선형대수 프로그래밍 과제 #1: Gauss-Jordan 소거법 - Forward Elimination

국민대학교 컴퓨터공학부
2014년 2학기

출제자: 김준호
조교: 김종훈

과제 설명:

Gauss-Jordan 소거법은 선형 시스템을 풀기 위한 대표적인 알고리즘으로 forward elimination과 backward elimination 두 단계로 구성되어 있다. 본 과제에서는 forward elimination을 Python 패키지인 NumPy를 통해 구현하는 것을 목표로 한다.

과제 목표:

풀고자 하는 $m \times n$ 선형 시스템이 $m \times (n+1)$ augmented matrix $\mathbf{M} = [\mathbf{A}|\mathbf{b}]$ 의 꼴로 주어졌다고 가정하자. 본 과제에서 작성하고자 하는 프로그램은 augmented matrix $\mathbf{M}$에 대해, forward elimination을 수행한 후, 그 결과에 대한 rank를 분석하여 원래의 선형 시스템이 해가 있는지(consistent), 그렇지 않은지(inconsistent)를 판단할 수 있어야 한다. 그리고, 해가 있다고 판단된 경우(consistent)에는 원래의 선형 시스템이 유일해(unique solution)를 가지는 경우인지, 무수히 많은 해(infinitely many solutions)를 가지는 경우인지를 판단할 수 있어야 한다.

과제의 목표를 만족하기 위해 다음의 세부 목표를 모두 만족하는 프로그램을 디자인 하도록 한다.

- 프로그램은 다음과 같은 형식으로 수행될 수 있어야 한다.
  - 실행 형식: 프로그램명 <input.txt> <output.txt>
  - 실행의 예: forward_elimination.py aug_mat_01.txt result_01.txt
- 입력 텍스트파일 <input.txt>는 우리가 풀고자 하는 $m \times n$ 선형 시스템에 대한 $m \times (n+1)$ augmented matrix $\mathbf{M} = [\mathbf{A}|\mathbf{b}]$를 가지고 있다고 가정한다.
- 입력으로 주어진 augmented matrix $\mathbf{M}$에 대해 forward elimination을 수행하여 행 사다리꼴 행렬(row echelon form)을 계산한다.
  - 단, forward elimination 과정에서 pivot 값은 반드시 1이 되도록 scaling을 수행한다.
- 행 사다리꼴 행렬에 대해, rank$\mathbf{A}$와 rank$\mathbf{M}$를 계산한다.
- 계산된 rank$\mathbf{A}$와 rank $\mathbf{M}$, 미지수의 개수 n을 이용하여, 주어진 $m \times n$ 선형 시스템이 ('Inconsistent', 'A unique solution', 'Infinity many solutions') 중 어느 경우에 해당하는지를 판단한다.
  - rank$\mathbf{A}$ < rank$\mathbf{M}$: Inconsistent
  - rank$\mathbf{A}$ = rank$\mathbf{M}$ = n : A unique solution
  - rank$\mathbf{A}$ = rank$\mathbf{M}$ < n : Infinitely many solutions
- 프로그램은 최종 결과를 결과 텍스트파일 <output.txt>에 저장한다. 저장 형식은 다음과 같다.
  - Rank 계산 결과
    - rank$\mathbf{A}$

- ◆ rank**M**
- ◆ n

■ Forward elimination 결과

- ◆ 'Inconsistent', 'A unique solution', 'Infinity many solutions' 중 하나

■ 행 사다리꼴 행렬 (row echelon form)

- [Figure 1]은 프로그램 실행 방법의 예를 보여주고 있다.
(단, [Figure 1]에서는 forward_elimination.py 파일과 sample_01.txt 파일이 모두 파이썬 인터프리터 python.exe가 있는 디렉토리 C:\Python34에 있는 경우를 보여줌. 출력 파일인 result_01.txt은 프로그램의 실행 결과로 생성되어야 함)

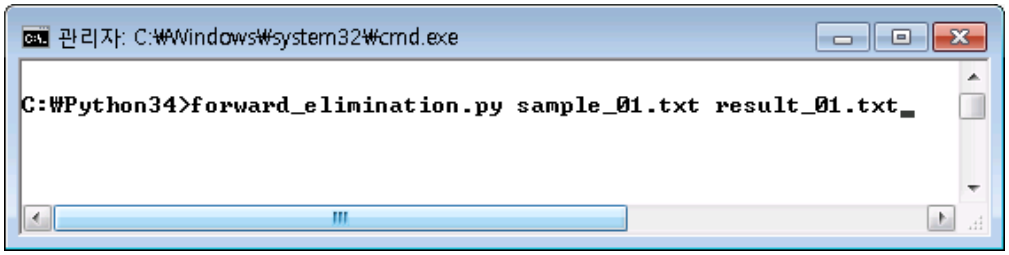

Figure 1. 프로그램 실행 방법

- [Figure 2]는 입력 텍스트파일과 그에 대한 출력 텍스트파일의 예를 보여주고 있다.

