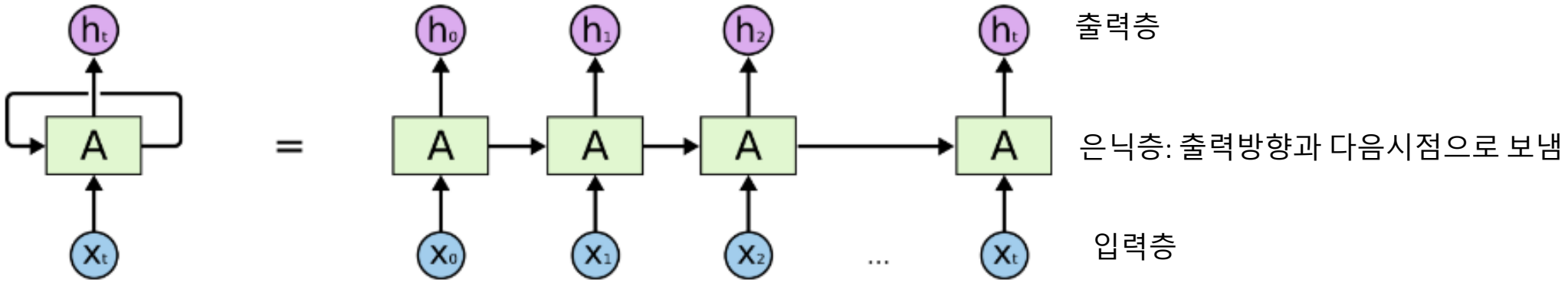
8. 순환신경망(Recurrent Neural Network, RNN)

1. **RNN(Recurrent Neural Network)**
2. **LSTM(Long Short-Term Memory)**
3. **GRU(Gate Recurrent Unit)**

1. RNN(Recurrent Neural Netwok, RNN)

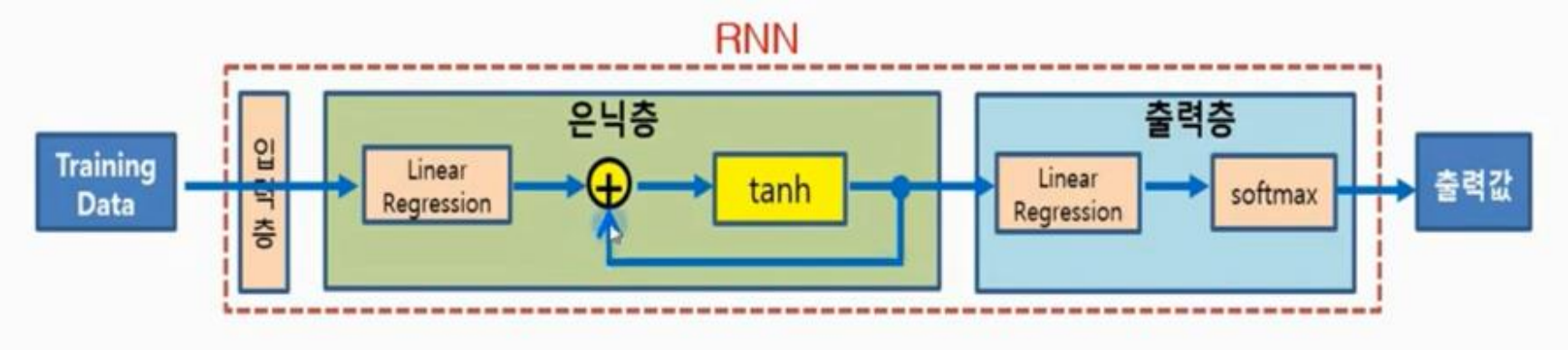
❖ 순환신경망(Recurrent Neural Network, RNN)

- **시계열 데이터와 같이 시간의 흐름에 따라 변화하는 데이터를 학습하기 위해** 등장한 인공신경망



1. RNN(Recurrent Neural Netwok, RNN)

❖ RNN의 구조와 원리



② 순서(sequence)가 있는 데이터를 처리하는 데 강점을 가진 신경망

순서(sequence)가 있는 데이터 ?

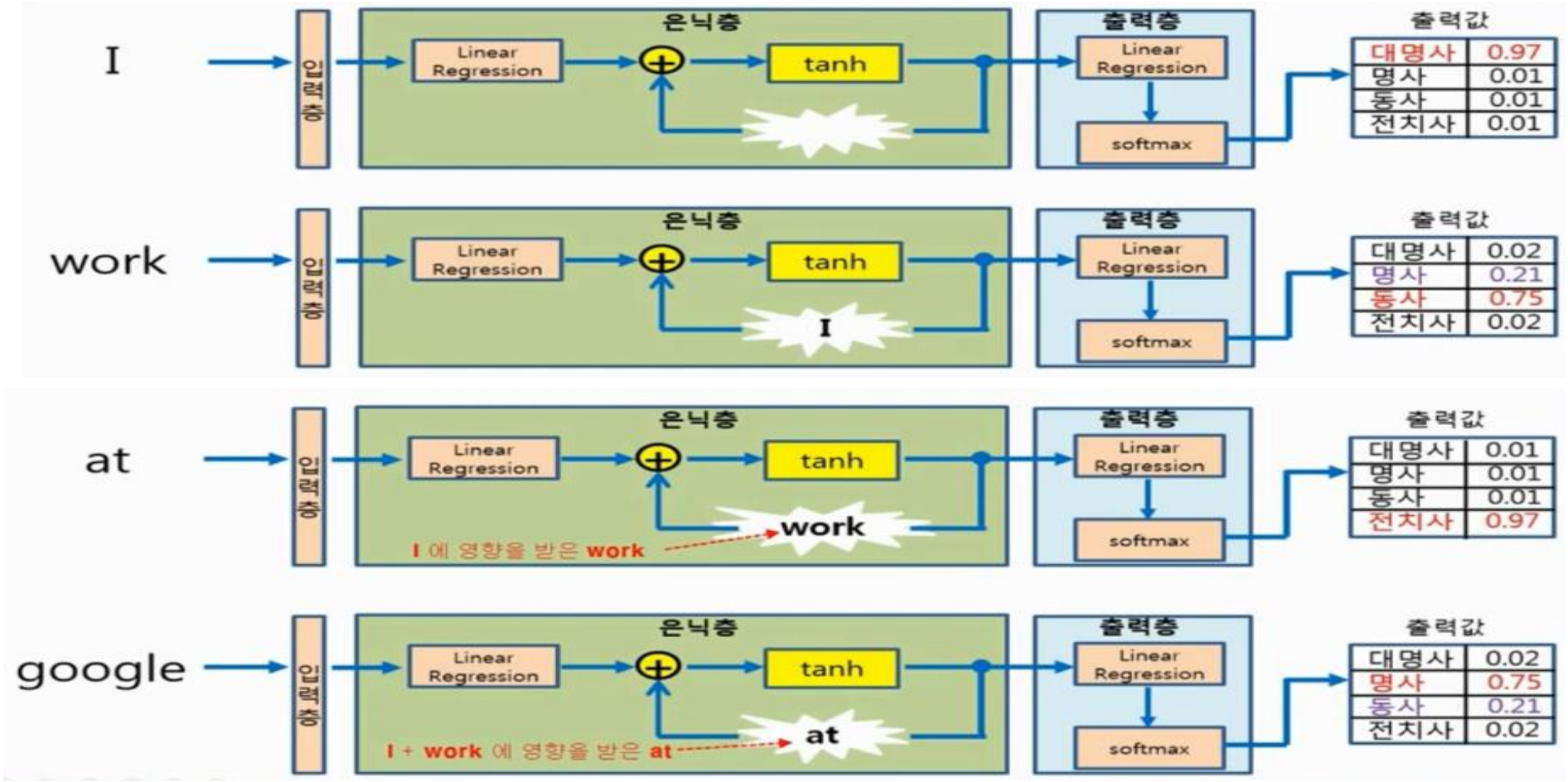
1. RNN(Recurrent Neural Netwok, RNN)

