3.2 좌표와 변환

이 절에서는 공간상에서 좌표를 정의하기 위해 필요한 개념을 살펴본다. 우선 벡터의 선형독립과 랭크의 개념을 알아보고 기저벡터와 좌표변환이 선형대수와 어떻게 연관이 있는지 공부한다. 좌표변환은 이미지 처리 작업뿐 아니라 다변수 확률변수를 분석하는데도 사용된다.

선형종속과 선형독립

벡터 집합 $x_1, x_2, \ldots, x_N$을 이루는 벡터의 선형조합이 영벡터가 되도록 하는 스칼라 계수 $c_1, c_2, \ldots, c_N$이 존재하면 이 벡터들이 **선형종속(linearly dependent)**이라고 한다. 단 $c_1 = c_2 = \cdots = c_N = 0$ 으로 계수가 모두 0인 경우는 제외한다.

$$c_1 x_1 + c_2 x_2 + \cdots + c_N x_N = 0 \tag{3.2.1}$$

반대로 벡터들의 선형조합이 0이 되면서 모두 0은 아닌 계수들이 존재하지 않으면 그 벡터들은 **선형독립(linearly independent)**이라고 한다. 선형독립을 논리 기호로 나타내면 다음과 같다.

$$c_1 x_1 + \cdots + c_N x_N = 0 \quad \rightarrow \quad c_1 = \cdots = c_N = 0 \tag{3.2.2}$$

왼쪽에서 오른쪽 방향 화살표의 의미는 벡터들의 선형조합이 0이면 반드시 계수들이 모두 0이라는 뜻이다.

선형독립을 다음처럼 표현하기도 한다.

$$c_1 x_1 + \cdots + c_N x_N = 0 \quad \leftrightarrow \quad c_1 = \cdots = c_N = 0 \tag{3.2.3}$$

오른쪽에서 왼쪽 방향 화살표의 의미는 모든 계수가 0일 때 선형조합이 0이 된다는 뜻이다. 이는 꼭 선형독립이 아니더라도 당연하게 성립한다.

예제

다음 벡터 x_1, x_2는 선형독립이다.

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{3.2.4}$$

벡터 x_1은 두 원소의 크기가 다른데, 벡터 x_2는 두 원소의 크기가 같기 때문에 어떤 계수를 사용해도 $c_1 x_1 + c_2 x_2 = 0$을 만들 수 없다.

예제

다음 벡터 x_1, x_2, x_3는 선형종속이다.

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \quad x_3 = \begin{bmatrix} 5 \\ 7 \end{bmatrix} \tag{3.2.5}$$

이는 다음과 같은 식으로 증명할 수 있다.

$$2x_1 + x_2 - x_3 = 0 \tag{3.2.6}$$

파이썬으로 계산한 결과도 영벡터이다.

```
import numpy as np

x1 = np.array([1, 2])
x2 = np.array([3, 3])
x3 = np.array([10, 14])
2 * x1 + x2 - 0.5 * x3
```

```
array([0., 0.])
```

연습 문제 3.2.1

다음 벡터들이 선형독립인지 선형종속인지 판별하라. 선형종속이면 영벡터를 만드는 계수값을 찾아라.

Contents

선형종속과 선형독립
예제
예제
연습 문제 3.2.1
연습 문제 3.2.2
연습 문제 3.2.3
선형독립과 선형 연립방정식
선형종속인 경우
경우 1: 벡터의 개수가 벡터의 차원보다 크면 선형종속이다.
경우 2: 값이 같은 벡터가 있으면 반드시 선형종속이다.
경우 3: 어떤 벡터가 다른 벡터의 선형조합이면 반드시 선형종속이다.
랭크
예제
예제
풀랭크
연습 문제 3.2.4
로우-랭크 행렬
연습 문제 3.2.5
벡터공간과 기저벡터
예제
예제
예제
연습 문제 3.2.6
연습 문제 3.2.7
랭크와 역행렬
벡터공간 투영
정규직교인 기저벡터로 이루어진 벡터공간
표준기저벡터
좌표
예제
변환행렬
좌표변환
연습 문제 3.2.8
이미지 변환
연습 문제 3.2.9

(1) $

$$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \longrightarrow \text{독립} \tag{3.2.7}$$

$

(2) $

$$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \longrightarrow \text{독립} \tag{3.2.8}$$

$

(3) $

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \tag{3.2.9}$$

$

연습 문제 3.2.2

서로 직교하는 영벡터가 아닌 N개의 벡터 $v_1, v_2, \cdots, v_N$은 선형독립임을 증명하라.

$$v_i^T v_j = 0 \quad (i \neq j) \tag{3.2.10}$$

연습 문제 3.2.3

(1) 선형독립인 2개의 2차원 벡터의 예를 들어라.

(2) 선형독립인 2개의 3차원 벡터의 예를 들어라.

(3) 선형독립인 3개의 2차원 벡터의 예를 들어라. 이러한 벡터들이 존재하는가?

(4) 선형독립인 3개의 3차원 벡터의 예를 들어라. 이러한 벡터들이 존재하는가?

(5) 선형독립인 4개의 3차원 벡터의 예를 들어라. 이러한 벡터들이 존재하는가?

선형독립과 선형 연립방정식

선형독립 관계를 행렬과 벡터의 곱으로 나타낼 수도 있다. 다음 식에서 c_i는 x_i에 대한 가중치 계수이고 c는 c_i를 원소로 가지는 가중치 벡터이다. X는 열벡터 $x_1, x_2, \ldots, x_N$를 열로 가지는 행렬이다. 이제부터는 벡터의 집합으로 모두 이런 식으로 행렬로 표시하겠다.

$$c_1 x_1 + \cdots + c_N x_N = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = Xc \tag{3.2.11}$$

행렬·벡터

따라서 어떤 벡터들이 선형독립인지 아닌지를 알아내는 문제는 선형 연립방정식을 푸는 문제와 같다.

$$Xc = 0 \tag{3.2.12}$$

이 연립방정식의 해가 영벡터밖에 없으면 선형독립이다. 만약 영벡터가 아닌 해가 존재하면 선형종속이다. 해가 무한히 많은 경우에는 그 중에 영벡터가 아닌 해가 반드시 존재하므로 선형 종속이다.

벡터 $x_1, x_2, \ldots, x_N$이 선형독립이라는 것을 논리기호로 나타내면 다음과 같다.

$$Xc = 0 \quad \rightarrow \quad c = 0 \tag{3.2.13}$$

$c = 0$이면 $Xc = 0$은 당연히 성립하므로 위 식 대신 다음처럼 쓰기도 한다.

또는

$$Xc = 0 \quad \leftrightarrow \quad c = 0 \tag{3.2.14}$$

선형종속인 경우

벡터가 선형종속이 되는 대표적인 세 경우를 알아보자. 예측모형을 만들기 위한 특징행렬 X의 열벡터들이 선형종속이거나 선형종속에 가까운 현상을 **다중공선성(multicollinearity)**이라고 부른다. 다중공선성이 발생하면 예측의 성능이 나빠지므로 되도록 이러한 경우가 발생하지 않도록 주의해야 한다.

