아니면서 동시에 인천이 아니면서… (중략) … 동시에 대전이 아니면서 동시에 대구가 아닌 상황에 경매가는?

Unit 01 | 임베딩 깊이 살펴보기

4-1) 계산의 한계 **3주차 나이브베이즈 중에서...**

문제점: 계산량이 많아짐

$$P(X = x|Y = y) = \text{for all } x,y \quad -> \quad (2^d - 1)k$$

변수가 늘어날 수록 기하급수적으로 연산량이 증가함

어떻게 얻은 변수들인데…….d의 개수를 줄이는 건 피하고 싶어

->>해결책: 조건부 독립을 가정!!!!

그러나 현실세계에서 모든 Feature(독립변수)가 실제로 독립이라고 쉽게 가정할 수 있을까?

→ 그나마 텍스트는 조건부 독립의 전제 관점에서 비교적 강점이 있다!

Unit 01 | 임베딩 깊이 살펴보기

알고 계셨나요? 우리나라는 17개의 시가 있습니다!

서울	부산	인천	…	대전	대구	경매가
1	0	0	…	0	0	**높음**
0	1	0	…	0	0	**높음**
0	0	1	…	0		
0	0	0	…	1		
0	0	0	…	0		
1	0	0	…	0		

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

*Naïve Bayes

Q. 어..? $P(Y = \text{높음}|X = \text{서울})$

$$P(x1 = 1, x2 = 0, \dots, x16 = 0, x17 = 0, |Y = \text{높음})$$

-> ~~서울이면서 부산이 아니면서 동시에 인천이 아니면서… (중략) … 동시에 대전이 아니면서 동시에 대구가 아닌 상황에 경매가는?~~

나이브 베이즈를 적용하면

$P(x1 = 1|Y = \text{높음})P(x2 = 0|Y = \text{높음}) \dots P(x17 = 0|Y = \text{높음})$

-> 높음이라는 조건에 따른 각각의 확률은?

조건부확률을 각자 계산해서 곱하기만 하면 된다!

Unit 01 | 임베딩 깊이 살펴보기

■ 단어 임베딩은 단순히 문자를 구분하는 데 그치지 않습니다.

1. 임베딩은 단어를 벡터로 표현하여 독립적인 존재로 만듭니다.
2. 이와 동시에 임베딩 자체가 Feature로써 머신러닝 모델의 학습 데이터로 사용됩니다.
3. 텍스트만의 특징을 이용하여 더 나은 모델을 선택할 수 있습니다. Ex) 나이브 베이즈
4. 텍스트 역시 전처리와 EDA가 필요하며 좋은 임베딩은 좋은 결과를 만듭니다.

   (토크나이징, 불용어 처리, 오탈자 처리 등등)

Unit 2.

Document(문서) 임베딩과 Classification(분류)

Unit 02 l 문서 임베딩과 분류

'단어'를 넘어서 '문서'로 시선을 높여 보아요

단어가 모여서 문장을 이루고, 문장이 모여서 문서를 이룹니다.

우리는 문서 자체를 다룰 줄 알아야 더 많은 일을 해낼 수 있습니다.

- Document Classification/Categorization

✓Assigning subject topics
✓Spam detection
✓Language identification
✓Sentiment analysis
✓ ...

Unit 02 l 문서 임베딩과 분류

Document Classification : Example

■ Spam or not ?

From : <tobigs13@gmail.com>

♛♛투빅스 음☆성 세미나♛♛가입시$$전원 1인당 1 강의☜☜과제100%증정※ ♜Speech to Text♜논문 리뷰무료증정￥ 특정조건 §§Librosa§§★박진혁의 유산★갓승현과 세미나 진행 기회@@@ 즉시이동 http://tobigs.gitbook.io/tobigs-audio-seminar/

From : <xnqlrtmwhdk@gmail.com>

[음성 세미나 커리큘럼 구체화 요청]
안녕하세요 12기 신윤종입니다. 음성 세미나 관련하여 3월 내로 커리큘럼 확정을 짓고자 합니다. 이와 관련하여 금일 13시에 미팅하도록 하겠습니다. 리뷰 논문 목록 초안을 첨부하였으니 참고바랍니다.

Unit 02 l 문서 임베딩과 분류

그렇다면 문서 임베딩은 어떻게 해야할까?

단어 임베딩의 예)

'**최악**' = [0, 1]

'**재미**' = [1, 0]

문서 임베딩의 예?

'진짜 **최악**이었다 절대 관람하지마세요' = ?

'**재미**있는 오락영화였다 강추' = ?

Unit 02 | 문서 임베딩과 분류

7주차 텍스트 기초 중에서...

우리는 문서 임베딩의 한 방법을 이미 배웠답니다

- **TDM(Term Document Matrix) : 문서별로 단어의 빈도를 정리한 표**

Ex) 문서1: 오늘은 밥을 먹었다
문서2: 어제도 밥, 오늘도 밥

	오늘	어제	밥	먹다
문서1	1	0	1	1
문서2	1	1	2	0

→ [1, 1, 2, 0]

Q. **오늘**은 **밥**이고 **어제**도 **밥**이었다. → 얘도 [1, 1, 2, 0] 같은데?

A. 문서 임베딩은 원-핫처럼 각자 구분 짓기보다 문서 간 유사도를 측정하는데 많이 쓰입니다. (Cosine, Jaccard 등등)

Unit 02 | 문서 임베딩과 분류

또 하나의 방법으로 단어의 임베딩 벡터를 단순하게 묶을 수도 있죠

Ex) 문서1: 오늘은 밥을 먹었다
문서2: 어제도 밥, 오늘도 밥

문서1 : [[1 0 0 0], - 오늘
[0 0 1 0], - 밥
[0 0 0 1]] - 먹다

문서2 : [[0 1 0 0], - 어제
[0 0 1 0], - 밥
[1 0 0 0], - 오늘
[0 0 1 0]] - 밥

*One-hot Encoding

오늘	어제	밥	먹다
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

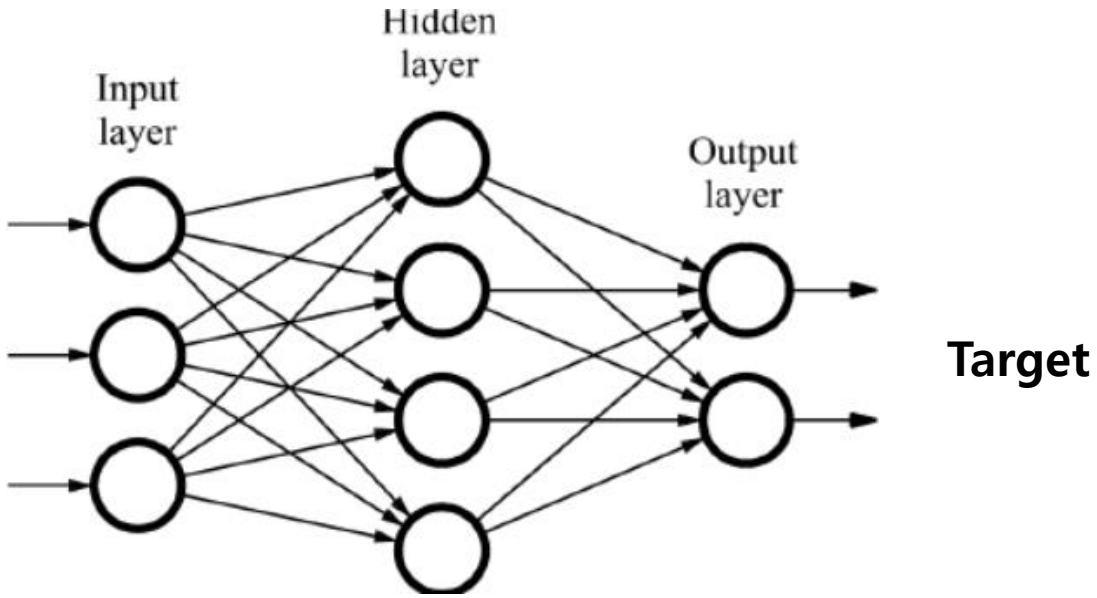
Can we use MLP?

■ 정형 데이터 적용 예시

	키	몸무게	나이
A	150	50	20
B	160	60	22
C	170	70	24

150
50
20

*** 평범한 다층 신경망**



Target

Unit 02 l 문서 임베딩과 분류

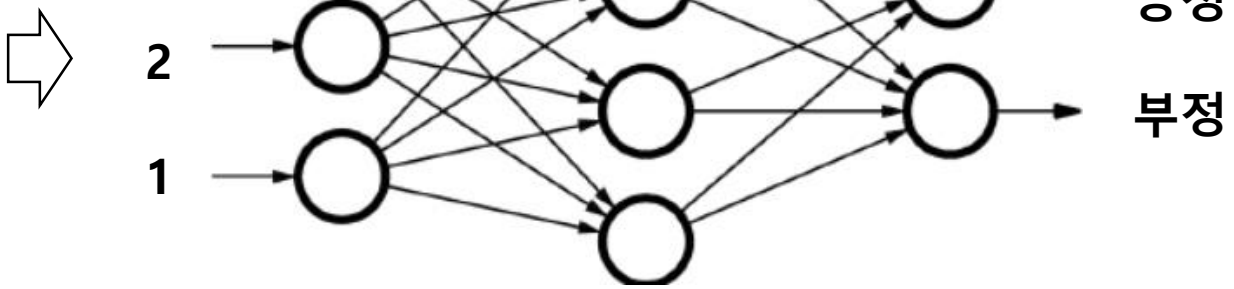
Can we use MLP?

■ 텍스트 데이터 적용 예시

1. 빈도기반 문서 임베딩

*** 영화 리뷰 데이터**

	영화	최고	최악
Doc A	2	2	1
Doc B	2	0	0
Doc C	0	3	1



Unit 02 l 문서 임베딩과 분류

Can we use MLP?

