LSTM을 이용한 주가 예측

IT융합공학부 윤세영

유투브 주소: https://youtu.be/6ZXMOL1PuTk

시계열 데이터란?

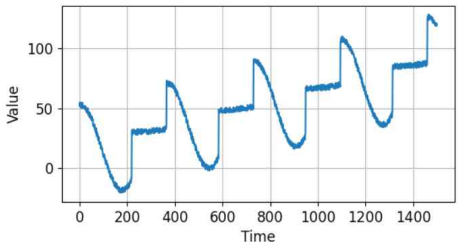
시계열 데이터 기초 코드

RNN과 LSTM

LSTM을 이용한 주가 예측 코드

시계열 데이터란?

- **경향성을 갖는 시계열 데이터**
- **계절성을 갖는 시계열 데이터**
- 노이즈를 갖는 시계열 데이터
- 자기 상관성을 갖는 시계열 데이터



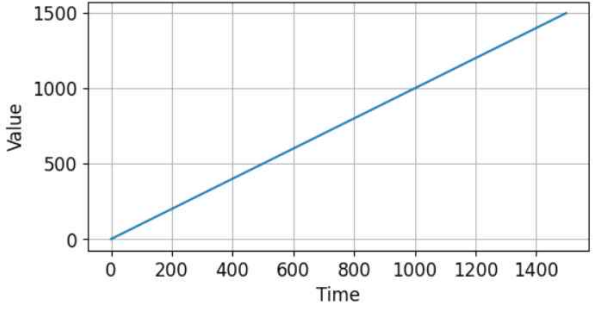
<자기 상관성을 갖는 시계열 데이터>

0. 시계열 데이터 시각화 함수

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

plt.style.use('default')
plt.rcParams['figure.figsize'] = (6, 3)
plt.rcParams['font.size'] = 12

def plot_series(time, series, format="-", start=0, end=None, label=None):
    plt.plot(time[start:end], series[start:end], format, label=label)
    plt.xlabel("Time")
    plt.ylabel("Value")
    if label:
        plt.legend(fontsize=14)
    plt.grid(True)
```

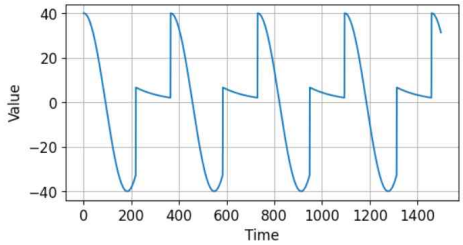
1. 경향성을 갖는 시계열 데이터

```
def trend(time, slope=0):
    return slope * time

time = np.arange(1500)
series = trend(time, slope=1) #양의 경향성

plot_series(time, series)
plt.show()
```



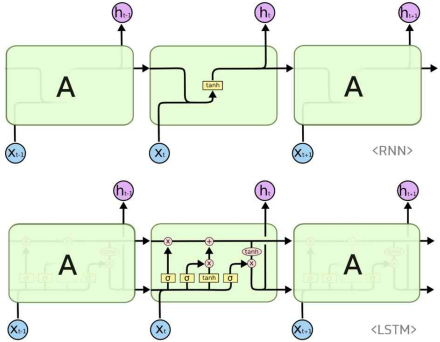
2. 계절성을 갖는 시계열 데이터

```
def seasonal_pattern(season_time):
    return np.where(season_time < 0.6,
                    np.cos(season_time * 2 * np.pi),
                    1 / np.exp(3 * season_time))
# np.where(a>10) == 조건 'a가 10 이상일 때'
# np.where(a>10, a*2, b) == 첫 번째 인자에 해당하는 조건을 보고 True면 두 번째 인자를, False면 세 번째 인자를 반환.

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

amplitude = 40
series = seasonality(time, period=365, amplitude=amplitude)

plot_series(time, series)
plt.show()
```



RNN과 LSTM

- RNN의 가장 큰 특징은 이전 cell의 정보를 다음 cell의 연산에서 사용한다는 것이다. (따라서 미래를 예측하는 데 좋은 모델이다.)
- LSTM(Long Short-Term Memory)은 RNN의 특별한 한 종류로, 긴 의존 기간을 필요로 하는 학습을 수행할 능력을 갖고 있다.
- LSTM은 RNN의 히든 state에 cell-state를 추가한 구조이다.



LSTM을 이용한 주가 예측 코드

```
# finance-datareader가 설치가 되지 않은 상황이라면 설치해준다.
pip install -U finance-datareader
```

LSTM을 이용한 주가 예측 코드

▾ 데이터 로드

```
[ ] import FinanceDataReader as fdr
    import numpy as np
    import matplotlib.pyplot as plt
```

```
[ ] # 삼성전자 종목에 대한 데이터('005930')를 2018년 5월 4일부터 2020년 1월 22일까지 로드한다.
    df = fdr.DataReader('005930', '2018-05-04', '2020-01-22')
    df.head()
```

Date	Open	High	Low	Close	Volume	Change
2018-05-04	53000	53900	51800	51900	39565391	-0.020755
2018-05-08	52600	53200	51900	52600	23104720	0.013487
2018-05-09	52600	52800	50900	50900	16128305	-0.032319
2018-05-10	51700	51700	50600	51600	13905263	0.013752
2018-05-11	52000	52200	51200	51300	10314997	-0.005814

```
[ ] print(df.shape)
```

```
(421, 6)
```

LSTM을 이용한 주가 예측 코드

▾ 데이터 전처리

```
[ ]  def MinMaxScaler(data):
         """최솟값과 최댓값을 이용하여 0 ~ 1 값으로 변환"""
         numerator = data - np.min(data, 0)
         denominator = np.max(data, 0) - np.min(data, 0)
         # 0으로 나누기 에러가 발생하지 않도록 매우 작은 값(1e-7)을 더해서 나눔
         return numerator / (denominator + 1e-7)
```

