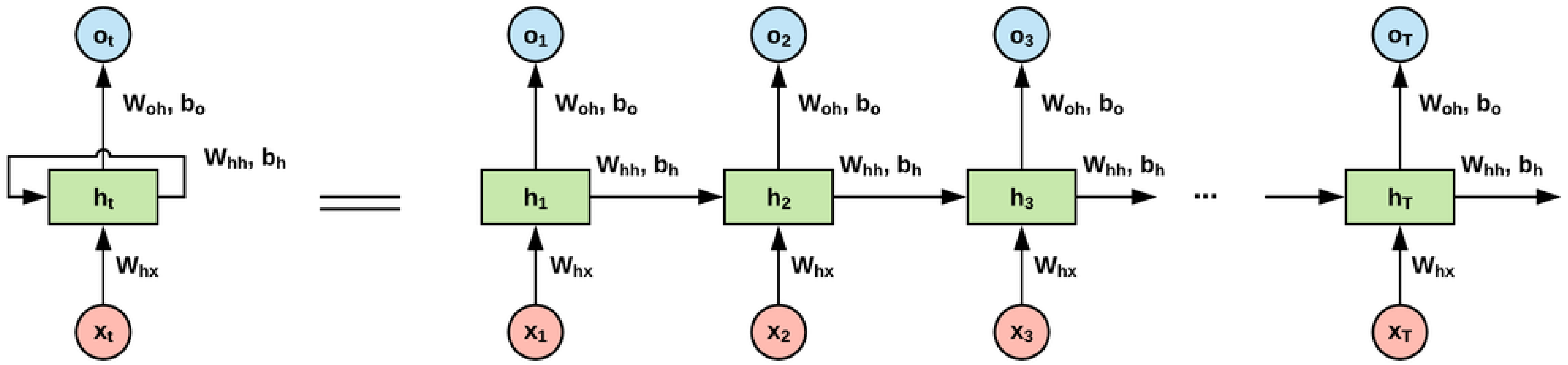
NLP분반 WEEK 5

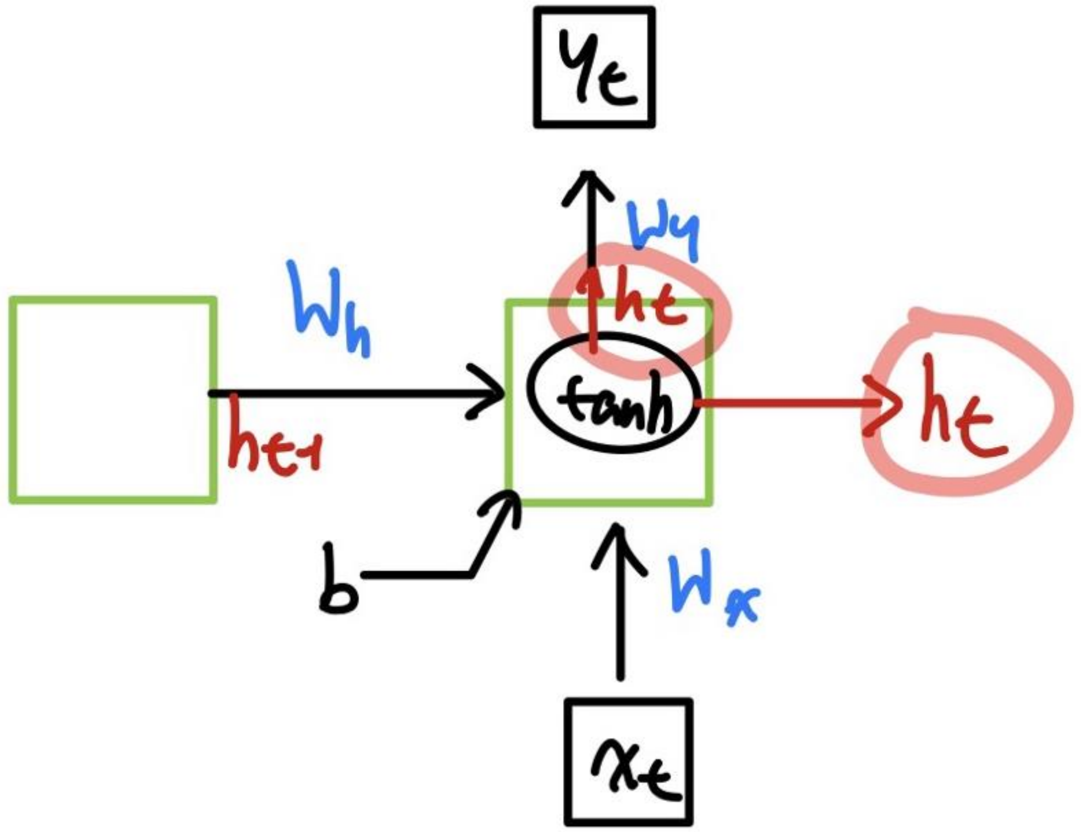
LSTM

Long Short-Term Memory

RNN 한계점

RNN 구조 복습





RNN 한계점 :

장기 의존성 문제

the problem of Long-Term Depdendencies

Give it back
now, Malfoy ……………………………………………………
………………… I was looking for the Troll, I 've read about
them thought I could handle it. But I was
wrong. …………….Give it here now, Malfoy or I'll knock
you off your broom.. ……………………….Hermione! Oh
now what are we going to do? ………………
Bravery. Your parents had it too. You bet _________
heard about this. This is servant stuff!

문장이 길어져도
빈칸에 들어갈 Malfoy가 남자라는 사실을
모델은 계속 기억할 수 있을까?

