**Chapter 1 벡터, 파트1: 벡터와 벡터의 기본 연산**

* 벡터의 정의, 해석 방법,
* 파이썬으로 벡터 활용
* 벡터 대수학과 내적(dot product) 등 주요 연한
* 벡터 분해

**1.1 NumPy로 벡터 생성 및 시각화하기**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 선형대수학에서 [ a ]는 **수를** **순서대로 나열**한 것을 의미한다. (벡터는 함수 등 다른 수학적 대상을 갖을 수 있으나 교재에서는 다루지 않는다) | | a. 벡터(vector) |
| 2 | 벡터는 다음의 두 가지 특징을 갖는다.  ① [ a ] : 벡터가 가진 원소의 수  ② [ b ] : 벡터가 열 방향인지 행 방향인지를 나타낸다. | | a. 차원(dimensionality)  b. 방향(orientation) |
| 3 | 차원은 [ a ] 로 표기하며 은 실수(Real number), 은 차원을 나타낸다. 참고로 는 복소수(실수와 허수의 합으로 이루어진 수)이다.  \*\* example \*\*  4차원 열 벡터, | | a. |
| 4 | 수학에서의 차원과 파이썬에서 차원은 다른 의미를 갖는다. 수학에서 차원은 [ a ]인 반면 파이썬에서 벡터와 행렬의 차원은 [ b ]이다.  예를 들어 NumPy는 원소의 수에 상관없이 **2차원 배열**로 간주한다. | | a. 원소의 수  b. 기하학적 차원 |
| 5 | 특정 방향이 없는 경우 파이썬에서 [ a ]로 설정된다. 출력은 행으로 출력되나 이는 [ b ]는 아니다. | | a. 1차원 배열  b. 행 벡터 |
| 6 | 수학적 차원인 벡터의 원소 수는 파이썬에서는 [ a ] 혹은 [ b ]이라 한다. | | a. 길이(length)  b. 모양(shape) |
| 7 | | 벡터는 다음과 같이 표기한다.  ① 진한 로마자 : [ a ]  ② 이탤릭체 : [ b ] or [ c ] | a. **v**  b.  c. |
| 8 | | 선형대수학에서 보통 벡터에 표시가 없다면 [ a ]을 가정한다. | a. 열 방향 |
| 9 | | 행 벡터의 경우 [ a ]로 표기한다. 는 전치 연산(transpose operation)을 나타낸다. | a. |
| 10 | | 파이썬 벡터는 여러 데이터 타입으로 나타낼 수 있다. **리스트 타입**이 가장 간편하기는 하지만 선형대수학 연산은 리스트에 잘 동작하지 않기 때문에 [ a ]로 생성하는 것이 가장 좋다. | a. NumPy 배열 |
| 11 | | **\*\* 벡터의 방향은 중요한가? \*\***   * 벡터는 세 개의 방향이 있다. **① 열 방향, ② 행 방향, ③ 방향이 없는 1차원 배열.** * 데이터를 **저장**할 때에는 벡터의 방향이 중요하지 않다. * 그러나 **연산**을 할 때에 벡터의 방향이 잘못되면 원하는 결과가 나오지 않는 경우가 발생할 수 있다. | |

**1.1.1 벡터의 기하학적 해석**

|  |  |  |
| --- | --- | --- |
| 1 | **< 벡터의 해석 >**  **① 대수학적 해석**   * 순서대로 [ a ]된 수 목록(ordered list of numbers). * 데이터 과학에 유용하다(시간별 판매 등).   **② 기하학적 해석**   * 특정 [ b ]와 [ c ]을 가진 직선. * 물리학과 공학에 유용하다. | a. 나열  b. 길이(크기, magnitude)  c. 방향(양의 x 축의 각도, angle) |

|  |  |  |
| --- | --- | --- |
| 2 | 벡터의 두 점은 **꼬리(시작)**와 **머리(끝)**으로 불리며 일반적으로 머리에 [ a ]가 달려있다. | a. 화살표 |
| 3 | **벡터**는 직선의 형태로 **좌표**가 인코딩 것으로 착각할 수 있다. 그러나 벡터와 좌표는 다른 개념이다.  단, 원점에서 시작하는 벡터는 좌표와 동일하며 이를 [ a ]이라 한다. | a. 기준 위치  (standard position) |
| 4 | * 위의 벡터는 모두 **동일한 벡터**이다. * 파란 원이 **머리(끝)**이고 노란 원이 **꼬리(시작)**이다. * 원점에서 시작하는 벡터는 **기준 위치(standard position)**로 좌표와 동일한 의미를 갖는다. | |

**1.2 벡터 연산**

벡터는 이야기의 주인공이다. 이러한 벡터(주인공)에게 부여되는 움직임을 연산(operation)이라 한다. 연산에는 간단한 덧셈부터 조금 어려운 특잇값 분해 등이 있다.