❖ 순서(sequence)가 있는 데이터

- 문장이나 음성같은 연속적인 데이터를 말함. 이런 데이터는 문장에서 놓여진 위치(순서)에 따라 의미가 달라지는 것을 알 수 있음
- 즉, 현재 데이터 의미를 알기 위해서는 이전에 놓여 있는 과거 데이터도 알고 있어야 함(I work/ I google[대명사+동사], at google/ at work[전치사+명사])
- RNN은 이러한 과거 데이터를 알기 위해서 ① 은닉층내에서 순환(Recurrent) 구조를 이용하여 과거의 데이터를 기억해 두고 있다가 ② 새롭게 입력된 데이터와 은닉층에서 기억하고 있는 데이터를 연결 시켜서 그 의미를 알아내는 기능을 가지고 있음

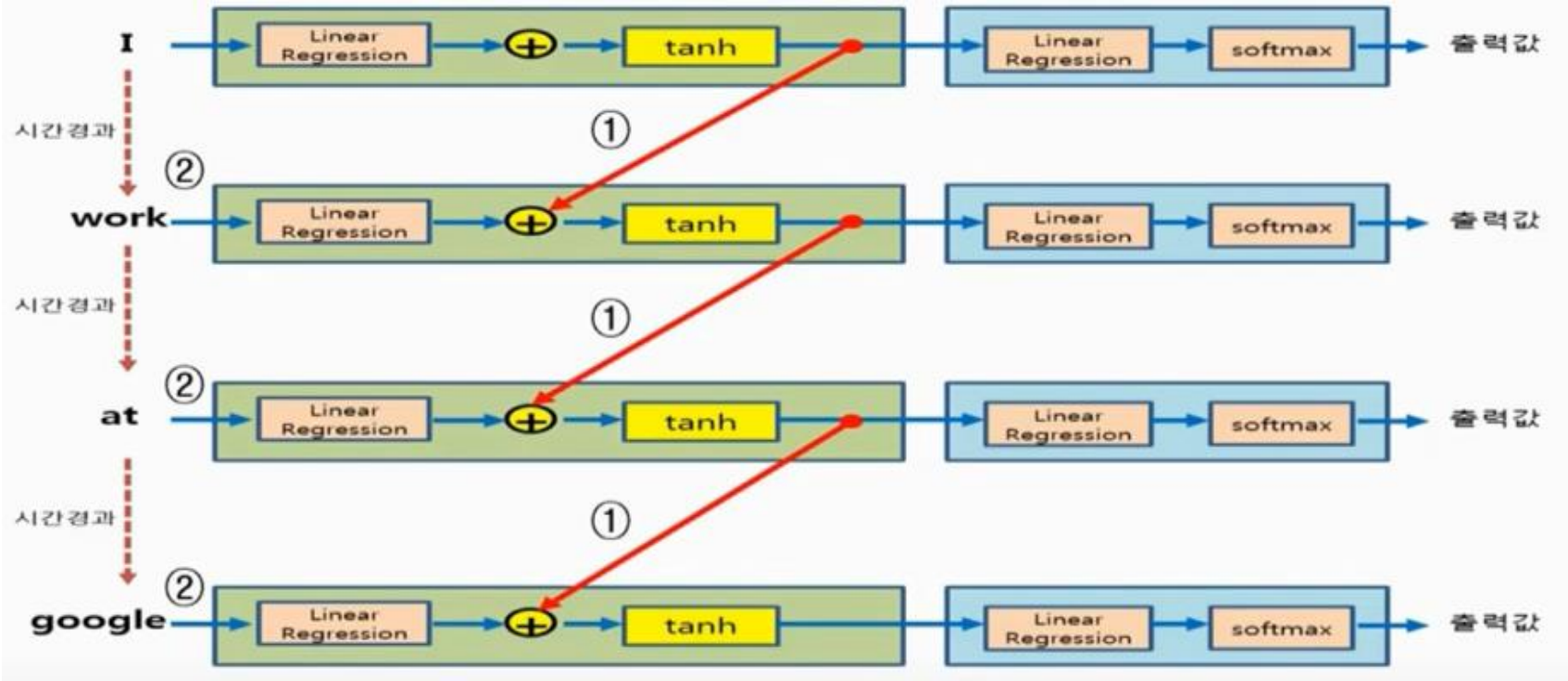
1. RNN(Recurrent Neural Netwok, RNN)