He / She

RNN 한계점

장기 의존성 문제

the problem of Long-Term Depdendencies

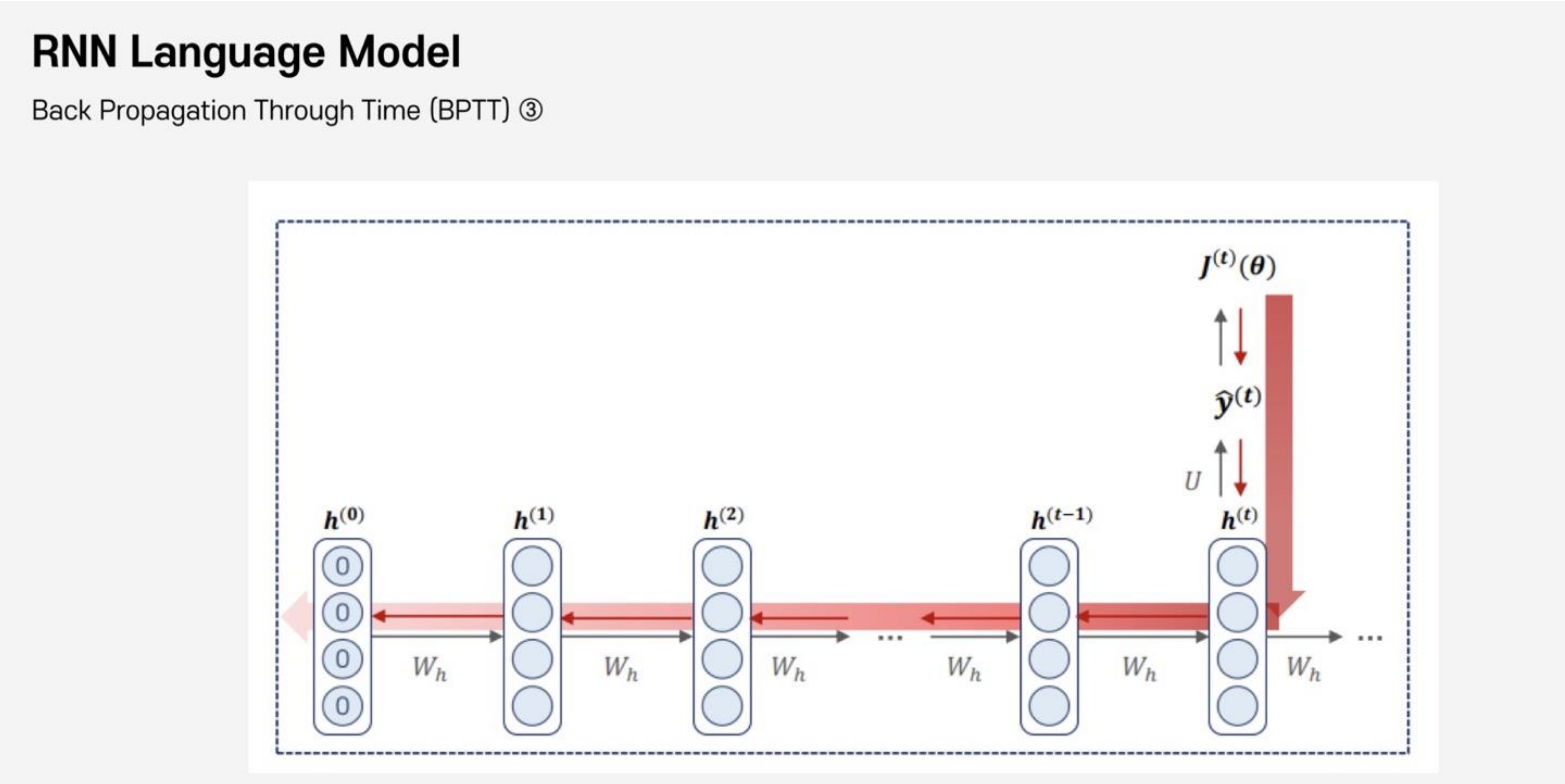
(Wh or Wx)
가중치
matrix

(실제값 - 예측값)

$$W = W - \alpha \cdot \frac{\partial E}{\partial W}$$

학습률
learning rate