1. 빈도기반 문서 임베딩의 문제점

Q. 이 영화의 리뷰의 감정은?

문서 A : 그 **영화**는 **최고**였는데, 이 **영화**는 **최악**이다.

문서 B : 그 **영화**는 **최악**인데, 이 **영화**는 **최고**야.

	영화	최고	최악
Doc A	2	1	1
Doc B	2	1	1

*** 단어의 순서를 파악할 수 없다**

Unit 02 l 문서 임베딩과 분류

Can we use MLP?

■ 텍스트 데이터 적용 예시

2. 원-핫 벡터 기반 문서 임베딩

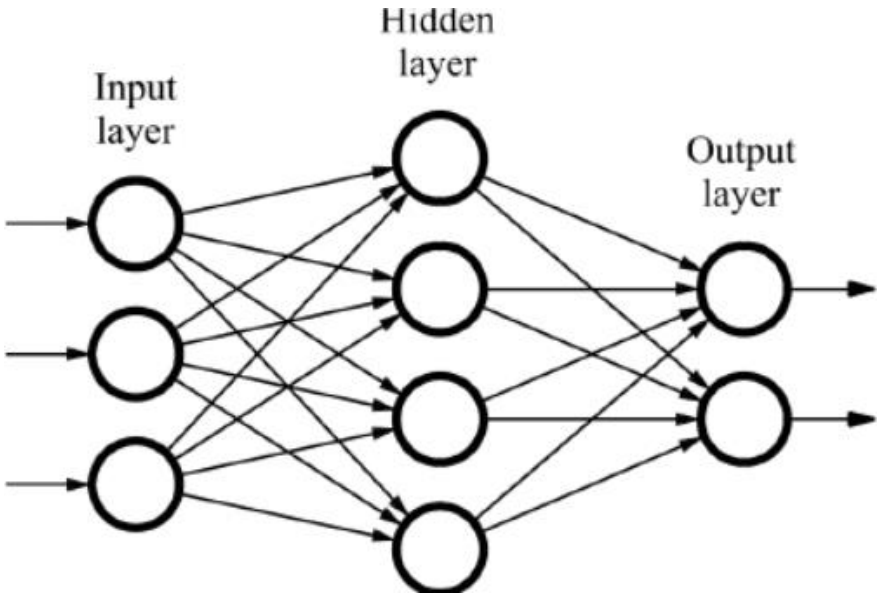
*** 아까 봤던 그 예시**

문서1 : [[1 0 0 0], - 오늘
[0 0 1 0], - 밥
[0 0 0 1]] - 먹다

[1 0 0 0]

[0 0 1 0]

[0 0 0 1]



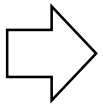
Unit 02 l 문서 임베딩과 분류

Can we use MLP?

1. 원-핫 벡터 기반 문서 임베딩의 문제점

어? 야 3개라며 !!!!

문서1 : [[1 0 0 0],
[0 0 1 0],
[0 0 0 1]]

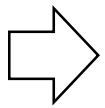


[1 0 0 0] x_2

[0 0 1 0] x_1

[0 0 0 1] x_0

문서2 : [[0 1 0 0],
[0 0 1 0],
[1 0 0 0],
[0 0 1 0]]



[0 1 0 0] x_2

[0 0 1 0] x_1

[1 0 0 0]

[0 0 1 0] x_0

*** 벡터의 행, 즉 단어 벡터의 개수가**
신경망 입력 노드의 개수와 맞지 않을 수 있다.

Unit 02 l 문서 임베딩과 분류

■ **문서 임베딩은 여러 단어의 임베딩이 모여 하나로 표현됩니다.**

1. 문서 임베딩은 단어의 조합으로써 문서 간의 유사도를 판별할 수 있습니다.
2. 대표적으로 빈도 기반 임베딩과 원-핫 벡터 기반 임베딩이 있습니다.
3. 빈도 기반 임베딩은 단어의 순서를 파악할 수 없다는 단점이 있습니다.
4. 원-핫 벡터 기반 임베딩은 신경망 학습을 위한 고정된 입력 노드 개수에 맞추기 어렵습니다.

Unit 3.

Recurrent Neural Networks (순환 신경망)

Unit 03 | 순환신경망

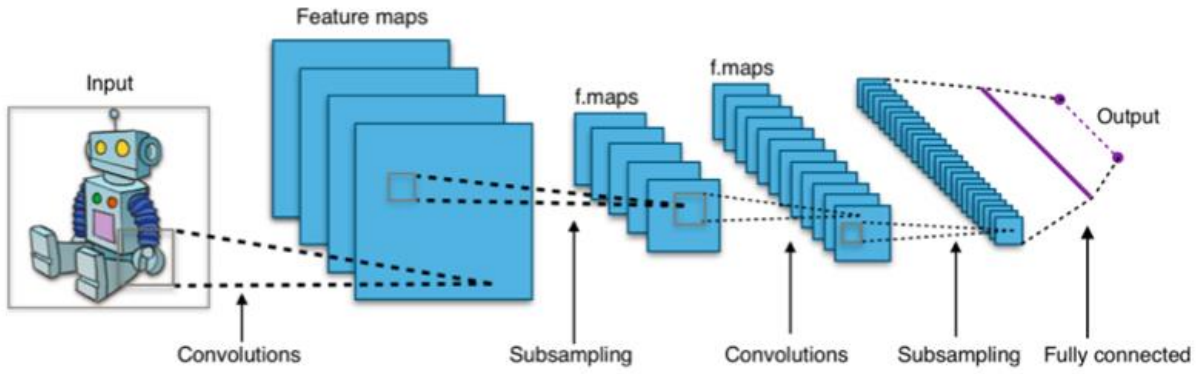
괜찮아요 비정형 데이터를 인공신경망과 접목시키는 일은 쉽지 않습니다

Image의 경우 2차원 특성을 보존하기 위해 **CNN**이 나왔어요

9주차 이미지 심화 중에서...

Convolutional Neural Network (합성곱 신경망)

- Locality를 고려한 구조
- Filter를 통한 연산으로 parameter의 수 감소



Unit 03 | 순환신경망

단순한 MLP로는 텍스트 데이터를 처리하기엔 역부족이에요 ㅠㅠ

1. 단어의 순서 정보를 잃지 말아야 하고
2. 입력 단어의 개수에 구애 받지 않아야 해요

그야말로 'Sequence'에 최적화된 신경망 어디 없나요!

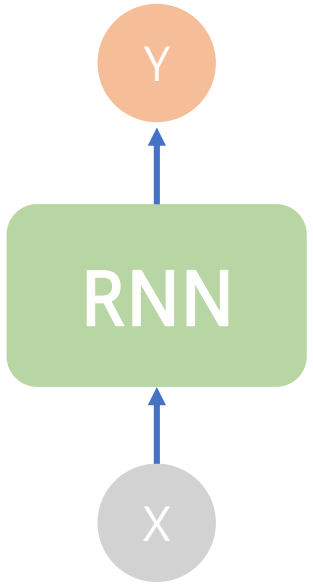
RNN

바로 'RNN'

Unit 03 | 순환신경망

What is RNN?

- **RNN의 입력을 이해하자**

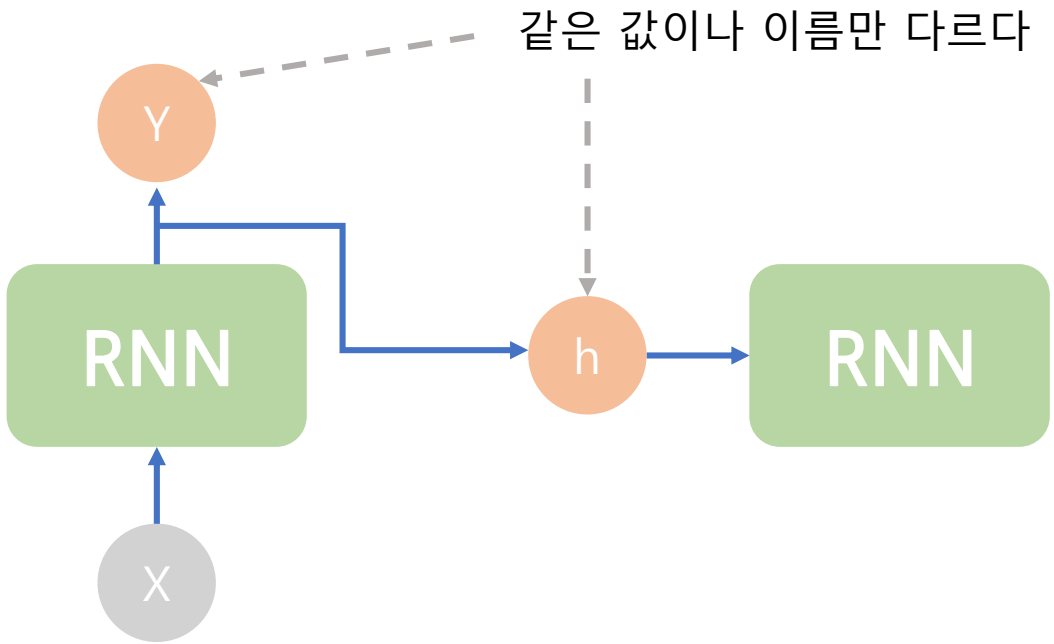


얘는 신경망인데 입력 노드는 단 **하나**
(주로 원-핫 벡터 하나 입력)

Unit 03 | 순환신경망

What is RNN?

- **RNN의 입력을 이해하자**



특이한 점은 출력값을 두 갈래로 나뉘어 한 쪽은 출력(Y)을,
나머지는 다음 스텝에서 RNN 자기 자신의 또 다른 입력(h)으로 사용한다
→신경망에게 '**기억**' 하는 기능을 부여했다

Unit 03 | 순환신경망

What is RNN?