❖ **RNN 구조와 원리 – 동작원리: 정성적 분석(I work at google)**



1. RNN(Recurrent Neural Netwok, RNN)

❖ 시간 개념을 포함한 RNN 구조

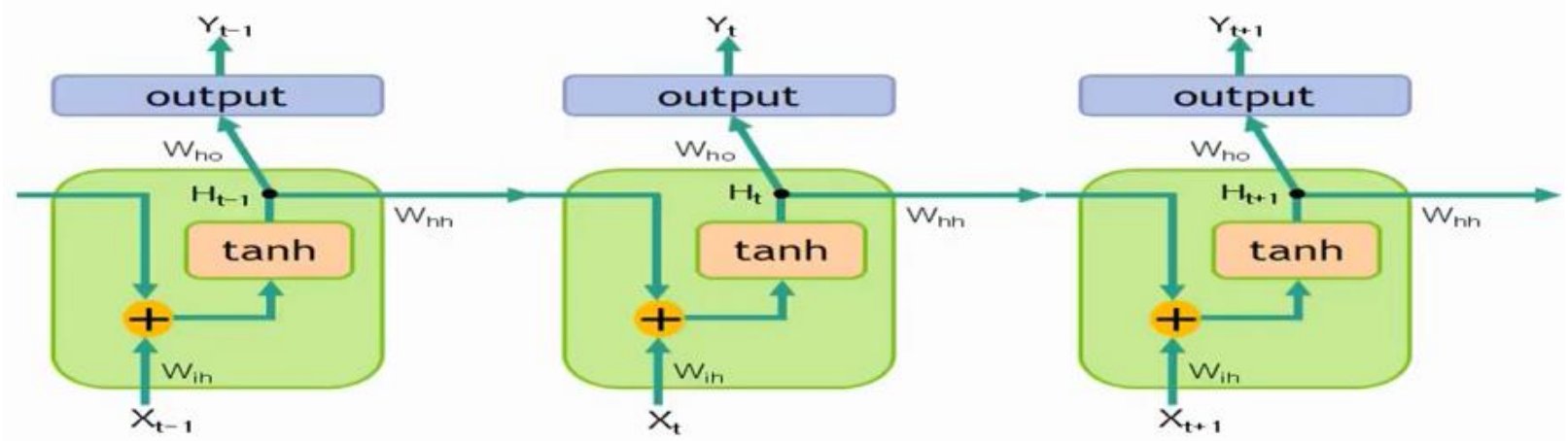


- 순환 구조를 ① 은닉층에서 기억하는 과거의 데이터(붉은색 화살표)와 ② 일정 시간이 지난 후에 입력되는 데이터를 연결시켜 주는 구조로 바꾸어서 생각해볼 수 있음
- 즉 문장이나 음성 같은 순서가 있는 데이터라는 것은 시간의 경과에 따라서 데이터가 순차적으로 들어온다는 것과 같은 의미라는 것을 알 수 있음

1. RNN(Recurrent Neural Netwok, RNN)

❖ SimpleRNN 레이어

- 가장 간단한 형태의 RNN 레이어이며 기본 구조 다음과 같다.



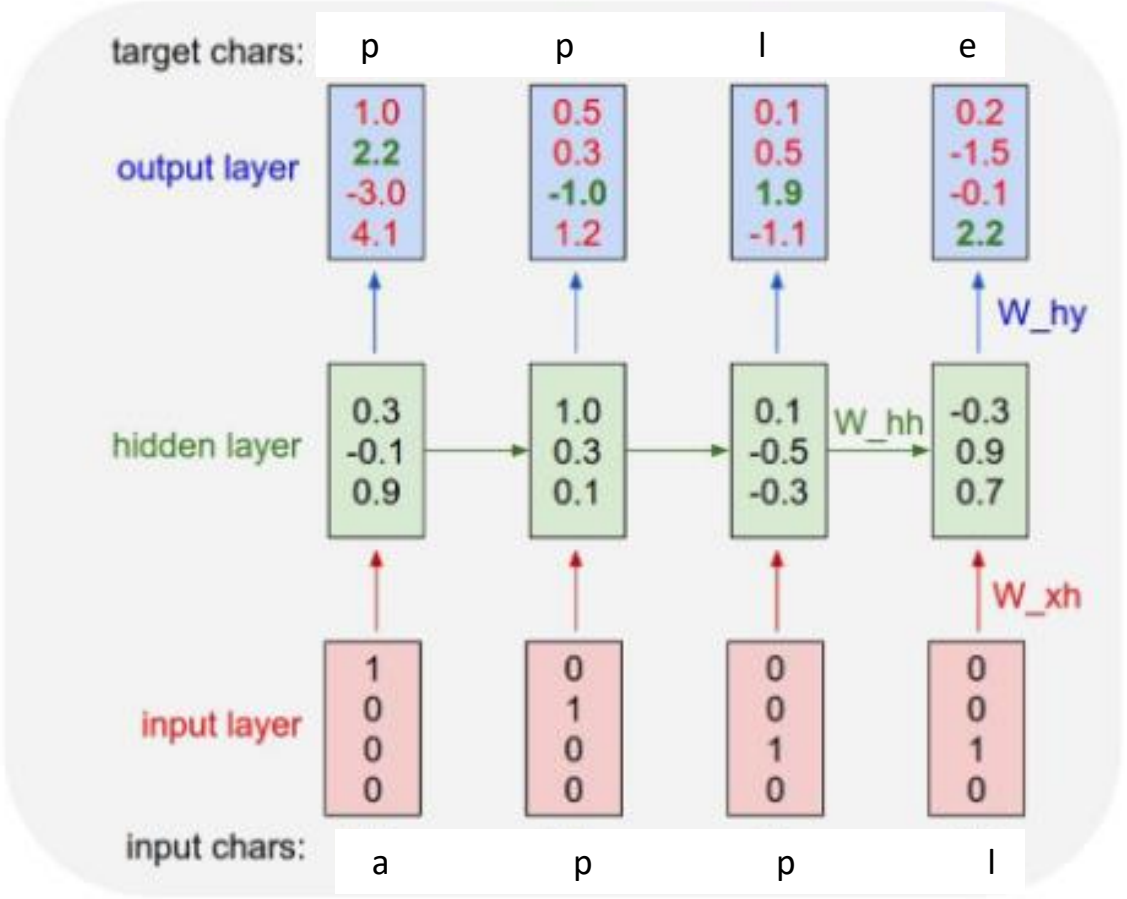
✓ X_{t-1}, X_t, X_{t+1} 은 입력 데이터를 나타내고 H_{t-1}, H_t, H_{t+1} 등은 은닉층 개념의 SimpleRNN 레이어 출력 값을, 그리고 Y_{t-1}, Y_t, Y_{t+1} 등은 출력층의 출력 값을 나타냄

✓ 학습 대상의 가중치는 ① 입력층과 은닉층 사이의 가중치 W_{ih} ② 시간 t 에서의 은닉층과 시간 t+1 에서의 은닉층 간의 가중치 W_{hh} ③ 은닉층과 출력층 사이의 가중치 W_{ho}

