

# 사물인터넷 (Internet of Things)

김태운

## 목차

- Sensing Layer: 센서 및 센싱/분석기술
  - 데이터 정제

## Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - 회귀/Regression
    - 단순 선형 회귀
    - D차원 선형 회귀
    - 선형 기저 함수 모델

# Sensing Layer: 데이터 정제

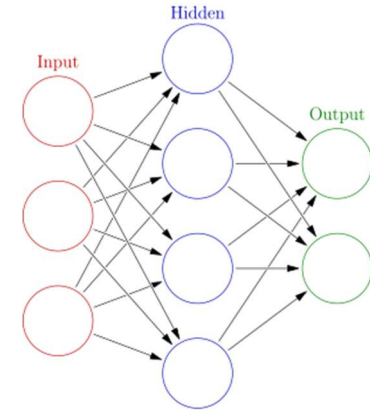
- 추가 토픽: 센싱 값 분석/예측
  - 딥러닝을 활용한 시계열 데이터 예측
  - 시계열 데이터(Time series data)
    - 일정 시간 간격으로 순차적으로 획득한/관측한 값의 집합
    - 획득/관측한 데이터가 시간적 순서를 가지며, 연속한 관측치는 서로 상관관계를 가지고 있음
    - 예) 1시간 간격으로 측정한 서울 기온
    - 예) 24시간 간격으로 측정한 한강 수위
    - 예) 10초 간격으로 측정한 CPU 사용률
    - 예) 최근 10년간 매일 기록한 코스피 지수(종가 기준)
  - 참고. 순차 데이터 (Sequential data)
    - 샘플에 순서가 있는 데이터
    - 예) 텍스트/문장은 순서에 맞게 배열된 단어로 구성됨

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

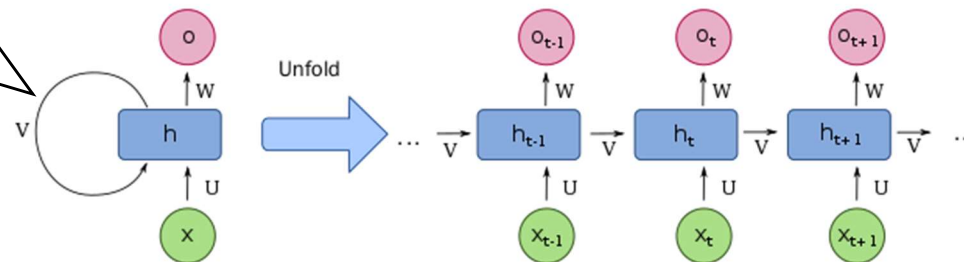
### • 순환 신경망(RNN, Recurrent Neural Network)

- 일반적인 인공 신경망은, 학습에 사용하는 각 샘플이 독립적이라고 가정하며, **Input Layer** → **Hidden Layer(s)** → **Output Layer**의 단방향으로 정보가 전달됨.
- 따라서, 전/후 데이터간 상관관계를 고려하지 못함
  - $N$ 번째 데이터와  $N - k$  또는  $N + t$  번째 데이터는 독립적으로 처리됨 ( $k, t > 0$ )
  - 일반적으로 딥러닝에서의 학습데이터는 IID 분포를 따른다고 가정



- 순환 신경망의 경우 은닉 계층의 출력 값이 다음 계층으로 넘어갈 뿐 아니라, 자기 자신에게 다시 되돌아옴
- 이를 통해, 시계열 데이터처럼 앞뒤 신호에 서로 상관도가 있는 경우, 순환 신경망은 인공 신경망의 정확도 및 성능을 더 높일 수 있음
- RNN은 자연어 처리, 번역, 로봇 제어, 시계열 예측, 음성 인식 등 연속성을 가지는 데이터 또는 작업에 주로 활용

은닉층의 출력 값이 다시 은닉층의 입력으로 활용됨  
→ 순환 구조를 통해 직전에 처리한 정보를 “기억”  
하고, 다음 데이터 처리에 활용할 수 있음