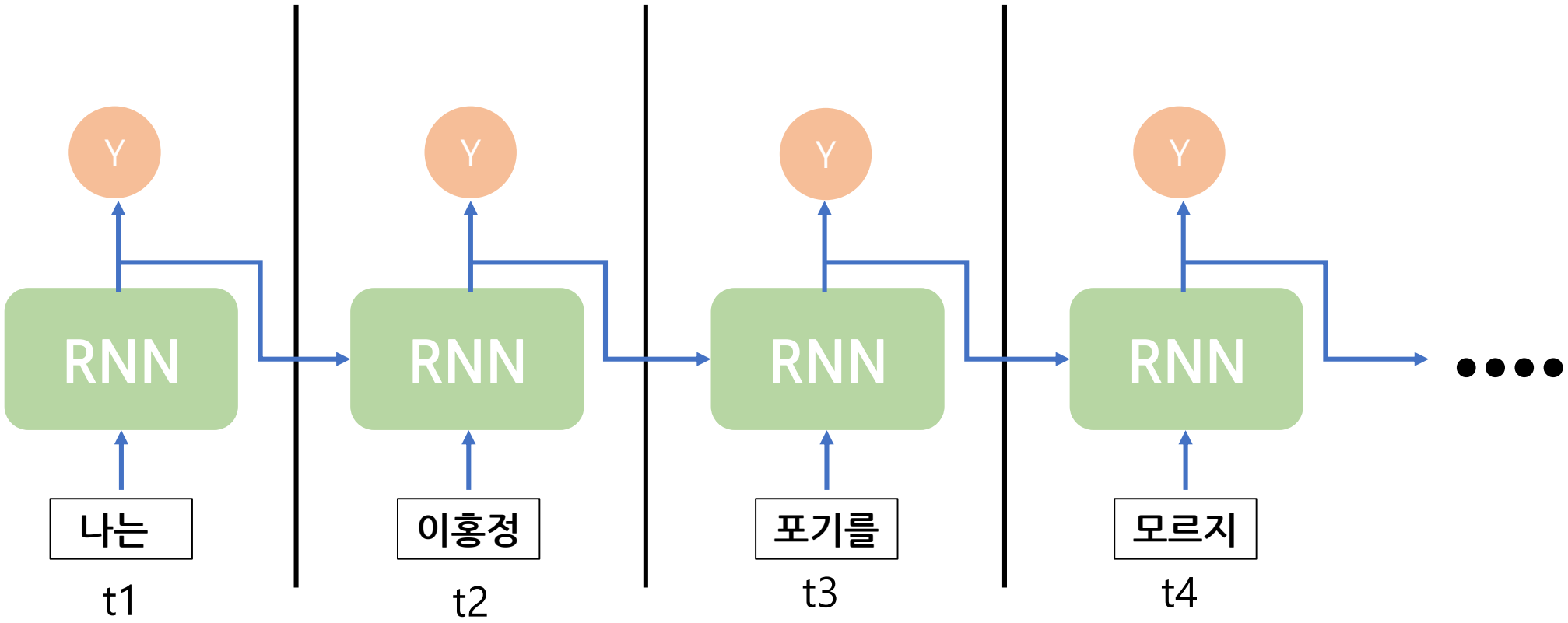
- **RNN의 입력을 이해하자**
- **Time Step : 학습할 문장을 끝까지 입력하는데 걸리는 시간**

Ex) [[나는], [이홍정], [포기를], [모르지]] -> 4 Time Step

Unit 03 | 순환신경망

What is RNN?

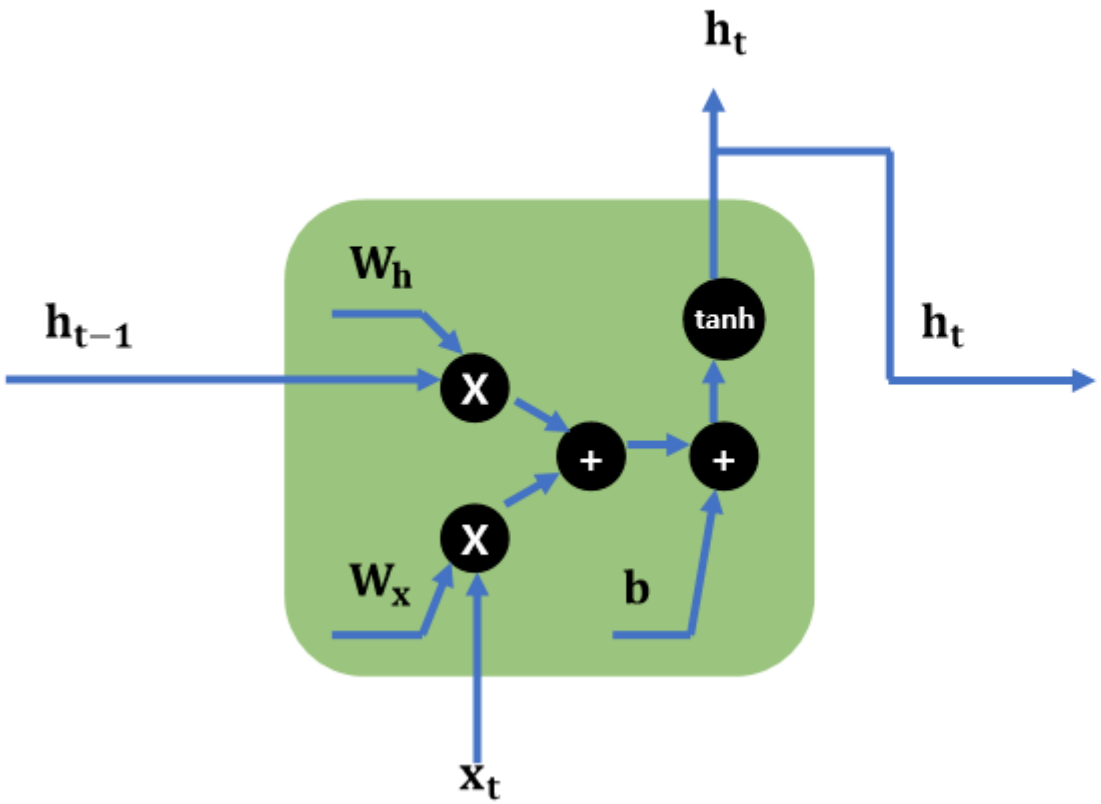
- **RNN의 입력을 이해하자**
- **Time Step : 학습할 문장을 끝까지 입력하는데 걸리는 시간**



Unit 03 | 순환신경망

- Recurrent Neural Networks (RNN) Forward

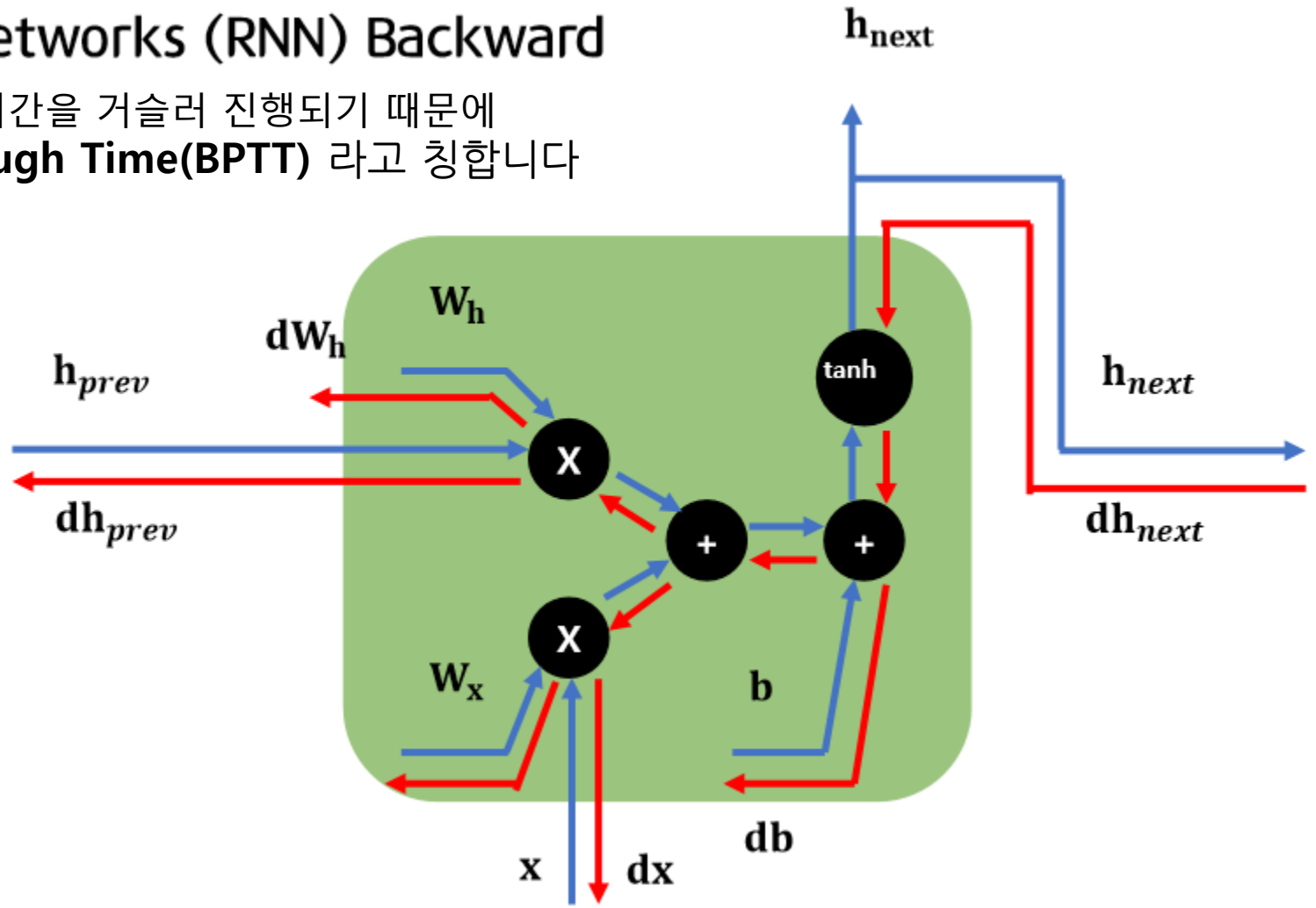
*일반적 퍼셉트론 : $f(x) = sigmoid(xW + b)$



$$\mathbf{h}_t = \tanh(\mathbf{h}_{t-1}\mathbf{W}_h + \mathbf{x}_t\mathbf{W}_x + \mathbf{b})$$

