선형대수 HW #3

2017027265 여주안

1. Code Summary

전체 코드는 4개의 메서드로 구성하였고 언어는 Python, IDE는 Jupyter Notebook을 사용하였다.
Grim-Schmidt 방법에서 반복되어 사용되는 벡터 연산을 위해 NumPy 라이브러리를 활용했다.

Method 1: **gram_schmidt_method(A)**

Gram-Schmidt 방법으로
Orthonormal Basis (Q) 계산

Method 2: **prove_orthonormality(Q)**

계산한 벡터들의 orthonormality 검증

Method 3: **run_gram_schmidt()**

사용자에게 입력을 받고
Method 1, 2를 차례로 시행

Method 4: **print_result(Q, orthonormality)**

프로그램 수행 결과 출력

[프로그램 실행 예시]

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 3
enter the Number of Vectors: 3
enter the Vector 1: 1 0 1
enter the Vector 2: 1 0 0
enter the Vector 3: 2 1 0

Orthonormal Basis Vectors
q1: [0.707 0.0 0.707]
q2: [0.707 0.0 -0.707]
q3: [0.0 1.0 -0.0]

Orthonormality Test Result: PASS
```

1. Code Summary

gram_schmidt_method(A) 메서드는 주어진 행렬(A)의 벡터들에 대하여 $A_n = a_n - \sum_{i=1}^{n-1}(q_i \cdot a_n)q_i$ 를 계산하고 최종적으로 $q_n = A_n \div \|A_n\|$ 을 구하여 Orthonormal Basis, Q 형태로 반환한다

np.dot : 내적 값 계산
np.square : 각 요소 제곱
np.sum : 모든 요소 총합
np.sqrt : 제곱근 값 계산

In [1]:
```
import numpy as np

A = [[1, 0, 1], [1, 0, 0], [2, 1, 0]]

def gram_schmidt_method(A):
    A = np.array(A, dtype='f')
    Q = []

    for i in range(len(A)):
        A_n = A[i]
        for j in range(i):
            A_n -= (Q[j].dot(A[i])) * Q[j]
        volume_An = np.sqrt(np.sum(np.square(A_n)))
        q_n = A_n / volume_An
        Q.append(q_n)

    Q = np.round(Q, 3)
    return Q

Q = gram_schmidt_method(A)
print(Q)
```

```
[[ 0.707  0.     0.707]
 [ 0.707  0.    -0.707]
 [ 0.     1.    -0.   ]]
```

$$A_n = a_n - \sum_{i=1}^{n-1}(q_i \cdot a_n)q_i$$
$$(A_1 = a_1)$$

$$q_n = A_n \div \|A_n\|$$

결과: Orthonormal Basis, Q를 반환

1. Code Summary

prove_orthonormality(Q) 메서드는 주어진 행렬(Q)의 orthonormality를 검증한다. Orthonormality 의 두가지 조건 중 하나라도 만족하지 않는다면 False를 반환한다

In [2]:
```
def prove_orthonormality(Q):
    orthonormality = True
    for i in range(len(Q)):
        volume_Qn = np.sum(np.square(Q[i]))
        if round(volume_Qn, 1) != 1:
            orthonormality = False

        for j in range(i):
            inner_product = Q[i].dot(Q[j])
            if round(inner_product, 1) != 0:
                orthonormality = False

    return orthonormality

print(prove_orthonormality(Q))
```

```
True
```

Orthonormality 조건 1:
벡터의 크기가 1이다

Orthonormality 조건 2:
나머지 벡터와 모두 수직이다 (내적 값이 0이다)

결과: True – Orthonormality 검증 성공
False – Orthonormality 검증 실패

1. Code Summary

run_gram_schmidt() 메서드는 사용자에게 Vector Dimension (n), Number of Vectors (m), Vectors(a) 를 입력받아 메서드 1, 2를 차례로 실행한다.

```
In [3]: def run_gram_schmidt():
            n = int(input('enter the Number of Vector Dimension: '))
            m = int(input('enter the Number of Vectors: '))
            i = m
            A = []
            while m > 0:
                v = list(input('enter the Vector ' + str(i - m + 1) + ": ").split())
                if len(v) == n:
                    A.append(v)
                    m -= 1
                else:
                    print('ERROR: wrong vector dimension - Please enter the correct vector...')