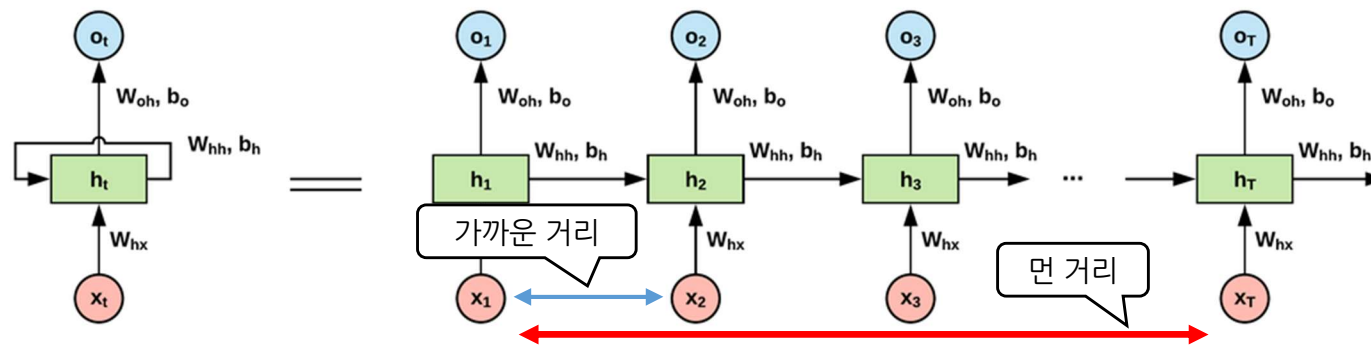
RNN의 구조

RNN을 시간의 흐름에 따라 펼친 결과

# Sensing Layer: 데이터 정제

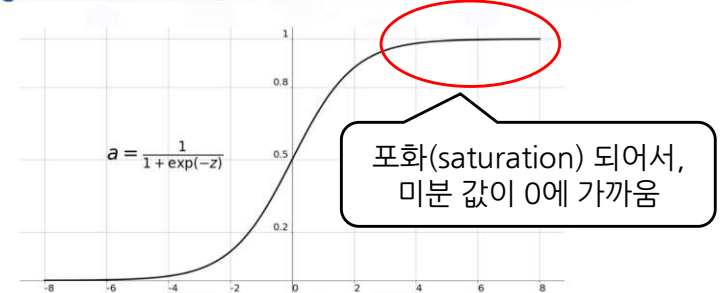
## ■ 추가 토픽: 센싱 값 분석/예측

- 순환 신경망(RNN, Recurrent Neural Network)
  - 일반적인 RNN의 경우 짧은 시퀀스를 처리할 경우 유리
    - 멀리 떨어진 입력 값사이의 관계를 파악하기 어려움(학습 능력이 저하됨)



- 활성화 함수는 입력 신호가 커지면 포화(saturation)되는 특성이 있기 때문에, 입력 값이 조금만 커져도 미분 값이 매우 작아지게 됨
- 미분 값이 작아지면 backpropagation 과정에서 미분 값을 곱한 결과를 사용하므로 인공신경망을 구성하는 가중치가 거의 업데이트 되지 않고, 학습이 거의 이루어지지 않음
- 이를 그래디언트 소실(gradient vanishing) 문제라고 하며, 이로 인해 RNN은 단기 기억(짧은 시퀀스)에는 적합하나 장기 기억 성능이 떨어짐

## Sigmoid Function

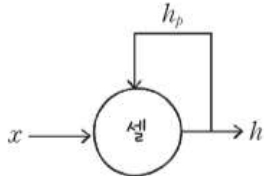
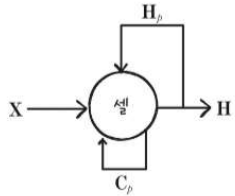
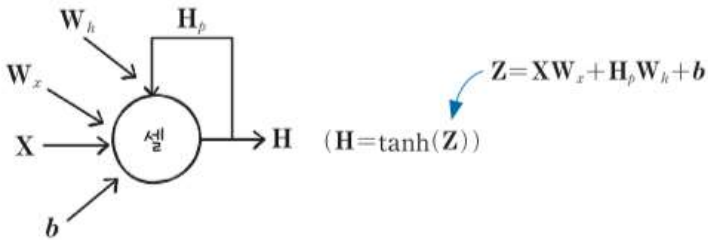
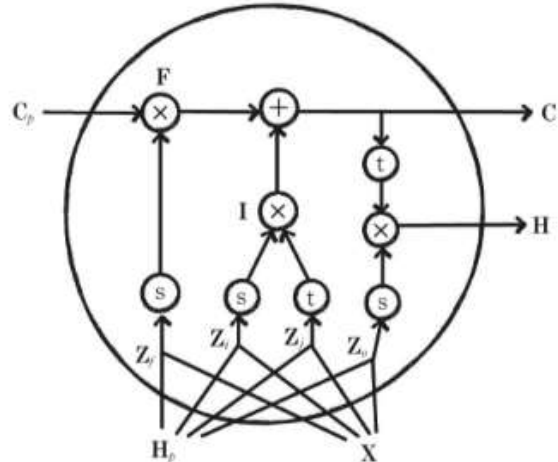


# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

### • LSTM(Long Short-Term Memory) 순환 신경망

- 1997년 호크라이터와 슈미트후버가 고안한 것으로, 그레디언트 소실 문제를 극복하여 긴 시퀀스를 모델링 할 수 있음
- RNN과 LSTM의 셀 구조 비교(참고: 순환신경망에서는 뉴런을 셀/cell 또는 메모리 셀이라고 부름)

RNN 순환 신경망의 셀 구조	LSTM 순환 신경망의 셀 구조
<ul style="list-style-type: none"> <li>• 셀의 입력: <math>x</math>, 셀의 출력: <math>h</math> (은닉 상태, hidden state)</li> <li>• 현재의 은닉 상태와 구분하기 위해, 직전의 은닉 상태를 <math>h_p</math>로 표현</li> </ul> 	<ul style="list-style-type: none"> <li>• RNN과 달리 2개의 출력이 순환되고, 그 중 하나만 다음 층으로 전달됨</li> <li>• 이전 셀로부터 전달받은 정보를 hidden state 또는 short-term memory라고 함</li> <li>• 셀로 순환만 되는 출력을 셀 상태(<math>C</math>, cell state)라고 하며, 또는 long-term memory 라고도 함</li> </ul> 
<ul style="list-style-type: none"> <li>• 활성화 함수로는 하이퍼볼릭탄젠트(tanh) 함수를 주로 사용</li> </ul> 	<ul style="list-style-type: none"> <li>• (s): sigmoid 활성화 함수, (t): tanh 활성화 함수</li> <li>• (x): 원소 별 곱셈, (+): 원소 별 덧셈</li> <li>• <math>H_p, C_p</math>: 이전 상태에서 만들어진 <math>H, C</math></li> </ul> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <math display="block">\begin{aligned} F &amp;= C_p \times \text{sigmoid}(Z_f) \\ I &amp;= \text{sigmoid}(Z_i) \times \tanh(Z_j) \\ C &amp;= F + I \\ H &amp;= \tanh(C) \times \text{sigmoid}(Z_o) \end{aligned}</math> </div> 



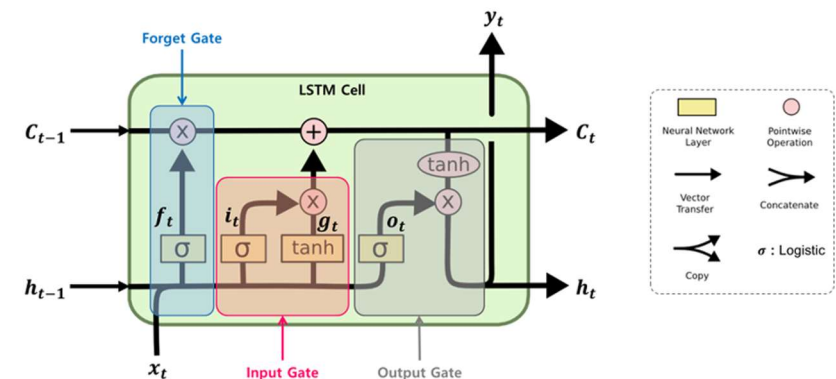
# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

### • LSTM(Long Short-Term Memory) 순환 신경망

- RNN에 기울기 정보 크기를 조절하기 위한 gate를 추가한 모델이며, 다음과 같이 3가지의 gate 종류가 있음

- 망각 게이트 forget gate ( $f_t$ ): 과거 정보를 얼마나 유지할 것인지를 결정
- Input gate ( $i_t$ ): 새로 입력된 정보를 얼마나 유지/활용할 것인지를 결정
- output gate ( $o_t$ ): 입력 정보( $h, x$ )를 계산하여 나온 출력 정보를 다음 셀로 얼마나 넘겨줄 것인지를 결정
- Logistic으로 표현된 sigmoid 함수의 출력은 0~1이고, 0에 가까울 수록 정보를 버리고, 1에 가까울 수록 정보를 보존하는 역할 (또는, 정보 전달/차단 양을 결정하는 게이트/필터 역할을 수행, gating mechanism)