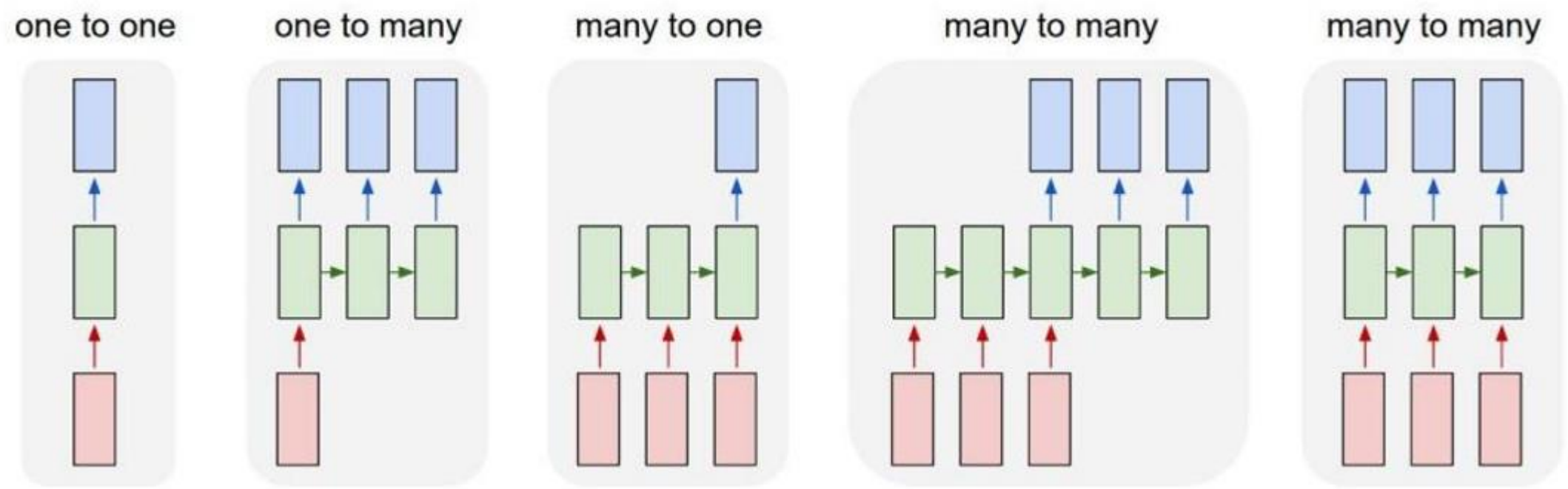
Unit 03 | 순환신경망

- Recurrent Neural Networks (RNN) Backward
  - RNN의 역전파가 시간을 거슬러 진행되기 때문에 **Backpropagation Through Time(BPTT)** 라고 칭합니다



Unit 03 | 순환신경망

- Recurrent Neural Networks (RNN) Architecture
  - Input과 Output 개수에 따라 RNN Architecture를 구분



*사용처 : Image Captioning | Text Classification | Machine Translation | Video classification

Unit 03 | 순환신경망

■ **RNN 덕분에 기존 퍼셉트론 구조의 한계를 극복하여 텍스트를 학습할 수 있게 되었습니다.**

1. RNN은 텍스트 뿐만 아니라 음성, 이미지 등 다양한 분야에 적용될 수 있는 시퀀스에 특화된 아키텍처입니다.
2. 학습 시 Time step t에서는 이전 출력인 h_{t-1} 가 사용됩니다. 즉 기억 능력을 가지고 있습니다
3. 역전파 시 순전파와 동일하게 h_{t-1}방향으로도 Gradient가 전달됩니다
4. 입력과 출력을 어떻게 구성하느냐에 따라 다양한 방식으로 사용될 수 있습니다.

Unit 4.

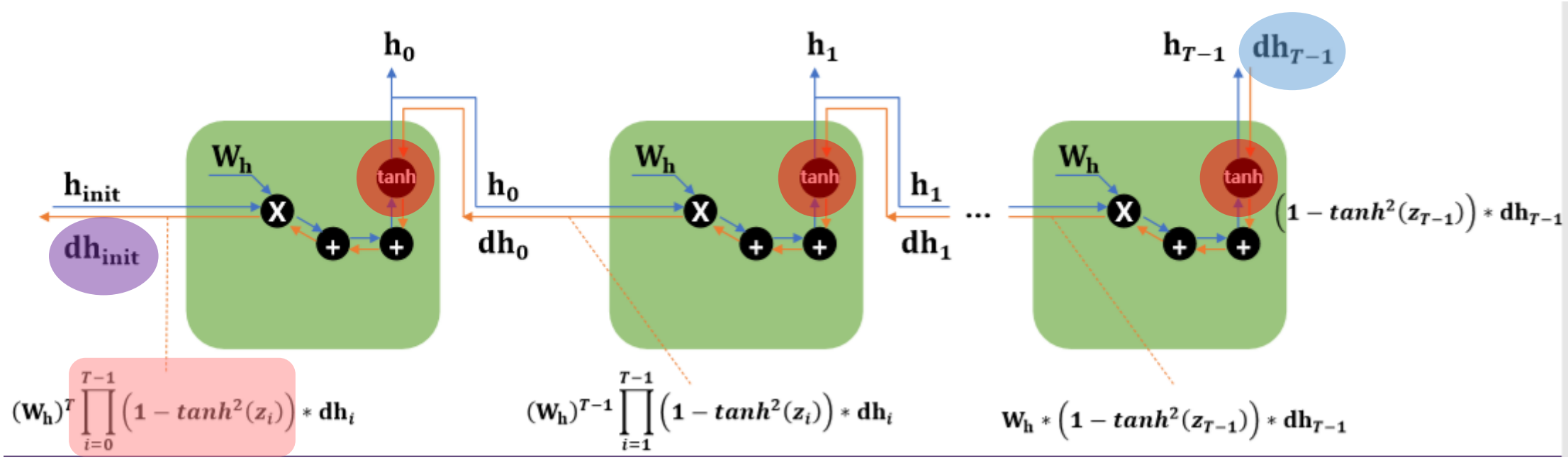
Long Short-Term Memory models (LSTM)

- 순환신경망의 진화

Unit 04 | LSTM

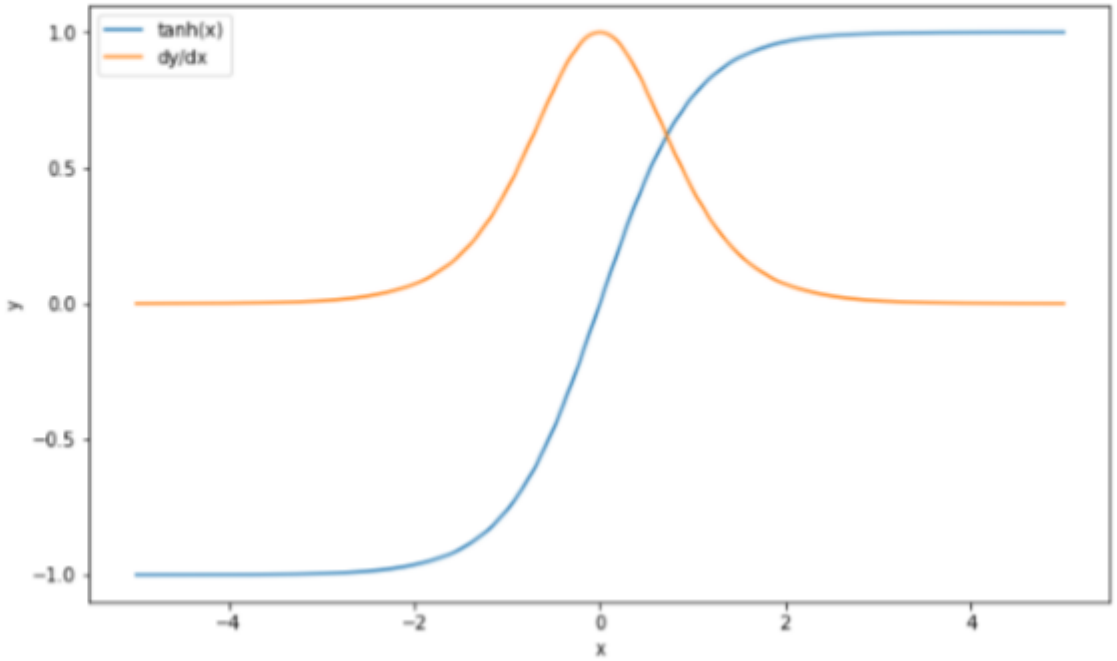
RNN이 정말 완벽히 학습할 수 있는지 지켜봐주세요

h_{T-1}의 Gradient(역전파의 출발점)가 h_{init}(도착점) 까지 만나는 tanh 활성화함수



Unit 04 | LSTM

- Vanishing(or Exploding) Gradient Problem
- 원인 : tanh()와 행렬곱 : tanh의 미분
- $\mathbf{0} < \mathbf{1} - \boldsymbol{tanh}^{\mathbf{2}}(\boldsymbol{z}_{\boldsymbol{T}-\mathbf{1}}) < \mathbf{1}$
- 1보다 작은 값이 계속 곱해져
  Gradient의 크기가 줄어든다



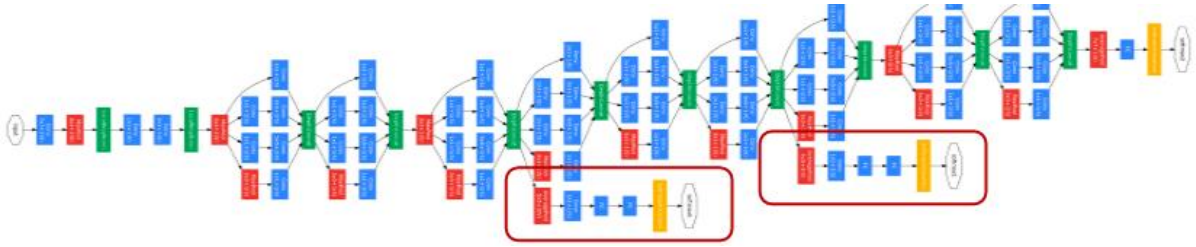
Unit 04 | LSTM

RNN에도 아직 부족한 점이 많습니다.

