Lecture5. Transposes, Permutations, Spaces R^n

▼ **1. Permutations**

- Permutation matrix(치환행렬)는 row exchange를 수행하는 행렬이다.
- 소거 과정에서 pivot이 0인 경우에 row exchange가 필요하고 이때 Permutation matrix가 사용된다. 한번 or 여러번 필요할 수 있다.
- A=LU Decomposition에서 행교환이 필요한 경우 $PA = LU$가 된다. 이 식은 어떠한 invertible A에도 적용이 된다.
- Permutation matrix의 속성 두가지가 있다.
  - 모든 P행렬은 invertible하다. 역행렬이 존재한다는 뜻이다.
  - P행렬의 역행렬은 Transpose행렬과 같다. 즉 $P^{-1} = P^T, P^T P = I$를 만족한다.

Lecture4의 Permutation 참고

▼ **2. Transpose**(전치)

- 행렬의 전치는 column의 차원과 row의 차원이 바뀐 것으로 볼 수 있다. 이에 대한 정의를 다음과 같이 쓸 수 있다.

  $Transpose : (A^T)_{ij} = A_{ij}$
  $i : index\ of\ row$
  $j : index\ of\ column$

▼ **3. Symmetric matrix**(대칭행렬)

- Symmetric Matrix는 transpose를 해도 원래 행렬과 동일하다는 특징을 가지고 있다.

  $$A^T = A$$
  $$\begin{bmatrix} 3 & 1 & 7 \\ 1 & 2 & 9 \\ 7 & 9 & 4 \end{bmatrix}$$

- 대각 원소를 기준으로 Upper triangular와 Lower Triangular 부분이 같은 것을 알 수 있다. 그렇기 때문에 transpose를 해도 변함이 없다.
- 어떤 임의의 행렬과 그 행렬의 transpose 행렬을 곱하면 항상 Symmetric Matrix를 얻을 수 있다.

  $(R^T R)^T = R^T R^{TT} = R^T R$

$$\begin{bmatrix} 1 & 3 \\ 2 & 3 \\ 4 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 2 & 4 \\ 3 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 11 & 7 \\ 11 & 13 & 11 \\ 7 & 11 & 17 \end{bmatrix}$$

$$R^T R = S$$

▼ 4. Vector spaces

▼ 1) 2-dimensional vector space

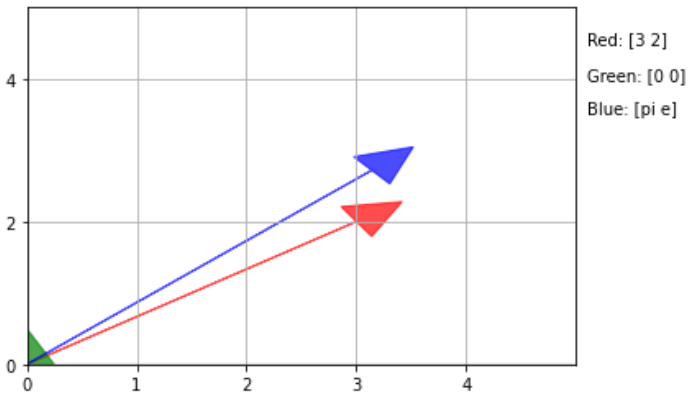
- 공간이란 다수의 벡터가 있고 벡터들이 모여 하나의 공간을 형성하는 것이다. 이때 Linear Combination 연산이 같은 공간상에 존재하는 벡터들이어야한다.
- x축,y축으로 이루어진 2차원 벡터공간을 다음과 같이 정의할 수 있다.

  $R^2 : all\ 2-dimentional\ real\ vectors$

  $: x-y\ plane$

  - R의 지수부분의 숫자는 차원을 의미한다.
- 다음과 같이 몇가지 예시를 생각해볼 수 있다.

  $\begin{bmatrix} 3 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \pi \\ e \end{bmatrix} \cdots$
- 이 예시들을 공간상에 표현해보면 다음과 같다.