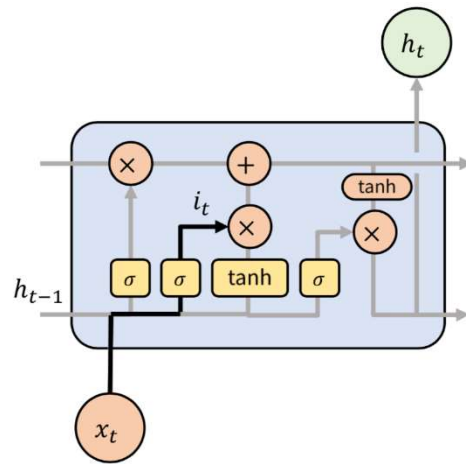


- RNN과 동일하게 은닉 상태(hidden state,  $h$ )를 가지며, 이에 더해 활성화 함수를 직접 거치지 않는 상태인 cell-state가 추가되었고, cell-state는 출력(output,  $y$ )으로 전달되지 않고 셀에서 셀로 순환만 되는 정보임
- 셀 상태 / cell-state ( $C$ )
  - 이전 시점의 셀에서 보존할 정보를 선택하고, 이를 현재 시점의 새로운 셀 정보와 입력 게이트 간의 원소 별 곱(element-wise product)을 통해 선택된 정보와 더하여 산출됨
  - 역전파 과정에서 활성화 함수를 거치지 않아 정보 손실이 없음 // RNN에 비해서 장기 기억 성능이 좋은 이유
  - 즉, 현재 입력된 정보를 얼마나 활용할지에 대한 비중을 결정하면서 동시에 예전 입력된 정보를 잃지 않음

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망: 3종 게이트 상세보기
- 1) Input gate / 입력 게이트



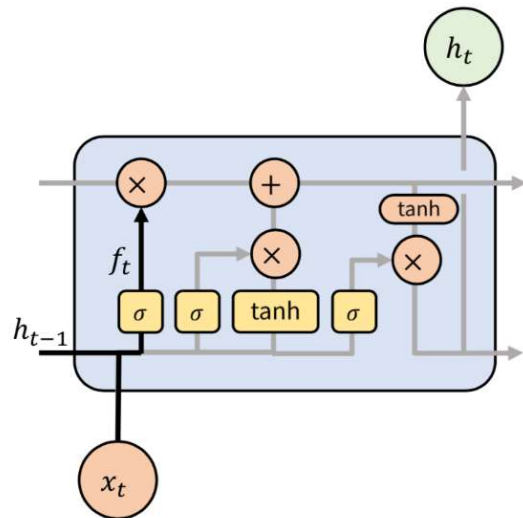
$$\begin{aligned}f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\h_t &= o_t \odot \tanh(c_t)\end{aligned}$$

- 이번 time-step에 입력된 정보 중에서 어떤 정보를 얼마나 보존/무시할지를 결정
- 0~1 범위의 값을 가지는 sigmoid activation function( $\sigma$ )을 사용하여 정보의 보존(1에 가까움) 또는 무시(0에 가까움)를 결정함

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망: 3종 게이트 상세보기
- 2) Forget gate / 망각 게이트



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

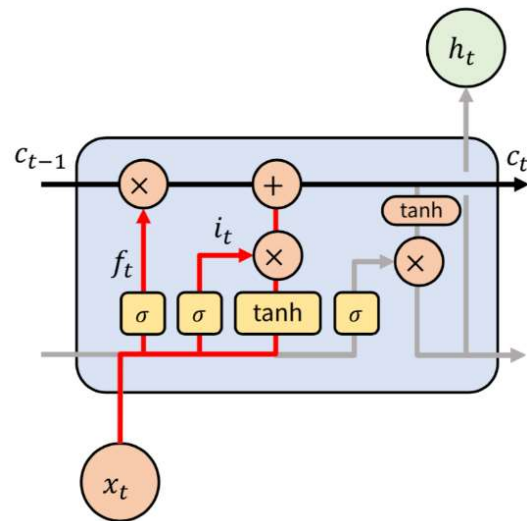
$$h_t = o_t \odot \tanh(c_t)$$

- Long-term memory(장기 기억)에서 어떤 정보를 얼마나 보존/무시할지를 결정
- 0~1 범위의 값을 가지는 sigmoid activation function( $\sigma$ )을 사용하여 정보의 보존(1에 가까움) 또는 무시(0에 가까움)를 결정함

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망: 3종 게이트 상세보기
- 2. 망각 게이트... 계속) Cell state (long-term memory) 계산



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

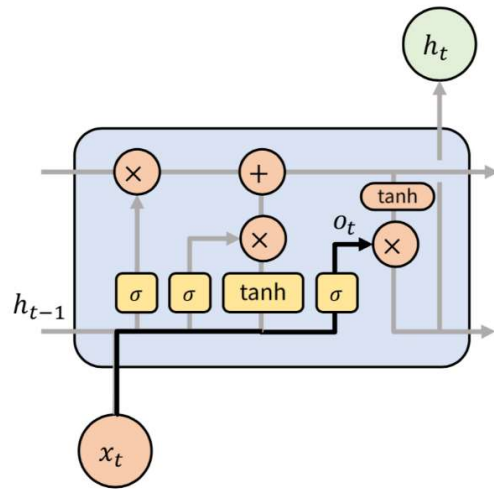
$$h_t = o_t \odot \tanh(c_t)$$

- Forget gate를 통과한 Long-term memory(장기 기억) 결과와, input gate를 통과한 입력 값 일부를 더한 결과를 cell state로 사용
- 곱셈이 아닌 덧셈 연산을 통해서 장기 기억이 가능
- $\odot$  기호는 원소 별 곱(element-wise multiplication)을 의미함