- Vanishing Gradient Problem : 역전파가 진행되면서 Gradient가 점차 줄어 학습 능력이 저하된다.
- 이미지에서는? (9주차 이미지 심화 참조!)

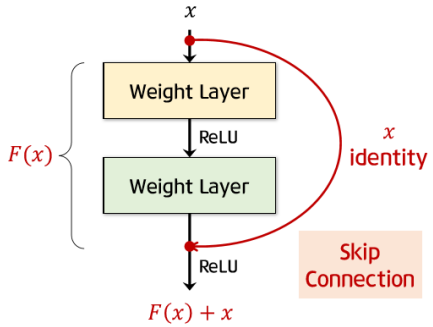
GoogLeNet (Inception v1)

- Auxiliary Classifier



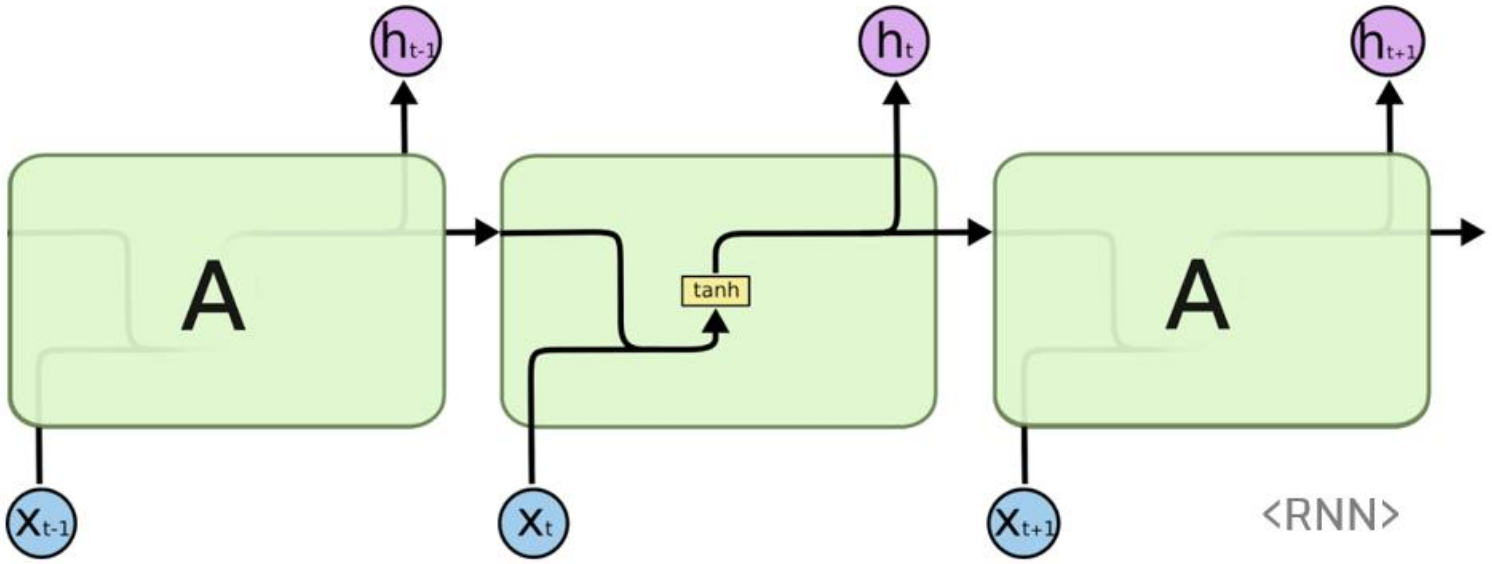
ResNet

- Residual Connection



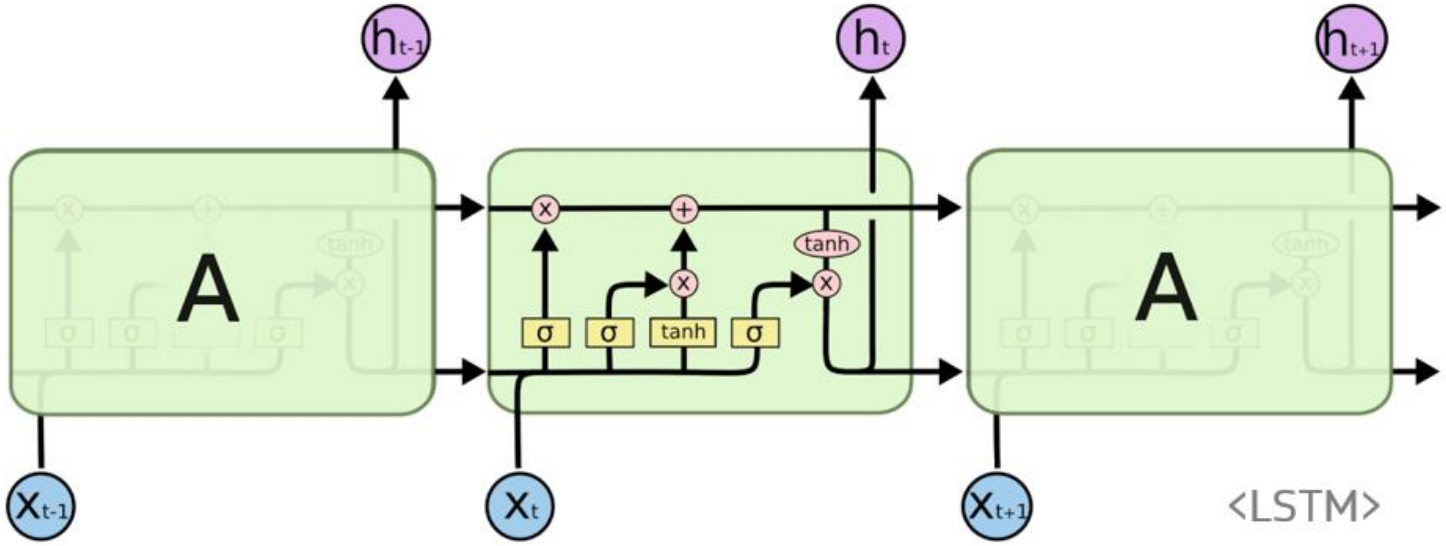
Unit 04 | LSTM

기존 RNN architecture



Unit 04 | LSTM

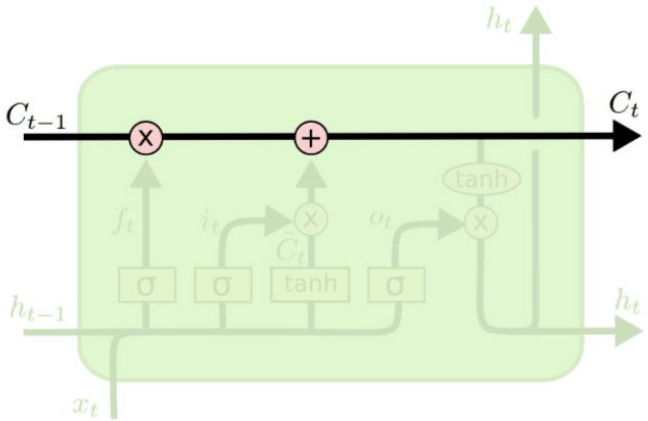
NEW! LSTM architecture

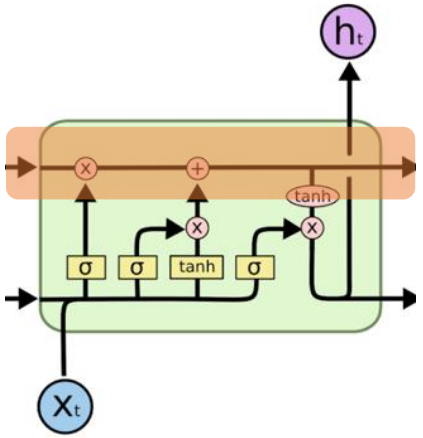


- X (Hadamard product) : 점 단위 곱하기 연산 ex) [1,2,3] X [1,2,3] = [1,4,9]
- Gate : 선택적으로 정보를 제어하는 관문 (σ 3개)
  - σ : sigmoid 활성화 레이어 (출력으로 0 ~ 1 출력)
- Tanh : 쌍곡 탄젠트 활성화 함수 – RNN의 그것과 역할이 같음 (출력으로 -1 ~ 1 출력)

Unit 04 | LSTM

LSTM architecture : Cell State



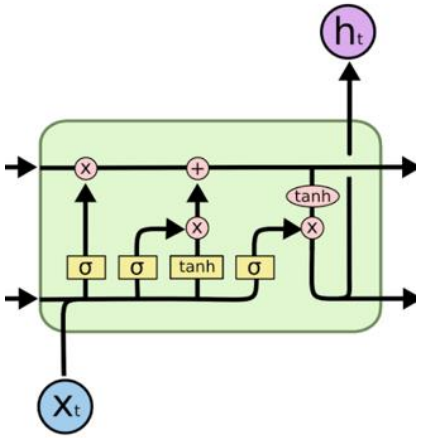


- 하나의 컨베이어 벨트 같이 전체 연결을 관통하여 다음 단계에 전달된다
- 게이트로부터 값이 더해지거나 곱해지면서 정보를 축적한다.
- RNN에 장기 기억(Long-term Memory) 기능을 추가한 것

→ LSTM에서 Vanishing Gradient를 다루기 위한 독자적인 구조!