- x축, y축은 2차원을 구성하는 구성요소이다. x,y 두가지 구성요소로 이루어진 벡터 공간을 x-y plane이라고 한다.
  - x-y plane을 2-dimensional vector space라고 하는 이유는 x, y로 구성된 모든 벡터들이 이 공간상에 존재할 수 있기 때문이다.
- 모든 차원의 벡터공간은 반드시 영벡터를 포함해야한다.

▼ 2) 3, and n-dimensional vector space

- 2차원 벡터의 구성요소에 비해 한개가 더 추가된 것이다. 기본적인 개념은 똑같다.

- 3개의 실수 구성요소(x, y, z)로 정의할 수 있는 모든 벡터들이 존재할 수 있는 공간이 3차원 공간이다.
- 3차원 벡터공간을 다음과 같이 정의할 수 있다.

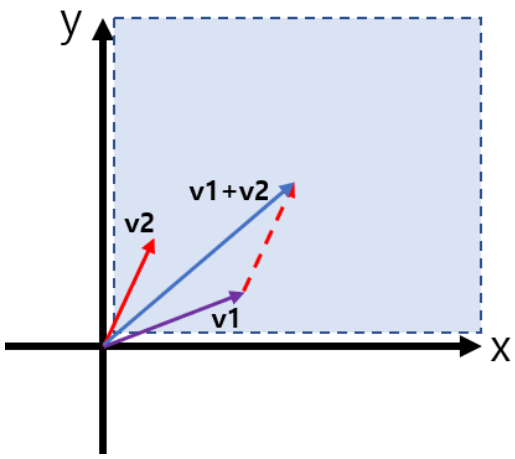
  $R^3 : \ all\ 3-dimentional\ real\ vectors\ with\ 3\ components$
- 벡터 공간을 n차원으로 확장하여 다음과 같이 정의할 수 있다.

  $R^n : \ all\ n-dimentional\ real\ vectors\ with\ n\ components$

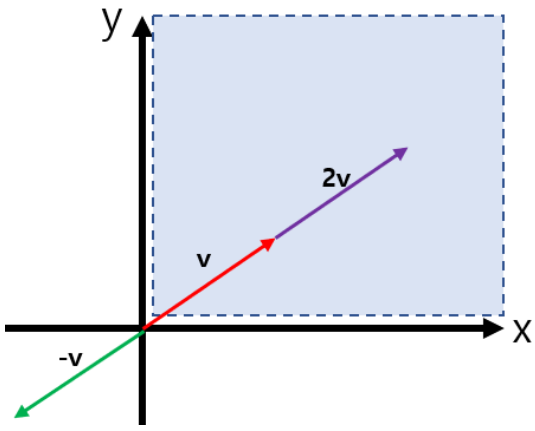
▼ **5. Subspace** (부분공간)

▼ **1) 벡터공간이 성립하지 않는 경우**

- 2차원 공간에서 1사분면만 취한다고 가정했을때 1사분면내에 어떠한 벡터끼리 덧셈을 해도 그 결과가 1사분면에 위치한다.



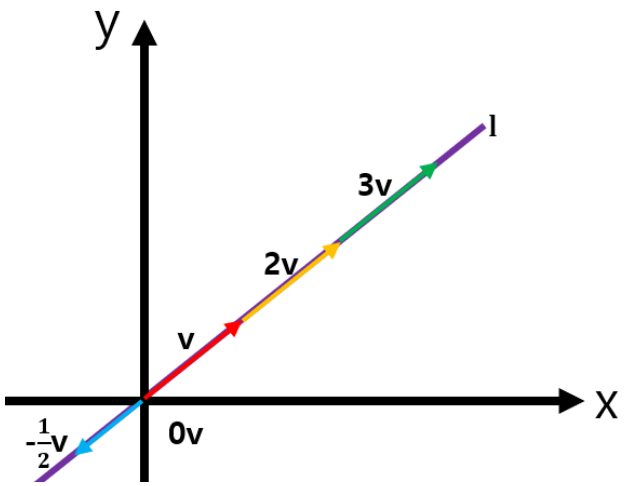
  - v1과 v2는 1사분면 공간 내의 원소들이기 때문에 x, y가 양의 값을 가진다. 따라서 이 공간 내의 어떤 벡터끼리 더해도 그 결과 벡터는 1사분면에 위치할 수밖에 없다.
  - 이러한 경우를 1사분면의 벡터 공간은 덧셈 연산에 대해 "닫혀있다"라고 한다.
- 1사분면내에 어떠한 벡터들에 대해 임의의 상수 값을 곱했을때 그 결과 벡터는 항상 1사분면에 위치해있지 않는다.