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망: 3종 게이트 상세보기
- 3) output gate / 출력 게이트



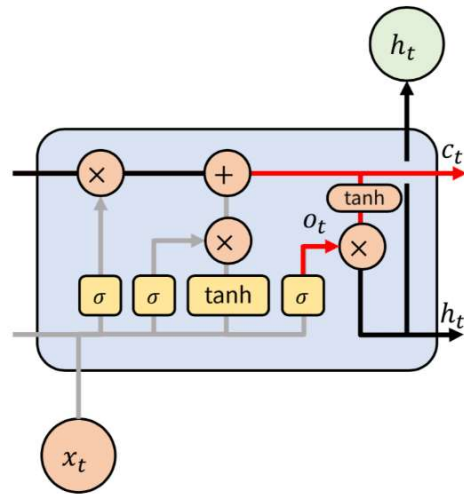
$$\begin{aligned}f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\h_t &= o_t \odot \tanh(c_t)\end{aligned}$$

- 출력 값을 결정하기 위한 게이트(어떤 정보를 얼마나 보존/무시 할지를 결정)
- 0~1 범위의 값을 가지는 sigmoid activation function( $\sigma$ )을 사용하여 정보의 보존(1에 가까움) 또는 무시(0에 가까움)를 결정함

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망: 3종 게이트 상세보기
- 3. 출력 게이트... 계속) hidden state 계산



$$\begin{aligned}f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\h_t &= o_t \odot \tanh(c_t)\end{aligned}$$

- 다음 time-step으로 전달할 hidden state를 계산
- 계산된 hidden state는 현재 셀의 출력 값(output)으로도 사용됨
- $\odot$  기호는 원소 별 곱(element-wise multiplication)을 의미함

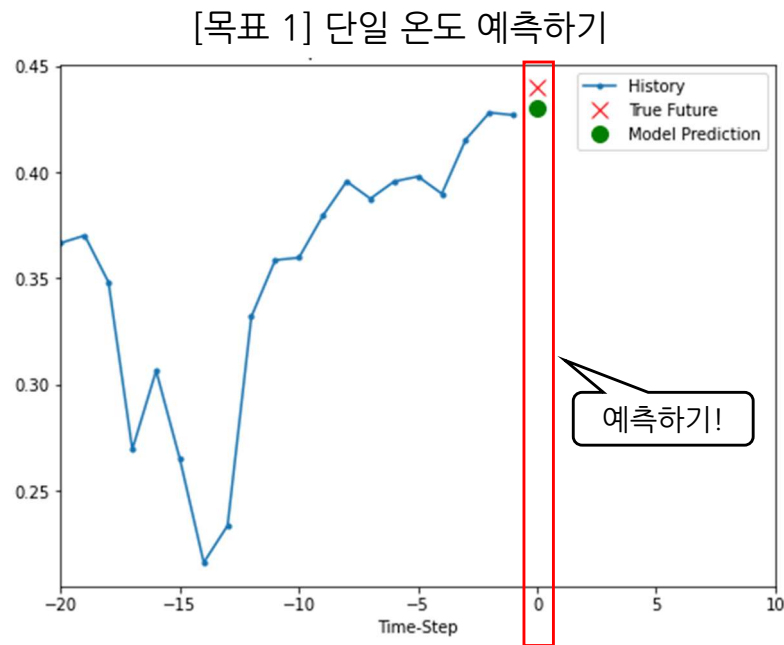
# Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제
    - 데이터 셋: Jena Climate Dataset
      - 독일의 Jena 라는 지역에서 측정된 시계열 날씨 데이터
      - <https://www.kaggle.com/datasets/mnassrib/jena-climate>
      - 측정시간을 포함하여 총 15개의 열로 구성(대기 온도, 대기압, 습도 등...)
      - 2009년부터 2016년까지 측정한 데이터(측정 간격: 10분)

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.80	0.72	1.50	136.1
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.20	1.88	3.02	1310.24	0.19	0.63	171.6
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.50	198.0
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309.00	0.32	0.63	214.3

# Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제
    - (가까운) 미래의 단일 온도 예측하기 // single value prediction



Goal:  
과거 일정 기간 동안의 온도 변화를  
분석하여 가까운 미래의 단일 온도  
예측하기

- 참고: TensorFlow 학습 - 시계열 예측 ([https://www.tensorflow.org/tutorials/structured\\_data/time\\_series](https://www.tensorflow.org/tutorials/structured_data/time_series))



## Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction

```
# 라이브러리 import
import os
import datetime

import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

# 날씨 데이터 셋 다운받기
zip_path = tf.keras.utils.get_file(
    origin='https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip',
    fname='jena_climate_2009_2016.csv.zip',
    extract=True)
csv_path, _ = os.path.splitext(zip_path)
```

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

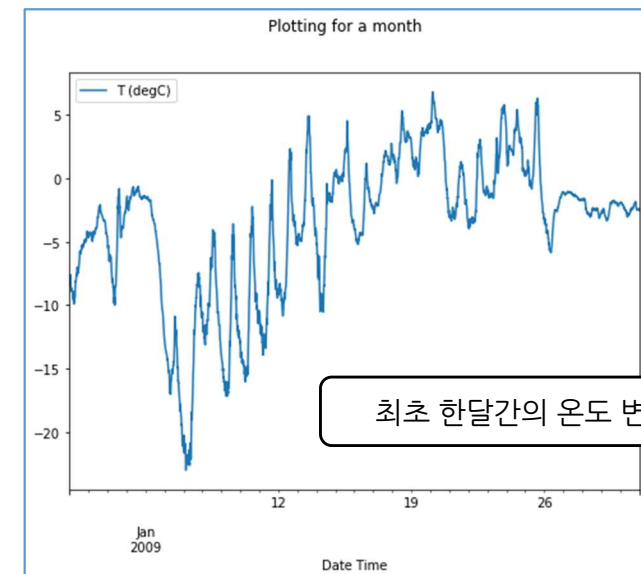
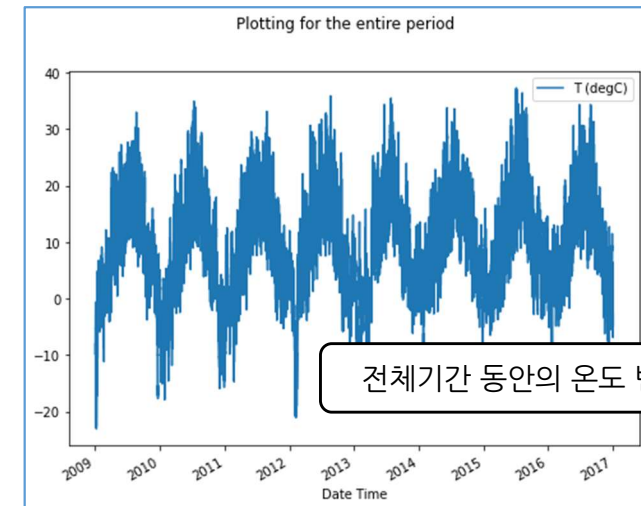
- LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction

```
# 온도 데이터를 plot 해서 그래프로 그리기
date_time = pd.to_datetime(df['Date Time'], format='%d.%m.%Y %H:%M:%S')

# 특정 column만 추출하여 plot 하기
plot_cols = ['T (degC)']
plot_features = df[plot_cols]

# 전체 기간에 대하여 plot 하기
plot_features.index = date_time
_ = plot_features.plot(subplots=True, title='Plotting for the entire period')

# 최초 30일간의 데이터 plot 하기
# - 10분 단위로 측정된 데이터니까...
# - 1시간(6개) * 24시간 * 30일
IND_END = 6*24*30
plot_features = df[plot_cols][:IND_END]
plot_features.index = date_time[:IND_END]
_ = plot_features.plot(subplots=True, title='Plotting for a month')
```



## Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction

```
# 데이터 정규화
col_names = df_train.columns.values

col_names = list(col_names)
col_names.remove('Date Time') # 시간을 기록한 열은 정규화 제외
#print(col_names)

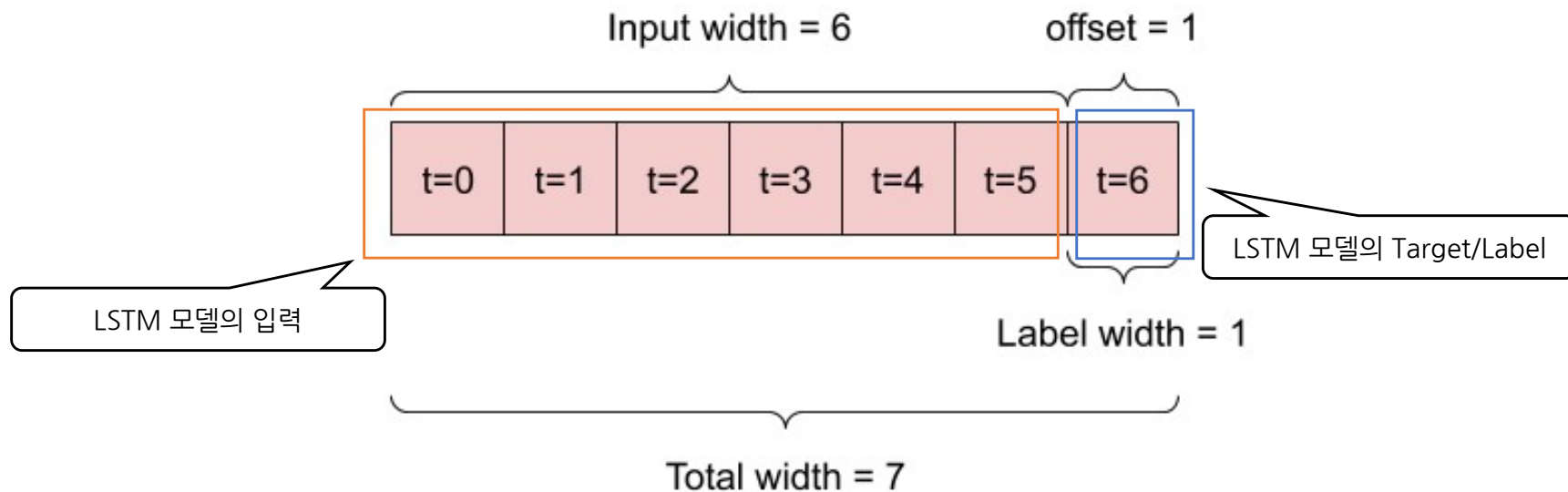
for col in col_names:
    col_mean = df_train[col].mean()
    col_std = df_train[col].std()

    df_train[col] = (df_train[col] - col_mean) / col_std
    df_test[col] = (df_test[col] - col_mean) / col_std
```

## Sensing Layer: 데이터 정제

### ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction
- 데이터 윈도우 생성 함수 정의하기
  - 시계열 데이터에서, N개의 연속된 데이터를 하나로 묶어서(데이터 윈도우) LSTM 모델의 입력으로 전달하고, N개의 데이터 이후의 데이터를 Target 또는 Label로 지정
  - N개의 데이터 윈도우를 입력으로, Target/Label을 예측하는 LSTM 예측 모델 개발



## Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction

```
"""
시계열 데이터를 다루는 LSTM에서는 데이터 윈도우 라는 개념을 사용함
○ 예: N개의 데이터를 입력으로 받고, 분석한 후, N+1번째 데이터를 예측
○ 예: N개의 데이터를 입력으로 받고, 분석한 후, [N+1:N+k] 데이터 시퀀스를 예측
이때, N개의 데이터를 데이터 윈도우 라고 하며, 주어진 전체 데이터 셋을 데이터 윈도우
크기로 분할하기 위해서 아래의 함수를 사용함
- history_size: 학습에 사용할 과거 데이터 수 (데이터 윈도우 크기)
- target_size: 예측해야 하는 값(얼마나 멀리 떨어진 값을 예측할지를 결정)
"""
def univariate_window_generator(dataset, start_index, end_index, history_size, target_size):
    data, labels = [], []

    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i-history_size, i)
        data.append(np.reshape(dataset[indices], (history_size, 1)))
        labels.append(dataset[i+target_size])
    return np.array(data), np.array(labels)
```

## Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction
  - LSTM 모델을 생성하고 학습

```
simple_lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(8, input_shape=x_train_uni.shape[-2:]),
    tf.keras.layers.Dense(1)
])

simple_lstm_model.compile(optimizer='adam', loss='mae')

EVAL_INTERVAL = 200
EPOCHS = 10

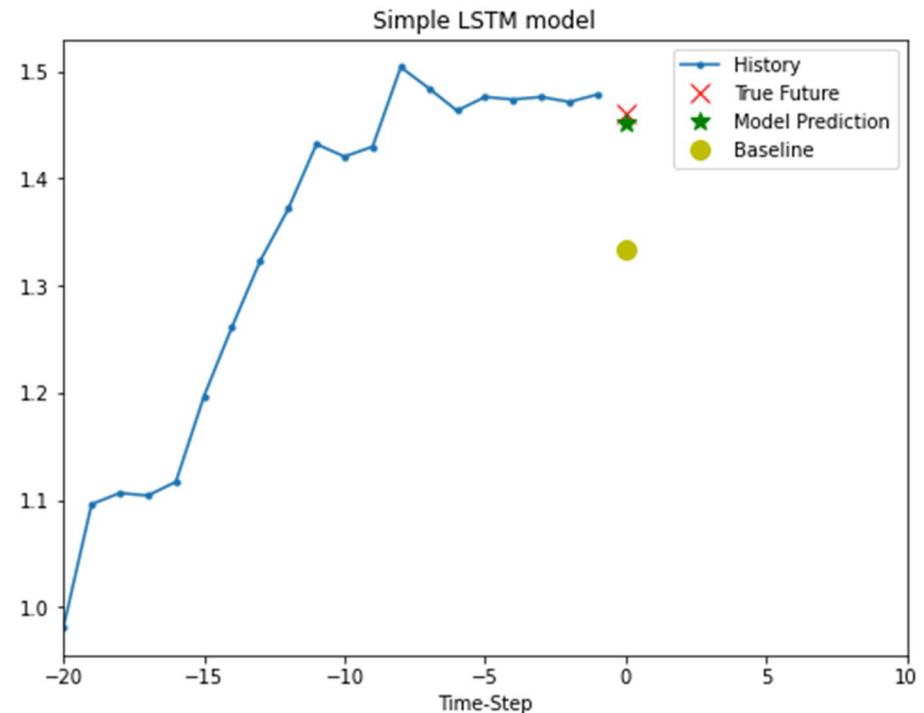
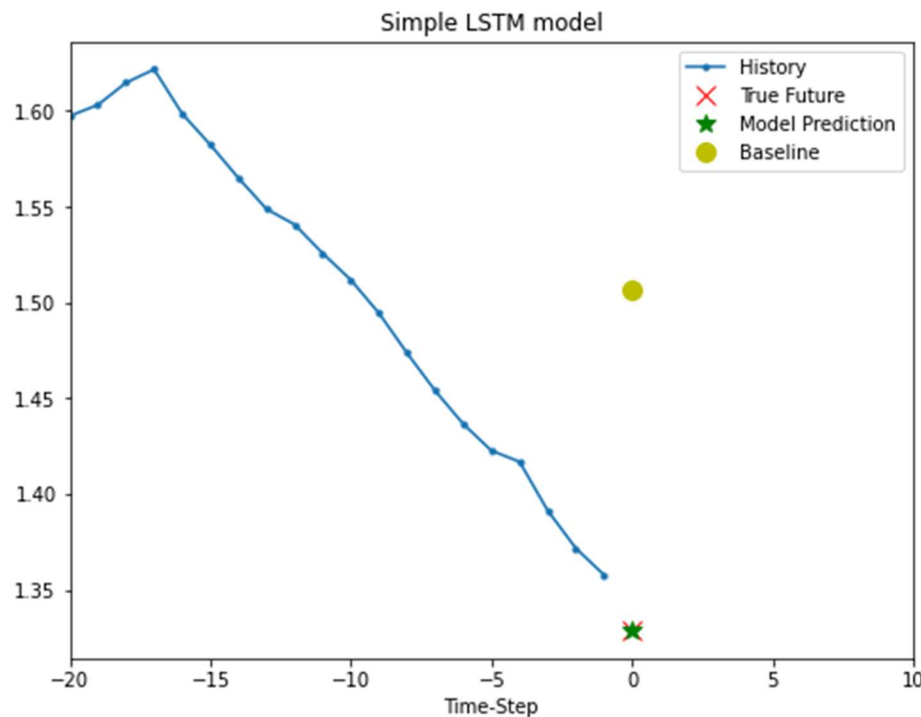
training_history_uni = simple_lstm_model.fit(train_univariate, epochs = EPOCHS,
                                             steps_per_epoch = EVAL_INTERVAL,
                                             validation_data = test_univariate,
                                             validation_steps = 50)
```

# Sensing Layer: 데이터 정제

## ■ 추가 토픽: 센싱 값 분석/예측

- LSTM(Long Short-Term Memory) 순환 신경망 예제: Single value prediction

- LSTM 모델의 예측 결과 확인하기



- 참고: 'Baseline'은 성능 비교를 위해 구현한 알고리즘으로, 최근 20개 온도의 평균을 온도 예측 값으로 사용하는 간단한 모델임

# Sensing Layer: 데이터 정제

- 추가 토픽: 센싱 값 분석/예측
  - LSTM(Long Short-Term Memory) 순환 신경망 예제
    - LSTM 모델을 확장하여 가까운 미래의 일정 기간 동안의 온도 변화 예측도 가능 // sequence prediction

일정 기간 동안의 온도 변화 예측하기