경우 1: 벡터의 개수가 벡터의 차원보다 크면 선형종속이다.

벡터의 차원보다 벡터의 수가 많으면 그 벡터를 행으로 가지는 행렬 X의 행의 개수보다 열의 개수가 많다. 따라서 이 행렬이 표현하는 연립방정식을 고려하면 미지수의 수가 방정식의 수보다 커서 해가 무한히 많다. 해가 무한히 많다는 것은 영벡터가 아닌 해 c도 존재한다는 뜻이다. 따라서 그 벡터들은 선형종속이다. 반대로 행의 개수가 열의 개수와 같거나 크면 대부분 선형독립이다. 우리가 분석할 대부분의 데이터는 데이터(행)의 수가 특징(열)의 수보다 많기 때문에 여기에 해당한다.

경우 2: 값이 같은 벡터가 있으면 반드시 선형종속이다.

만약 i번째 벡터 x_j와 j번째 벡터 x_j가 같으면 $c_j = -c_i$로 놓고 다른 c값은 모두 0으로 하면

$$\begin{aligned} & 0 \cdot x_1 + \cdots + c_i \cdot x_i + \cdots + c_j \cdot x_j + \cdots + 0 \cdot x_N \\ & = 0 \cdot x_1 + \cdots + c_i \cdot x_i + \cdots + (-c_i) \cdot x_j + \cdots + 0 \cdot x_N \\ & = 0 \end{aligned} \tag{3.2.15}$$

따라서 다음처럼 중복된 데이터가 있으면 선형종속이다.

$$\begin{bmatrix} 1 & 5 & 1 \\ 3 & 6 & 3 \\ 4 & 6 & 4 \end{bmatrix} \tag{3.2.16}$$

벡터 x_j가 벡터 x_i의 실수배인 경우도 마찬가지이다.

예측 모형을 위한 특징행렬을 만들 때 실수로 위와 같은 행렬을 만드는 경우가 종종 있다. 우리가 실무에서 다루게 되는 데이터는 그 양이 크기 때문에 엑셀 등으로 한 눈에 볼 수 없는 경우가 많아서 위와 같은 실수를 하는 경우에도 빨리 발견하기 어렵다.

경우 3: 어떤 벡터가 다른 벡터의 선형조합이면 반드시 선형종속이다.

예를 들어 벡터 x_1과 다른 벡터 x_2, x_3 사이에 다음 관계가 성립한다고 하자.

$$x_1 = 2x_2 - 3x_3 \tag{3.2.17}$$

그러면 $c_1 = -1, c_2 = 2c_3 = -3$일 때

$$-1 \cdot x_1 + 2x_2 - 3x_3 = 0 \tag{3.2.18}$$

이므로 선형종속이다.

이 경우도 데이터 분석에서 흔히 하는 실수이다. 예를 들어 국어, 영어, 수학 점수를 각각 별도의 데이터로 포함하면서 이 세 점수에 의존하는 총점수나 평균을 다시 데이터로 포함하면 선형종속이 된다.

랭크

행렬의 열벡터 중 서로 독립인 열벡터의 최대 개수를 **열랭크(column rank)**라고 하고 행벡터 중 서로 독립인 행벡터의 최대 개수를 **행랭크(row rank)**라고 한다. 행랭크와 열랭크에 대해서는 다음 정리가 성립한다.

> **[정리] 행랭크와 열랭크는 항상 같다.**

따라서 행 랭크나 열 랭크를 그냥 **랭크(rank)**라고 하기도 한다. 행렬 A의 랭크는 기호로 $\text{rank}A$와 같이 표시한다.

행랭크는 행의 개수보다 커질 수 없고 열랭크는 열의 개수보다 커질 수 없기 때문에 **행의 개수가 N이고 열의 갯수가 M인 행렬의 랭크는 행의 개수 N과 열의 개수 M 중 작은 값보다 커질 수 없다.**

$$\text{rank}A \leq \min(M, N) \tag{3.2.19}$$

$$A \in \mathbf{R}^{N \times M} \tag{3.2.20}$$

예제

다음 행렬 X_1의 두 열벡터는 선형독립이기 때문에 열랭크는 2다.

$$X_1 = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix} \tag{3.2.21}$$

예제

다음 행렬 X_2의 세 열벡터는 선형종속이므로 열랭크는 3보다는 작다. 그런데 이 열벡터 중 앞의 두 개는 서로 독립이므로 X_2의 랭크는 2다.

$$X_2 = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 3 & 7 \end{bmatrix} \quad (3.2.22)$$

넘파이 linalg 서브 패키지의 `matrix_rank()` 함수로 행렬의 랭크를 계산할 수 있다.

```
X1 = np.array([[1, 3], [2, 4]])
np.linalg.matrix_rank(X1)
```

```
2
```

```
X2 = np.array([[1, 3, 5], [2, 3, 7]])
np.linalg.matrix_rank(X2)
```

```
2
```

풀랭크

위에서 예로 든 행렬 X_1나 X_2처럼 랭크가 행의 개수와 열의 개수 중 작은 값과 같으면 **풀랭크(full rank)**라고 한다.

$$\text{rank} A = \min(M, N) \quad (3.2.23)$$

선형독립인 벡터들을 행 또는 열로 가지는 행렬을 만들면 정의에 의해 항상 풀랭크다.

연습 문제 3.2.4

다음 행렬의 랭크를 구하고 풀랭크인지 아닌지 말하라.

(1) $

$$A = \begin{bmatrix} 1 & 5 & 6 \\ 2 & 6 & 8 \\ 3 & 11 & 14 \\ 1 & 4 & 5 \end{bmatrix} \quad (3.2.24)$$

$

(2) $

$$B = \begin{bmatrix} 1 & 5 & 6 \\ 2 & 6 & 8 \\ 3 & 11 & 14 \\ 1 & 4 & 8 \end{bmatrix} \quad (3.2.25)$$

$

로우-랭크 행렬

N차원 벡터 x 하나를 이용하여 만들어지는 다음과 같은 행렬을 **랭크-1 행렬(rank-1 matrix)**이라고 한다.

$$xx^T \in \mathbf{R}^{N \times N} \quad (3.2.26)$$

이 행렬의 열벡터들은 x라고 하는 하나의 벡터를 x_1배, x_2배, $\cdots$ x_n배한 벡터이므로 독립적인 열벡터는 1개다. 따라서 **랭크-1 행렬의 랭크는 1**이다.

$$\begin{aligned} xx^T &= x \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \\ &= \begin{bmatrix} x_1 x & x_2 x & \cdots & x_n x \end{bmatrix} \end{aligned} \quad (3.2.27)$$

선형독립인 두 개의 N차원 벡터 x_1, x_2를 이용하여 만든 다음과 같은 행렬은 **랭크-2 행렬(rank-2 matrix)**이라고 한다.

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_1^T \\ x_2^T \end{bmatrix} = x_1 x_1^T + x_2 x_2^T \quad (3.2.28)$$

앞서와 비슷한 방법으로 **랭크-2 행렬의 랭크는 2**임을 보일 수 있다.