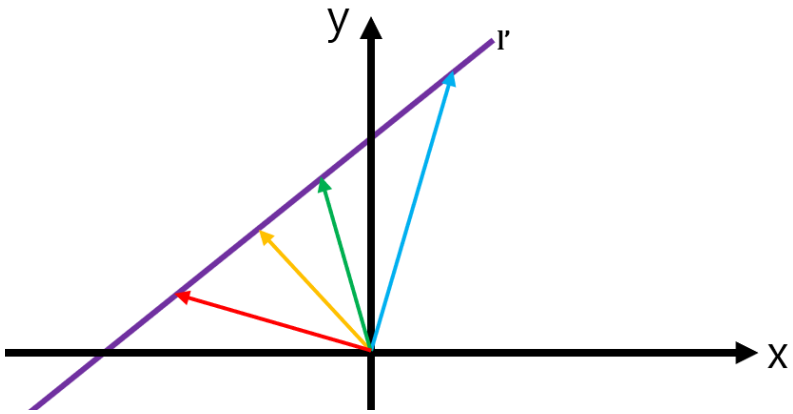
    - v에 임의의 상수 2를 곱해서 벡터의 길이를 늘린 2v는 여전히 1사분면에 위치해 있다. 그러나 반대 부호인 -1을 곱했을 경우 이 결과 벡터인 -v는 1사분면을 벗어난다.
    - 이러한 경우 1사분면 공간의 벡터에 대해 곱셈 연산은 "닫혀있지않다"라고 한다.
- **임의의 차원의 벡터 공간이 성립하기 위해선 Linear Combination 연산에 대해 성립하고 닫혀있어야 한다.**

▼ **2) Subspace of R2**

- Subspace란 임의의 n차원 공간에 대해, n차원 공간에 포함되면서 n차원 벡터들에 대해 Linear Combination 연산이 성립하는 작은 공간이다.
- Subspace안의 벡터들끼리는 Linear Combination을 할 수 있어야한다.
- Subspace는 n차원 공간에 포함되고 subspace내의 벡터들을 활용해 Linear Combination을 했을때 그 결과 벡터가 subspace에 존재할 경우 성립한다.
- 벡터 v가 2차원 공간의 어떤 subspace에 위치해 있다고 가정하고 벡터v에 임의의 상수를 곱한 벡터들을 공간 상에 나타내면 다음과 같다.



    - 이 부분공간에는 v, 2v, 3v -$\frac{1}{2}$v, 영벡터 등이 존재한다. 이러한 수많은 벡터들을 나열한 것이 직선 l이다. 따라서 이 부분공간은 직선 l이 된다.
    - 직선 l에 존재하는 벡터들끼리 더해도 같은 공간에 존재하기 때문에 직선 l은 R2의 subspace가 맞다.
- R2에 존재하는 모든 직선은 전부 subspace가 될 수 있을까? No.