✓ 시간 t 에서 은닉층 SimpleRNN 레이어 출력 $H_t = \tanh(X_tW_{ih} + H_{t-1}W_{hh})$

1. RNN(Recurrent Neural Netwok, RNN)

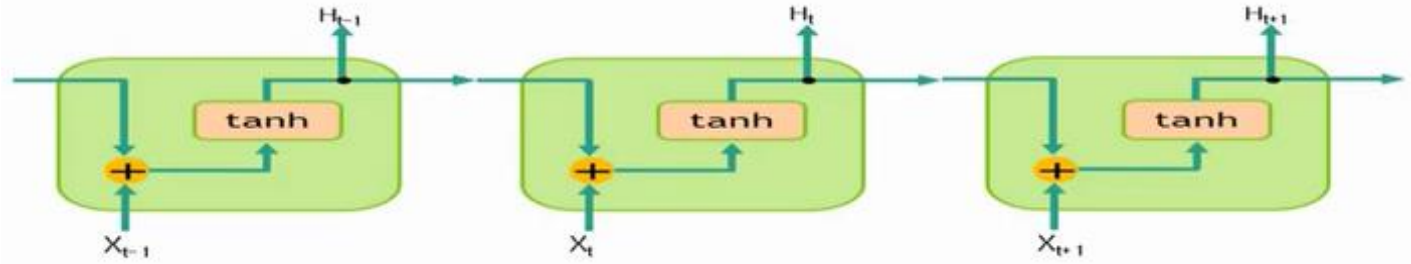
❖ **RNN 내부 학습 과정**



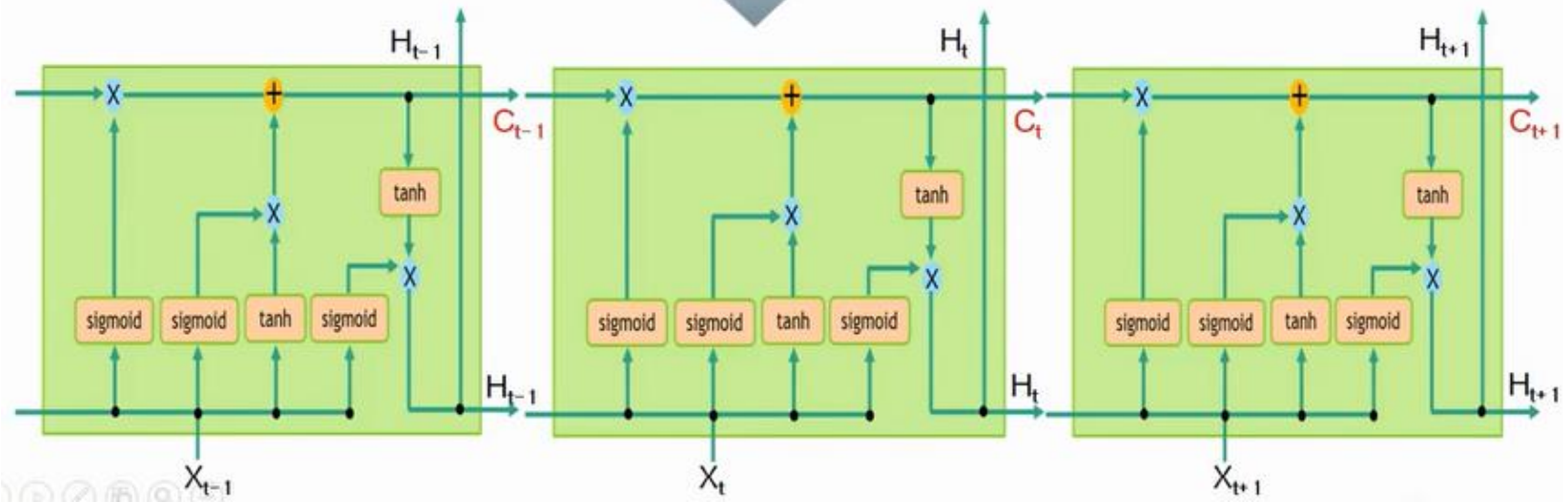
2. LSTM(Long Short-Term Memory)

❖ RNN vs LSTM

- LSTM 레이어시간 t에서의 출력값 Ht 이외에 , LSTM레이어 사이에서 공유되는 셀 상태(cell state) Ct라는 변수가 추가적으로 공유되는 특징이 있음



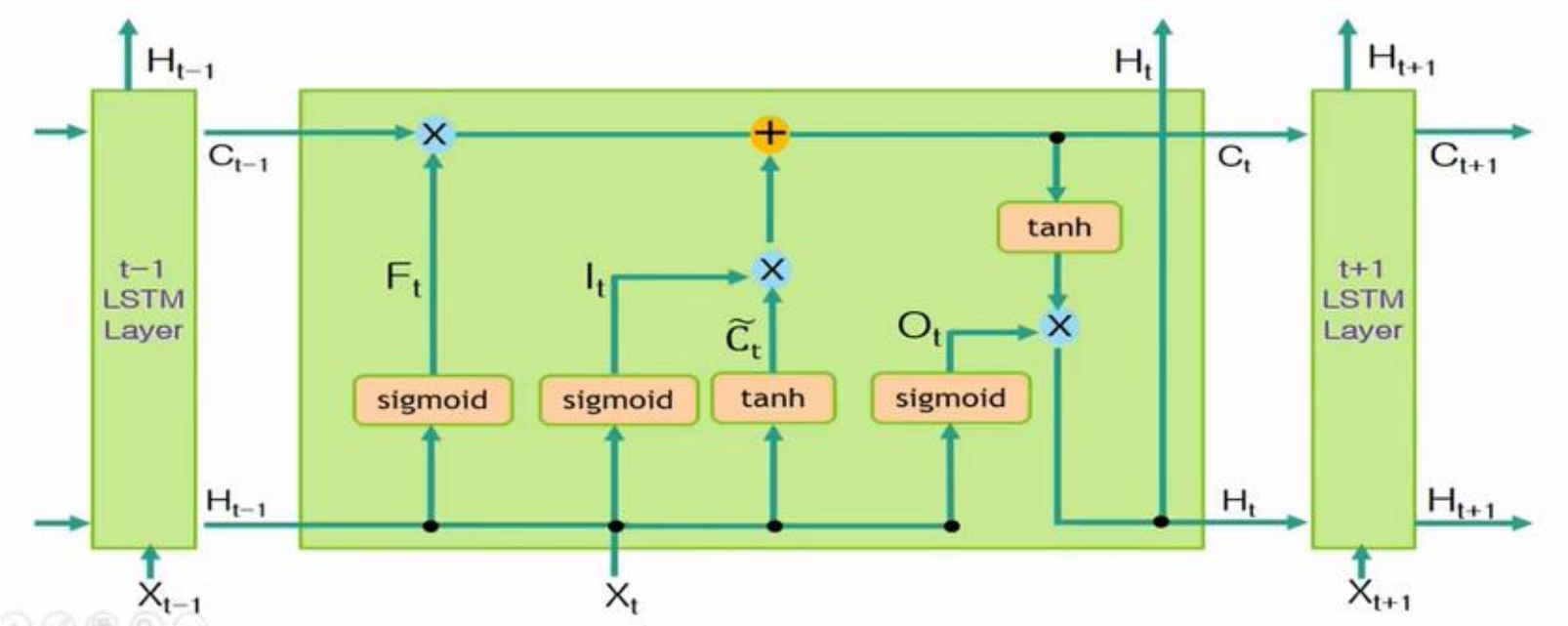
LSTM 의 셀 상태가 다음 레이어로 전달되면서 기존의 상태를 보존하므로 장기의존성 문제를 해결할 수 있음