만약 M개의 N차원 벡터 $x_1, x_2, \cdots, x_M$을 이용하면 **랭크-M 행렬(rank-M matrix)**이 된다.

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_M \end{bmatrix} \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_M^T \end{bmatrix} = x_1 x_1^T + x_2 x_2^T + \cdots + x_M x_M^T = \sum_{i=1}^{M} x_i x_i^T \quad (3.2.29)$$

이러한 행렬들을 가리켜 **로우-랭크 행렬(low-rank matrix)**이라고 한다. 로우-랭크 행렬은 나중에 특이분해(singular value decomposition)와 PCA(principal component analysis)에서 사용된다.

연습 문제 3.2.5

(1) 다음 벡터로 랭크-1 행렬을 만들고 NumPy로 랭크를 계산하여 실제로 1이 나오는지 확인하라.

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (3.2.30)$$

(2) 다음 두 개의 벡터로 랭크-2 행렬을 만들고 NumPy로 랭크를 계산하여 실제로 2가 나오는지 확인하라.

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (3.2.31)$$

벡터공간과 기저벡터

여러 벡터를 선형조합을 하면 다른 벡터를 만들 수 있다. 벡터 N개가 서로 선형독립이면 이 벡터들을 선형조합하여 만들어지는 모든 벡터의 집합을 **벡터공간(vector space)** V라 하고 이 벡터공간의 차원을 N이라고 한다. 그리고 그 벡터들을 벡터공간의 **기저벡터(basis vector)**라고 한다.

$$V = \{c_1 x_1 + \cdots + c_N x_N \mid c_1, \ldots, c_N \in \mathbf{R}\} \quad (3.2.32)$$

벡터공간의 차원(dimension)이 벡터의 차원(길이)가 아니라 **기저벡터의 개수**로 정의된다는 점에 유의해야 한다.

N차원 벡터 N개 $x_1, x_2, \cdots, x_N$이 선형독립인 경우에는 다음 정리가 성립한다.

> **[정리] N개의 N차원 벡터 $x_1, x_2, \cdots, x_N$이 선형독립이면 이를 선형조합하여 모든 N차원 벡터를 만들 수 있다.**
> ↳ 기저벡터.

다음과 같이 증명한다. 임의의 벡터 x가 있다고 하자. 기저벡터 $x_1, x_2, \cdots, x_N$와 이 벡터 x를 열벡터로 사용하여 만든 행렬

$$X = [x_1, x_2, \cdots, x_N, x] \quad (3.2.33)$$

는 크기가 $N \times (N+1)$이므로 랭크값은 N보다 커질 수는 없다. 그런데 N개의 선형독립인 열벡터가 있기 때문에 랭크값은 N이고 풀랭크다. 따라서 어떠한 N차원 벡터를 생각하더라도 기저벡터의 조합으로 표현할 수 있다.

예제

다음 벡터의 집합은 선형독립이므로 2차원 벡터공간의 기저벡터이다.

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad (3.2.34)$$

따라서 이 기저벡터 $\{x_1, x_2\}$를 선형조합하면 어떠한 2차원 벡터도 만들 수 있다.

예제

다음 벡터의 집합은 선형독립이 아니므로 벡터공간의 기저벡터가 되지 않는다.

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad x_3 = \begin{bmatrix} 5 \\ 7 \end{bmatrix} \quad (3.2.35)$$

예제

다음 벡터의 집합은 선형독립이므로 벡터공간의 기저벡터이다.

$$x_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \tag{3.2.36}$$

하지만 이 벡터공간은 3차원 벡터공간이 아니라 2차원 벡터공간이라고 한다. 예를 들어 이 벡터 x_1, x_2를 어떻게 선형조합해도 다음 벡터는 만들 수 없다.

$$c_1 x_1 + c_2 x_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.2.37}$$

벡터공간의 차원을 기저벡터의 차원과 다르게 정의하는 이유는 선형독립인 기저벡터를 선형조합했을 때 이렇게 만들어낼 수 없는 벡터들이 존재하기 때문이다.

연습 문제 3.2.6

(1) 다음 기저벡터 x_1, x_2를 선형조합하여 벡터 y_1, y_2를 만들어라. $

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{3.2.38}$$

$

$$y_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \quad y_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{3.2.39}$$

(2) 2차원 벡터공간을 만드는 2차원 기저벡터의 또다른 예를 들어라.

(3) 2차원 벡터공간을 만드는 3차원 기저벡터의 또다른 예를 들어라.

(4) 3차원 벡터공간을 만드는 3차원 기저벡터의 예를 들어라.

연습 문제 3.2.7

N개의 N차원 벡터 $x_1, x_2, \cdots, x_N$이 기저벡터이다. 이 벡터 $x_1, x_2, \cdots, x_N$ 각각에 대해 모두 수직인 영벡터가 아닌 벡터 x가 존재하지 않는다는 것을 증명하라.

랭크와 역행렬

정방행렬의 랭크와 역행렬 사이에는 다음과 같은 정리가 성립한다.

> [정리] **정방행렬이 풀랭크면 역행렬이 존재한다. 역도 성립한다. 즉, 정방행렬의 역행렬이 존재하면 풀랭크**다.

따라서 다음 두 문장은 같은 뜻이다.

> **정방행렬이 풀랭크다 $\leftrightarrow$ 역행렬이 존재한다**

다음과 같이 증명한다.

(1) 우선 왼쪽에서 오른쪽 방향 즉, 정방행렬이 풀랭크이면 역행렬이 존재한다는 것을 증명하자. 정방행렬이 풀랭크이면 선형독립이고 기저벡터가 되므로 어떠한 벡터에 대해서도 그 벡터를 만들 수 있는 선형조합을 생각할 수 있다. 예를 들어 다음과 같은 벡터 $e_1, \cdots, e_N$을 만들기 위한 조합 $c_1, \cdots, c_N$도 있을 수 있다.

$$Xc_1 = e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.2.40}$$

$$Xc_2 = e_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{3.2.41}$$

이 식들을 모으면 다음과 같아진다.

$$X \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix} = XC = I \tag{3.2.42}$$

정방행렬의 경우 $XC = I$이면 $CX = I$가 성립한다.(연습문제 2.4.4) 따라서

$$XC = CX = I \tag{3.2.43}$$

인 행렬 C가 존재한다. 이 행렬이 역행렬이다.

(2) 다음으로 오른쪽에서 왼쪽 방향 즉, 역행렬이 존재하면 풀랭크라는 것을 증명하자. 역행렬이 존재하는 경우에 다음 식이 성립한다는 것을 증명하면 된다.

$$Xc = 0 \quad \leftrightarrow \quad c = 0 \tag{3.2.44}$$

(i) 우선 역행렬이 존재하든 말든 $c = 0$이면 $Xc = 0$는 당연하다. 따라서 오른쪽에서 왼쪽 방향은 증명된다.