* 역전파 관련 자세한 내용은 뒤에서 다시 또 다루겠습니다.

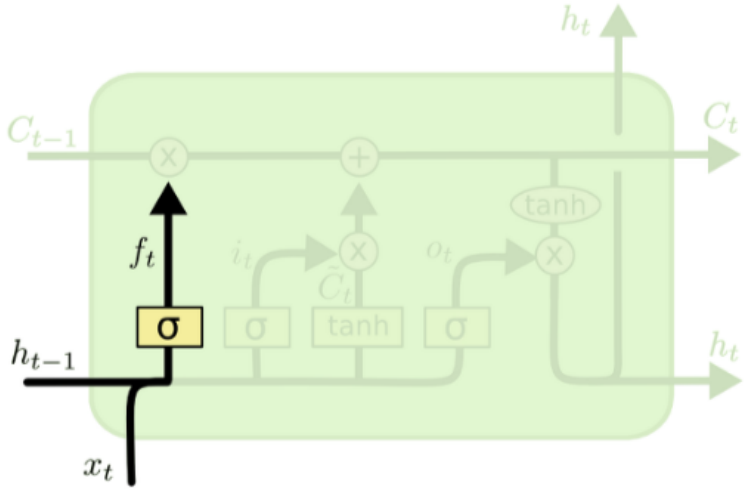
Unit 04 | LSTM

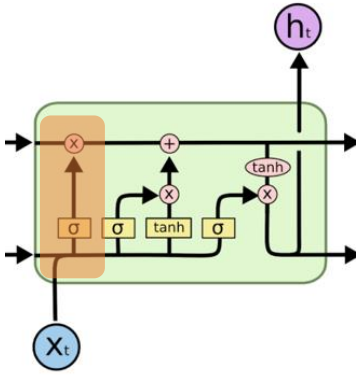


- LSTM (Long Short-Term Memory)
  - 시점 t에서, hidden state $\mathbf{h_t}$ 와 cell state $\mathbf{c_t}$ 로 구성
    - LSTM은 Long-Term Memory인 cell $\mathbf{c_t}$로부터 정보를 **지우고**(잊고), **쓰고**, **읽을** 수 있다.(**erase**, **write**, **read**)
  - Gate를 추가하여 어떤 정보를 지우고/쓰고/읽을지를 조절함.
    - 각 시점 t에서 gate의 원소들은 각 위치에 해당하는 원소들(정보)을 얼마나 사용할 지, 그 개방 정도를 0~1사이의 가중치로 표현함.
    - 현재의 맥락 정보를 바탕으로 gate의 개방 여부가 계산됨

Unit 04 | LSTM

Gate1. Forget gate





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

역할 : Cell state에서 어떤 정보를 잊을지 결정한다

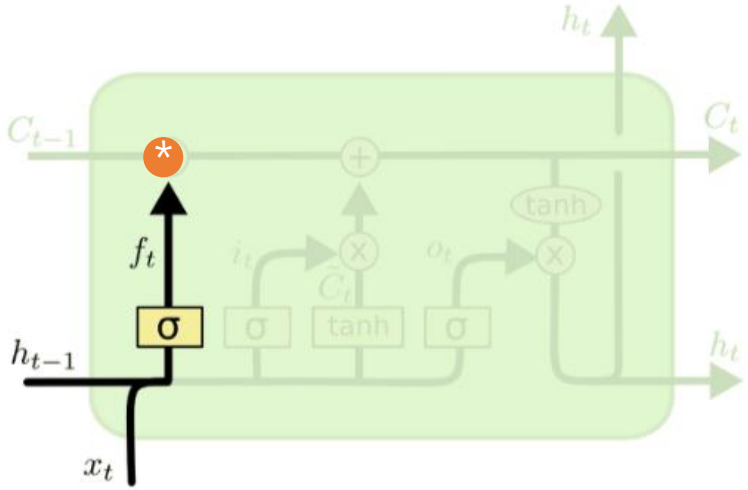
입력 : 현재 입력 정보 (h_{t-1}와 x_t)

출력 : sigmoid 출력(0과 1 사이의 출력값)

→ 1에 가까울 수록 "값을 유지해라", 0에 가까울 수록 "값을 버려라"

Unit 04 | LSTM

Gate1. Forget gate



$$\mathbf{f_t} \quad \begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix}$$

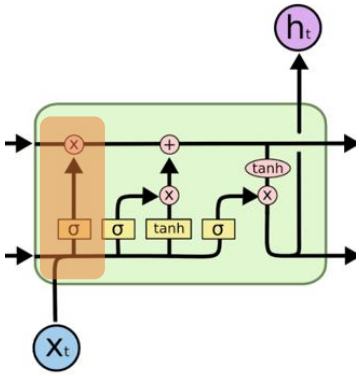
$$* \qquad *$$

$$\mathbf{c_{t-1}} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\| \qquad \|$$

$$\mathbf{f_t} * \mathbf{c_{t-1}} \quad \begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix}$$

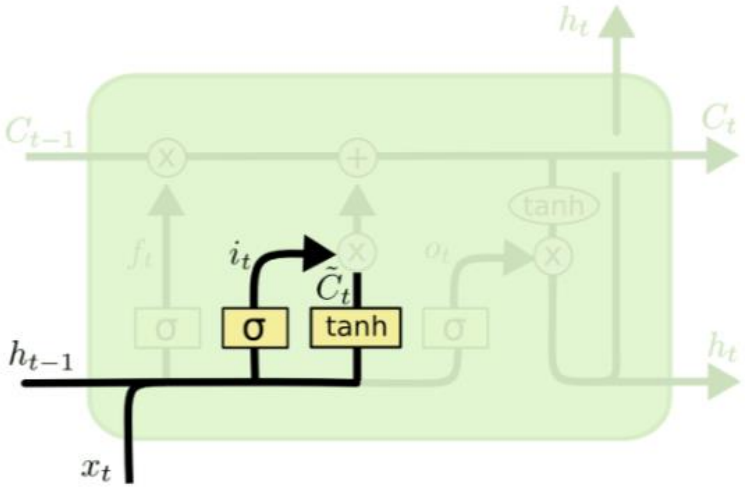
망각 유지

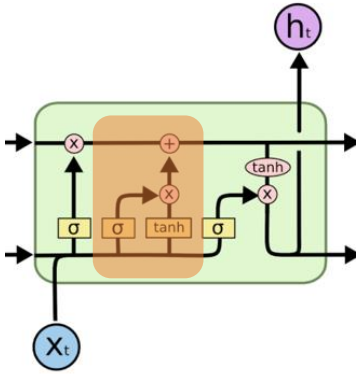


- 계산된 forget gate f_t를 이전 시점의 장기기억 C_{t-1}에 요소별로 곱(Hadamard product)하여 이전 시점의 각 정보의 망각(또는 유지)를 결정

Unit 04 | LSTM

Gate2. Input gate





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

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

역할 : Cell state에 어떤 정보를 더해줄지 결정한다

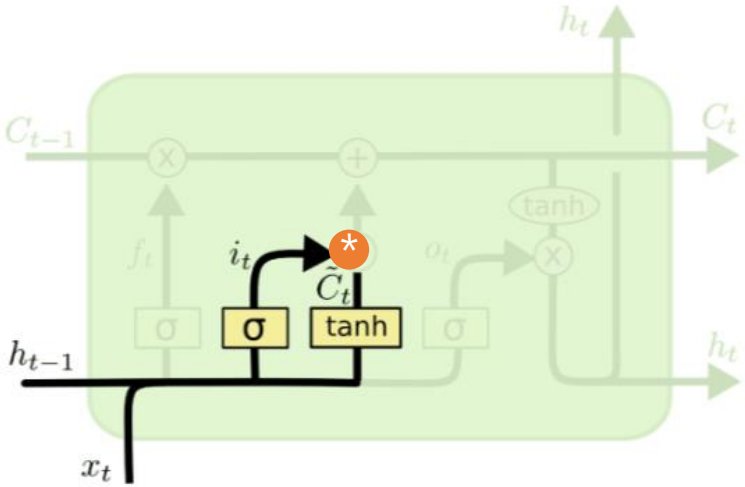
입력 : 현재 입력 정보 (h_{t-1}와 x_t)

출력 : tanh의 출력(-1 ~ 1) → 필요한 정보를 추출

sigmoid의 출력(0 ~ 1) → tanh에서 추출한 정보를 취사선택

Unit 04 | LSTM

Gate1. Forget gate



$$\tilde{\mathbf{c}}_t \begin{bmatrix} -1 & 0.8 \\ -0.5 & 0.9 \end{bmatrix}$$

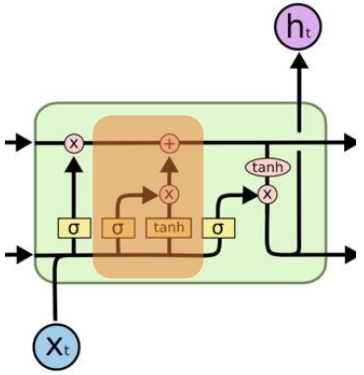
$$* \quad * \quad \begin{bmatrix} 0.9 & 0.1 \\ 0.8 & 0 \end{bmatrix}$$