2. LSTM(Long Short-Term Memory)

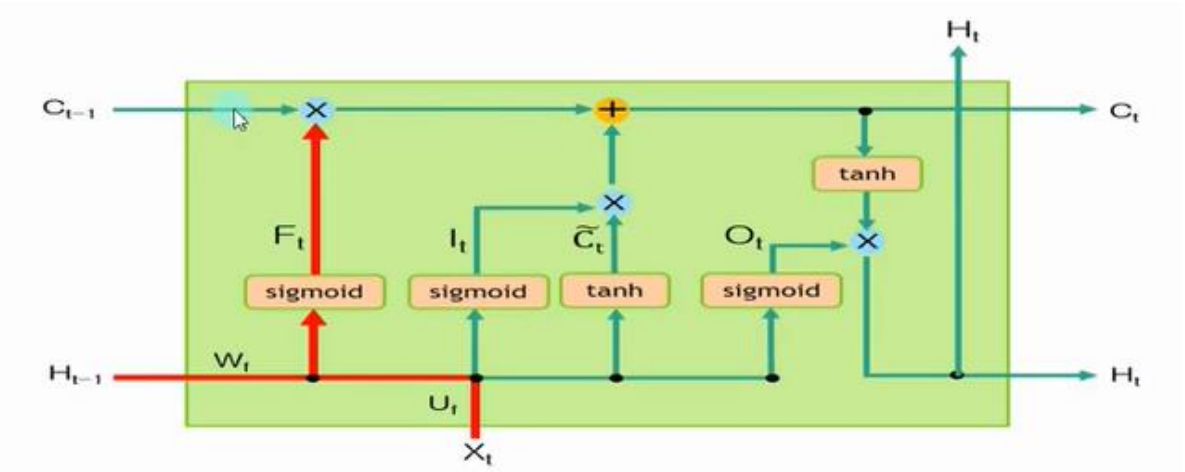
❖ LSTM 구조 - 개요

- LSTM 핵심은 이전 단계 정보를 memory cell에 저장하여 다음 단계로 전달하는 것임
  - LSTM은 현재 시점의 정보를 바탕으로 과거 내용을 얼마나 잊을지 또는 기억할지 등을 계산하고, 그 결과에 현재 정보를 추가해서 다음 시점으로 정보를 전달
  - 이러한 기능을 구현하기 위해 LSTM을 forget gate, input gate, output gate 등으로 구성되며, 이러한 gate는 memory cell에 정보를 저장하고 다음 단계로 전달하는 역할 수행



2. LSTM(Long Short-Term Memory)

❖ LSTM 구조 – forget gate



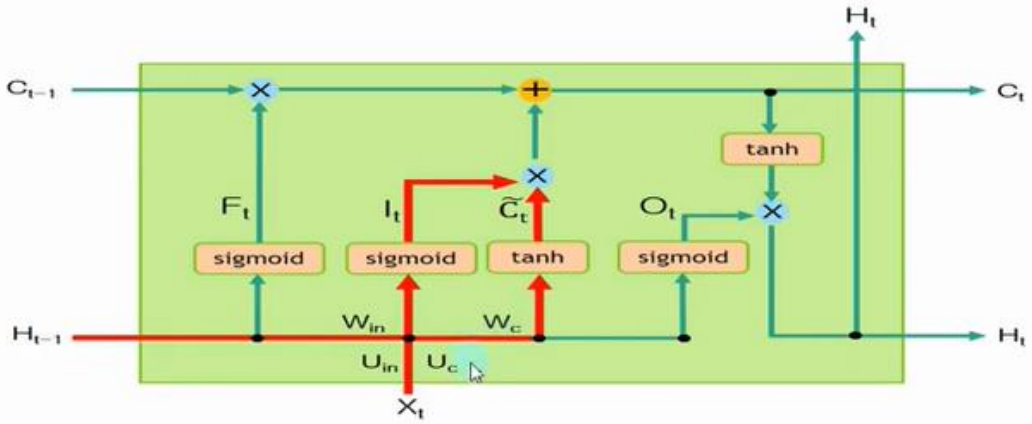
$$F_t = \text{sigmoid}(U_f X_t + W_f H_{t-1} + b_f)$$

- forget gate는 과거의 정보를 얼마나 잊을지(또는 기억할지) 결정하는 게이트이며, ① 현 시점의 데이터 X_t 와 과거의 은닉층 값 H_{t-1}에 각각의 가중치 W_f, U_f 곱한 후에 ② 그 값들을 더한 후 sigmoid 함수를 적용하는 과정임

- sigmoid 함수 값은 0~1 사이 값을 가지므로, 계산 값이 1에 가깝다면 과거 정보를 많이 활용한다는 의미이고, 만약 sigmoid 값이 0에 가깝다면 과거 정보를 많이 잃게 되는 원리임