(ii) 다음으로 역행렬이 존재할 때 $Xc = 0$이면

$$X^{-1}Xc = c = 0 \tag{3.2.45}$$

이므로 왼쪽에서 오른쪽 방향도 증명된다. 따라서 역행렬이 존재하면 풀랭크다.

벡터공간 투영

M개의 N차원 기저벡터 $v_1, v_2, \cdots, v_M$ 가 존재한다고 하자. M은 N보다 작다. 이 때 모든 N차원 벡터 x에 대해 기저벡터 $v_1, v_2, \cdots, v_M$를 선형조합하여 만든 벡터 $x^{\parallel v}$와 원래 벡터 x의 차 $x - x^{\parallel v}$가 모든 기저벡터에 직교하면 그 벡터 $x^{\parallel v}$를 $v_1, v_2, \cdots, v_M$ **벡터공간에 대한 투영벡터**라 하고 차이 벡터 $x - x^{\parallel v} = x^{\perp v}$를 **벡터공간에 대한 직교벡터**라 한다.

$$(x - x^{\parallel V}) \perp \{v_1, v_2, \cdots, v_M\} \tag{3.2.46}$$

다음 그림은 $N = 3, M = 2$ 즉 3차원 벡터를 2차원 벡터공간에 투영하는 예를 보인 것이다.

그림 3.2.1 : 3차원 벡터를 2차원 벡터공간에 투영하는 예

정규직교인 기저벡터로 이루어진 벡터공간

만약 기저벡터 $v_1, v_2, \cdots, v_M$가 정규직교(orthonormal)이면 투영벡터 $x^{\parallel v}$는 각 기저벡터에 대한 내적값으로 표현된다.

$$x^{\parallel V} = (x^T v_1)v_1 + (x^T v_2)v_2 + \cdots + (x^T v_M)v_M \tag{3.2.47}$$

그리고 투영벡터의 길이의 제곱은 각 기저벡터와의 내적의 제곱합이다.

$$\|x^{\parallel V}\|^2 = \sum_{i=1}^{M} (x^T v_i)^2 \tag{3.2.48}$$

벡터 x에서 이 벡터 $x^{\parallel V}$를 뺀 벡터 $x - x^{\parallel V}$, 즉 직교벡터 $x^{\perp V}$가 기저벡터 $v_1, v_2, \cdots, v_M$에 모두 직교한다는 것은 다음처럼 증명할 수 있다.

$$\begin{aligned} v_i^T(x - x^{\parallel v}) &= v_i^T x - v_i^T \left((x^T v_1)v_1 + (x^T v_2)v_2 + \cdots + (x^T v_M)v_M \right) \\ &= v_i^T x - \left((x^T v_1)v_i^T v_1 + (x^T v_2)v_i^T v_2 + \cdots + (x^T v_M)v_i^T v_M \right) \\ &= v_i^T x - x^T v_i \\ &= 0 \end{aligned} \tag{3.2.49}$$

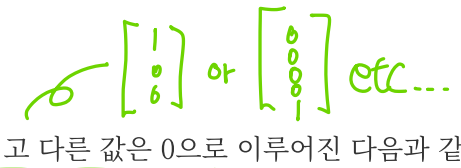
따라서 **직교벡터 $x^{\perp V}$는 기저벡터 $v_1, v_2, \cdots, v_M$으로 이루어진 벡터공간의 모든 벡터에 대해 직교한다**.

이 사실로부터 **벡터 x의 투영벡터 $x^{\parallel V}$은 기저벡터 $v_1, v_2, \cdots, v_M$으로 이루어진 벡터공간의 모든 벡터 중에서 가장 벡터 x와 가까운 벡터**라는 것도 알 수 있다.

기저벡터 $v_1, v_2, \cdots, v_M$으로 이루어진 벡터공간의 어떤 벡터를 y라고 하자. 그러면 $x^{\parallel V}$와 y의 차이 벡터 $x^{\parallel v} - y$도 $v_1, v_2, \cdots, v_M$으로 이루어진 벡터공간에 존재하므로 직교벡터 $x^{\perp V}$와 직교한다.

$$\begin{aligned} \|x - y\|^2 &= \|x - x^{\parallel V} + (x^{\parallel V} - y)\|^2 \\ &= \|x^{\perp V} + (x^{\parallel V} - y)\|^2 \\ &= \|x^{\perp V}\|^2 + \|(x^{\parallel V} - y)\|^2 \\ &\geq \|x^{\perp V}\|^2 \end{aligned} \tag{3.2.50}$$

표준기저벡터



기저벡터 중에서도 원소 중 하나만 값이 1이고 다른 값은 0으로 이루어진 다음과 같은 기저벡터를 **표준기저벡터**(standard basis vector)라고 한다.

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots, \quad e_N = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{3.2.51}$$

표준기저벡터를 열로 가지는 행렬은 항등행렬이 된다.

$$[\, e_1 \quad e_2 \quad \cdots \quad e_N \,] = I_N \tag{3.2.52}$$

좌표

어떤 벡터의 **좌표(coordinate)**는 기저벡터를 선형조합하여 그 벡터를 나타내기 위한 계수벡터를 말한다.

예를 들어 다음처럼 기저벡터 $\{e_1, e_2\}$를 선형조합하여 벡터 x를 나타낼 수 있다고 가정하자.