            Q = gram_schmidt_method(A)
            orthonormality = prove_orthonormality(Q)
            print_result(Q, orthonormality)
```

[사용자 입력 예시]

```
enter the Number of Vector Dimension: 3
enter the Number of Vectors: 3
enter the Vector 1: 1 0 0 0
ERROR: wrong vector dimension - Please enter the correct vector...

enter the Vector 1: 1 0 0
```

[사용자 입력 완료 예시]

```
enter the Number of Vector Dimension: 3
enter the Number of Vectors: 3
enter the Vector 1: 1 0 1
enter the Vector 2: 1 0 0
enter the Vector 3: 2 1 0
```

1. Code Summary

print_result() 메서드는 프로그램이 계산한 Orthonormal Basis 벡터를 차례로 출력하고 Orthonormality 검증 결과를 PASS / FAIL 로 보여준다.

In [3]:
```
def run_gram_schmidt():
    n = int(input('enter the Number of Vector Dimension: '))
    m = int(input('enter the Number of Vectors: '))
    i = m
    A = []
    while m > 0:
        v = list(input('enter the Vector ' + str(i - m + 1) + ": ").split())
        if len(v) == n:
            A.append(v)
            m -= 1
        else:
            print('ERROR: wrong vector dimension - Please enter the correct vector...')

    Q = gram_schmidt_method(A)
    orthonormality = prove_orthonormality(Q)
    print_result(Q, orthonormality)

def print_result(Q, orthonormality):
    print('\nOrthonormal Basis Vectors')
    for n in range(len(Q)):
        print('q' + str(n+1) + ': [' + ' '.join(str(e) for e in Q[n]) + ']')
    if orthonormality:
        print('\nOrthonormality Test Result: PASS')
    else:
        print('\nOrthonormality Test Result: FAIL')

run_gram_schmidt()
```

[프로그램 출력 예시]

```
Orthonormal Basis Vectors
q1: [1.0 0.0 0.0]
q2: [0.0 0.0 1.0]
q3: [0.0 1.0 0.0]

Orthonormality Test Result: PASS
```

1. Code Summary

- Orthonormal Basis를 계산할 때 float 형식을 어떻게 변환하여 사용할지 기준을 세우는 것이 중요하다. 이 프로그램에서는 Basis Vector 를 계산할 때는 소수점 셋째 자리까지, 0 과 1 같은 정수와 일치 여부를 판단할 때는 소수점 첫째 자리까지 반올림하였다.
- 앞에서 언급했듯이 반복적인 벡터 연산에는 NumPy 메서드들을 활용하여 코드 간결성을 유지하고 가독성을 높였다.
- 프로그램의 input 은 Vector Dimension (n), Number of Vectors (m), Vectors(a – 사용자가 입력한 숫자 m 만큼 입력을 요구함) 이며 output 은 Orthonormal Basis Vectors, Orthonormality 검증 결과이다.
- Orthonormality는 prove_orthonormality 메서드에 의해 판정되며 모든 벡터의 크기가 1이고 모든 벡터가 서로 수직일 때 True 를 반환한다

2. Execution (example 1)

프로그램을 실행한다. 첫번째 예시는 교재에 나오는 예 [1, 0, 0], [1, 0, 0], [2, 1, 0] 을 입력했다.

[실행 결과]

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 3
enter the Number of Vectors: 3
enter the Vector 1: 1 0 1
enter the Vector 2: 1 0 0
enter the Vector 3: 2 1 0

Orthonormal Basis Vectors
q1: [0.707 0.0 0.707]
q2: [0.707 0.0 -0.707]
q3: [0.0 1.0 -0.0]

Orthonormality Test Result: PASS
```

Orthonormal Basis Vectors

q1 = $[\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}]$, q2 = $[\frac{\sqrt{2}}{2}, 0, -\frac{\sqrt{2}}{2}]$, q3 = $[0, 1, 0]$

Prove Orthonormality

Orthonormality가 입증되었다

2. Execution (example 1)

프로그램이 성공적으로 작동했는지 확인한다.

