선형대수 프로그래밍 과제 #2: Gauss-Jordan 소거법을 이용하여 역행렬 구하기

국민대학교 컴퓨터공학부
2014년 2학기

출제자: 김준호
조교: 김종훈

과제 설명:

본 과제에서는 Gauss-Jordan 소거법을 이용하여 입력으로 주어진 $n \times n$ 정사각행렬 (square matrix) $\mathbf{A}$에 대한 역행렬(inverse matrix)을 구하는 프로그램을 작성한다.

과제 #1에서 이미 구현한 forward elimination을 재활용하고, 본 과제를 통해 backward elimination을 추가적으로 구현함으로써 Gauss-Jordan elimination을 완성하도록 한다. 완성된 Gauss-Jordan elimination을 활용하여 주어진 $n \times n$ 정사각행렬 $\mathbf{A}$에 대한 역행렬을 구하도록 하자.

과제 목표:

입력으로 $n \times n$ 정사각행렬 $\mathbf{A}$가 주어졌다고 가정하자. 본 과제에서 작성하고자 하는 프로그램은 입력으로 들어온 정사각행렬 $\mathbf{A}$로부터 augmented matrix $\mathbf{M} = [\mathbf{A}|\mathbf{I}]$을 만든 후, augmented matrix $\mathbf{M}$에 대한 Gauss-Jordan elimination을 수행하여, 입력으로 들어왔던 행렬 $\mathbf{A}$에 대한 역행렬의 존재 여부를 판단하고, 역행렬이 존재한다면 역행렬 $\mathbf{A}^{-1}$를 계산할 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: matrix_inversion.py mat_01.txt result_01.txt
- 입력 텍스트파일 <input.txt>은 $n \times n$ 정사각행렬 $\mathbf{A}$를 가지고 있다고 가정한다.
- 입력으로 주어진 $n \times n$ 정사각행렬 $\mathbf{A}$로부터 augmented matrix $\mathbf{M} = [\mathbf{A}|\mathbf{I}]$를 생성한다.
- 생성된 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 $\mathbf{A}$를 계산한다.
- 계산된 rank $\mathbf{A}$와 n을 이용하여, 주어진 $n \times n$ 정사각행렬 $\mathbf{A}$가 역행렬을 가질 수 있는지 (Invertible) 아니면 역행렬을 가질 수 없는지(Singular) 여부를 판단한다.
  - rank $\mathbf{A}$ = n Invertible
  - rank $\mathbf{A}$ < n Singular
- $n \times n$ 정사각행렬 $\mathbf{A}$가 역행렬을 가질 수 있다면, 현재 augmented matrix $\mathbf{M}$에 대해 backward elimination을 수행하여 $\mathbf{A}$의 역행렬인 $\mathbf{A}^{-1}$를 구한다.
  - Backward elimination을 통해 augmented matrix는 최종적으로 $[\mathbf{I}|\mathbf{A}^{-1}]$의 꼴을 가지게

된다.

- 프로그램은 최종 결과를 텍스트파일 <output.txt>에 저장한다. 저장 형식은 다음과 같다.
  - 역행렬 가능 유무
    - 'Invertible', 'Singular' 중 하나
  - 역행렬($\mathbf{A^{-1}}$: 입력 행렬 $\mathbf{A}$의 역행렬)
- [Figure 1]은 프로그램 실행 방법의 예를 보여주고 있다.
  (단, [Figure 1]에서는 matrix_inversion.py 파일과 mat_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)

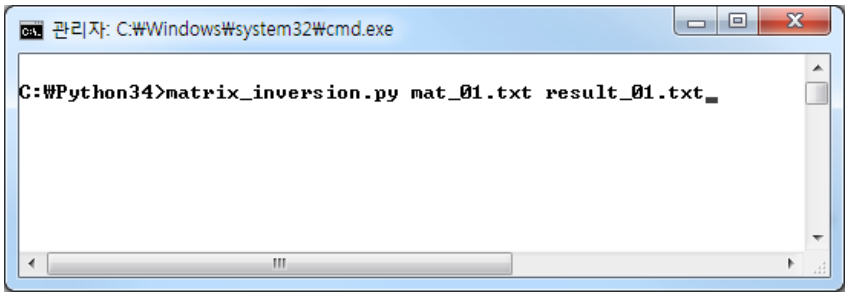

Figure 1. 프로그램 실행 방법

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

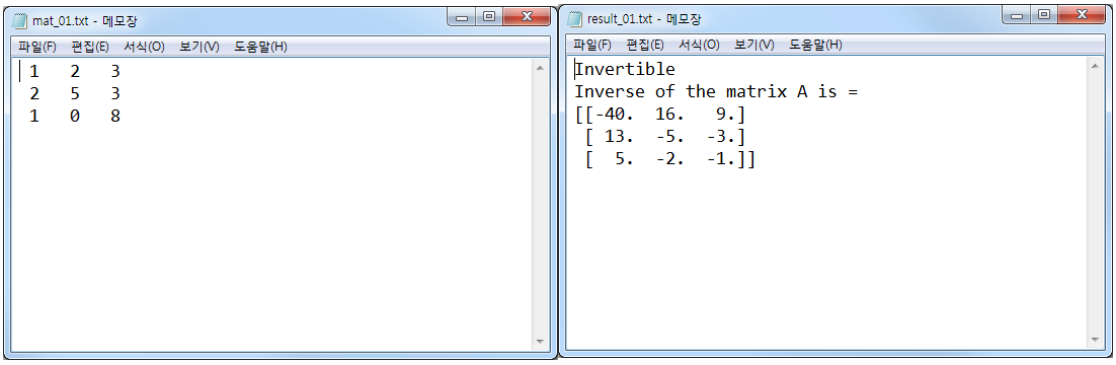

Figure 2. 프로그램의 입력값(왼쪽)과 출력값(오른쪽)

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 #2의 개략적인 코드 전개

matrix_inversion.py

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

# 주어진 n × n 정사각행렬 A로부터 augmented matrix M = [A|I]를 생성한다.
# 생성된 augmented matrix M에 대해 forward elimination을 수행하여 행 사다리꼴 행렬(row echelon form)을 계산한다.
# 행 사다리꼴 행렬에 대해, rank A를 계산한다.
# 계산된 rank A와 n을 이용하여, 주어진 n × n 정사각행렬 A가 역행렬을 가질 수 있는지(Invertible) 아니면 역행렬을 가질 수 없는지
(Singular) 여부를 판단한다.
# 정사각행렬 A가 역행렬을 가질 수 있다면, 현재 augmented matrix M에 대해 backward elimination을 수행한다.
# backward elimination을 수행하여 얻은 [I|A^-1]에서 A의 역행렬인 A^-1을 구한다.
def matrix_inversion(matrix):
        m, n  = matrix.shape

        aug_mat = numpy.mat(adjoin_I_to_A(matrix))

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

        rank_A = rank(aug_mat[0:n, 0:n])

        if (rank_A == n):
                result = "Invertible"
                backward_elimination(aug_mat)
                matrix = numpy.mat(readoff_inv_A(aug_mat))
        elif(rank_A < n):
                result = "Singular"

        return result, matrix

# n × n square matrix A 의 오른쪽에 같은 크기의 identity matrix I 을 인접하여 [A|I] 형태로 만든다.
# 인접한 [A|I] 행렬의 행의 수는 n, 열의 수는 2n.
def adjoin_I_to_A(matrix):
        m, n = matrix.shape
        identity_matrix = numpy.mat(numpy.identity(n))
        aug_mat = numpy.mat(numpy.arange(n*n*2).reshape((n, n*2)) , dtype = float)
        aug_mat[0:n,0:n] = matrix
        aug_mat[0:n,n:n*2] = identity_matrix

        return aug_mat

# Gauss-Jordan 소거법이 적용된 [I|A^-1]에서 A^-1을 구한다.
def readoff_inv_A(aug_mat):
        m, n = aug_mat.shape

        return aug_mat[0:m, m:m*2]

# forward elimination
def forward_elimination(aug_mat):
        ##########################################################
        # 아래 라인을 지우고 forward elimination을 구현하세요.
        ##########################################################
        pass
```

```
# backward elimination
def backward_elimination(aug_mat):
        ###########################################################
        # 아래 라인을 지우고 backward 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

# 주어진 matrix에 대한 rank를 구하는 알고리즘
def rank(matrix):
        m, n = matrix.shape
        rank = 0

        # 주어진 matrix에 대하여 rank 측정.
        for r in range(0, m):
                if (not is_zerorow(matrix, r)):
                        rank +=1

        return rank


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

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

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

# 입력 square matrix A는
# matrix_inversion()함수를 수행한 후,
```

```
# result('Invertible', 'Singular' 중 하나)와 A의 역행렬인 A^-1를 반환한다.
result, inverse_matrix = matrix_inversion(square_matrix)

##############################################
# 출력파일의 형식을 맞추기 위해서 다음 코드는 절대로 건들지 마세요.
##############################################
f = open(output_file, 'w')
f.write(result)
if(result=="Invertible"):
        f.write("\nInverse of the matrix A is = \n")
        f.write(str(inverse_matrix))
f.close()
##############################################
```

참고할 정보:

- 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 대한 개략적인 사용법 소개는 [가상강의실]-[자료실]에 업로드 되어 있는 PDF파일(NumPy_Introduction.pdf)을 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.
- matrix_inversion.py 안의 소스를 완벽히 구현한다.
- matrix_inversion.py의 소스를 표준압축파일형식(.zip)으로 압축하여 제출한다.
  - 제출형식: 학번_이름.zip
- **마감일은 2014년 11월 11일(화) 자정(23:59)이며, 가상대학을 통해 제출하도록 한다.**