$$x = x_{e_1} e_1 + x_{e_2} e_2 \tag{3.2.53}$$

이 때 벡터 x_e

$$x_e = \begin{bmatrix} x_{e_1} \\ x_{e_2} \end{bmatrix} \tag{3.2.54}$$

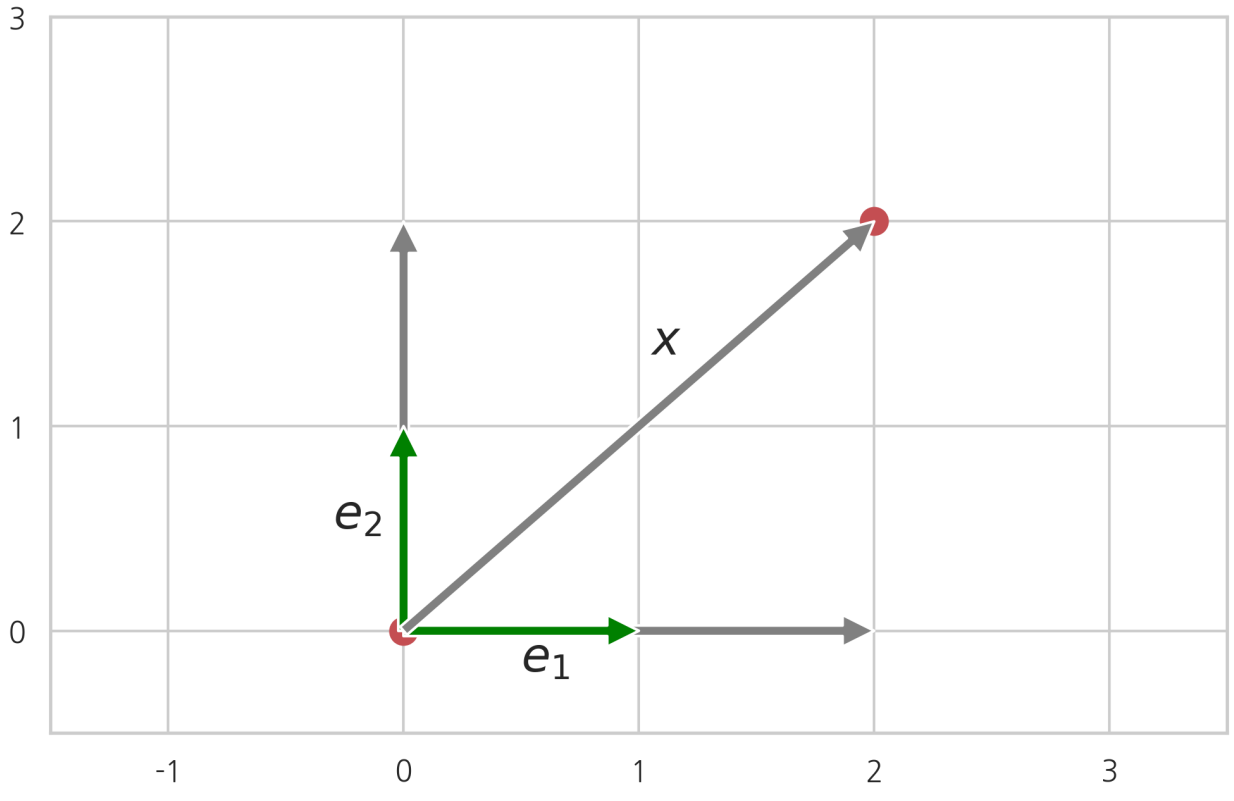
를 벡터 x의 기저벡터 $\{e_1, e_2\}$에 대한 좌표벡터 혹은 간단히 **좌표(coordinate)**라고 한다. 벡터와 기저벡터 그리고 좌표의 관계는 다음과 같다.

$$x = [e_1 e_2] \begin{bmatrix} x_{e_1} \\ x_{e_2} \end{bmatrix} = [e_1 e_2] x_e \tag{3.2.55}$$

표준기저벡터를 모아놓은 행렬이 항등행렬이기 때문에 표준기저벡터에 대한 벡터의 좌표 x_e는 원래 벡터 x와 같다. 하지만 같은 벡터라도 다른 기저벡터를 사용하면 좌표가 달라진다. 따라서 하나의 벡터도 기저벡터에 따라 여러 좌표를 가질 수 있다.

```
import matplotlib.pylab as plt

gray = {"facecolor": "gray"}
black = {"facecolor": "black"}
red = {"facecolor": "red"}
green = {"facecolor": "green"}
blue = {"facecolor": "blue"}
lightgreen = {"facecolor": "lightgreen"}
```

```
e1 = np.array([1, 0])
e2 = np.array([0, 1])
x = np.array([2, 2])

plt.annotate('', xy=2 * e1, xytext=(0, 0), arrowprops=gray)
plt.annotate('', xy=2 * e2, xytext=(0, 0), arrowprops=gray)
plt.annotate('', xy=e1, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=e2, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=x, xytext=(0, 0), arrowprops=gray)

plt.plot(0, 0, 'ro', ms=10)
plt.plot(x[0], x[1], 'ro', ms=10)

plt.text(1.05, 1.35, "$x$", fontdict={"size": 18})
plt.text(-0.3, 0.5, "$e_2$", fontdict={"size": 18})
plt.text(0.5, -0.2, "$e_1$", fontdict={"size": 18})

plt.xticks(np.arange(-2, 4))
plt.yticks(np.arange(-1, 4))
plt.xlim(-1.5, 3.5)
plt.ylim(-0.5, 3)
plt.show()
```



예제

다음 기저벡터 $\{g_1, g_2\}$를 사용하면,

$$g_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ g_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad (3.2.56)$$

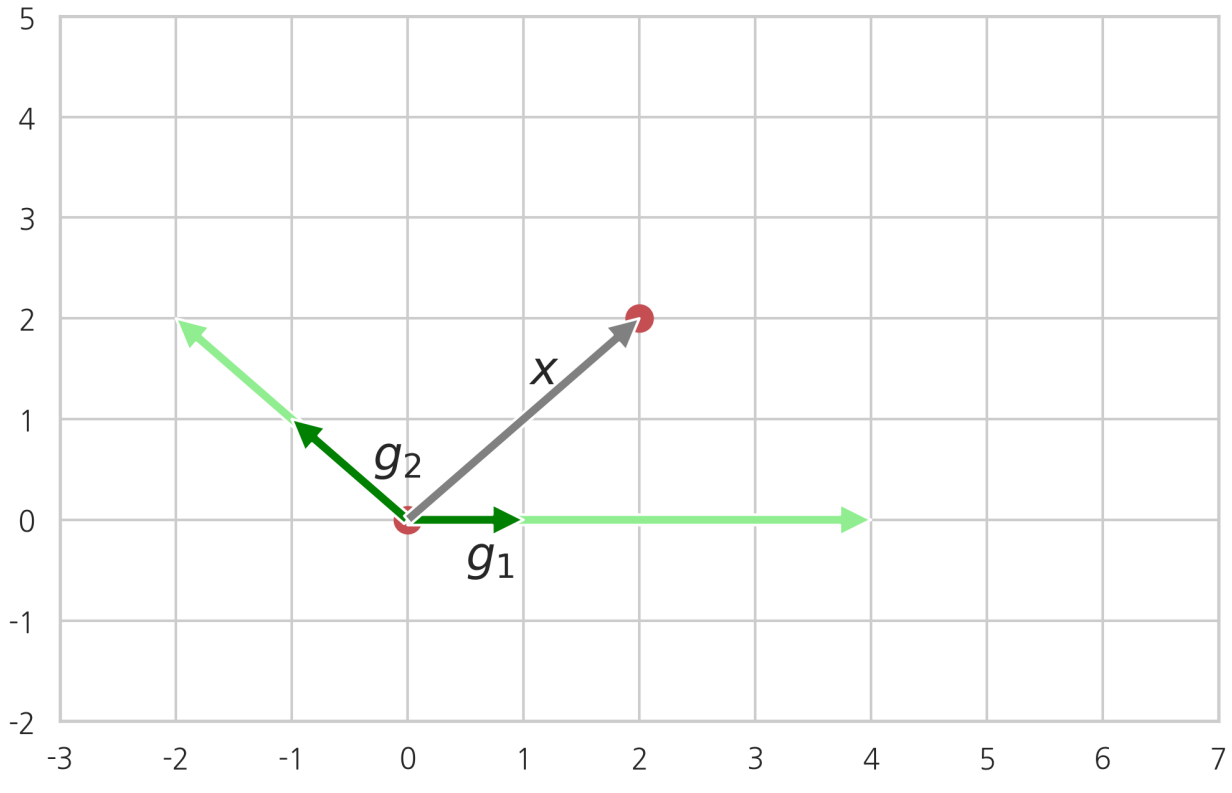
위에 예로 든 벡터 x는 기저벡터 $\{g_1, g_2\}$를 다음처럼 선형조합하여 표현할 수 있다.

