7️⃣ 시계열 분석

1. 시계열 문제

시계열 분석: 시간에 따라 변하는 데이터를 사요아여 추이를 분석 → 추세를 파악하거나 향후 전망 등을 예측하기 위한 용도

데이터 변동 유형에 따른 시계열 형태

- 불규칙 변동: 시계열 자료에서 시간에 따른 규칙적인 움직임과 달리 어떤 규칙성 없이 예측 불가능하고 우연적으로 발생하는 변동 (전쟁, 홍수, 화재, 지진, 파업)
- 추세 변동: 시계열 자료가 갖는 장기적인 변화 추세 (GDP, 인구 증가율)
- 순환 변동: 2~3년 정도의 일정한 기간을 주기로 순환적으로 나타나는 변동 (경기 변동)
- 계절 변동: 계절적 영향과 사회적 관습에 따라 1년 주기로 발생하는 것

⇒ 규칙적 시계열과 불규칙적 시계열로 나뉨

불규칙적 시계열 데이터에 규칙성을 부여하는 방법 → AR, MA, ARMA, ARIMA 모델

2. AR, MA, ARMA, ARIMA

시계열 분석 → 시간을 독립 변수로 사용하여 종속 변수 예측

(1) AR 모델

AR(자기 회귀) 모델: 이전 관측 값이 이후 관측 값에 영향을 준다는 아이디어

$$\underset{①}{Z_t} = \underset{②}{\Phi_1 Z_{t-1} + \Phi_2 Z_{t-2} + \cdots + \Phi_p Z_{t-p}} + \underset{③}{a_t}$$

- 1은 시계열 데이터에서 현재 시점을 의미
- 2는 과거가 현재에 미치는 영향을 나타내는 모수(파이)에 시계열 데이터의 과거 시점을 곱한 것
- 3은 시계열 분석에서 오차항을 의미 (백색 잡음)

⇒ p시점을 기준으로 그 이전의 데이터에 의해 현재 시점의 데이터가 영향을 받는 모형

(2) MA 모델

MA(이동 평균) 모델: 트랜드(y값)가 변화하는 상황에 적합한 회귀 모델

$$\underset{①}{Z_t} = \underset{②}{\theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_p a_{t-p}} + \underset{③}{a_t}$$

- 1은 시계열 데이터에서 현재 시점을 의미
- 2는 매개변수(세타)에 과거 시점의 오차를 곱한 것
- 3은 오차항

⇒ AR 모델처럼 이전 데이터의 '상태'에서 현재 데이터의 상태를 추론하는 것이 아닌, 이전 데이터의 '오차'에서 현재 데이터의 상태를 추론하겠다는 의미

(3) ARMA 모델

ARMA(자기 회귀 이동 평균) 모델은 AR과 MA를 섞은 모델 → 두가지 관점에서 과거의 데이터를 사용하는 것

$$Z_t = a + \Phi_1 Z_{t-1} + \cdots + \Phi_p Z_{t-p} + \theta_1 a_{t-1} + \cdots + \theta_q a_{t-q} + a_t$$