    - Subspace가 되기 위해선 어떠한 수를 곱하거나, 같은 공간내의 어떠한 벡터들을 더해도 그 결과가 subspace내에 존재해야 한다. 위 그림에서 초록색 벡터에 0을 곱했을때 직선 l'위에 존재하지 않는다. 다른 벡터들도 마찬가지이다. 따라서 직선은 R2의 subspace가 아니다.
    - 어떠한 벡터도 0을 곱하면 zero vector가 되기 때문에 **임의의 R^n의 subspace는 반드시 zero vector를 포함해야 한다.**
- R2에서 가능한 subspace는 다음과 같은 것들이 있다.
    - All of R^2
        - R^2 그 자체가 하나의 subspace가 될 수 있다.
    - Any line through zero vector
        - 원점을 지나는 어떠한 직선은 R^2의 subspace가 될 수 있다.
    - Zero vector only
- R3에서 가능한 subspace는 다음과 같은 것들이 있다.
    - All of R^2
    - Plane through zero vector
    - Line through zero vector
    - Zero vector only

  등 여러가지가 있을 수 있다.

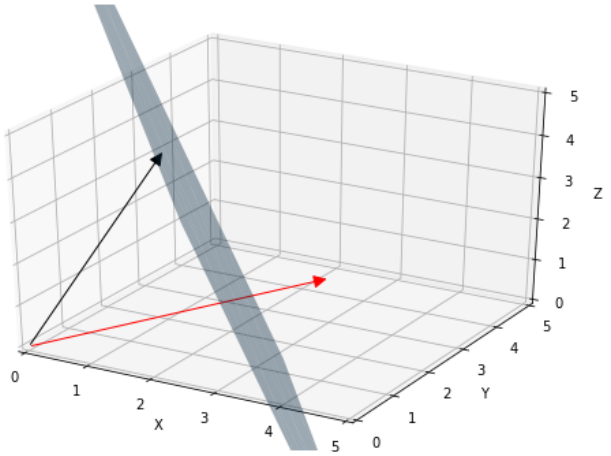
▼ 6. Subspace of matrix

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 3 \\ 4 & 1 \end{bmatrix}$$

- 행렬 A로부터 subspace인 column space를 뽑아낼 수 있다.
- **임의의 행렬A에서, 모든 column의 Linear Combination은 subspace을 형성하고 이를 column space라 부르고 C(A)로 쓴다.**

- 두 column 벡터들에 임의의 상수를 곱하고 이를 더해 Linear Combination을 한 결과를 그려보면 다음과 같다.

$$lc = c_1 \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} + c_2 \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$$



검정 : $\begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$ 빨강 : $\begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$

▼ 코드

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')

class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0,0), (0,0), *args, **kwargs)
        self._verts3d = xs, ys, zs

    def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0],ys[0]),(xs[1],ys[1]))
        FancyArrowPatch.draw(self, renderer)
print ("ingrese coordenada inicial")

a = Arrow3D([0,1],[0,2],[0,4], mutation_scale=20, lw=1, arrowstyle="-|>", color="k")
b = Arrow3D([0,3],[0,3],[0,1], mutation_scale=20, lw=1, arrowstyle="-|>", color="r")

ax.add_artist(a)
ax.add_artist(b)

ax.set_xlim([0, 5])
```

```
ax.set_ylim([0, 5])
ax.set_zlim([0, 5])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

point  = np.array([1, 2, 4])
normal = np.array([3, 3, 1])

# a plane is a*x+b*y+c*z+d=0
# [a,b,c] is the normal. Thus, we have to calculate
# d and we're set
d = -point.dot(normal)

# create x,y
xx, yy = np.meshgrid(range(5), range(5))

# calculate corresponding z
z = (-normal[0] * xx - normal[1] * yy - d) * 1. /normal[2]

# plot the surface
ax.plot_surface(xx, yy, z, alpha=0.4)

#ax.view_init(50,50)

plt.show()
```

- 위의 예시를 통해 R3의 subspace가 평면이라는 것을 확인할 수 있었다.
- Subspace를 정의할때 임의의 행렬의 column 원소들의 가능한 모든 Linear Combination을 생각해보면 된다.