[실행 결과]

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 3
enter the Number of Vectors: 3
enter the Vector 1: 1 0 1
enter the Vector 2: 1 0 0
enter the Vector 3: 2 1 0

Orthonormal Basis Vectors
q1: [0.707 0.0 0.707]
q2: [0.707 0.0 -0.707]
q3: [0.0 1.0 -0.0]

Orthonormality Test Result: PASS
```

입력한 벡터는 계산한 Orthonormal Basis 벡터의 Linear Combination 으로 표현할 수 있다

[1, 0, 1] = 1.414 * q1
[1, 0, 0] = 0.707 * q1 + 0.707 * q2
[2, 1, 0] = 1.414 * q1 + 1.414 * q2 + q3

Orthonormal Basis 벡터의 크기는 1이다

$$\|q1\|^2 = (0.707)^2 + (0)^2 + (0.707)^2 = 1$$
$$\|q2\|^2 = (0.707)^2 + (0)^2 + (-0.707)^2 = 1$$
$$\|q3\|^2 = (0)^2 + (1)^2 + (0)^2 = 1$$

Orthonormal Basis 벡터는 서로 모두 수직이다

$$\mathrm{q}1 \cdot q2 = (0.707)(0.707) + (0)(0) + (0.707)(-0.707) = 0$$
$$\mathrm{q}1 \cdot q3 = (0.707)(0) + (0)(1) + (0.707)(0) = 0$$
$$\mathrm{q}2 \cdot q3 = (0.707)(0) + (0)(1) + (-0.707)(0) = 0$$

3. Execution (example 2)

프로그램을 실행한다. 두번째 예시는 벡터의 차원을 4로 설정하여 [1, 1, 1, 1], [-1, 4, 4, -1], [4, -2, 2, 0], [2, 2, 0, 0] 을 입력한다

[실행 결과]

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 4
enter the Number of Vectors: 4
enter the Vector 1: 1 1 1 1
enter the Vector 2: -1 4 4 -1
enter the Vector 3: 4 -2 2 0
enter the Vector 4: 2 2 0 0

Orthonormal Basis Vectors
q1: [0.5 0.5 0.5 0.5]
q2: [-0.5 0.5 0.5 -0.5]
q3: [0.5 -0.5 0.5 -0.5]
q4: [0.5 0.5 -0.5 -0.5]

Orthonormality Test Result: PASS
```

벡터의 차원을 4로 설정

Orthonormal Basis Vectors

q1 = $[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$, q2 = $[-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}]$, q3 = $[\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}]$,
q4 = $[\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}]$

Prove Orthonormality

Orthonormality가 입증되었다

3. Execution (example 2)

프로그램이 성공적으로 작동했는지 확인한다.

[실행 결과]

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 4
enter the Number of Vectors: 4
enter the Vector 1: 1 1 1 1
enter the Vector 2: -1 4 4 -1
enter the Vector 3: 4 -2 2 0
enter the Vector 4: 2 2 0 0

Orthonormal Basis Vectors
q1: [0.5 0.5 0.5 0.5]
q2: [-0.5 0.5 0.5 -0.5]
q3: [0.5 -0.5 0.5 -0.5]
q4: [0.5 0.5 -0.5 -0.5]