$$\mathbf{i}_t \quad =$$

$$\| \quad \begin{bmatrix} -0.9 & 0.08 \\ -0.4 & 0 \end{bmatrix}$$

$$\tilde{\mathbf{c}}_t * \mathbf{i}_t$$

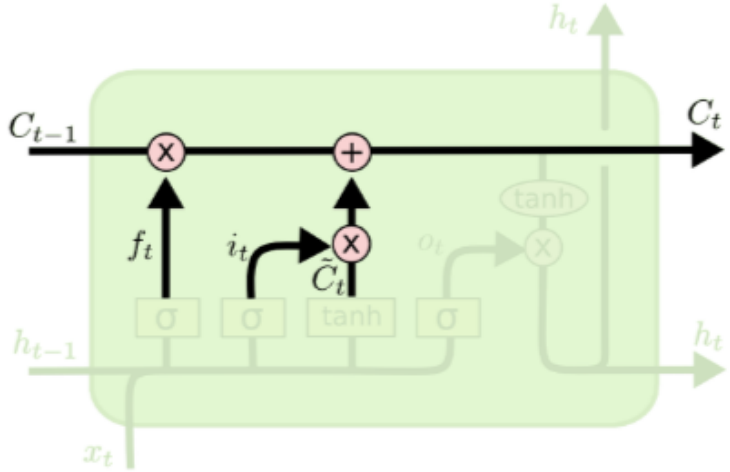
유지 망각



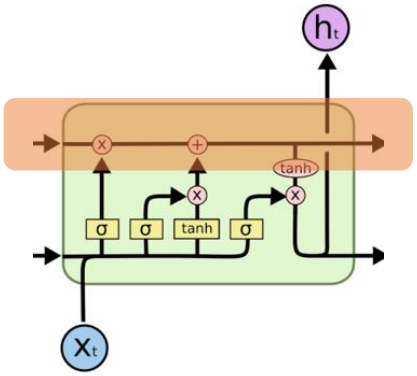
• 계산된 input gate i_t 를 현재의 입력 정보를 가공한 새로운 $\tilde{C}_t$ 에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 장기 기억($f_t * C_{t-1}$)에 더할 것인지를 결정

Unit 04 | LSTM

Cell State 돌아보기



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

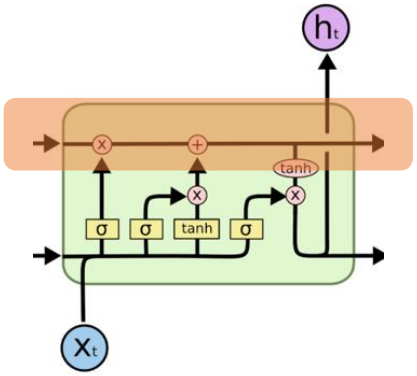


■ 지금까지의 과정

1. Cell state(C_{t-1})와 Forget Gate(f_t)를 pointwise(요소 별) 곱 해준다 → 망각효과
2. Cell state(f_t*C_{t-1})와 Input Gate(i_t*$\tilde{C}_{t-1}$)를 합한다 → 정보 입력효과
3. 망각과 정보 입력을 마친 현재의 Cell state(C_t)는 다음 Time Step으로 넘어간다

Unit 04 | LSTM

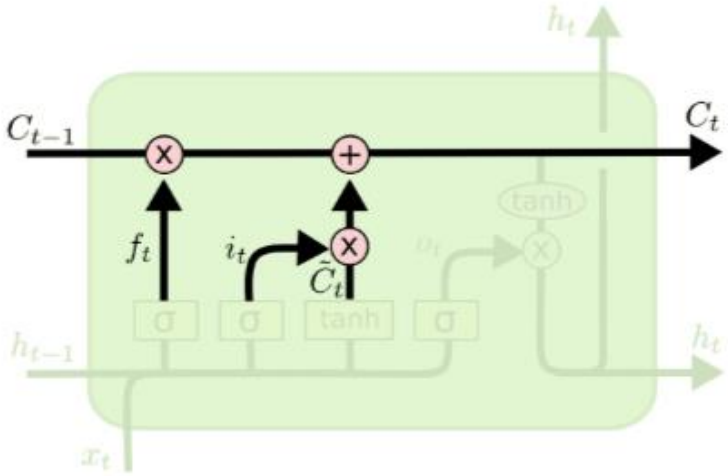
Cell State 돌아보기



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

$$\begin{bmatrix} 0 & 0.8 \\ 0 & 0.9 \end{bmatrix} + \begin{bmatrix} -0.9 & 0.08 \\ -0.4 & 0 \end{bmatrix} = \begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}$$

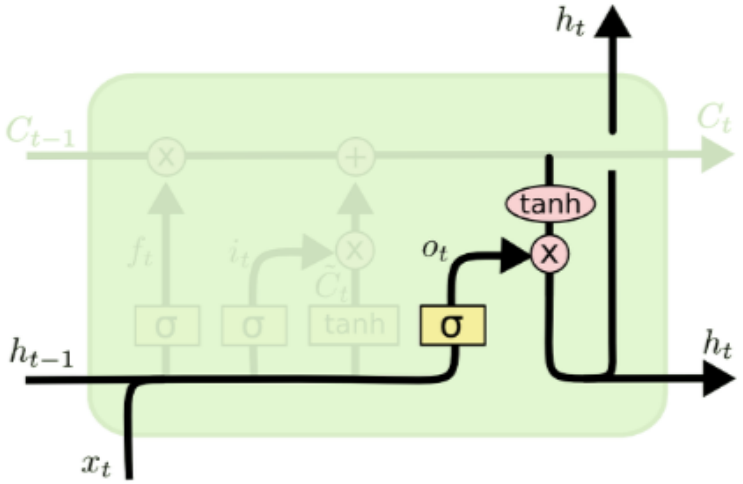
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

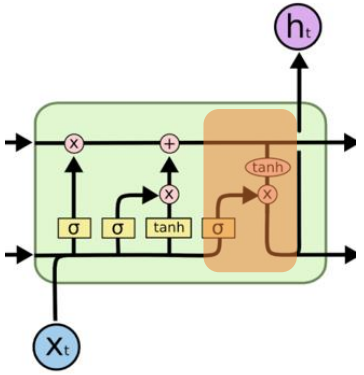


$$\begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}$$

Unit 04 | LSTM

Gate3. Output gate





$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

$$tanh\left(\begin{bmatrix} -0.9 & 0.88 \\ -0.4 & 0.9 \end{bmatrix}\right)$$

역할 : Cell state와 현재 입력으로부터 출력값 (h_t)정보를 선택

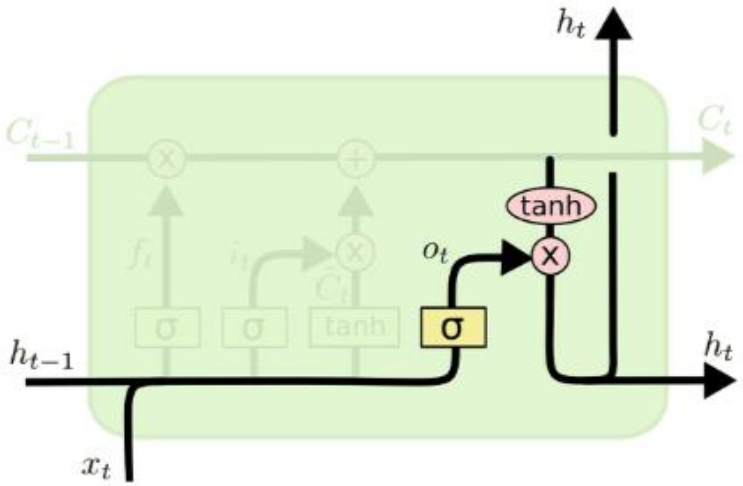
입력 : 현재 입력 정보 (h_{t-1}와 x_t)와 망각, 입력 게이트를 거친 Cell state(C_t)

출력 : Cell state의 tanh 출력값과 현재 입력의 sigmoid 출력값의 합 (0 ~ 1)

→ 장기기억으로부터 정보를 추출하고, 현재 입력과 대비해서 중요한 것만 선별하자

Unit 04 | LSTM

Gate3. Output gate



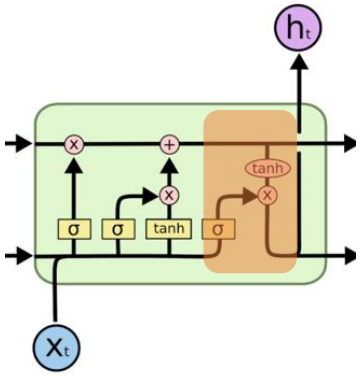
$$\mathbf{h_t^{(raw)}} \begin{bmatrix} -0.9 & 0.9 \\ -0.7 & 0.7 \end{bmatrix}$$

$$* \qquad *$$

$$\mathbf{o_t} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\| \qquad \|$$

$$\mathbf{h_t} \begin{bmatrix} 0 & 0 \\ -0.7 & 0.7 \end{bmatrix}$$

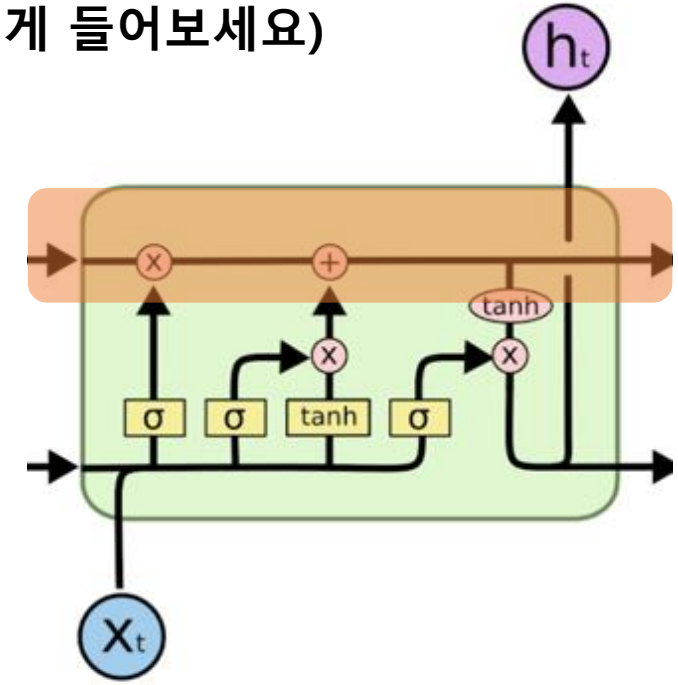


■ 계산된 output gate o_t 를 현재의 장기기억(기억셀)으로부터 꺼내온 $h_{t(raw)}$ $(\tanh(C_t))$에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 단기기억(h_t)으로 사용할지를 결정