(4) ARIMA 모델

ARIMA(자기 회귀 누적 이동 평균) 모델 : 자기 회귀와 이동 평균을 둘 다 고려하는 모형

→ ARMA와 달리 곽 데이터의 선형 관계뿐만 아니라 추세까지 고려

<코드>

ARIMA(p,d,q) 파라미터

- p: 자기 회귀 함수
- d: 차분 차수
- q: 이동 평균 차수

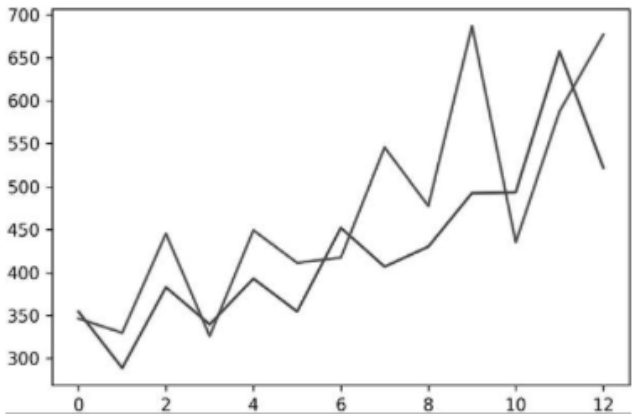
```
ARIMA Model Results
==============================================================================
Dep. Variable:                D.Sales   No.Observations:                   35
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -197.350
Method:                       css-mle   S.D. of innovations             66.436
Date:                Sun, 02 Aug 2020   AIC                            408.699
Time:                        10:28:58   BIC                            419.587
Sample:                    02-01-1991   HQIC                           412.458
                         - 12-01-1993
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.4256      3.774      3.292      0.001       5.028      19.823
ar.L1.D.Sales -1.0850      0.188     -5.764      0.000      -1.454      -0.716
ar.L2.D.Sales -0.6688      0.283     -2.365      0.018      -1.223      -0.114
ar.L3.D.Sales -0.4426      0.297     -1.489      0.136      -1.025       0.140
ar.L4.D.Sales -0.0495      0.288     -0.172      0.864      -0.614       0.515
ar.L5.D.Sales  0.1652      0.197      0.840      0.401      -0.220       0.551
                                    Roots
==============================================================================
                  Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1           -1.1401           -0.4612j            1.2298           -0.4388
AR.2           -1.1401           +0.4612j            1.2298            0.4388
AR.3            0.0222           -1.2562j            1.2564           -0.2472
AR.4            0.0222           +1.2562j            1.2564            0.2472
AR.5            2.5355           -0.0000j            2.5355           -0.0000
------------------------------------------------------------------------------
```

ARIMA() 함수를 사용한 예측을 진행

```
import numpy as np
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

def parser(x):
    return datetime.strptime('199'+x, '%Y-%m')

series = read_csv('../chap7/data/sales.csv', header=0, parse_dates=[0], index_col=0,
                  squeeze=True, date_parser=parser)
X = series.values
X = np.nan_to_num(X)
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)] ------ train과 test로 데이터셋 분리
history = [x for x in train]
predictions = list()
for t in range(len(test)): ------ test 데이터셋의 길이(13번)만큼 반복하여 수행
    model = ARIMA(history, order=(5,1,0)) ------ ARIMA() 함수 호출
    model_fit = model.fit(disp=0)
    output = model_fit.forecast() ------ forecast() 메서드를 사용하여 예측 수행
    yhat = output[0] ------ 모델 출력 결과를 yhat에 저장
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs)) ------ 모델 실행 결과를 predicted로 출력하고, test로 분리해 둔 데이터를 expected로 사용하여 출
error = mean_squared_error(test, predictions) ------ 손실 함수로 평균 제곱 오차 사용
print('Test MSE: %.3f' % error)
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```



→ 데이터가 우상향 추세를 나타내고 있으므로, 자전거 판매가 향후에도 계속 증가할 것임을 예측할 수 있음

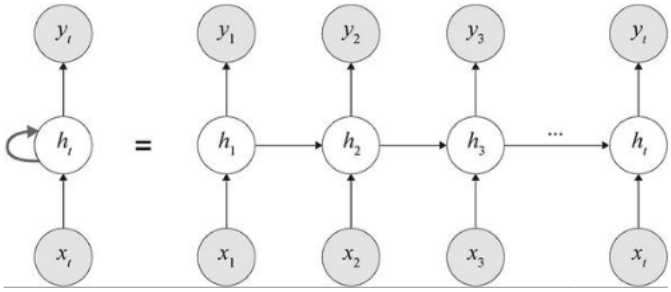
⇒ ARIMA를 사용할 경우 데이터 경향을 파악해서 미래를 예측할 수 있다.

3. 순환 신경망(RNN)

RNN: 시간적으로 연속성이 있는 데이터를 처리하려고 고안된 인공 신경망

기존 네트워크와 다른 점 : '기억(memory)'을 갖는다. (기억은 현재까지 입력 데이터를 요약한 정보)

⇒ 새로운 입력이 네트워크로 들어올 때마다 기억은 조금씩 수정되며, 결국 최종적으로 남겨진 기억은 모든 입력 전체를 요약한 정보가 됨.



첫 번째 입력이 들어오면 첫 번째 기억(h1)이 만들어지고, 두 번째 입력이 들어오면 기존 기억(h1)과 새로운 입력을 참고하여 새 기억(h2)을 만듦.

⇒ RNN은 외부 입력과 자신의 이전 상태를 입력받아 현재 상태를 갱신하는 것!