Orthonormality Test Result: PASS
```

입력한 벡터는 계산한 Orthonormal Basis 벡터의 Linear Combination 으로 표현할 수 있다

[1, 1, 1, 1] = 2 * q1
[-1, 4, 4, -1] = 3 * q1 + 5 * q2
[4, -2, 2, 0] = 2 * q1 - 2 * q2 + 4 * q3
[2, 2, 0, 0] = 2 * q1 + 2 * q4

Orthonormal Basis 벡터의 크기는 1이다

$$\|q1\|^2 = (0.5)^2 + (0.5)^2 + (0.5)^2 + (0.5)^2 = 1$$
$$\|q2\|^2 = (-0.5)^2 + (0.5)^2 + (0.5)^2 + (-0.5)^2 = 1$$
$$\|q3\|^2 = (0.5)^2 + (-0.5)^2 + (0.5)^2 + (-0.5)^2 = 1$$
$$\|q4\|^2 = (0.5)^2 + (0.5)^2 + (-0.5)^2 + (-0.5)^2 = 1$$

Orthonormal Basis 벡터는 서로 모두 수직이다

$$\mathrm{q1} \cdot q2 = (0.5)(-0.5) + (0.5)(0.5) + (0.5)(0.5) + (0.5)(-0.5) = 0$$
$$\mathrm{q1} \cdot q3 = (0.5)(0.5) + (0.5)(-0.5) + (0.5)(0.5) + (0.5)(-0.5) = 0$$
$$\mathrm{q1} \cdot q4 = (0.5)(0.5) + (0.5)(0.5) + (0.5)(-0.5) + (0.5)(-0.5) = 0$$
$$\mathrm{q2} \cdot q3 = (-0.5)(0.5) + (0.5)(-0.5) + (0.5)(0.5) + (-0.5)(-0.5) = 0$$
$$\mathrm{q2} \cdot q4 = (-0.5)(0.5) + (0.5)(0.5) + (0.5)(-0.5) + (-0.5)(-0.5) = 0$$
$$\mathrm{q3} \cdot q4 = (0.5)(0.5) + (-0.5)(0.5) + (0.5)(-0.5) + (-0.5)(-0.5) = 0$$

4. Execution (example 3)

프로그램을 실행한다. 마지막 예시는 10 차원 벡터들의 Orthonormal Basis 를 프로그램으로 구해보았다.

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 10
enter the Number of Vectors: 10
enter the Vector 1: 1 0 2 4 0 0 1 1 3 3
enter the Vector 2: 2 0 3 1 4 0 0 0 1 2
enter the Vector 3: 1 1 1 0 2 1 3 0 0 0
enter the Vector 4: 1 0 0 0 0 1 1 0 0 0
enter the Vector 5: 4 2 1 2 2 0 3 1 3 2
enter the Vector 6: 1 3 0 3 2 4 0 2 1 1
enter the Vector 7: 2 2 2 2 0 0 1 1 1 3
enter the Vector 8: 1 0 0 3 0 4 1 1 2 4
enter the Vector 9: 3 1 0 2 0 1 0 0 0 0
enter the Vector 10: 1 1 1 1 1 1 3 1 1 2

Orthonormal Basis Vectors
q1: [0.156 0.0 0.312 0.625 0.0 0.0 0.156 0.156 0.469 0.469]
q2: [0.302 0.0 0.401 -0.213 0.812 0.0 -0.104 -0.104 -0.109 0.094]
q3: [0.07 0.288 -0.029 -0.045 0.104 0.288 0.882 0.018 -0.063 -0.181]
q4: [0.699 -0.31 -0.122 -0.084 -0.252 0.559 -0.107 -0.044 -0.04 0.052]
q5: [0.507 0.452 -0.412 -0.164 0.032 -0.455 -0.052 0.132 0.337 -0.039]
q6: [-0.166 0.515 -0.223 0.264 0.231 0.543 -0.377 0.301 -0.059 -0.064]
q7: [0.115 0.473 0.357 -0.17 -0.38 -0.08 -0.035 0.057 -0.503 0.445]
q8: [-0.259 -0.084 -0.45 -0.34 0.161 0.153 0.106 -0.123 0.126 0.719]
q9: [0.137 0.052 -0.344 0.541 0.124 -0.16 0.022 -0.587 -0.42 0.071]
q10: [0.088 -0.343 -0.24 0.149 0.16 -0.22 0.132 0.701 -0.449 0.102]

Orthonormality Test Result: PASS
```

10 x 10 행렬을 input 으로 입력

Orthonormal Basis Vectors

q1 = [0.156, 0, 0.312, 0.625, 0, 0, 0.156, 0.156, 0.469, 0.469]

q2 = [0.302, 0, 0.401, -0.213, 0.812, 0, -0.104, -0.104, -0.109, 0.094]

q3 = [0.07, 0.288, -0.029, -0.045, 0.104, 0.288, 0.882, 0.018, -0.063, -0.181]