```
[ ]  dfx = df[['Open','High','Low','Volume', 'Close']]
     dfx = MinMaxScaler(dfx)
     dfy = dfx[['Close']]
     dfx = dfx[['Open','High','Low','Volume']]
     dfx
```

Date	Open	High	Low	Volume
2018-05-04	0.633401	0.646825	0.601610	0.595060
2018-05-08	0.617108	0.619048	0.605634	0.316465
2018-05-09	0.617108	0.603175	0.565392	0.198390
2018-05-10	0.580448	0.559524	0.553320	0.160765
2018-05-11	0.592668	0.579365	0.577465	0.100000
...	...	...	...	...
2020-01-16	0.881874	0.916667	0.891348	0.168830
2020-01-17	0.995927	0.968254	0.971831	0.196653
2020-01-20	1.000000	1.000000	1.000000	0.137469
2020-01-21	1.000000	0.984127	0.979879	0.114009
2020-01-22	0.938900	0.992063	0.947686	0.185040

421 rows × 4 columns

LSTM을 이용한 주가 예측 코드

```
[ ] dfx.describe()
```

	Open	High	Low	Volume
count	421.000000	421.000000	421.000000	421.000000
mean	0.372409	0.374434	0.373457	0.117439
std	0.175834	0.171931	0.171432	0.084973
min	0.000000	0.000000	0.000000	0.000000
25%	0.262729	0.267857	0.269618	0.064016
50%	0.350305	0.349206	0.352113	0.101418
75%	0.443992	0.452381	0.444668	0.146095
max	1.000000	1.000000	1.000000	1.000000

```
[ ] dfy.head()
```

Date	Close
2018-05-04	0.579158
2018-05-08	0.607214
2018-05-09	0.539078
2018-05-10	0.567134
2018-05-11	0.555110

```
[ ] # 두 데이터를 리스트 형태로 저장
    X = dfx.values.tolist()
    y = dfy.values.tolist()
```

LSTM을 이용한 주가 예측 코드

```
[ ]  # 데이터 전처리 - 10일 동안의 OHLVC 데이터로 다음 날의 종가를 예측

     window_size = 10

     data_X = []
     data_y = []
     for i in range(len(y) - window_size):
         _X = X[i : i + window_size] # 다음 날 종가(i+windows_size)는 포함되지 않음
         _y = y[i + window_size]     # 다음 날 종가
         data_X.append(_X)
         data_y.append(_y)
     print(_X, "->", _y)
```

```
[ ]  # 10일간의 OHLVC 데이터
     data_X[0]
```

```
[[0.6334012219933466,
  0.64682539682283,
  0.6016096579452651,
  0.5950598479352758],
 [0.6171079429710097,
  0.6190476190451625,
  0.6056338028144642,
```

```
[ ]  # data_X[0]을 넣었을 때의 정답인 data_y[0]
     data_y[0]
```

```
[0.503006012022032]
```

LSTM을 이용한 주가 예측 코드

훈련 데이터와 테스트 데이터를 분리

```
[ ]  print('전체 데이터의 크기 :', len(data_X), len(data_y))
```

```
전체 데이터의 크기 : 411 411
```

```
[ ]  train_size = int(len(data_y) * 0.7)
     train_X = np.array(data_X[0 : train_size])
     train_y = np.array(data_y[0 : train_size])

     test_size = len(data_y) - train_size
     test_X = np.array(data_X[train_size : len(data_X)])
     test_y = np.array(data_y[train_size : len(data_y)])

     print('훈련 데이터의 크기 :', train_X.shape, train_y.shape)
     print('테스트 데이터의 크기 :', test_X.shape, test_y.shape)
```

```
훈련 데이터의 크기 : (287, 10, 4) (287, 1)
테스트 데이터의 크기 : (124, 10, 4) (124, 1)
```

LSTM을 이용한 주가 예측 코드

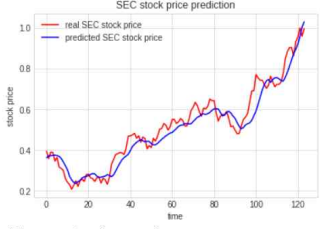
```
[ ]  from tensorflow.keras import Sequential
     from tensorflow.keras.layers import Dense, LSTM, Dropout
```

```
[ ]  model = Sequential()
     model.add(LSTM(units=20, activation='relu', return_sequences=True, input_shape=(10, 4)))
     model.add(Dropout(0.1))
     model.add(LSTM(units=20, activation='relu'))
     model.add(Dropout(0.1))
     model.add(Dense(units=1))
     model.summary()
```

```
[ ]  model.compile(optimizer='adam', loss='mean_squared_error')
     model.fit(train_X, train_y, epochs=70, batch_size=30)
     pred_y = model.predict(test_X)
```

LSTM을 이용한 주가 예측 코드

```
[ ] # 주가 예측 시각화

    pred_y = model.predict(test_X)

    plt.figure()
    plt.plot(test_y, color='red', label='real SEC stock price')
    plt.plot(pred_y, color='blue', label='predicted SEC stock price')
    plt.title('SEC stock price prediction')
    plt.xlabel('time')
    plt.ylabel('stock price')
    plt.legend()
    plt.show()

    print("내일 SEC 주가 :", df.Close[-1] * pred_y[-1] / dfy.Close[-1], 'KRW')

    4/4 [==============================] - 0s 4ms/step
```




```
내일 SEC 주가 : [64433.91] KRW
```

출처:
[시계열 데이터 기초]: https://codetorial.net/tensorflow/time_series_forecasting/time_series_data.html
[주가 예측 코드]: https://wikidocs.net/173005

Q & A