$$x = 4g_1 + 2g_2 = [\, g_1 \quad g_2 \,] \begin{bmatrix} 4 \\ 2 \end{bmatrix} = [\, g_1 \quad g_2 \,]\, x_g \quad (3.2.57)$$

따라서 기저벡터 $\{g_1, g_2\}$에 대한 x의 좌표는 다음과 같다.

$$x_g = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \quad (3.2.58)$$

```
g1 = np.array([1, 0])
g2 = np.array([-1, 1])
x = np.array([2, 2])

plt.annotate('', xy=4 * g1, xytext=(0, 0), arrowprops=lightgreen)
plt.annotate('', xy=2 * g2, xytext=(0, 0), arrowprops=lightgreen)
plt.annotate('', xy=g1, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=g2, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=x, xytext=(0, 0), arrowprops=gray)

plt.plot(0, 0, 'ro', ms=10)
plt.plot(x[0], x[1], 'ro', ms=10)

plt.text(1.05, 1.35, "$x$", fontdict={"size": 18})
plt.text(-0.3, 0.5, "$g_2$", fontdict={"size": 18})
plt.text(0.5, -0.5, "$g_1$", fontdict={"size": 18})

plt.xticks(np.arange(-10, 10))
plt.yticks(np.arange(-10, 10))
plt.xlim(-3, 7)
plt.ylim(-2, 5)
plt.show()
```



변환행렬

원래의 기저벡터가 아닌 새로운 기저벡터가 있다고 하자. 이 새로운 기저벡터들의 기존 기저벡터에 대한 좌표를 열벡터로 보고 이를 행렬로 묶은 행렬 A를 생각하자.

예를 들어, 기존의 기저벡터가 $\{e_1, e_2\}$이고 새로운 기저벡터 $\{g_1, g_2\}$간에 다음과 같은 관계가 성립한다면,

$$
\begin{aligned}
g_1 &= \frac{1}{\sqrt{2}} e_1 + \frac{1}{\sqrt{2}} e_2 \\
g_2 &= -\frac{1}{\sqrt{2}} e_1 + \frac{1}{\sqrt{2}} e_2
\end{aligned}
\tag{3.2.59}
$$

e_1, e_2에 대한 g_1, g_2의 좌표벡터는 다음처럼 열벡터로 나타낼 수 있다.

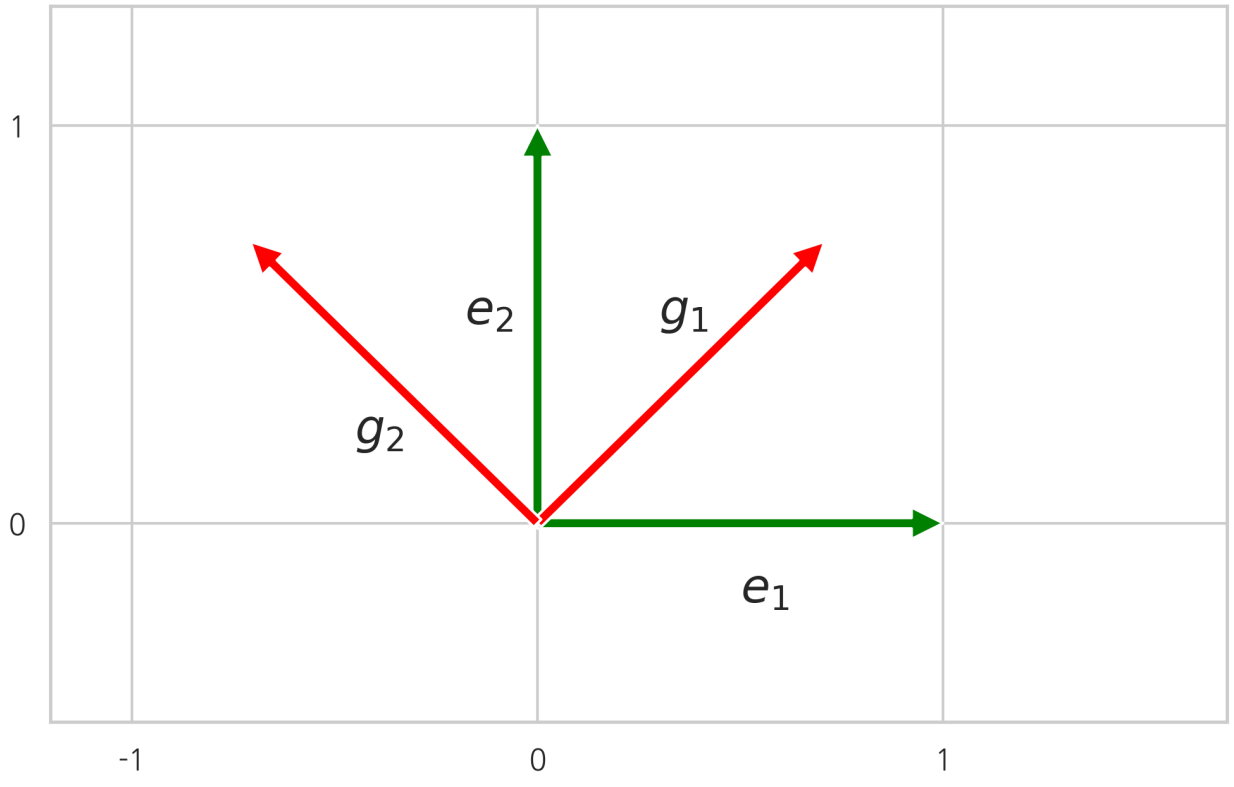
$$
g_{1e} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad g_{2e} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{3.2.60}
$$

두 좌표벡터들을 합쳐서 행렬로 표시하면 다음과 같다.

$$
[\, g_1 \quad g_2 \,] = [\, e_1 \quad e_2 \,] [\, g_{1e} \quad g_{2e} \,] = [\, e_1 \quad e_2 \,] A \tag{3.2.61}
$$

$$
A = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{3.2.62}
$$

```
e1 = np.array([1, 0])
e2 = np.array([0, 1])
a = np.array([2, 2])
g1 = np.array([1, 1]) / np.sqrt(2)
g2 = np.array([-1, 1]) / np.sqrt(2)

plt.annotate('', xy=e1, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=e2, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=g1, xytext=(0, 0), arrowprops=red)
plt.annotate('', xy=g2, xytext=(0, 0), arrowprops=red)

plt.text(-0.18, 0.5, "$e_2$", fontdict={"size": 18})
plt.text(0.5, -0.2, "$e_1$", fontdict={"size": 18})
plt.text(0.3, 0.5, "$g_1$", fontdict={"size": 18})
plt.text(-0.45, 0.2, "$g_2$", fontdict={"size": 18})

plt.xticks(np.arange(-2, 4))
plt.yticks(np.arange(-1, 4))
plt.xlim(-1.2, 1.7)
plt.ylim(-0.5, 1.3)
plt.show()
```



좌표변환

새로운 기저벡터에 대해 좌표를 계산하는 것을 **좌표변환(coordinate transform)**이라고 한다.