2. LSTM(Long Short-Term Memory)

❖ **LSTM 구조 –input gate**



$$\tilde{C}_t = \tanh(U_c X_t + W_c H_{t-1} + b_c)$$

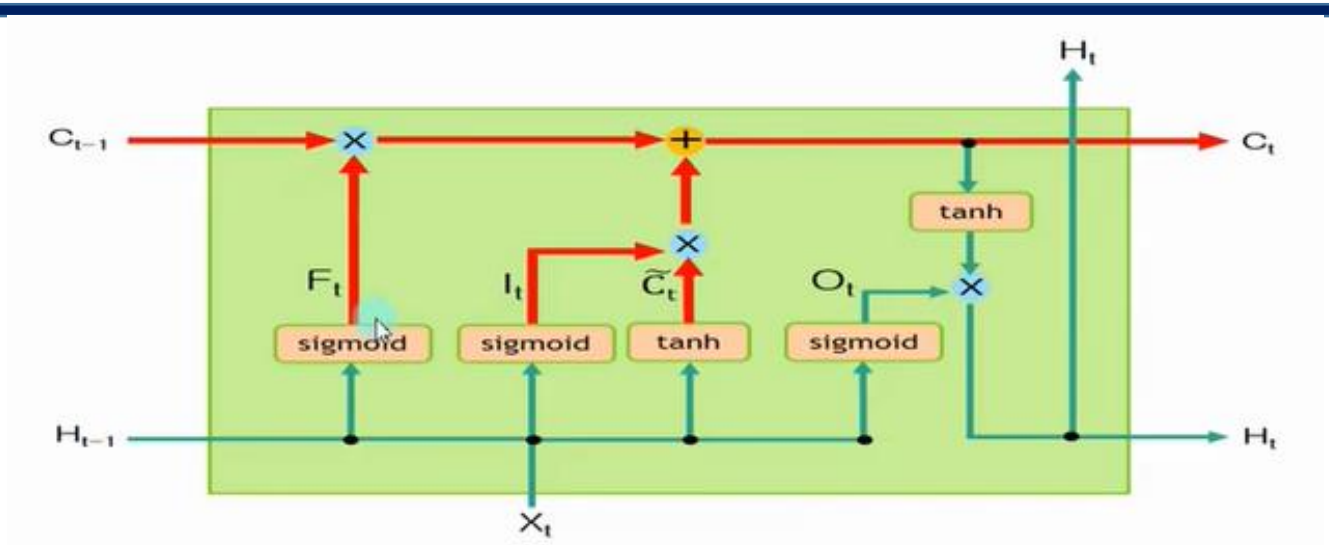
$$I_t = \text{sigmoid}(U_{in} X_t + W_{in} H_{t-1} + b_i)$$

- input gate는 ① 현재 시점의 데이터 X_t와 과거의 은닉층 값 H_{t-1}에 각각의 가중치 W_{in}, U_{in} 곱하고 더한 결과에 sigmoid 함수를 적용하여, 어떤 정보를 업데이트 할 지 결정하고 (I_t)
  ② 현재 시점의 데이터 X_t와 과거 은닉층 값 H_{t-1}에 각각의 가중치 W_c, U_c 곱하여 더한 후 tanh 함수를 적용하여 현재 시점의 새로운 정보를 생성함 ($\tilde{C}_t$)

- 즉 현 시점에서 실제로 갖고 있는 정보가 얼마나 중요한지를 반영하여 cell에 기록함

2. LSTM(Long Short-Term Memory)

❖ **LSTM 구조 – cell state**



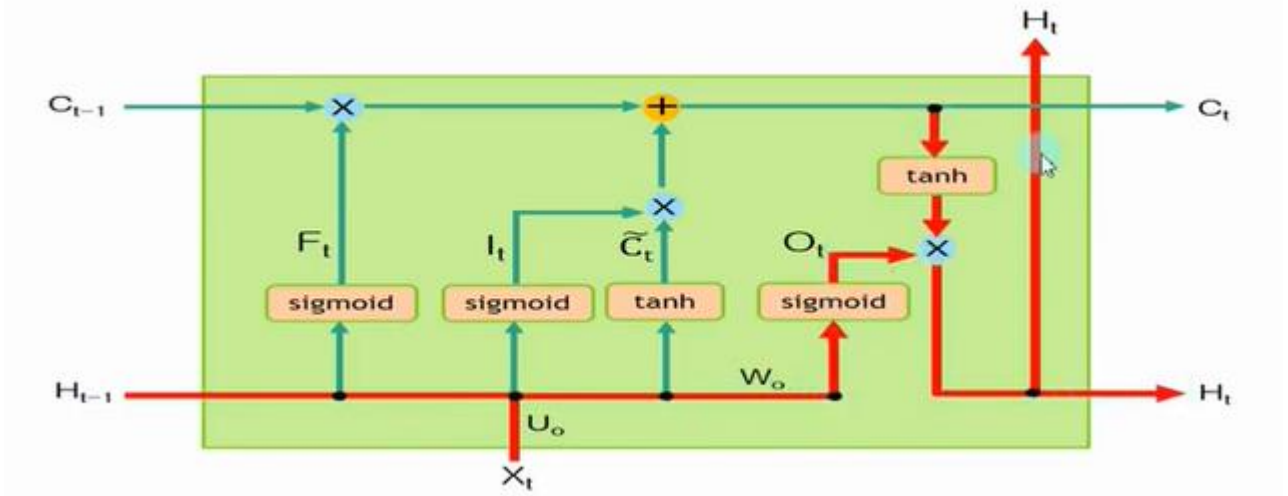
$$C_t = F_t C_{t-1} + I_t \tilde{C}_t$$

- cell state는 forget gate 출력 값 F_t, input gate 출력 I_t, $\tilde{C}_t$ 값을 이용하여 memory cell에 저장하는 단계임
- 즉 과거의 정보를 forget gate 에서 계산된 만큼 잊고(또는 기억하고), 현 시점의 정보 값에 입력 게이트의 중요도 만큼 곱해준 것을 더해서 현재 시점 기준의 memory cell 값을 계산.