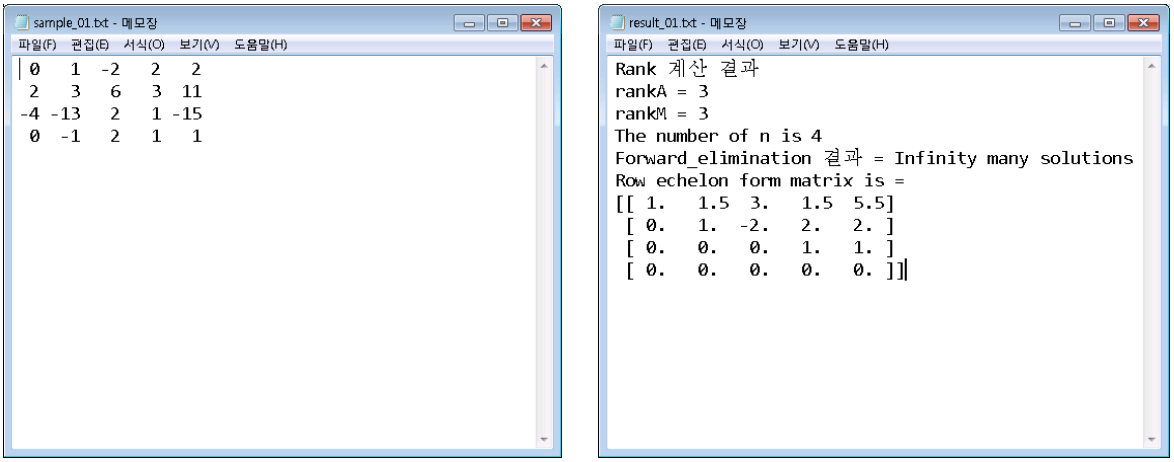

Figure 2. 프로그램의 입력값(왼쪽)과 출력값(오른쪽)

과제의 목표를 달성하도록 하는 프로그램의 개략적인 코드 전개는 [Table 1]과 같다. (**주의사항**: [Table 1]은 프로그램의 개략적인 뼈대만을 알려줄 뿐이며, 학생 개개인은 [Table 1]을 참조하여 과제의 요구사항 및 목표를 만족하도록 자신만의 코드를 구현해야 함.)