2차원의 경우를 예로 들어보자. 벡터 x의 기저벡터 $\{e_1, e_2\}$에 대한 좌표 x_e를 새로운 기저벡터 $\{g_1, g_2\}$에 대한 좌표 x_g로 변환하고자 한다.

새로운 기저벡터에 대한 좌푯값이 가리키는 실제 위치는 원래의 벡터가 가리키는 실제 위치와 같아야 되므로

$$x = x_{e1}e_1 + x_{e2}e_2 = x_{g1}g_1 + x_{g2}g_2 \tag{3.2.63}$$

$$x = \begin{bmatrix} e_1 & e_2 \end{bmatrix} x_e = \begin{bmatrix} g_1 & g_2 \end{bmatrix} x_g \tag{3.2.64}$$

이 식에

$$\begin{bmatrix} g_1 & g_2 \end{bmatrix} = \begin{bmatrix} e_1 & e_2 \end{bmatrix} A \tag{3.2.65}$$

를 대입하면

$$x = \begin{bmatrix} e_1 & e_2 \end{bmatrix} x_e = \begin{bmatrix} e_1 & e_2 \end{bmatrix} A x_g \tag{3.2.66}$$

이 된다. 이 식으로부터 다음 식이 성립한다.

$$x_e = A x_g \tag{3.2.67}$$

$$x_g = A^{-1} x_e = T x_e \tag{3.2.68}$$

이 때 A의 역행렬 $T = A^{-1}$을 **변환행렬(transform matrix)**이라고 한다.

예를 들어 벡터 x의 표준기저벡터에 대한 좌표가 다음과 같다고 하자.

$$x = 2e_1 + 2e_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = x_e \tag{3.2.69}$$

표준기저벡터에 대한 새로운 기저벡터의 좌표가 다음과 같다면

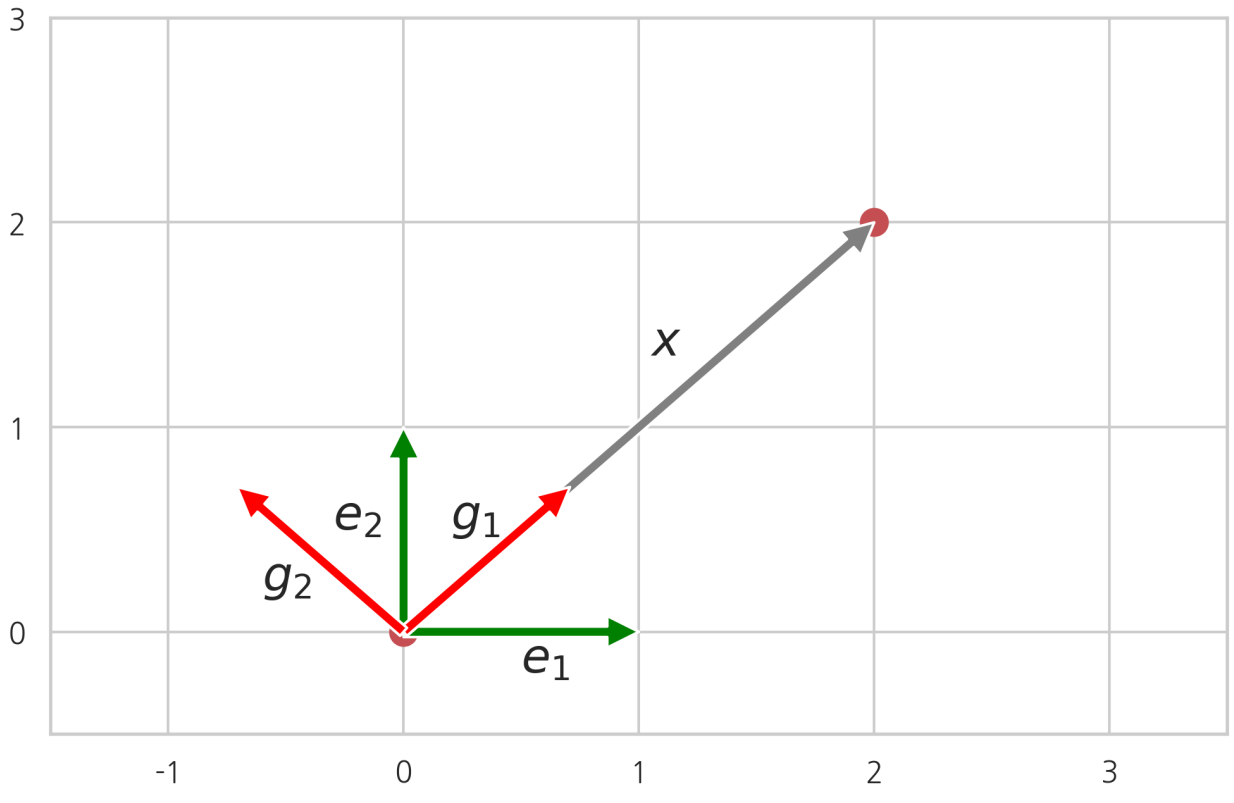
$$g_{1e} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad g_{2e} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{3.2.70}$$

새로운 기저벡터에 대한 벡터 a의 좌표는 위의 공식을 이용하여 다음처럼 계산할 수 있다.

$$x_g = A^{-1} x_e = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2\sqrt{2} \\ 0 \end{bmatrix} \tag{3.2.71}$$

다음 그림은 이 변환을 나타낸 것이다.

```
e1 = np.array([1, 0])
e2 = np.array([0, 1])
x = np.array([2, 2])
g1 = np.array([1, 1]) / np.sqrt(2)
g2 = np.array([-1, 1]) / np.sqrt(2)

plt.annotate('', xy=e1, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=e2, xytext=(0, 0), arrowprops=green)
plt.annotate('', xy=x, xytext=(0, 0), arrowprops=gray)
plt.annotate('', xy=g1, xytext=(0, 0), arrowprops=red)
plt.annotate('', xy=g2, xytext=(0, 0), arrowprops=red)

plt.plot(0, 0, 'ro', ms=10)
plt.plot(x[0], x[1], 'ro', ms=10)

plt.text(1.05, 1.35, "$x$", fontdict={"size": 18})
plt.text(-0.3, 0.5, "$e_2$", fontdict={"size": 18})
plt.text(0.5, -0.2, "$e_1$", fontdict={"size": 18})
plt.text(0.2, 0.5, "$g_1$", fontdict={"size": 18})
plt.text(-0.6, 0.2, "$g_2$", fontdict={"size": 18})

plt.xticks(np.arange(-2, 4))
plt.yticks(np.arange(-1, 4))
plt.xlim(-1.5, 3.5)
plt.ylim(-0.5, 3)
plt.show()
```



NumPy를 사용하면 다음처럼 계산할 수 있다.

```
A = np.vstack([g1, g2]).T
A
```

```
array([[ 0.70710678, -0.70710678],
       [ 0.70710678,  0.70710678]])
```

```
Ainv = np.linalg.inv(A)
Ainv
```

```
array([[ 0.70710678,  0.70710678],
       [-0.70710678,  0.70710678]])
```

```
Ainv.dot(x)
```

```
array([2.82842712, 0.        ])
```

즉, **새로운 좌표벡터는 원래의 좌표벡터에 변환행렬을 곱하여** 구할 수 있다.