※ 곱하기 표시는 모두 pointwise operation 을 나타냄

2. LSTM(Long Short-Term Memory)

❖ **LSTM – output gate**



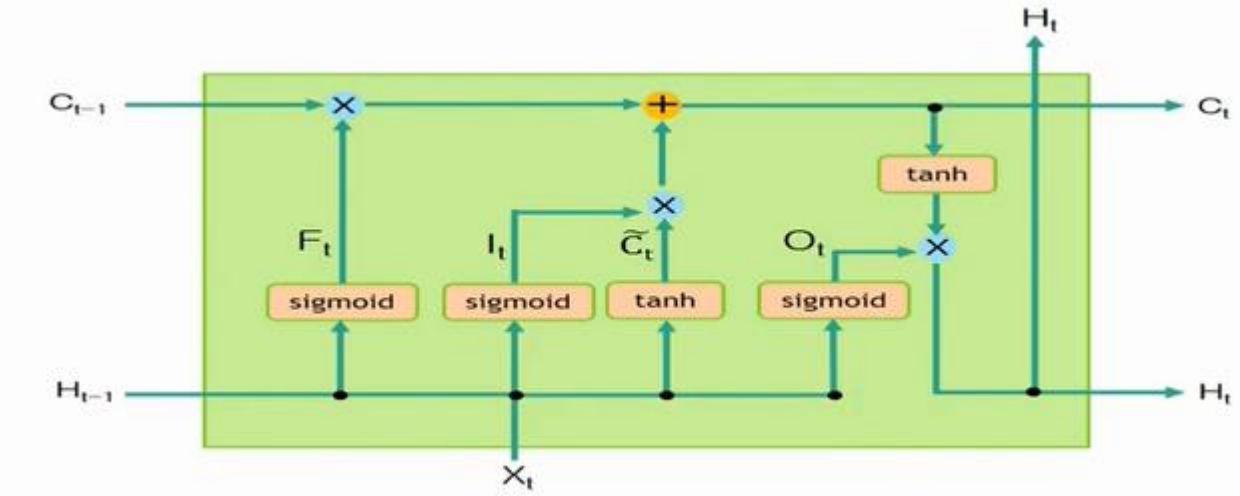
$$H_t = O_t * \tanh(C_t)$$

$$O_t = \text{sigmoid}(U_o X_t + W_o H_{t-1} + b_o)$$

- output gate는 forget gate와 input gate에 의해서 변경된 현재 시점의 memory cell state (C_t) 값을, 얼마나 빼내서 다음 레이어로 전달할지 결정하는 단계
- 이때 현재 시점의 LSTM 출력 값 $H_t = O_t * \tanh(C_t)$ 수식에서의 곱하기 표시는 pointwise operation 을 나타냄

2. LSTM(Long Short-Term Memory)

❖ **LSTM summary**

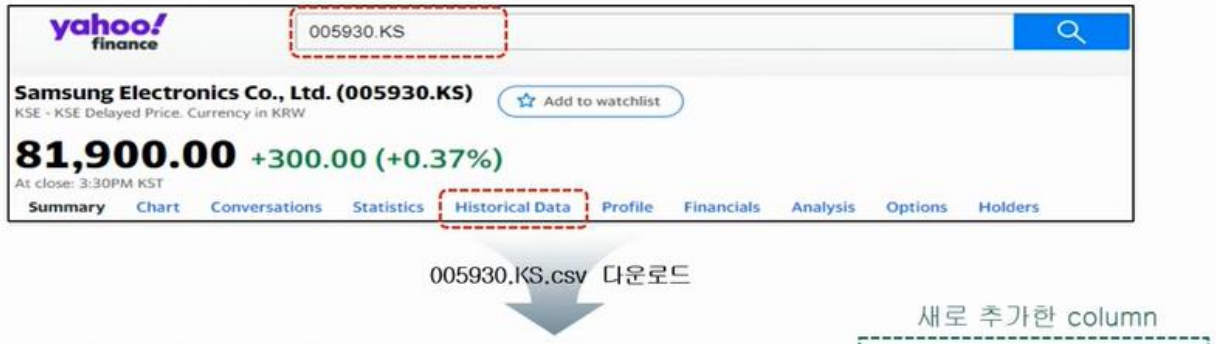


계층		수식	학습 파라미터
LSTM 계층	forget gate	$F_t = \text{sigmoid}(U_f X_t + W_f H_{t-1} + b_f)$	U_f , W_f, b_f
	input gate	$\tilde{C}_t = \tanh(U_c X_t + W_c H_{t-1} + b_c)$ $I_t = \text{sigmoid}(U_{in} X_t + W_{in} H_{t-1} + b_i)$	U_c , W_c, b_c U_{in} , W_{in}, b_i
	cell state	$C_t = F_t C_{t-1} + I_t \tilde{C}_t$	—
	output gate	$H_t = O_t * \tanh(C_t)$ $O_t = \text{sigmoid}(U_o X_t + W_o H_{t-1} + b_o)$	U_o , W_o, b_o
출력 계층		$y_t = activation\_function(V_{out} H_t + b_{out})$	V_{out} , b_{out}

2. LSTM(Long Short-Term Memory)

❖ LSTM 활용한 삼성전자 주가 예측

- 삼성전자 주식가격(2000-01-04 ~ 2021.06.18)
- 삼성전자 주가 데이터 yahoo finance에서 csv 파일로 다운로드 할 수 있음
- Csv 파일에서 7개의 column(Date, Open, High, Low, Close, Adj Close, Volume) 만 존재하나, 예측의 정확도를 높이기 위해 3일 이동평균선(3MA), 5일 이동평균선(5MA) 데이터를 추가함

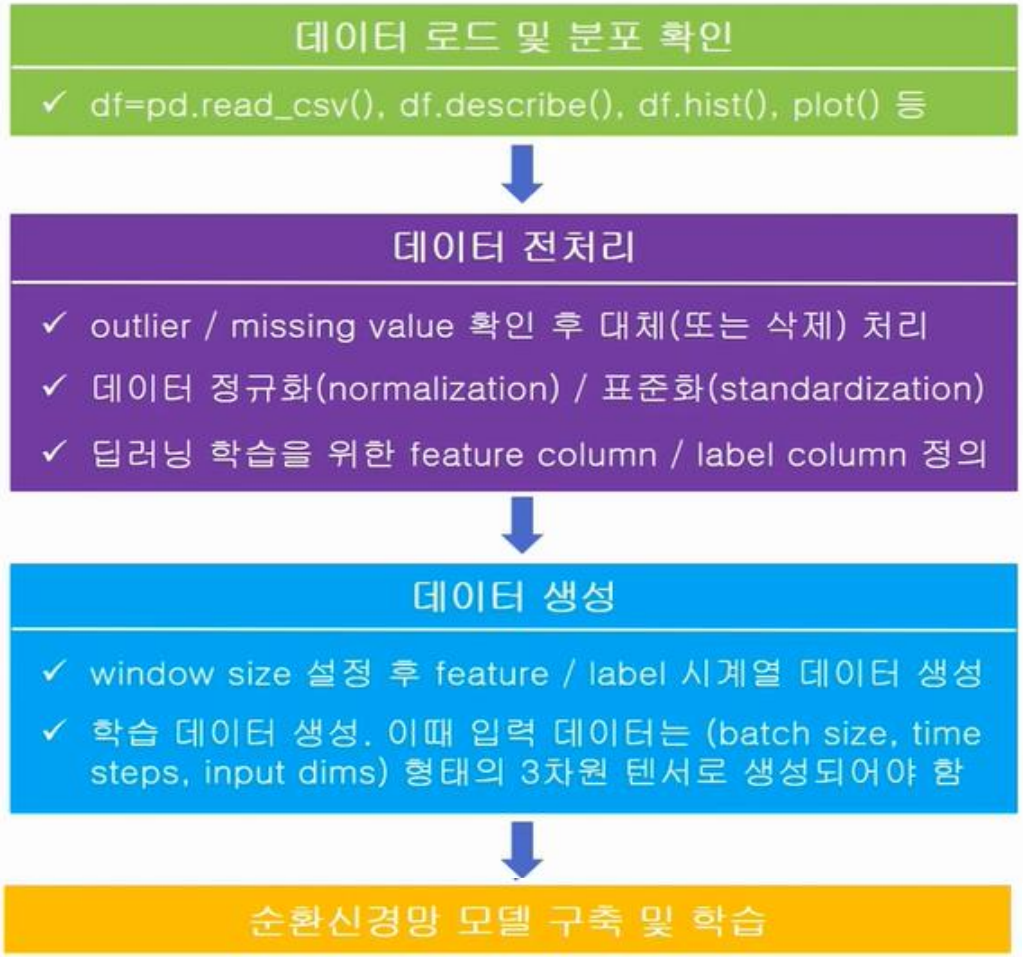


새로 추가한 column

Date	Open	High	Low	Close	Adj Close	Volume	3MA	5MA
2000-01-04	6000.0	6110.0	5660.0	6110.0	4740.119629	74195000.0	NaN	NaN
2000-01-05	5800.0	6060.0	5520.0	5580.0	4328.947754	74680000.0	NaN	NaN
2000-01-06	5750.0	5780.0	5580.0	5620.0	4359.979492	54390000.0	4476.348958	NaN
2000-01-07	5560.0	5670.0	5360.0	5540.0	4297.916992	40305000.0	4328.948079	NaN
2000-01-10	5600.0	5770.0	5580.0	5770.0	4476.349121	46880000.0	4378.081868	4440.662598

2. LSTM(Long Short-Term Memory)

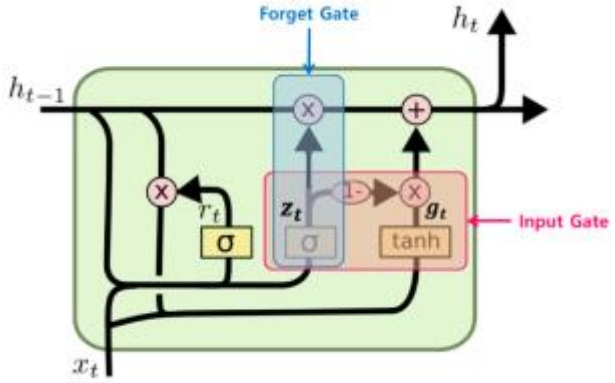
❖ 개발과정 – 시계열 데이터 분석 및 예측

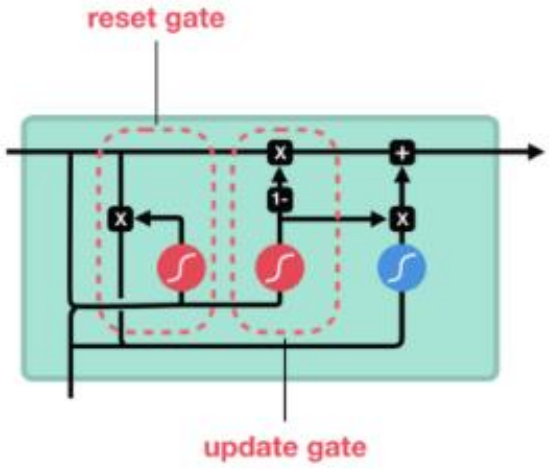


3. GRU(Gate Recurrent Unit)

❖ GRU

- GRU는 LSTM과 비슷한 이유로 만들어짐, LSTM을 구성하는 Time-Step의 Cell을 조금 더 간소화한 버전.
- GRU는 LSTM보다 학습 속도가 빠르다고 알려져있지만, 여러 평가에서 GRU는 LSTM과 비슷한 성능을 보인다고 알려짐
- 데이터 양이 적을 때는, 매개 변수의 양이 적은 GRU가 조금 더 낫고,
- 데이터 양이 더 많으면 LSTM이 더 낫다고 알려져 있음.





3. GRU(Gate Recurrent Unit)

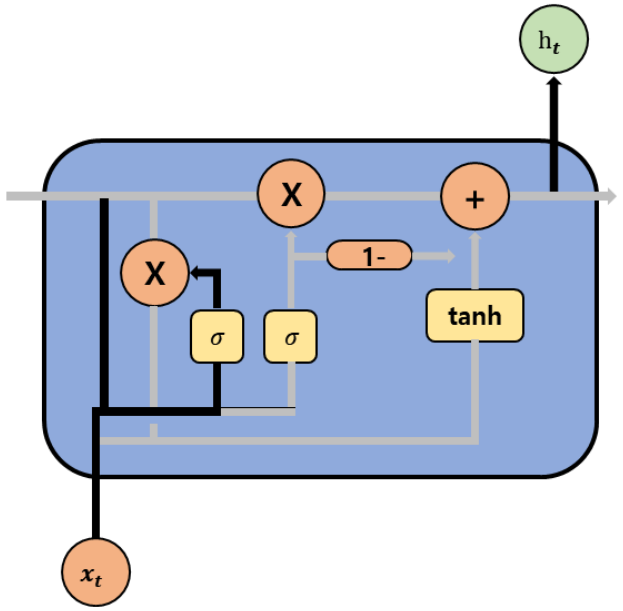
❖ LSTM과 GRU 차이점

- GRU는 LSTM과 다르게 Gate가 2개이며, Reset Gate(r)과 Update Gate(z)로 이루어짐.
  - Reset Gate는 이전 상태를 얼마나 반영할지
  - Update Gate는 이전 상태와 현재 상태를 얼마만큼의 비율로 반영할지
- 또한, LSTM에서의 Cell State와 Hidden State가 Hidden State로 통합됨
- Update Gate가 LSTM에서의 forget gate, input gate를 제어함.
- GRU에는 Output Gate가 없음.

3. GRU(Gate Recurrent Unit)

❖ Reset Gate

- Forget Gate와 Input Gate를 합친 Gate
- 이전의 정보를 얼마나 통과시킬지 결정하는 Gate
- 입력 값과 hidden state 값을 각각 가중치에 곱하고 sigmoid 함수에 통과시켜 Forget Gate로 사용한다.
- 이 값을 1에서 빼 Input Gate로 사용한다.



3. GRU(Gate Recurrent Unit)

❖ Update Gate

- 이전 상태의 hidden state와 현재 상태의 x를 받아 sigmoid 처리
- LSTM의 forget gate, input gate와 비슷한 역할을 하며,
- 이전 정보와 현재 정보를 각각 얼마나 반영할 것인지에 대한 비율을 구하는 것이 핵심이다.
  - 즉, update gate의 계산 한 번으로 LSTM의 forget gate + input gate의 역할을 대신할 수 있다.
- 따라서, 최종 결과는 다음 상태의 hidden state로 보내지게 된다.