Table 1: 과제 #1의 개략적인 코드 전개

forward_elimination.py

```
##############################################
# 학과:
# 학번:
# 이름:
##############################################

# forward elimination
# 1. 초기 pivot의 위치는 [0, 0]으로 한다.
# 2. pivot의 위치를 재설정 한다.
#           주어진 augmented matrix M에 대하여 행과 열의 수는 m과 n+1이라고 한다.
#           2-1. 현재 pivot의 위치가 [i, j] (단, i < m, j < n)에 있다고 가정하자.
#            2-2. 만일 [i, j]의 값이 0이 아닌 값이면, 현재 pivot의 위치는 재설정할 필요가 없으므로, 3의 과정으로 간다.
#            2-3. 만일 [i, j]의 값이 0이면, pivot의 아래 행을 방문하며 [p, j] (단, p > i) 값을 검사한다.
#                 만일 [p, j]의 값이 0이 아닌 값이면 i행과 p행을 서로 교체(interchange)한다.
#            2-4. 만일 2-3 과정에서 [p, j]의 값이 0이 아닌 값을 못 찾았을 경우, pivot의 위치를 한 열 옆으로
#                 이동(j <- j+1) 한다.
#            2-5. 0값이 아닌 pivot의 위치를 찾을 때까지 2-1 ~ 2-4의 과정을 반복한다.
#                  만일, pivot의 위치가 행렬의 마지막 행과 끝에서 두 번째 열인 경우에는 주어진 행렬의 마지막 pivot이 되며,
#                  이후 다음 pivot은 설정하지 않고 forward elimination을 종료한다.
# 3. pivot의 값이 a라면 그 행의 모든 값에 1/a를 곱하여 scaling을 수행한다.
# 4. pivot의 위치보다 아래 행들의 같은 열의 값이 0이 되도록 replacement를 수행한다.
# 5. replacement 과정이 모두 끝나면 pivot의 위치를 다시 재설정 한다.
#    기존의 pivot위치에서 행은 한 행 아래로 내려가고, 열은 아래 행들 중에 왼쪽으로부터 모든 열의 값에서 하나라도 0이 아닌
곳으로 가도록 설정한다.
# 6. 모든 행렬이 행 사다리꼴(row echelon form)이 될 때까지 과정[2 ~ 5]를 반복한다.
def forward_elimination(matrix):
        ##############################################
        # 아래 라인을 지우고 forward elimination을 구현하세요.
        ##############################################
        pass

# r_i↔ r_j
def interchange(matrix, row_i, row_j):
        m, n = matrix.shape

        for i in range(0, n):
                matrix[row_i, i] ,matrix[row_j, i] = matrix[row_j, i] ,matrix[row_i, i]
        return matrix

# r_i ← cr_i
def scaling(matrix, row_i, c):
        matrix[row_i, :] *= c
        return matrix

# r_i ← r_i − mr_j
def replacement(matrix, row_i, row_j, m):
        matrix[row_i, :] = matrix[row_i,:] - m*matrix[row_j, :]
        return matrix

# 행렬의 한 요소 값이 절대값 0.00001 보다 작으면 0과 가까운 숫자라고 판단한다.
def is_nearzero(value, tol = 1e-05):
        if(abs(value) < tol):
                return True
        else:
                return False

# i번째 행(i-th row)의 모든 요소가 0인지 검사한다.
def is_zerorow(matrix, ith_row):
        m, n = matrix.shape
        for r in range(0, n):
```

```
                        if(not is_nearzero(matrix[ith_row, r])):
                                return False
        return True


# forward elimination이 적용된 사다리꼴 행렬(row echelon form)을 이용하여 system matrix A와 augmented matrix M의 rank를 구한다.
def rank(aug_mat):
        m, n = aug_mat.shape
        rank_A = 0
        rank_M = 0

        sys_mat = aug_mat[:, 0:n-1]

        #System matrix A의 rank 측정.
        for r in range(0, m):
                if (not is_zerorow(sys_mat, r)):
                        rank_A +=1
        #Augmented matrix M의 rank 측정.
        for r in range(0, m):
                if (not is_zerorow(aug_mat, r)):
                        rank_M +=1

        return rank_A, rank_M


################################################
# 프로그램 메인 파트
################################################
import numpy, sys

input_file = sys.argv[1]
output_file = sys.argv[2]

aug_matrix = numpy.mat(numpy.loadtxt(input_file))

# 입력 augmented matrix (aug_matrix)는
# 아래의 forward_elimination() 함수를 수행한 후,
# row echelon form으로 바뀜.
forward_elimination(aug_matrix)

m, n = aug_matrix.shape
rank_A, rank_M = rank(aug_matrix)
################################################
# rank_A와 rank_M을 이용해서 주어진 선형 시스템이
# ('Inconsistent', 'A unique solution', 'Infinity many solutions') 중
# 어느 경우에 해당하는지를 판단한다.
#
# Augmented Matrix(M) = [A|b]
# rank_A < rank_M then, Linear system is inconsistent
# rank_A = rank_M = n, Linear system has a unique solution
# rank_A = rank_M < n, Linear system has infinity many solutions
#
# 아래 라인을 지우고 위의 내용을 구현하세요.
################################################
pass

################################################
# 출력파일의 형식을 맞추기 위해서 다음 코드는 절대로 건들지 마세요.
################################################
f = open(output_file, 'a')
f.write("Rank 계산 결과\n")
f.write("rankA = ")
f.write(str(rank_A))
f.write("\nrankM = ")
```

```
f.write(str(rank_M))
f.write("\nThe number of n is ")
f.write(str(n))
f.write("\nForward_elimination 결과 = ")
f.write(str(FE_result)) # FE_result는 문자열로써, 'Inconsistent', 'A unique solution', 'Infinity many solutions' 중 하나임.
f.write("\nRow echelon form matrix is = \n")
f.write(str(aug_matrix))
f.close()
################################################
```

참고할 정보:

- Python, NumPy에 대한 개략적인 사용법 소개는 [가상강의실]-[자료실]에 업로드 되어 있는 PDF파일(NumPy_Introduction.pdf)을 참고한다.
- 테스트를 하기 위한 입력 텍스트 파일(input_OO.txt)과 그에 따른 계산결과 파일(output_OO.txt)은 가상강의실에 있으니 디버깅 시 참고한다.
- Python, NumPy에 대해 더 자세히 알고 싶은 경우, 아래 사이트를 참고한다.
  - http://wiki.scipy.org/Tentative_NumPy_Tutorial/
  - https://wikidocs.net/book/1/
  - http://www.numpy.org/
  - https://www.python.org/

과제 수행 시 주의사항:

본 과제를 성공적으로 수행하기 위해 아래의 요소들을 고려해야 한다.

- **학과, 학번은 반드시 소스 상단에 주석으로 표기하도록 한다.**
- 반드시 **Python 3.0 이상의 버전**을 사용하도록 한다.
- forward_elimination.py 안의 소스를 완벽히 구현한다.
  - forward_elimination(): 가우스 소거법 전체 알고리즘을 구현한다.

    1. 초기 pivot의 위치는 [0, 0]으로 한다.

    2. pivot의 위치를 재설정 한다.

       주어진 augmented matrix **M**에 대하여 행과 열의 수는 **m**과 **n+1**이라고 한다.

       2-1. 현재 pivot의 위치가 [i, j] (단, i < m, j < n)에 있다고 가정하자.

       2-2. 만일 [i, j]의 값이 0이 아닌 값이면, 현재 pivot의 위치는 재설정할 필요가 없으므로, 3의 과정으로 간다.

       2-3. 만일 [i, j]의 값이 0이면, pivot의 아래 행을 방문하며 [p, j] (단, p > i)값을 검사한다. 만일 [p, j]의 값이 0이 아닌 값이면 i행과 p행을 서로 교체 (interchange)한다.

       2-4. 만일 2-3 과정에서 [p, j]의 값이 0이 아닌 값을 못 찾았을 경우, pivot의 위치를 한 열 옆으로 이동(j <- j + 1) 한다.

2-5. 0값이 아닌 pivot의 위치를 찾을 때까지 2-1 ~ 2-4과정을 반복한다. 만일, pivot의 위치가 행렬의 마지막 행과 끝에서 두 번째 열인 경우에는 주어진 행렬의 마지막 pivot이 되며, 이후 다음 pivot은 설정하지 않고 forward elimination을 종료하도록 한다.

3. pivot의 값이 a라면 그 행의 모든 값에 1/a를 곱하여 scaling을 수행한다.

4. pivot보다 아래 행들의 같은 열의 값이 0이 되도록 replacement를 수행한다.

5. replacement 과정이 모두 끝나면 pivot의 위치를 다시 재설정 한다. 기존의 pivot위치에서 행은 한 줄 아래로 내려가고, 열은 아래 행들 중 왼쪽으로부터 모든 열의 값에서 하나라도 0이 아닌 곳으로 가도록 설정한다.

6. 모든 행렬이 행 사다리꼴(row echelon form)이 될 때까지 과정[2 ~ 5]를 반복한다.

- interchange(): 두 행을 교환한다.
- scaling(): 0이 아닌 상수를 한 행에 곱한다.
- replacement(): *i*-번째 행에서 상수 *m*배한 *j*-번째 행을 뺀 결과를 새로운 형태의 i-번째 행으로 교체한다.
- is_nearzero(): 행렬의 한 요소 값이 0과 가까운 숫자인지 판단한다.
- is_zerorow(): 주어진 행의 모든 요소가 0에 가까운 숫자로 이루어졌는지 검사한다.
- rank(): forward elimination이 적용된 사다리꼴 행렬(row echelon form)을 이용하여 system matrix **A**와 augmented matrix **M**의 rank를 구한다.

- 이 파일은 **10월 7일 오후 6시 이후로 업데이트** 되었으니, 만약 업데이트 하기 전의 파일을 받았던 학생은 전 파일을 폐기 처분해 주기 바람.
- forward_elimination.py의 소스를 표준압축파일형식(.zip)으로 압축하여 제출한다.
  - 제출형식: 학번_이름.zip
- **마감일은 2014년 10월 17일(금) 자정(23:59)이며, 가상대학을 통해 제출하도록 한다.**