[EEC3600-001] 수치해석		
소속: 전기전자공학부	학번: 12191529	이름: 장준영
Term Project		Prob #3

1. Problem

a. 문제

Problem 3. [*Least-squares Methods (programming part)*] Auto-regressive time series model. Suppose that $z_1, z_2, \ldots$ is a time series. An auto-regressive model (also called AR model) for the time series has the form

$$\hat{z}_{t+1} = \theta_1 z_t + \cdots + \theta_M z_{t-M+1}, \quad t = M, M+1, \ldots$$

where M is the memory or lag of the model. Here $\hat{z}_{t+1}$ is the prediction of z_{t+1} made at time t (when $z_t, \ldots, z_{t-M+1}$ are known). This prediction is a linear function of the previous M values of the time series. With good choice of model parameters, the AR model can be used to predict the next value in a time series, given the current and previous M values. This has many practical uses.

We can use least squares (or regression) to choose the AR model parameters, based on the observed data $z_1, \ldots, z_T$, by minimizing the sum of squares of the prediction errors $z_t - \hat{z}_t$ over $t = M+1, \ldots, T$, i.e.,

$$(z_{M+1} - \hat{z}_{M+1})^2 + \cdots + (z_T - \hat{z}_T)^2$$

(We must start the predictions at $t = M+1$, since each prediction involves the previous M time series values, and we do not know $z_0, z_{-1}, \ldots$.)

The AR model can be put into the general linear in the parameters model form by taking

$$y^{(i)} = z_{M+i}, \quad x^{(i)} = (z_{M+i-1}, \ldots, z_i), \quad i = 1, \ldots, T-M$$

We have $N = T - M$ examples, and $n = M$ features.

Consider the time series of hourly temperature data given in the file

`temperature_data.ipynb`

with length $31 \times 24 = 744$.

(a) Fit an AR model with memory $M = 8$ using least squares, with $N = 31 \times 24 - 8 = 736$ samples.

(b) What is the RMS error of this predictor?

$$\mathrm{RMS}(e) = \sqrt{\frac{\sum_{t=9}^{744} (z_t - \hat{z}_t)^2}{N}}$$

(c) Repeat the computations of Parts (a) and (b), i.e., (a) Model fitting and (b) Computation of the RMS error, for $M = 4, 8, 12, 24$. Plot M vs. RMS.

본 문제에서는 2016 년 5 월 한 달간 LAX 의 시간당 온도 데이터를 기반으로, Auto-Regressive (AR) 모델을 구성하고, Least-Squares 방법을 통해 계수 벡터 θ를 추정한다.

2. Solution (a), (b) : M = 8

■ 데이터 정의

```python
# Step 1: Define data function
# ----------------------------------------
def temperature_data():
    t = np.array([
        59.0, 57.9, 57.0, 57.0, 55.9, 57.0, 57.0, 60.1, 62.1, 64.0, 64.9, 66.0, 69.1, 68.0, 69.1, 66.0, 64.0, 62.1, 60.1, 59.0, 57.9, 57.9, 59.0, 57.9,
        57.9, 57.9, 57.0, 57.0, 57.0, 57.9, 57.9, 59.0, 61.0, 62.1, 64.0, 64.0, 64.0, 64.9, 64.9, 64.9, 64.0, 62.1, 61.0, 59.0, 59.0, 59.0, 59.0, 59.0,
        59.0, 59.0, 57.9, 57.9, 57.9, 57.0, 59.0, 62.1, 64.0, 64.9, 66.0, 66.9, 66.9, 66.0, 68.0, 66.9, 64.0, 61.0, 60.1, 59.0, 57.9, 57.0, 57.9, 57.9,
        57.0, 57.0, 57.0, 57.0, 57.9, 57.9, 57.9, 59.0, 59.0, 57.9, 59.0, 62.1, 64.0, 64.0, 64.0, 64.0, 63.0, 61.0, 61.0, 60.1, 60.1, 60.1, 60.1, 60.1,
        60.1, 59.0, 59.0, 59.0, 60.1, 60.1, 60.1, 60.1, 61.0, 63.0, 63.0, 63.0, 64.9, 66.0, 68.0, 66.9, 66.9, 64.0, 63.0, 62.1, 62.1, 60.1, 60.1, 59.0,
        57.9, 57.9, 59.0, 59.0, 57.9, 57.9, 59.0, 60.1, 60.1, 61.0, 63.0, 64.9, 64.9, 66.0, 66.9, 66.9, 64.9, 64.0, 62.1, 60.1, 60.1, 60.1, 60.1, 59.0,
        59.0, 60.1, 59.0, 59.0, 59.0, 59.0, 57.0, 57.9, 59.0, 60.1, 59.0, 64.0, 66.0, 66.0, 66.0, 64.9, 64.9, 64.0, 62.1, 61.0, 61.0, 60.1, 60.1, 60.1,
        60.1, 59.0, 59.0, 57.9, 57.9, 57.9, 57.9, 59.0, 61.0, 63.0, 64.0, 64.9, 66.0, 66.0, 64.9, 64.9, 64.0, 63.0, 62.1, 61.0, 61.0, 60.1, 60.1, 60.1,
        59.0, 60.1, 59.0, 59.0, 57.9, 57.9, 59.0, 60.1, 61.0, 61.0, 62.1, 63.0, 64.0, 64.9, 64.0, 64.9, 64.0, 63.0, 61.0, 60.1, 59.0, 59.0, 59.0, 59.0,
        59.0, 59.0, 57.9, 57.9, 59.0, 59.0, 60.1, 60.1, 61.0, 62.1, 63.0, 64.0, 64.9, 66.9, 66.9, 66.9, 64.9, 63.0, 62.1, 61.0, 60.1, 60.1, 60.1, 60.1,
        60.1, 60.1, 59.0, 60.1, 59.0, 60.1, 60.1, 61.0, 61.0, 62.1, 63.0, 64.9, 66.9, 66.9, 66.0, 64.9, 63.0, 62.1, 61.0, 61.0, 60.1, 59.0, 59.0, 57.9,
        57.9, 57.9, 57.0, 57.9, 57.9, 57.9, 57.9, 59.0, 61.0, 64.0, 64.0, 64.0, 64.9, 64.9, 64.9, 64.0, 62.1, 61.0, 61.0, 60.1, 60.1, 60.1, 60.1, 60.1,
        59.0, 59.0, 60.1, 60.1, 60.1, 60.1, 60.1, 61.0, 62.1, 62.1, 63.0, 64.0, 69.1, 68.0, 68.0, 66.0, 64.9, 63.0, 63.0, 62.1, 62.1, 62.1, 62.1, 61.0,
        63.0, 62.1, 61.0, 61.0, 61.0, 61.0, 61.0, 62.1, 62.1, 64.9, 64.9, 64.9, 66.9, 66.9, 66.0, 66.0, 64.9, 64.0, 62.1, 61.0, 61.0, 61.0, 61.0, 62.1,
        63.0, 62.1, 62.1, 62.1, 62.1, 62.1, 62.1, 62.1, 63.0, 64.0, 66.0, 68.0, 70.0, 69.1, 68.0, 66.9, 66.9, 64.9, 64.0, 62.1, 61.0, 61.0, 61.0, 61.0,
        61.0, 62.1, 62.1, 62.1, 61.0, 62.1, 62.1, 63.0, 63.0, 63.0, 64.9, 66.9, 68.0, 68.0, 68.0, 68.0, 66.9, 64.9, 64.0, 62.1, 63.0, 62.1, 62.1, 62.1,
        62.1, 61.0, 61.0, 61.0, 61.0, 61.0, 62.1, 62.1, 64.0, 64.0, 66.0, 66.9, 66.9, 68.0, 68.0, 66.9, 66.0, 64.9, 63.0, 62.1, 62.1, 62.1, 61.0, 61.0,
        62.1, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 62.1, 62.1, 63.0, 63.0, 62.1, 63.0, 62.1, 62.1, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0,
        61.0, 62.1, 61.0, 61.0, 61.0, 61.0, 61.0, 62.1, 62.1, 63.0, 64.0, 64.9, 64.9, 66.9, 66.9, 66.9, 64.9, 63.0, 62.1, 61.0, 61.0, 61.0, 61.0, 60.1,
        60.1, 60.1, 60.1, 59.0, 57.9, 59.0, 59.0, 61.0, 62.1, 66.0, 64.0, 64.0, 64.0, 64.9, 64.9, 62.1, 63.0, 61.0, 61.0, 59.0, 59.0, 57.9, 57.9, 57.9,
        57.0, 57.0, 55.9, 57.0, 57.0, 55.9, 57.0, 60.1, 62.1, 62.1, 64.0, 64.0, 64.9, 64.0, 64.0, 63.0, 63.0, 62.1, 61.0, 60.1, 60.1, 60.1, 59.0, 59.0,
        57.9, 57.9, 57.0, 54.0, 55.0, 54.0, 57.0, 59.0, 61.0, 64.9, 66.0, 66.9, 66.9, 66.0, 66.0, 66.0, 64.9, 64.0, 64.0, 61.0, 61.0, 60.1, 60.1, 59.0,
        57.9, 57.0, 57.9, 57.0, 55.9, 57.0, 57.9, 61.0, 63.0, 64.0, 66.9, 69.1, 70.0, 70.0, 70.0, 66.0, 66.0, 63.0, 62.1, 60.1, 59.0, 59.0, 57.9, 57.9,
        57.9, 57.9, 57.9, 57.9, 57.9, 57.9, 59.0, 62.1, 62.1, 63.0, 64.0, 66.0, 66.0, 64.9, 64.9, 64.9, 64.9, 63.0, 61.0, 61.0, 60.1, 60.1, 60.1, 59.0,
        57.9, 57.9, 57.9, 57.0, 57.0, 57.0, 57.9, 60.1, 61.0, 62.1, 63.0, 64.0, 64.0, 64.0, 64.0, 66.0, 64.9, 64.0, 62.1, 60.1, 60.1, 59.0, 57.9, 57.9,
        57.9, 57.9, 57.9, 55.9, 55.9, 57.0, 59.0, 61.0, 63.0, 64.0, 66.0, 66.0, 68.0, 68.0, 66.9, 64.9, 64.0, 63.0, 63.0, 61.0, 60.1, 60.1, 60.1, 60.1,
        59.0, 59.0, 59.0, 59.0, 57.9, 57.9, 60.1, 61.0, 63.0, 62.1, 64.9, 66.0, 66.9, 68.0, 68.0, 66.9, 66.0, 64.0, 61.0, 61.0, 60.1, 60.1, 60.1, 60.1,
        60.1, 61.0, 61.0, 60.1, 60.1, 60.1, 60.1, 61.0, 62.1, 63.0, 63.0, 64.0, 66.0, 66.9, 66.0, 64.0, 63.0, 62.1, 61.0, 60.1, 59.0, 59.0, 59.0, 60.1,
        60.1, 60.1, 60.1, 60.1, 60.1, 59.0, 60.1, 61.0, 62.1, 64.9, 64.9, 66.0, 66.9, 66.9, 66.0, 66.0, 66.0, 64.0, 62.1, 61.0, 60.1, 60.1, 60.1, 59.0,
        59.0, 59.0, 57.9, 57.9, 59.0, 60.1, 60.1, 61.0, 63.0, 63.0, 64.0, 64.9, 66.9, 68.0, 66.0, 66.0, 64.9, 63.0, 62.1, 61.0, 60.1, 61.0, 60.1, 59.0,
        59.0, 59.0, 59.0, 59.0, 60.1, 60.1, 61.0, 61.0, 62.1, 63.0, 63.0, 64.9, 68.0, 68.0, 68.0, 66.0, 64.9, 63.0, 62.1, 62.1, 62.1, 62.1, 62.1, 63.0 ]
    return t
```

- LAX 의 5 월 온도 시계열 데이터를 벡터 형태로 반환하는 함수이다.
- 전체 길이는 $T = 31 \times 24 = 744$개로 구성되어 있다.

■ 데이터 로딩 및 파라미터 설정

```python
# ----------------------------------------
# Step 2: Load data
# ----------------------------------------
z = temperature_data()        # Time series data (length 744)
T = len(z)                    # Total length T = 744
M = 8                         # AR model memory size
N = T - M                     # Number of usable samples = 736
```

- 시계열데이터를 불러오고, AR 모델의 메모리 길이를 $M = 8$로 설정한다.
- 이에 따라 학습 가능한 입력-출력 쌍은 736 개로 제한된다.

■ 디자인 행렬 X 및 목표 벡터 y 구성

```python
# ----------------------------------------
# Step 3: Build design matrix X and target vector y
# ----------------------------------------
X = np.zeros((N, M))          # Input: previous M values
y = np.zeros(N)               # Output: next time point

for i in range(N):
    X[i, :] = z[i:i+M][::-1]  # Most recent to oldest
    y[i] = z[i + M]           # Actual next value
```

- 각 입력 X_i는 8 개의 과거 온도값으로 구성되며, 출력 y_i는 그 다음 시점의 온도이다.
- 이렇게 구성된 $X \in \mathbb{R}^{736\times 8}, y \in \mathbb{R}^{736}$ 는 Linear-regression 문제로 정리 가능하다.

■ 최소제곱 해 구하기

```
# ---------------------------------------
# Step 4: Estimate theta using least squares
# ---------------------------------------
theta = np.linalg.lstsq(X, y, rcond=None)[0]  # Solve Normal Equation
```

- 최소제곱 해법 $\theta = (X^T X)^{-1} X^T y$를 통해 AR 계수를 추정한다.
- 이는 NumPy 의 "lstsq()" 함수를 사용하여 수치적으로 안정적으로 계산된다.

■ 예측 및 RMS 오차 계산

```
# Compute predictions
y_pred = X @ theta

# ---------------------------------------
# Step 5: Compute RMS error
# ---------------------------------------
rms_error = np.sqrt(np.mean((y - y_pred) ** 2))
```

- 예측값은 $\hat{y} = X\theta$로 계산된다.
- 실제값과의 차이에 대한 Root Mean Square (RMS) 오차는 모델의 예측 정확도를 나타낸다.

■ 예시 실행 및 결과 확인

```
# ---------------------------------------
# Step 6: Display results and plot
# ---------------------------------------
print("AR(8) coefficients (theta):")
print(theta)
print(f"RMS Error: {rms_error:.4f}")
```

```python
# Plot actual vs. predicted values (optional)
plt.figure(figsize=(10, 4))
plt.plot(range(N), y, label='Actual', alpha=0.7)
plt.plot(range(N), y_pred, label='Predicted', linestyle='--')
plt.xlabel('Time Step')
plt.ylabel('Temperature')
plt.title('AR(8) Prediction vs. Actual')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

- 실제 온도와 예측 온도를 함께 시각화하여 예측 성능을 직관적으로 확인할 수 있다.

[결과 분석]

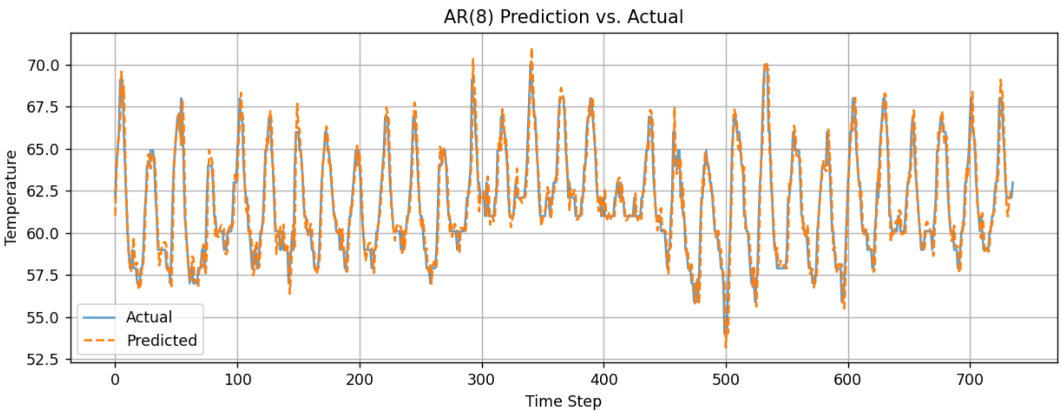
1) AR(8) 계수 θ

```
AR(8) coefficients (theta):
[ 1.23376549  0.00791052 -0.21489375  0.04341608 -0.11817636 -0.1318138
  0.09147989  0.08816761]
```

$$\theta = [1.2338, 0.0079, -0.2149, 0.0434, -0.1182, -0.1318, 0.0915, 0.0882\ ]^T$$

계수는 최근 값 (1-step lag)에 큰 비중을 두고, 나머지는 비교적 낮거나 음수 계수로 반영되어 있다. 이는 최근 온도 정보가 미래 예측에 가장 중요함을 시사한다.

2) RMS 오차




```
RMS Error: 1.0130
```

평균적으로 약 $\pm 1.01°\mathrm{F}$ 오차 내에서 예측이 이루어지고 있다. 시계열의 추세를 잘 따라가며, 예측 정확도가 준수한 수준임을 보여준다.

3. Solution (c): M=4, 8, 12, 24

1) 시뮬레이션 목적 및 방법

문제 3-(a), (b)에서 수행한 Auto-Regressive (AR) 모델 학습과 RMS 오차 계산 과정을, 서로 다른 메모리 크기 $M \in \{4, 8, 12, 24\}$에 대해 반복 수행한다. 이 과정을 통해 모델의 메모리 길이에 따른 예측 정확도 (RMS Error) 변화를 관찰하고자 한다.

각 메모리 길이 M 에 대해 다음 과정을 반복한다:

1) 총 744 개의 온도 시계열 데이터에서, M 개의 과거 값으로 구성된 입력 벡터 X 와 다음 시점의 실제값 y를 구성한다:
2) 선형 최소제곱법을 이용하여 AR 계수 $\theta \in \mathbb{R}^M$을 추정한다.
3) 예측값 $\hat{y} = X\theta$를 계산하고, 실제 y와 비교하여 RMS 오차를 구한다.
4) 이렇게 얻어진 RMS 오차를 M 에 따라 정리하고, 그래프 (Memory size vs. RMS Error)를 시각화한다.

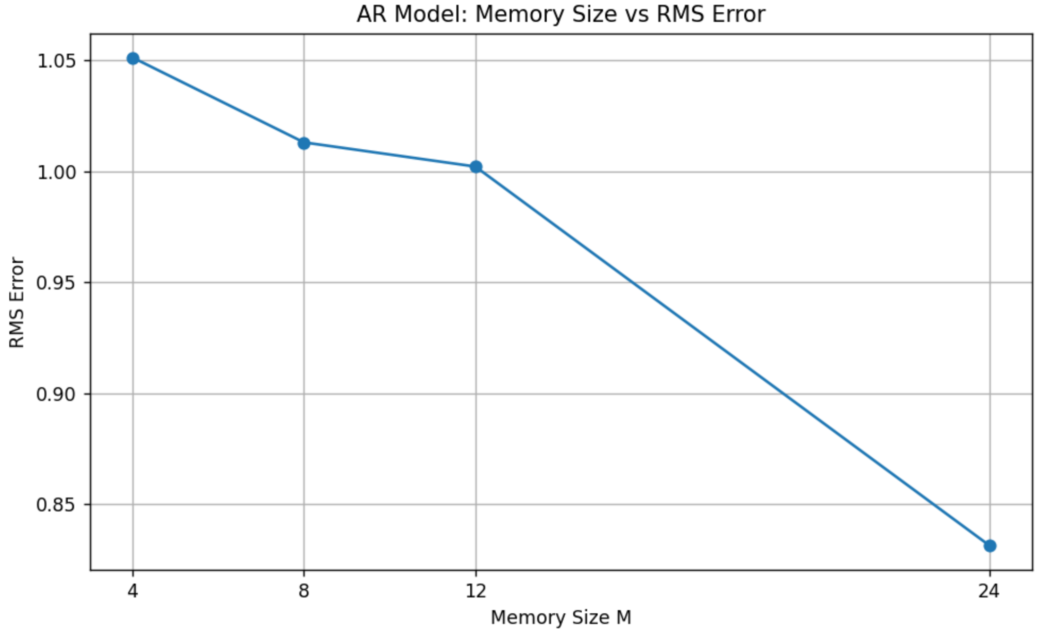
2) Python Code 설명

```python
# Step 3: Define memory sizes to test
M_values = [4, 8, 12, 24]
rms_errors = []

# Step 4: Fit AR(M) models and compute RMS for each M
for M in M_values:
    N = T - M
    X = np.zeros((N, M))
    y = np.zeros(N)

    for i in range(N):
        X[i, :] = z[i:i+M][::-1]  # reverse: most recent first
        y[i] = z[i + M]

    theta = np.linalg.lstsq(X, y, rcond=None)[0]
    y_pred = X @ theta
    rms = np.sqrt(np.mean((y - y_pred) ** 2))
    rms_errors.append(rms)
```

- 각 M에 대해 design matrix X와 target y를 구성하고, 최소제곱해 θ를 구하여 예측을 수행한다.
- 예측 오차를 RMS 기준으로 평가하여 rms_errors 리스트를 작성한다.

앞서 (a)에서 다음과 같은 선형 시스템 전개식을 유도하였다:

3) 시뮬레이션 결과 및 그래프 해석



이번 문제에서는 AR 모델의 메모리 크기 (M)에 따라 예측 성능이 어떻게 달라지는지를 분석하였다. 이를 위해 M=4, 8, 12, 24 의 네 가지 값을 설정하고, 각각의 경우에 대해 AR 모델을 최소제곱법으로 학습한 후, 예측값과 실제값의 차이를 Rood Mean Square (RMS) 오차로 평가하였다. 그 결과, M=4 일 때는 RMS 오차가 약 1.05 로 가장 높았으며, 이는 과거 데이터의 양이 부족해 시계열의 패턴을 충분히 반영하지 못한 것으로 해석된다. 반면, M=8, M=12 에서는 각각 RMS Error 가 약 1.02 와 1.00 으로 다소 감소하였고, 이는 메모리 길이가 증가함에 따라 모델이 더 많은 시계열 정보를 활용하여 예측력을 개선하였음을 의미한다.

가장 흥미로운 결과는 M=24 에서 나타났는데, 이 경우 RMS 오차가 약 0.83 까지 감소하였다. 이는 모델이 하루치 온도(24 시간)를 하나의 패턴으로 인식하여 이를 효과적으로 학습했다는 점을 시사하며, 실제 온도 시계열의 일변 주기성과도 잘 부합한다. 즉, 충분한 과거 정보를 제공하면 AR 모델은 계절성 또는 주기성을 포착하여 정확한 예측을 수행할 수 있다.

이러한 결과는 AR 모델의 성능이 메모리 길이에 따라 큰 영향을 받는다는 사실을 보여준다. 작은 M 은 정보 부족으로 인해 정확도가 떨어지는 반면, 큰 M 은 더 많은 시계열 정보를 포착할 수 있어 성능 향상에 기여한다. 다만 일반적으로 메모리 크기를 무한히 늘리는 것이 항상 좋은 결과를 보장하지는 않으며, 데이터의 주기성과 잡음 수준, 그리고 계산 복잡도 등을 고려하여 최적의 M 값을 선택하는 것이 중요하다.