**1.2.1 두 벡터의 덧셈**

|  |  |  |
| --- | --- | --- |
| 1 | 벡터의 덧셈과 뺄셈은 동일한 [ a ]을 갖는 벡터끼리만 가능하다(). | a. 차원 |
| 2 | 벡터의 덧셈과 뺄셈은 서로 [ a ]되는 원소끼리 연산이 행해진다.  \*\* example \*\* | a. 대응 |
| 3 | 벡터의 방향이 다른 벡터의 덧셈과 뺄셈을 시행하는 경우 파이썬에서는 [ a ]이라는 연산을 시행한다.  \*\* example \*\* | a. 브로드캐스팅  (broadcasting) |
| 4 | **< 결론 >**  두 벡터의 **차원**과 **방향**이 같을 때에만 벡터의 덧셈과 뺄셈이 가능하다. |  |

**1.2.2 벡터의 덧셈과 뺄셈의 기하학적 해석**

|  |  |  |
| --- | --- | --- |
| 1 | **< 두 벡터를 기하학 적으로 더함 >**   * 첫 번째 벡터의 [ a ] → 두 번째 벡터의 [ b ]로 이어지는 선 * 모든 벡터를 더하면 첫 번째 [ a ] → 마지막[ b ]로 이어지는 선이 된다.   \*\* example \*\* | a. 꼬리  b. 머리 |
| 2 | **< 두 벡터를 기하학적으로 뺌 >**   * 두 벡터의 꼬리를 [ a ]에 둔다(기준 좌표 이용) * 빼기의 결과는 ‘두 번째 벡터의 [ b ] → 첫 번째 벡터의 [ b ]’로 이어지는 선이 된다.   \*\* example \*\* | a. 같은 좌표  b. 머리 |
| 3 | 벡터의 뺄셈은 다음의 이유로 기하학적으로 중요하다.  벡터의 뺄셈  ↓  직교벡터 분해의 기초  ↓  선형 최소제곱법의 기초  ↓  과학과 공학에서 활용되는 선형대수학의 가장 중요한 응용 | |

**1.2.3 스칼라-벡터 곱셈**

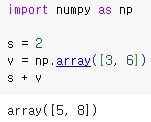
|  |  |  |
| --- | --- | --- |
| 1 | 선형대수학에서 [ a ]란 벡터나 행렬에 포함된 숫자가 아닌 **수 그 자체**이다. | a. 스칼라(scalar) |
| 2 | 스칼라는 일반적으로 [ a ] 그리고 [ b ]와 같은 그리스어 소문자로 나타낸다. | a.  b. |
| 3 | 스칼라와 벡터의 곱은 다음처럼 매우 간단한다. | |
| 4 | 스칼라를 곱할 때에는 data type이 중요하다. 다음의 두 가지 경우는 파이썬 연산에서 자주 사용되므로 알아 두자.  ① **list**에 스칼라 연산을 하면 list의 요소를 [ a ]한다.  ② **NumPy 배열**에 스칼라 연산을 하면 기존의 방식대로 스칼라를 [ b ]한다.  **※** int와 float의 type 또한 맞추어 주어야 함을 주의하자  \*\* example \*\* | a. 반복  b. 곱 |

**1.2.4 스칼라-벡터 덧셈**

기본적으로 **선형대수학**에서 스칼라와 벡터를 더하는 것은 **불가**하다. 그 이유는 스칼라와 벡터는 서로 다른 개념이기 때문이다.

그러나 **파이썬**에서는 벡터와 스칼라를 **더할 수 있다**. 더하는 방법은 스칼라-벡터 곱과 같이 벡터의 각 원소에 스칼라를 더한다.

\*\* example \*\*

****

**\*\* 스칼라-벡터 곱셉의 기하학적 해석 \*\***

|  |  |  |
| --- | --- | --- |
| 1 | 스칼라를 scalar라고 부르는 이유는 스칼라는 벡터의 [ a ]을 바꾸지 않고 벡터의 [ b ]만 조정하기 때문이다.  (scalar는 ‘방향의 구별은 없고 하나의 수치만으로 완전히 표시되는’의 의미가 있다. | a. 방향  b. 크기 |
| 2 | 스칼라-벡터 곱셈 결과는 스칼라가 ① 1 보다 크다, ② 0과 1 사이이다, ③ 0 이다, ④ 음수이다 에 따라 다르다. |  |
| 3 | 스칼라가 음수인 경우 벡터가 음의 방향으로 바뀌는 듯 보이는데 아래의 흐름을 따라가면 그렇지 않다는 것을 알 수 있다.  **< 왜 음의 스칼라-벡터 곱은 방향이 바뀌는 것이 아닌가 >**  벡터는 원점을 통과하여 **양방향의 무한대로 가는 무한히 긴 선**이라는 해석이 존재한다.  → ‘회전된’벡터는 위의 의미에서 여전히 동일한 **무한한 선**을 가리킨다.  → 음의 스칼라로 인하여 회전된 벡터 또한 여전히 **동일한 방향**이다.   * 위의 해석은 **행렬 공간, 고유벡터, 그리고 특이벡터**에서 중요하다. | |
| 4 | 벡터 덧셈과 스칼라-벡터 곱셈을 이용하면 벡터의 평균(vector average)를 구할 수 있다. N개의 벡터를 가정하면 N개의 벡터를 모두 더하고 스칼라 [ a ]을 곱해준다.  [ b ] | a.  b. |