Unit 04 | LSTM

LSTM Back-Prop (어려운 이야기입니다 맞아요. 그러니 가볍게 들어보세요)

*** Cell state가 왜 기울기 소실을 방지하는가?**



- Cell state의 역전파 과정에서 만나는 것은 딱 2가지
  1. 덧셈
  2. 요소 별 곱셈

- 덧셈 → 기울기를 그대로 흘린다 . 신경 쓰지 말자
- 곱셈 → 생각을 조금 해보자

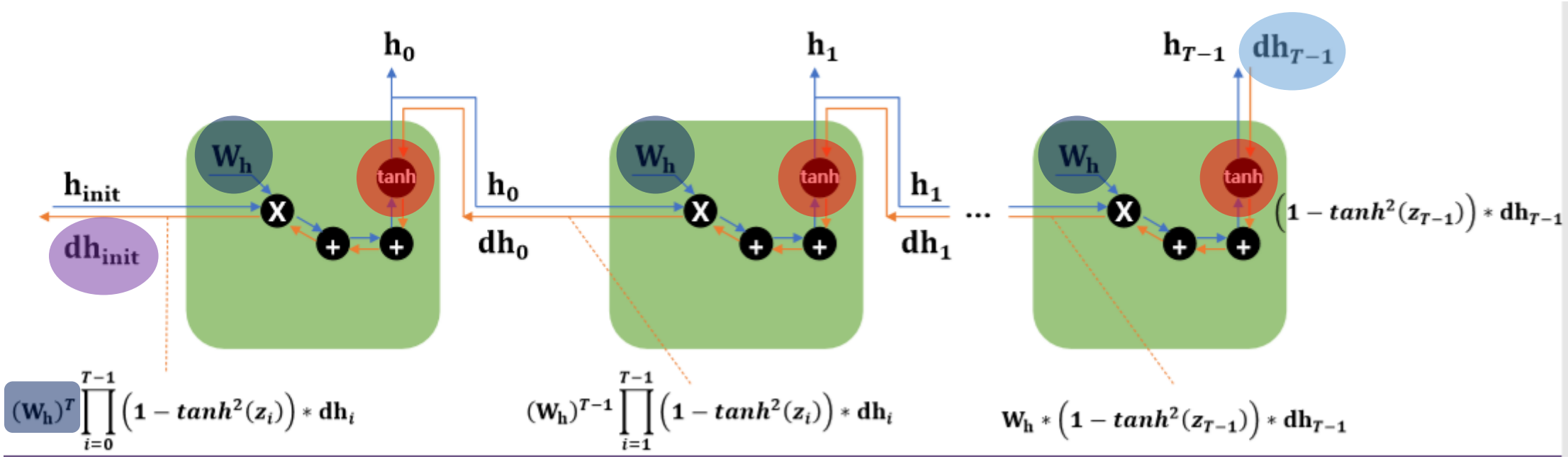
Unit 04 | LSTM

RNN은 왜 문제가 생겼는지 다시 떠올려보면...

1. 수 많은 활성화 함수(tanh)를 거쳐서 기울기 소실 발생 → 이건 알고있지

그러나 여기서 새로이 밝히는 문제점 또 하나

2. 똑같은 가중치를 반복해서 곱해주고 있다.



Ctrl C + Ctrl V

Unit 04 | LSTM

이거는 아주 당연하고 쉬운 이야기에요

Case 1) 1보다 큰 수의 똑같은 행렬을 계속 곱하면?

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \text{무지하게 커진다!!!!}$$

Case 2) 1보다 작고 0보다 큰 수의 행렬을 계속 곱하면?

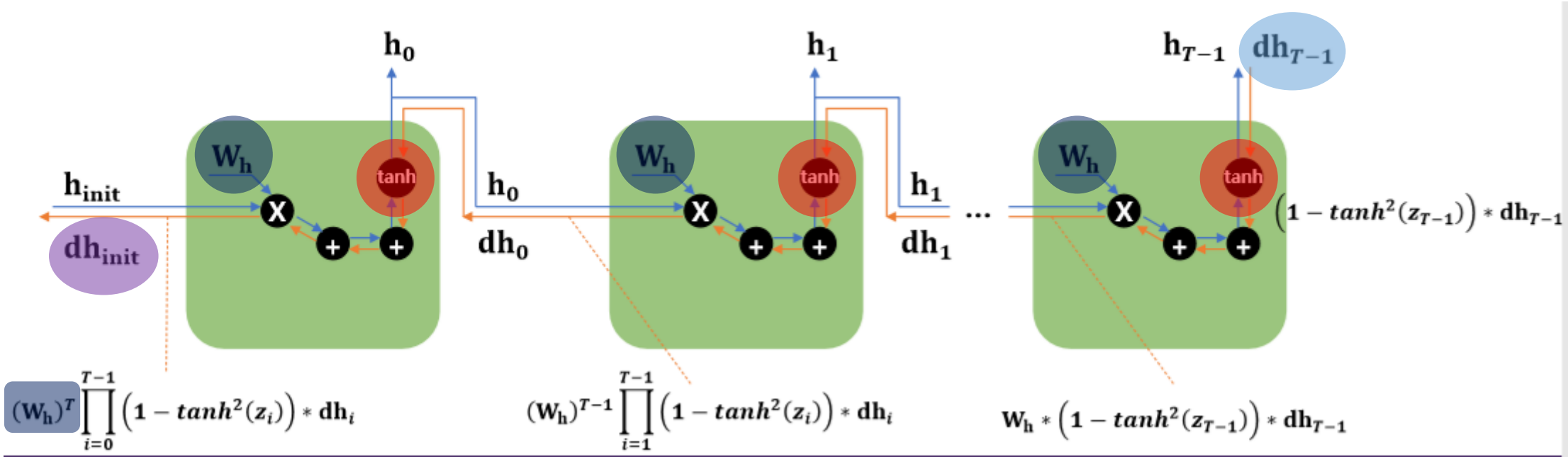
$$\begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} = \text{엄청나게 작아진다...}$$

Unit 04 | LSTM

RNN은 왜 문제가 생겼는지 다시 생각해볼게요

2. 똑같은 가중치를 반복해서 곱해주고 있다.

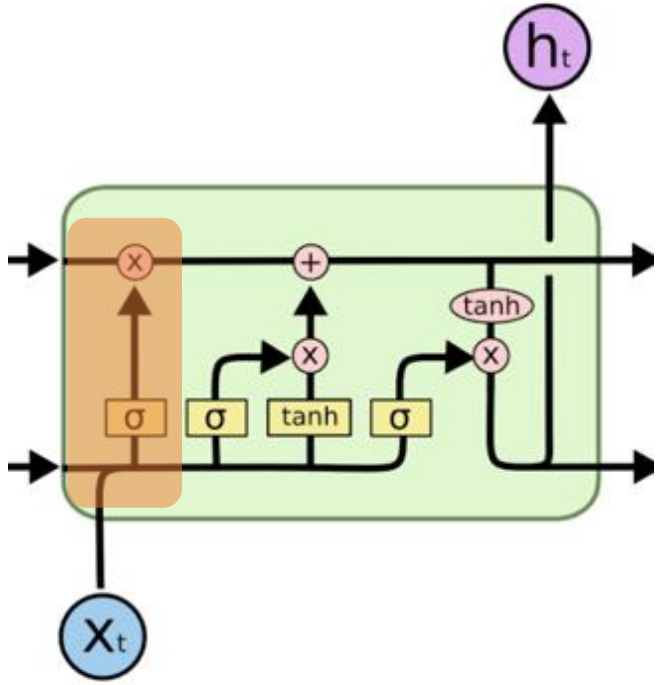
→ 무지하게 커지든 엄청나게 작아지든 둘 중 하나



Unit 04 | LSTM

그래서 LSTM은?

Q. Cell state가 곱셈에서 만나는 Forget gate(f_t) 값도
매 Time Step 마다 똑같지 않나?



A. **No!** (45번 Page에서 Forget gate 참조)

역할 : Cell state에서 어떤 정보를 잊을지 결정한다
입력 : 현재 입력 정보 (h_{t-1}와 x_t)
출력 : sigmoid 출력(0과 1 사이의 출력값)
→ 똑같은 W_t가 아닌 매번 입력에 따른 0 ~ 1사이
sigmoid 활성화 함수의 출력값이 곱해진다.
→ 적어도 똑같은 행렬을 반복적으로 곱하는 RNN보다 훨씬 낫다

Unit 03 | 순환신경망

■ **LSTM은 RNN의 기울기 소실 문제를 해결해주었습니다.**

1. LSTM은 Cell state라는 장기 기억 장치를 만들어 두었습니다
2. Cell state는 특성 상 기울기 소실 문제에서 비교적 RNN보다 자유롭습니다
3. 3가지 Gate(망각, 입력, 출력)로 정보의 흐름 강도를 조절하는 기능이 있습니다
4. RNN 계열의 신경망은 여러 층을 쌓을 수도 있습니다.
5. 최종 Output에 어떤 Classifier를 다느냐에 따라 다양한 Task를 수행할 수 있습니다.

References

참고자료

투빅스 10기 이준걸님의 강의자료

투빅스 11기 김대웅님의 10주차 강의자료

투빅스 12기 김탁영님의 이미지심화 강의자료

투빅스 12기 김태한님의 나이브베이즈 강의자료

투빅스 12기 김태욱님의 텍스트 기초 강의자료

투빅스 12기 김주호님의 EDA와 전처리 강의자료

고려대학교 산업공학과 강필성 교수님의 강의자료 : https://github.com/pilsung-kang/text-analytics

스탠포드 강의 cs224n

Ratsgo님 블로그 자료 https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

Colah's Blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/

부족한 강의이지만
들어주셔서 감사합니다.