연습 문제 3.2.8

만약 새로운 기저벡터의 좌표가 다음과 같다면 원래의 좌표 (1, 0), (1, 2), (-1, 2)는 각각 어떤 좌푯값이 될지 계산하라.

$$g_1 = \begin{bmatrix} 1 \\ 0.75 \end{bmatrix}, \quad g_2 = \begin{bmatrix} -1 \\ 0.75 \end{bmatrix} \quad (3.2.72)$$

이미지 변환

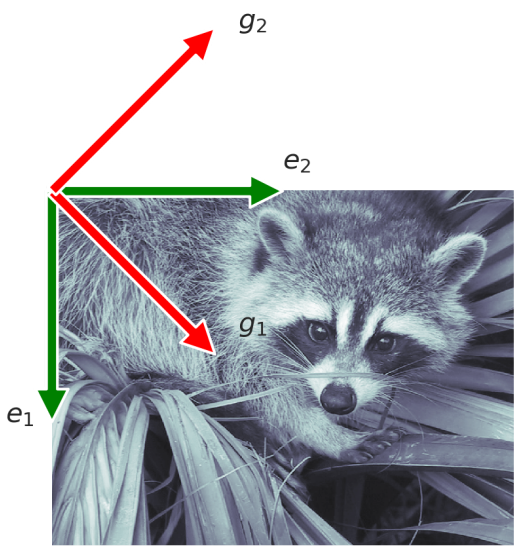
새로운 기저벡터에 대한 좌표변환을 응용하면 이미지를 자유롭게 변환할 수도 있다. 파이썬에서는 scipy.ndimage 패키지의 `affine_transform()` 명령을 사용한다. 이 명령은 이미지를 이루는 픽셀을 새로운 좌표로 이동시킨다. 인수로는 이미지 데이터와 **변환행렬의 역행렬(위에서 A로 표시한 행렬)**을 받는다. 단 파이썬 이미지에서는 다음과 같은 표준기저벡터를 사용하고 (x_1이 아래를 향하는 세로축, x_2가 오른쪽을 향하는 가로축) **원점이 왼쪽 상단의 점**이라는 점에 유의한다.

$$e_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \;\; e_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.2.73)$$

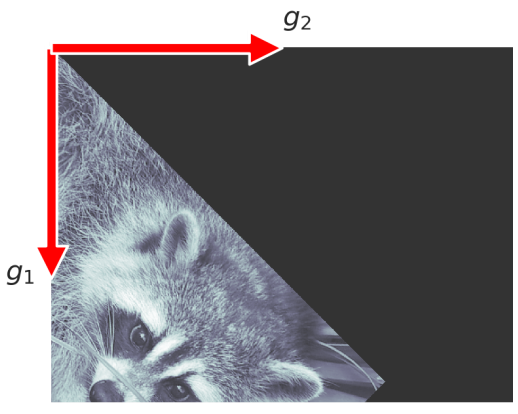
다음은 위에서 예로 든 기저벡터 $\{g_1, g_2\}$로 이미지 변환한 예다(그림에서 기저벡터의 크기는 설명을 위해 과장하여 크게 표시하였다).

```
import scipy as sp
import scipy.misc
import scipy.ndimage

f = sp.misc.face(gray=True)

e1 = np.array([0, 1]) # matplotlib의 이미지 좌표규약으로 부호변경
e2 = np.array([1, 0])
E = np.vstack([e1, e2]).T
g1 = np.array([1, 1]) / np.sqrt(2)
g2 = np.array([-1, 1]) / np.sqrt(2)
A = np.vstack([g1, g2]).T
gc1 = E @ g1
gc2 = E @ g2

plt.subplot(121)
plt.imshow(f, cmap=mpl.cm.bone, alpha=0.9)
plt.annotate('', xy=500*e1, xytext=(0,0), arrowprops=green)
plt.annotate('$e_1$', xy=500*e1, xytext=500*e1 + [-100,0])
plt.annotate('', xy=500*e2, xytext=(0,0), arrowprops=green)
plt.annotate('$e_2$', xy=500*e2, xytext=500*e2 + [0, -50])
plt.annotate('', xy=500*gc1, xytext=(0, 0), arrowprops=red)
plt.annotate('$g_1$', xy=500*gc1, xytext=500*gc1 + [50, -50])
plt.annotate('', xy=500*gc2, xytext=(0, 0), arrowprops=red)
plt.annotate('$g_2$', xy=500*gc2, xytext=500*gc2 + [50, 0])
plt.axis("off")
plt.xlim(-200, 1000)
plt.ylim(800, -500)
plt.title("좌표변환전")

f1 = sp.ndimage.affine_transform(f, A)

plt.subplot(122)
plt.imshow(f1, cmap=mpl.cm.bone, alpha=0.8)
plt.annotate('', xy=500*e1, xytext=(0,0), arrowprops=red)
plt.annotate('$g_1$', xy=500*e1, xytext=500*e1 + [-100,0])
plt.annotate('', xy=500*e2, xytext=(0,0), arrowprops=red)
plt.annotate('$g_2$', xy=500*e2, xytext=500*e2 + [0, -50])
plt.axis("off")
plt.xlim(-200, 1000)
plt.ylim(800, -500)
plt.title("좌표변환후")
plt.show()
```

좌표변환전



좌표변환후



연습 문제 3.2.9

다음 기저벡터를 이용하여 앞의 이미지를 변환하라. 변환한 이미지를 만들기 전에 어떤 이미지가 나올지 생각해보자.

$$g_1 = \begin{bmatrix} 1 \\ 0.75 \end{bmatrix}, \quad g_2 = \begin{bmatrix} -1 \\ 0.75 \end{bmatrix} \quad (3.2.74)$$

2 Comments - *powered by utteranc.es*

chiwanii commented on 2021년 6월 8일

연습문제 3.2.8

```
import numpy as np
import matplotlib.pylab as plt

e1 = np.array([1, 0])
e2 = np.array([0, 1])
x1 = np.array([1, 0])
x2 = np.array([1, 2])
x3 = np.array([-1, 2]) ## 애만 바꿈
g1 = np.array([1, 0.75])
g2 = np.array([-1, 0.75])

A = np.vstack([g1, g2]).T
A
Ainv = np.linalg.inv(A)
Ainv
A1=Ainv.dot(x1)
A2=Ainv.dot(x2)
A3=Ainv.dot(x3)
```

lysnjn commented 3 months ago

3.2.8 에서 '기존의 기저벡터에대한' 새로운기저벡터의 좌표인지, 'x에 대한' 새로운기저벡터의 좌표인지 명시가 안되어있는건 오류인가요? 그냥 새로운기저벡터의 좌표라해서 많이 헷갈렸네요 ㅜ

Write Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub