[EEC3600-001] 수치해석		
소속: 전기전자공학부	학번: 12191529	이름: 장준영
Term Project		**Prob #6**

1. Problem

a. 문제

Problem 6. [*Optimization (programming)*]

We continue a linear-equality constrained least-squares problem where the goal is to purchase advertising in n different channels so as to achieve (or approximately achieve) a target set of customer views or impressions in m different demographic groups. We denote the n-vector of channel spending as s; this spending results in a set of views (across the demographic groups) given by the m-vector $v = Rs$.

We want to minimize the sum of squares of the deviation from the target set of views, given by v_{des}. In addition, we fix our total advertising spending, with the constraint $\sum_{i=1}^{n} s_i = \mathbf{1}^\top s = b$, where b is a given total advertising budget and $\mathbf{1} = [1\ 1\ \cdots\ 1]^\top \in \mathbb{R}^n$ denotes the all-ones n-vector. (This can also be described as allocating a total budget b across the n different channels.) This leads to the constrained least squares problem

$$\begin{aligned} &\text{minimize} && \|Rs - v_{\text{des}}\|^2 \\ &\text{subject to} && \mathbf{1}^\top s = b \end{aligned}$$

The solution $\hat{s}$ of this problem is not guaranteed to have nonnegative entries, as it must to make sense in this application. But we ignore this aspect of the problem here.

(a) Write down two first-order necessary conditions we studied in the class. Namely, (1) primal condition and (2) primal-dual condition. Both are linear systems.

(b) Make a python code to compute the solution $\hat{s}$ for given $R \in \mathbb{R}^{m\times n}$, $v_{\text{des}} \in \mathbb{R}^m$, and $b \in \mathbb{R}$. Run your code with the data file `advertizing_budget_data.ipynb`.

(c) Compute and compare the RMS errors with different budgets,

$$b = 1200, 1300, \cdots, 1700$$

where the RMS error is defined as

$$\text{RMS} = \frac{\|R\hat{s} - v_{\text{des}}\|}{\sqrt{m}}.$$

For comparisons, plot b vs. RMS.

(Hint!)

[17]:
```
#advertising budget
R = np.matrix([[.97,1.86,.41],[1.23,2.18,.53],[.8,1.24,.62],[1.29,.98,.
↪51],[1.1,1.23,.69],[.67,.34,.54],[.87,.26,.62],[1.1,.16,.48],[1.92,.22,.
↪71],[1.29,.12,.62]])
m,n = np.shape(R)
vdes = 1e3 * np.ones(m)
s = npl.lstsq(R,vdes)[0]
s
#will be continued in 16 re: how to add a constraint like a total budget
```

```
/Users/kwangkikim/anaconda3/envs/py-numanal/lib/python3.7/site-
packages/ipykernel_launcher.py:5: FutureWarning: `rcond` parameter will
change
to the default of machine precision times ``max(M, N)`` where M and N are
the
input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  """
```

[17]:
```
array([  62.07662454,   99.98500403, 1442.83746254])
```

[18]:
```
rms = lambda x: np.sqrt(np.mean(np.square(x))) #not a built in numpy
function
rms(np.matmul(R,s) - vdes)
```

[18]:
```
132.6381902632653
```

본 문제는 광고 예산을 여러 채널에 분배하여 특정 인구통계 그룹에 대한 도달량을 최대한 정확히 달성하는 **선형 등식 제약 하의 최소제곱(Least Squares with Equality Constraint)** 문제이다.
우리는 다음과 같은 모델을 고려한다:

- $R \in \mathbb{R}^{m \times n}$: 채널별 도달률 (행: 인구 집단, 열: 채널)
- $s \in \mathbb{R}^{n}$: 채널별 광고 예산 분배
- $v_{des} \in \mathbb{R}^{n}$: 원하는 인구 집단별 도달량
- $s \in \mathbb{R}^{n}$: 총 광고 예산

본 문제는 다음과 같은 **제약조건 최소제곱** 문제로 표현할 수 있다:

$$minimize \ \|Rs - v_{des}\|^2 \ subject \ to \ 1^T s = b$$

이는 총 예산 b에서 인구 집단별 도달량과의 오차를 최소화하는 s를 찾는 문제이다.

2. Solution (a)

1) Prime Condition

주어진 문제는 다음과 같다:

$$minimize \ \|Rs - v_{des}\|^2 \qquad subject\ to \quad 1^T s = b$$

이는 선형 등식 제약이 포함된 최소제곱 문제이며, 목적함수는 convex 하므로 KKT 조건이 1 차 필요조건이자 충분조건이다.

우선 목적함수 $f(s) = \|Rs - v_{des}\|^2$의 gradient 는 다음과 같다:

$$\nabla f(s) = 2R^T(Rs - v_{des})$$

여기에 등식 제약 $1^T s = b$를 라그랑주 승수 λ와 함께 반영하면, 다음과 같은 Lagrangian 을 구성할 수 있다:

$$\mathcal{L}(s, \lambda) = \|Rs - v_{des}\|^2 + \lambda(1^T s - b)$$

2) Primal-Dual Condition (KKT 조건)

KKT 조건을 적용하면 다음의 선형 시스템을 얻을 수 있다:

$$\begin{cases} 2R^T(Rs - v_{des}) + \lambda \cdot 1 = 0 \\ 1^T s - b = 0 \end{cases}$$

이를 행렬로 정리하면 다음과 같은 선형 시스템이 된다:

$$\begin{bmatrix} 2R^T R & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} s \\ \lambda \end{bmatrix} = \begin{bmatrix} 2R^T v_{des} \\ b \end{bmatrix}$$

3. Solution (b)

■ 데이터 함수 정의 (advertising_budget_data())

```python
import numpy as np

# ----------------------------
# Step 1: 데이터 함수 정의
# ----------------------------
def advertising_budget_data():
    """
    Return R matrix (reach rate) and v_des vector (target views).
    R is shape (m, n), v_des is length m.
    """
    R = np.array([
        [0.97, 1.86, 0.41],
        [1.23, 2.18, 0.53],
        [0.80, 1.24, 0.62],
        [1.29, 0.98, 0.51],
        [1.10, 1.23, 0.69],
        [0.67, 0.34, 0.54],
        [0.87, 0.26, 0.62],
        [1.10, 0.16, 0.48],
        [1.92, 0.22, 0.71],
        [1.29, 0.12, 0.62]
    ])
    m, n = R.shape
    v_des = 1e3 * np.ones(m)
    return R, v_des
```

- 목적: 문제에 주어진 광고 채널 도달률 행렬 $R \in \mathbb{R}^{m \times n}$과, 목표 도달률 벡터 $v_{des} \in \mathbb{R}^{m}$를 초기화한다.
- R : 각 행은 인구 집단별로 광고 채널 3 개에 대한 도달률을 의미한다.
- v_{des}: 각 인구 집단별로 1000 뷰를 목표로 하므로, 모든 항목이 1000 인 벡터로 구성된다.

■ 제약 최소제곱 해법 (solve_least_squares_with_constraint())

```python
# ----------------------------
# Step 2: 제약 최소제곱 해법
# ----------------------------
def solve_least_squares_with_constraint(R, v_des, b):
    """
    Solve min ||Rs - v_des||^2 subject to 1^T s = b
    using KKT system.
    """
    m, n = R.shape
    RtR = 2 * R.T @ R
    Rtv = 2 * R.T @ v_des
    ones = np.ones((n, 1))

    # KKT 시스템 구성
    KKT_matrix = np.block([
        [RtR,    ones],
        [ones.T, np.zeros((1, 1))]
    ])
    rhs = np.concatenate([Rtv, [b]])

    # 선형 시스템 풀기
    sol = np.linalg.solve(KKT_matrix, rhs)
    s_opt = sol[:n]
    return s_opt
```

- 목적: 제약 조건 $1^T s - b = 0$ 하에서 최소제곱 문제 $min\|Rs - v_{des}\|^2$의 최적 해 $\hat{s}$를 구한다.
- 구현 원리: 수업에서 유도한 KKT 조건 기반 선형 시스템을 구성하고 풀도록 한다.
- KKT 시스템 구성:

$$\begin{bmatrix} 2R^T R & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} s \\ \lambda \end{bmatrix} = \begin{bmatrix} 2R^T v_{des} \\ b \end{bmatrix}$$

  - RtR: $2R^T R$ 목적 함수의 Hessian.
  - Rtv: $2R^T v_{des}$Gradient 항.
  - ones: 제약 조건 $1^T s - b = 0$ 표현을 위한 벡터.
  - KKT_matrix: 전체 선형 시스템 좌변 행렬
  - rhs: 우변 벡터
  - sol: $[\hat{s}; \lambda]$를 포함한 해.
  - s_opt: 최적 해 $\hat{s}$만 추출하여 반환.

■ 실행 및 결과 출력

```
52  # ------------------------------
53  # Step 3: 실행
54  # ------------------------------
55  R, v_des = advertising_budget_data()
56  b = 1300
57
58  s = solve_least_squares_with_constraint(R, v_des, b)
59
60  # 결과 출력
61  print("Optimal s:", s)
62  print("1^T s =", np.sum(s))
63
64  # RMS 오차 계산
65  m = R.shape[0]
66  rms = np.linalg.norm(R @ s - v_des) / np.sqrt(m)
67  print("RMS Error:", rms)
```

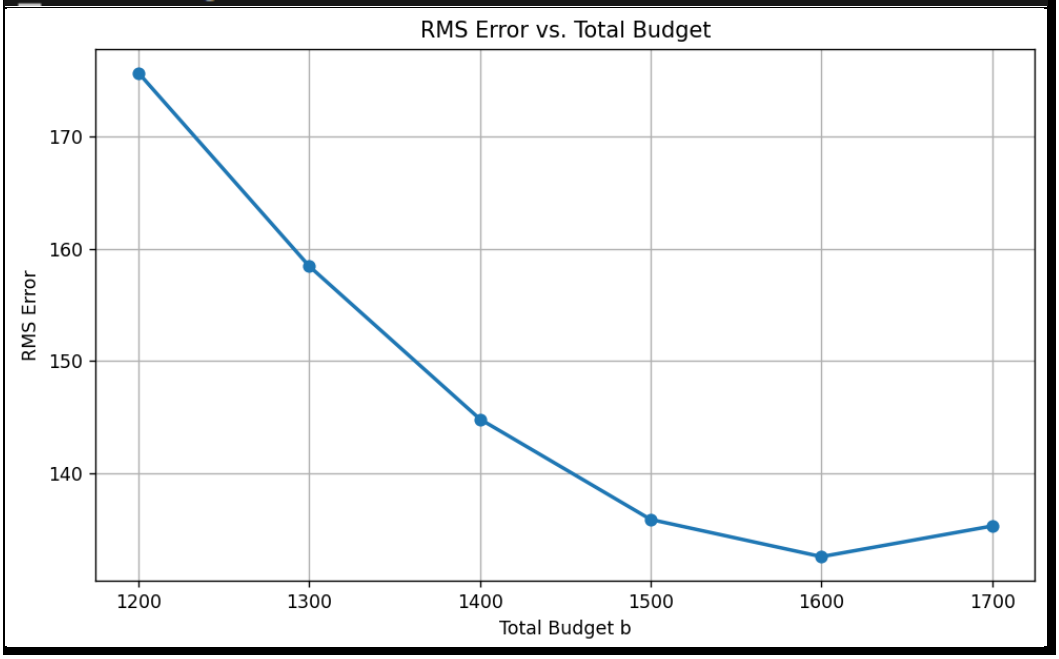
- 입력 데이터 로딩 및 예산 $b = 1300$ 설정.
- 함수 호출로 최적 광고비 분배 $\hat{s}$ 계산.
- 최종 목표 도달률과 실제 도달률 간의 RMS 오차 계산:

$$RMS = \frac{\|R\hat{s} - v_{des}\|}{\sqrt{m}}$$

4. Solution (c): 시뮬레이션 결과 및 그래프 해석

```
51  for b in budgets:
52      s = solve_least_squares_with_constraint(R, v_des, b)
53      rms = np.linalg.norm(R @ s - v_des) / np.sqrt(m)
54      rms_list.append(rms)
55      print(f"b = {b}, RMS = {rms:.4f}")
```

```
b = 1200, RMS = 175.5962
b = 1300, RMS = 158.4337
b = 1400, RMS = 144.8577
b = 1500, RMS = 135.9472
b = 1600, RMS = 132.6455
b = 1700, RMS = 135.3638
```



문제 6-(c)에서는 총 광고 예산 bbb 의 값을 1200 부터 1700 까지 100 단위로 변화시키면서, 각 예산에 대해 최소제곱 해 s^₩hat{s}s^를 구하고 실제 도달률과 목표 도달률 사이의 RMS 오차를 계산하였다. 실험 결과, 예산이 증가함에 따라 RMS 오차는 전반적으로 감소하는 경향을 보였다. 구체적으로 예산이 1200 일 때 RMS 오차는 약 175.60 으로 가장 컸으며, 예산을 1300, 1400, 1500 으로 순차적으로 늘릴수록 RMS 오차는 각각 약 158.43, 144.86, 135.95 로 점진적으로 줄어들었다. 이와 같은 경향은 더 많은 예산이 확보될수록 각 인구 집단의 도달 목표를 더 정확하게 맞출 수 있다는 것을 보여준다.

특히 예산이 1600 일 때 RMS 오차는 약 132.65 로 최저점을 기록하였고, 그 이후 1700 에서는 오히려 RMS 가 약간 상승하여 135.36 이 되었다. 이는 예산이 일정 수준 이상으로 증가했을 때, 도달률 개선 효과가 포화 상태에 이르며 일부 채널에 과도한 비용이 분배되어 비효율성이 발생할 수 있음을 시사한다. 결국 예산을 무작정 증가시키는 것이 항상 성능 향상으로 이어지지는 않으며, 주어진 문제에서는 예산이 1500~1600 사이일 때 도달률 정확도와 자원 효율성 사이의 균형이 가장 잘 맞는 것으로 해석할 수 있다.