$$\frac{\partial J(\theta)}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial W_h}$$



RNN 한계점 :

장기 의존성 문제

the problem of Long-Term Depdendencies

(Wh or Wx)
가중치 matrix

(실제값 - 예측값)

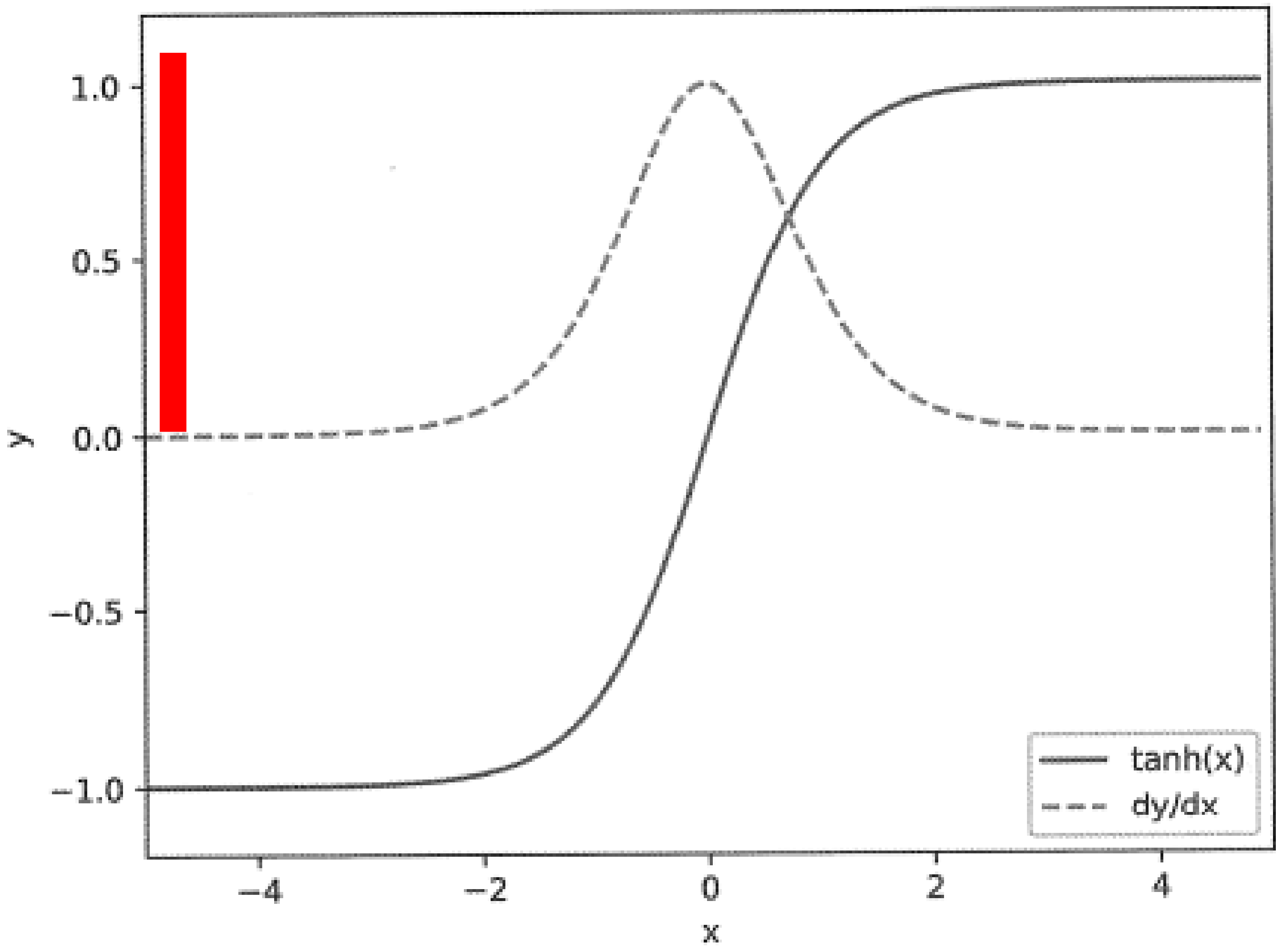
$$W = W - \alpha \cdot \frac{\partial E}{\partial W}$$

학습률
learning rate

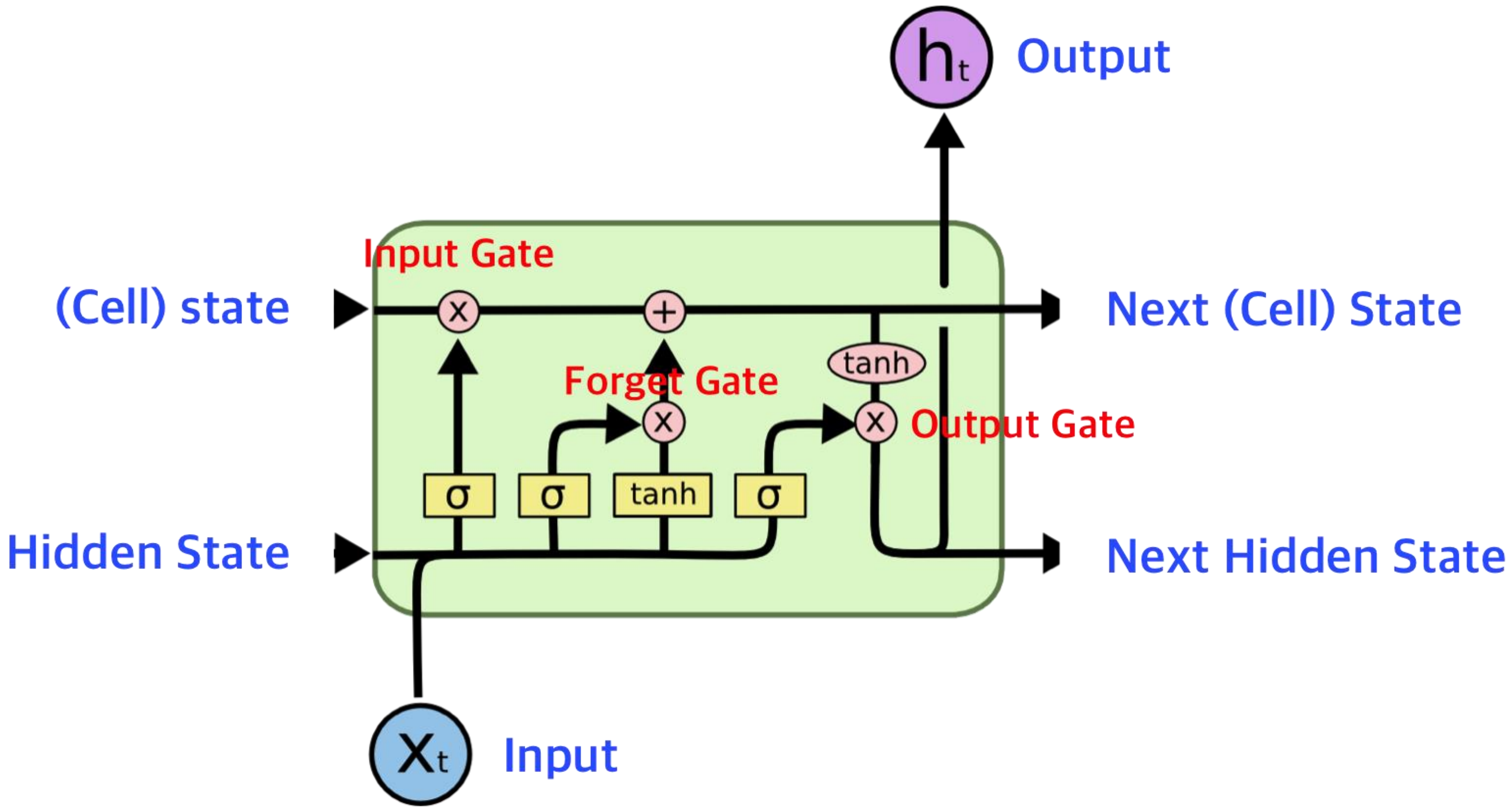
$$\frac{\partial J(\theta)}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial W_h}$$