…

Prove Orthonormality

Orthonormality가 입증되었다

4. Execution (example 3)

프로그램이 성공적으로 작동했는지 확인한다.

```
run_gram_schmidt()
```

```
enter the Number of Vector Dimension: 10
enter the Number of Vectors: 10
enter the Vector 1: 1 0 2 4 0 0 1 1 3 3
enter the Vector 2: 2 0 3 1 4 0 0 0 1 2
enter the Vector 3: 1 1 1 0 2 1 3 0 0 0
enter the Vector 4: 1 0 0 0 0 1 1 0 0 0
enter the Vector 5: 4 2 1 2 2 0 3 1 3 2
enter the Vector 6: 1 3 0 3 2 4 0 2 1 1
enter the Vector 7: 2 2 2 2 0 0 1 1 1 3
enter the Vector 8: 1 0 0 3 0 4 1 1 2 4
enter the Vector 9: 3 1 0 2 0 1 0 0 0 0
enter the Vector 10: 1 1 1 1 1 1 3 1 1 2

Orthonormal Basis Vectors
q1: [0.156 0.0 0.312 0.625 0.0 0.0 0.156 0.156 0.469 0.469]
q2: [0.302 0.0 0.401 -0.213 0.812 0.0 -0.104 -0.104 -0.109 0.094]
q3: [0.07 0.288 -0.029 -0.045 0.104 0.288 0.882 0.018 -0.063 -0.181]
q4: [0.699 -0.31 -0.122 -0.084 -0.252 0.559 -0.107 -0.044 -0.04 0.052]
q5: [0.507 0.452 -0.412 -0.164 0.032 -0.455 -0.052 0.132 0.337 -0.039]
q6: [-0.166 0.515 -0.223 0.264 0.231 0.543 -0.377 0.301 -0.059 -0.064]
q7: [0.115 0.473 0.357 -0.17 -0.38 -0.08 -0.035 0.057 -0.503 0.445]
q8: [-0.259 -0.084 -0.45 -0.34 0.161 0.153 0.106 -0.123 0.126 0.719]
q9: [0.137 0.052 -0.344 0.541 0.124 -0.16 0.022 -0.587 -0.42 0.071]
q10: [0.088 -0.343 -0.24 0.149 0.16 -0.22 0.132 0.701 -0.449 0.102]

Orthonormality Test Result: PASS
```

입력한 벡터는 계산한 Orthonormal Basis 벡터의 Linear Combination 으로 표현할 수 있다

Vector 1 = 6.41 * q1
Vector 2 = 3.284 * q1 + 4.926 * q2
…

Orthonormal Basis 벡터의 크기는 1이다

$\|q1\|^2 = (0.156)^2 + (0)^2 + (0.312)^2 + (0.625)^2 + (0)^2 + (0)^2$
$+ (0.156)^2 + (0.156)^2 + (0.469)^2 + (0.469)^2 = 1$
$\|q2\|^2 = (0.302)^2 + (0)^2 + (0.401)^2 + (-0.213)^2 + (0.812)^2 + (0)^2$
$+ (-0.104)^2 + (-0.104)^2 + (-0.109)^2 + (0.094)^2 = 1$
…

Orthonormal Basis 벡터는 서로 모두 수직이다

$\mathrm{q1} \cdot q2$
$= (0.156)(0.302) + (0)(0) + (0.312)(0.401) + (0.625)(-0.213)$
$+ (0)(0.812) + (0)(0) + (0.156)(-0.104) + (0.156)(-0.104)$
$+ (0.469)(-0.109) + (0.469)(0.094) = 0$
…

5. Full Source Code

```
import numpy as np

def gram_schmidt_method(A):
    A = np.array(A, dtype='f')
    Q = []

    for i in range(len(A)):
        A_n = A[i]
        for j in range(i):
            A_n -= (Q[j].dot(A[i])) * Q[j]
        volume_An = np.sqrt(np.sum(np.square(A_n)))
        q_n = A_n / volume_An
        Q.append(q_n)

    Q = np.round(Q, 3)
    return Q

def prove_orthonormality(Q):
    orthonormality = True
    for i in range(len(Q)):
        volume_Qn = np.sum(np.square(Q[i]))
        if round(volume_Qn, 1) != 1:
            orthonormality = False

        for j in range(i):
            inner_product = Q[i].dot(Q[j])
            if round(inner_product, 1) != 0:
                orthonormality = False

    return orthonormality
```

5. Full Source Code

```python
def run_gram_schmidt():
    n = int(input('enter the Number of Vector Dimension: '))
    m = int(input('enter the Number of Vectors: '))
    i = m
    A = []
    while m > 0:
        v = list(input('enter the Vector ' + str(i - m + 1) + ": ").split())
        if len(v) == n:
            A.append(v)
            m -= 1
        else:
            print('ERROR: wrong vector dimension - Please enter the correct vector...')

    Q = gram_schmidt_method(A)
    orthonormality = prove_orthonormality(Q)
    print_result(Q, orthonormality)

def print_result(Q, orthonormality):
    print('\nOrthonormal Basis Vectors')
    for n in range(len(Q)):
        print('q' + str(n+1) + ': [' + ' '.join(str(e) for e in Q[n]) + ']')
    if orthonormality:
        print('\nOrthonormality Test Result: PASS')
    else:
        print('\nOrthonormality Test Result: FAIL')

run_gram_schmidt()
```