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

그림 6-6 $y = \tanh(x)$의 그래프(점선은 미분)



LSTM 구조



기억 / 망각의 정도를 학습.
무조건 t-1을 다 반영 X

새로운 메모리 셀 C

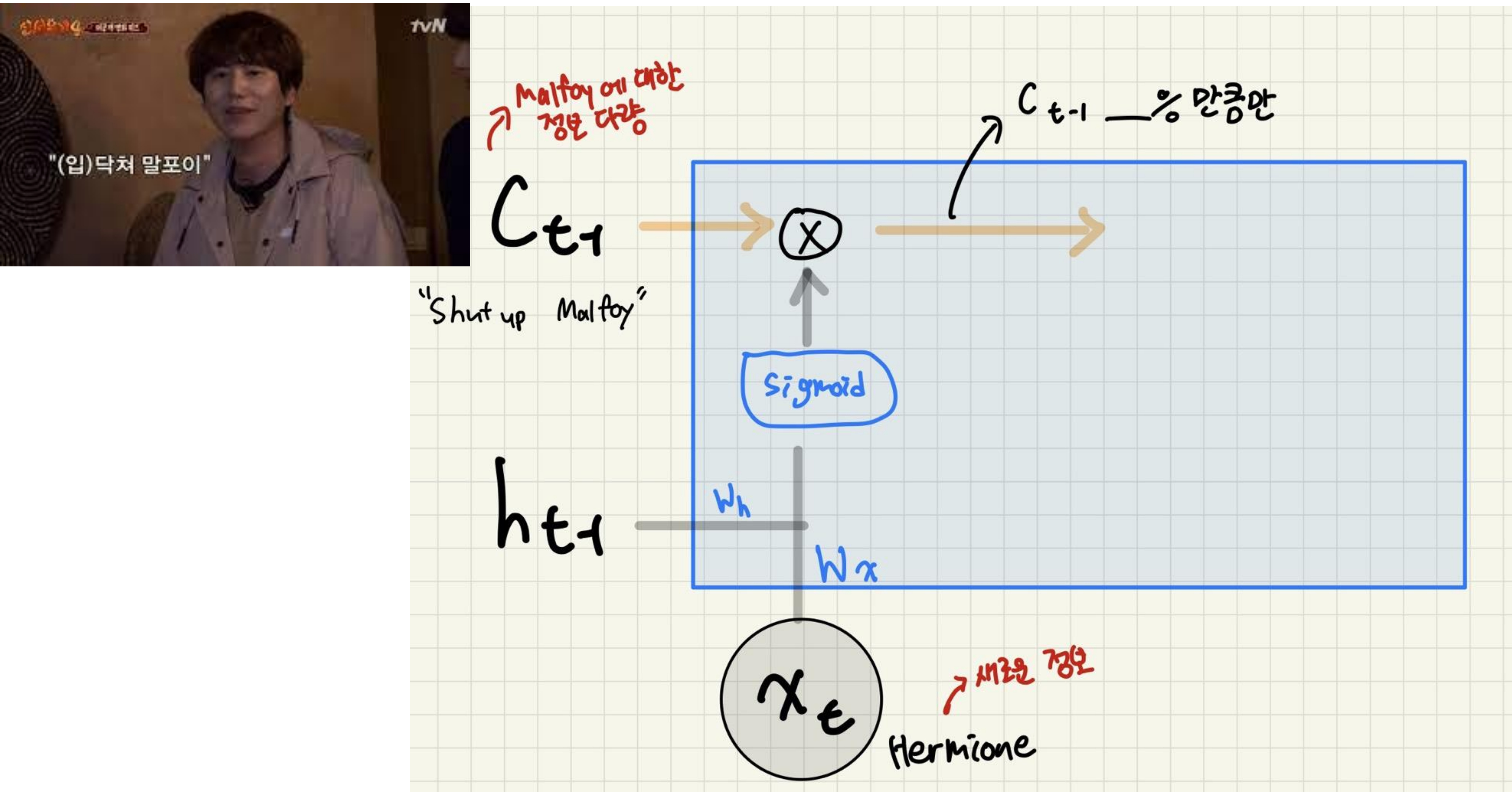
3가지 Gate

정보 섞는 정도를 결정

LSTM 구조

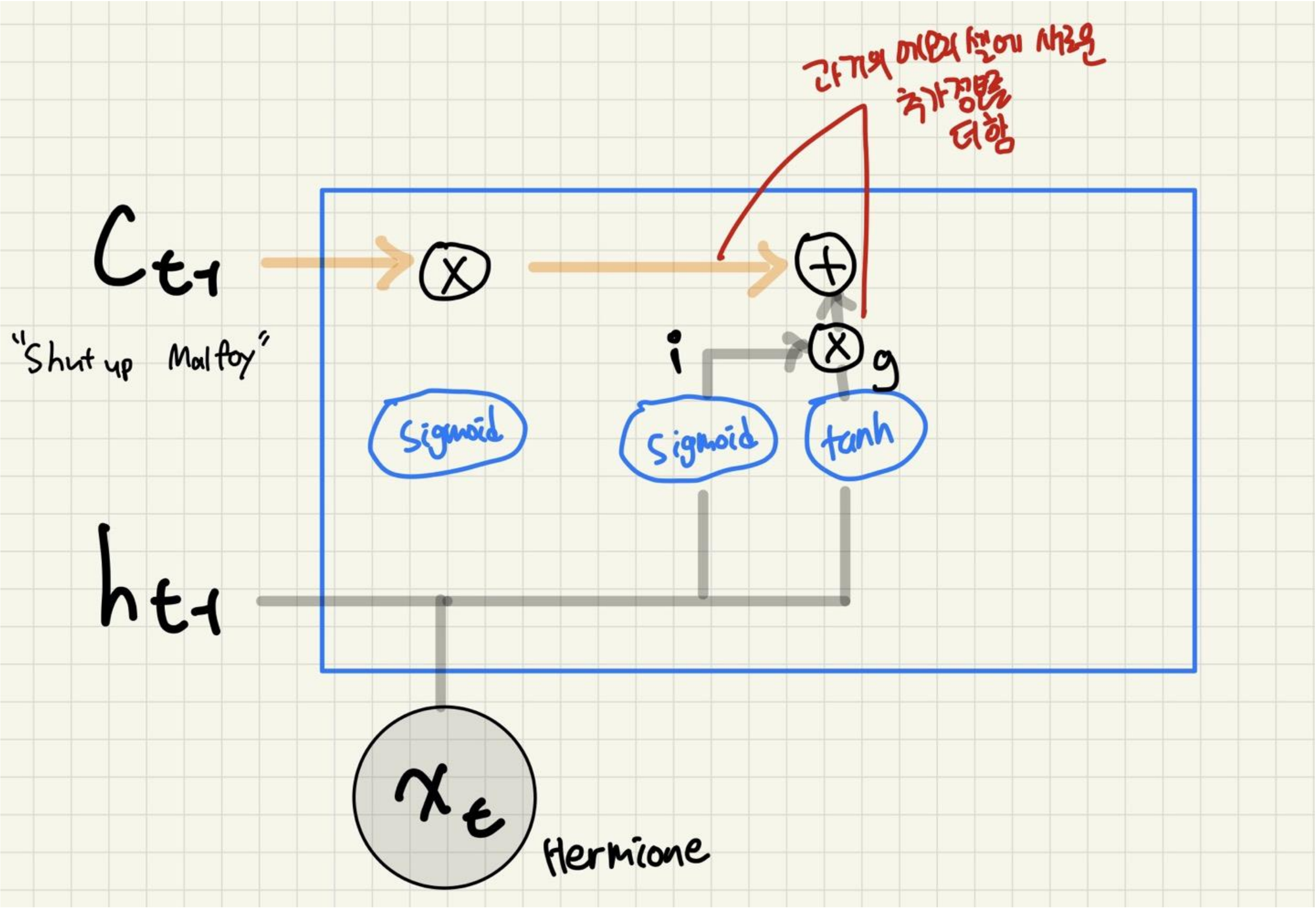
1) 삭제 게이트

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



LSTM 구조

2) 입력 게이트

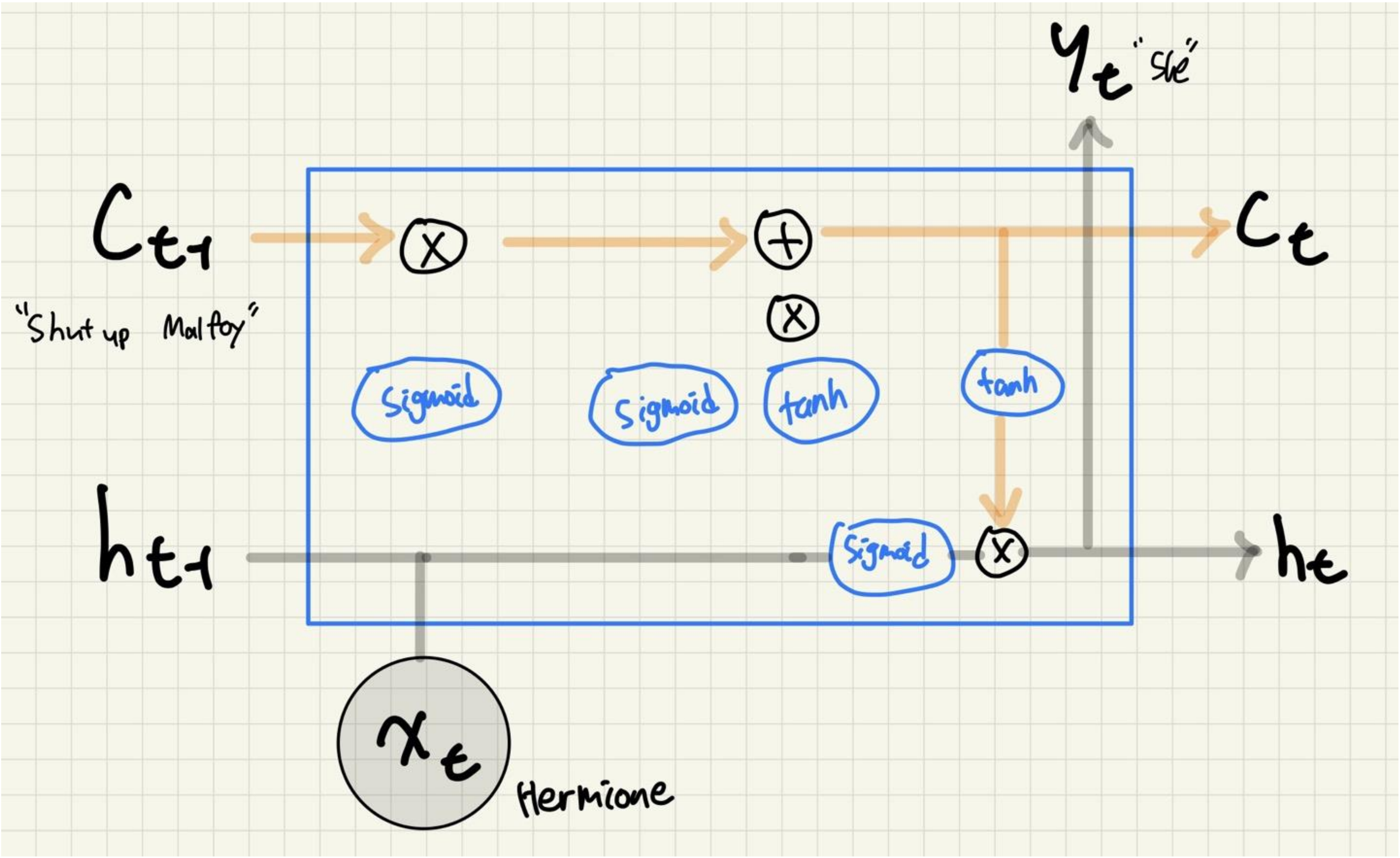


$$\mathbf{i} = \sigma(\mathbf{x}_t \mathbf{W}_x^{(\mathbf{i})} + \mathbf{h}_{t-1}\mathbf{W}_{\mathbf{h}}^{(\mathbf{i})} + \mathbf{b}^{(\mathbf{i})})$$

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

LSTM 구조 :

3) 출력 게이트



$$\mathbf{o} = \sigma(\mathbf{x}_t \mathbf{W}_\mathbf{x}^{(o)} + \mathbf{h}_{t-1} \mathbf{W}_\mathbf{h}^{(o)} + \mathbf{b}^{(o)})$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

ht에서 곱은 행렬 곱이 아닌 원소별 곱 (아마다르 곱)이다.

LSTM 실습 :

[Intro to Recurrent Neural Networks LSTM | GRU | Kaggle]

Date	Open	High	Low	Close	Volume	Name
2006-01-03	47.47	47.85	46.25	47.58	7582127	AMZN
2006-01-04	47.48	47.73	46.69	47.25	7440914	AMZN
2006-01-05	47.16	48.20	47.11	47.65	5417258	AMZN
2006-01-06	47.97	48.58	47.32	47.87	6154285	AMZN
2006-01-09	46.55	47.10	46.40	47.08	8945056	AMZN

```python
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional
from tensorflow.keras.optimizers import SGD
import math
from sklearn.metrics import mean_squared_error
```

```python
def plot_predictions(test,predicted):
    plt.plot(test, color='red',label='Real AMAZON Stock Price')
    plt.plot(predicted, color='blue',label='Predicted AMAZON Stock Price')
    plt.title('AMAZON Stock Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('AMAZON Stock Price')
    plt.legend()
    plt.show()

def return_rmse(test,predicted):
    rmse = math.sqrt(mean_squared_error(test, predicted))
    print("The root mean squared error is {}.".format(rmse))
```

LSTM 실습

[Intro to Recurrent Neural Networks LSTM | GRU | Kaggle]()

```
training_set = dataset[:'2016'].iloc[:,1:2].values
test_set = dataset['2017':].iloc[:,1:2].values

dataset["High"][:'2016'].plot(figsize=(16,4),legend=True)
dataset["High"]['2017':].plot(figsize=(16,4),legend=True)
plt.legend(['Training set (Before 2017)','Test set (2017 and beyond)'])
plt.title('AMAZON stock price')
plt.show()
```

```
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set
```

LSTM 실습

[Intro to Recurrent Neural Networks LSTM | GRU | Kaggle]

```
X_train = []
y_train = []
for i in range(60,2768):
    X_train.append(training_set_scaled[i-60:i,0])
    y_train.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train), np.array(y_train)

X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))

print(X_train.shape)
# (2708, 60, 1)
```

LSTM 실습 :

[Intro to Recurrent Neural Networks LSTM | GRU | Kaggle]

```python
# The LSTM architecture
regressor = Sequential()
# First LSTM layer with Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))
# Second LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Third LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Fourth LSTM layer
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
# The output layer
regressor.add(Dense(units=1))

# Compiling the RNN
regressor.compile(optimizer='rmsprop',loss='mean_squared_error')
# Fitting to the training set
regressor.fit(X_train,y_train,epochs=50,batch_size=32)

regressor.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 dropout (Dropout)           (None, 60, 50)            0

 lstm_1 (LSTM)               (None, 60, 50)            20200

 dropout_1 (Dropout)         (None, 60, 50)            0

 lstm_2 (LSTM)               (None, 60, 50)            20200

 dropout_2 (Dropout)         (None, 60, 50)            0

 lstm_3 (LSTM)               (None, 50)                20200

 dropout_3 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

LSTM 실습 :

[Intro to Recurrent Neural Networks LSTM | GRU | Kaggle]()

```
dataset_total = pd.concat((dataset["High"][:'2016'],dataset["High"]['2017':]),axis=0)
inputs = dataset_total[len(dataset_total)-len(test_set) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = sc.transform(inputs)
```

```
X_test = []
for i in range(60,311):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
return_rmse(test_set,predicted_stock_price)
# The root mean squared error is 146.4590971388303.
```

```
plot_predictions(test_set,predicted_stock_price)
```

얼레?