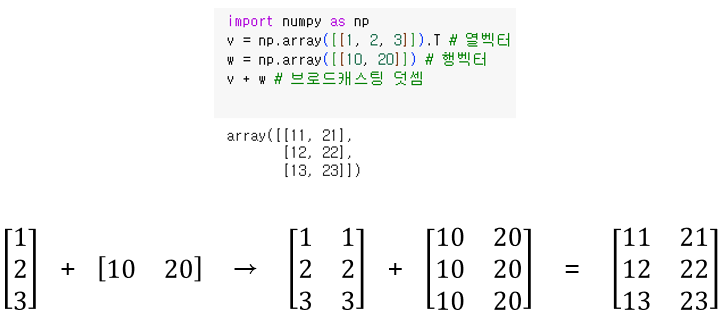
**1.2.5 전치**

|  |  |  |
| --- | --- | --- |
| 1 | [ a ] 연산은 열 벡터를 행 벡터로 혹은 반대로 변환한다. | a. 전치(transpose) |
| 2 | 예를 들어 행, 열을 갖는 행렬 의 전치(transpose) 결과는 다음과 같다.  = [ a ] | a. |
| 3 | **< 중요 규칙 >**  벡터에 두 번 전치(transpose) 연산을 하면 벡터는 원래 방향이 된다. 즉, 다음과 같다.  = [ a ]  ※ 위의 규칙은 데이터 과학과 머신러닝에서 중요한 증명의 핵심 근거가 된다. 예를 들면 이 된다. | a. |

**1.2.6 파이썬에서 벡터 브로드캐스팅**

브로드캐스팅 연산은 전통적인 선형대수학 교과서에는 존재하지 않으며 **현대 컴퓨터 기반 선형대수학**에만 존재하는 연산이다.

**\*\* example \*\***



**1.3 벡터 크기와 단위벡터**

|  |  |  |
| --- | --- | --- |
| 1 | 벡터의 크기는 **기하학적 길이** 혹은 [ a ]이라고도 한다. 벡터의 크기를 구하는 방식은 일반적으로 [ b ] 거리 공식을 사용한다. 마지막으로 벡터의 크기는 [ c ]로 표기한다. | a. 노름(norm)  b. 표준 유클리드(Euclidean)  c. |
| 2 | 벡터의 크기를 수식으로 표현하면 다음과 같다.  = [ a ] | a. |
| 3 | 앞서 언급하였듯 파이썬과 선형대수학 사이에는 용어의 불일치가 존재한다. 이를 정리하면 다음과 같다.   |  |  | | --- | --- | | **파이썬** | **선형대수학** | | 파이썬 함수 [ a ] | 배열의 차원 | | 파이썬 함수 [ b ] | 기하학적 길이 | | a. len()  b. np.norm() |
| 4 | 벡터의 **기하학적 길이(노름, norm)이 1인 벡터**를 [ a ]라 한다. [ a ]는 직교 행렬, 회전 행렬, 고유벡터, 특이벡터 등 응용에 사용된다. 단위벡터는 [ b ]로 정의된다. | a. 단위벡터(unit vector)  b. |

|  |  |  |
| --- | --- | --- |
| 5 | 비단위벡터의 경우 같은 방향의 단위벡터로 만들 수 있다. 방법은 비단위벡터에 [ a ]의 역수를 곱하며 [ b ]로 표기한다. | a. 벡터 노름  b. |
| 6 | 단위벡터 생성을 수식으로 표현하면 다음과 같다.  [ a ]  **※** 단, 모든 비단위벡터가 연관된 단위벡터를 갖는 것은 아님을 주의해야 한다. | a. |
| 7 | \*\* 비단위벡터의 단위벡터화 \*\* | |

**1.4 벡터-내적**

|  |  |  |
| --- | --- | --- |
| 1 | [ a ](점곱 & 스칼라곱)는 하나의 숫자로 **두 벡터 사이의 관계**를 나타내며 선형대수학에서 가장 중요한 연산이다. | a. 내적(dot product) |
| 2 | 내적의 표기는 [ a ], , 그리고 등이 있으며 교재에서는 [ a ]를 사용한다. | a. |
| 3 | 내적을 계산하는 방법은 두 벡터의 대응되는 원소끼리 곱한 다음 모든 결과를 더한다.  두 벡터 a와 b가 있을 때 a와 b의 내적을 식으로 표현하면 다음과 같다.  = [ a ]  **※** 내적은 **동일한 차원**의 두 벡터 사이에서만 성립한다. | a. |
| 4 | 파이썬에서 벡터의 내적은 [ a ] 함수를 통해 쉽게 구할 수 있다.  **※** 단, [ a ]는 실제로 벡터-내적을 구현하지는 않으며 **첫 번째 입력이 행벡터**이고, **두 번째 입력이 열벡터**인 경우에만 결과를 출력한다. | a. np.dot() |
| 5 | 벡터에 스칼라를 곱하면 내적도 그만큼 커지는 특성을 갖는다. 예를 들어 = 70 일 때 는 [ a ]이 된다. | a. 700 |
| 6 | 내적은 두 벡터 사이의 [ a ] 또는 [ b ]의 척도로 해석할 수 있다.  예를 들어 몸무게와 키는 상관관계가 있을 것이므로 내적 또한 클 것이다. | a. 유사성(similarity)  b. 매핑(mapping) |

|  |  |  |
| --- | --- | --- |
| 7 | 데이터에서 두 변수의 벡터의 내적을 구할 경우 [ a ]에 영향을 받을 수 있다. | a. 단위 |
| 8 | 두 변수의 벡터의 내적을 구할 때 단위의 영향을 받는다면 **정규화 계수**로 단위의 차이를 제거할 수 있다.  이렇게 **정규화 된 내적**은 [ a ]라 불리며 데이터 과학에서 가장 중요한 분석이다. | a. 피어슨 상관계수 (Pearson correlation coefficient) |

**1.4.1 내적의 분배 법칙**

|  |  |  |
| --- | --- | --- |
| 1 | 내적의 분배 법칙은 다음과 같다.  = [ a ]  \*\* example \*\* | a. |

**1.4.2 내적의 기하학적 해석**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 다음은 벡터 내적의 **기하학적 정의**이다. 기하학적 정의의 식은 그 모양이 다르지만 원래 벡터 내적식과 수학적으로 동일하다.   |  |  | | --- | --- | | **내적 식** | **기하학적 내적 식** | |  | [ a ] | | a. |
| 2 | 벡터의 크기는 양수이다. 그러나 기하학적 정의에서는 **cos이 -1과 1사이의 값**을 갖기에 [ a ]인 경우가 존재한다.  이는 기하학적 정의에서 [ b ]가 두 벡터 사이의 **기하학적 정의임**을 보인다. | a. 음수  b. 내적의 부호 |

|  |  |  |
| --- | --- | --- |
| 3 | 두 벡터 사이의 각에 따라 내적 부호는 다섯 가지의 사례가 존재한다. | |
| 4 | **< 암기하세요. 직교벡터의 내적은 0입니다 >**  ① 두 벡터가 [ a ]한다.  ↕  ② 두 벡터는 내적이 [ b ]이다.  ↕  ③ 두 벡터 사이의 각은 [ c ]이다. | a. 직교  b. 0  c. 90° |

**1.5 그 외 벡터 곱셈**

**1.5.1 아다마르곱**

|  |  |  |
| --- | --- | --- |
| 1 | [ a ]은 두 벡터의 **대응되는 각 원소를 곱**한다. 곱의 결과는 두 벡터와 [ b ] 차원의 벡터이다.  **※ 동일한 차원**의 벡터만 아다마르곱을 할 수 있다. | a. 아다마르곱  (Hardamard product)  b. 같은 |
| 2 | 아다마르곱(Hardamard product)는 여러 [ a ]를 곱할 때 편리하다. 예를 들면 다음과 같다. | a. 스칼라 |

**1.5.2 외적**

|  |  |  |
| --- | --- | --- |
| 1 | 외적은 열벡터와 행벡터를 이용해 [ a ]을 만든다.   * 외적 행렬의 각 행 : [ b ] 스칼라에 대응되는 열벡터 원소의 곱 * 외적 행렬의 각 열 : [ c ] 스칼라에 대응되는 행벡터 원소의 곱. | a. 행렬  b. 행벡터  c. 열벡터 |
| 2 | **< 외적과 내적의 차이 >**   |  |  |  | | --- | --- | --- | |  | **외적** | **내적** | | **결과** | 행렬 | 스칼라 | | **동일 차원 필요** | X | O | | **표기** |  |  | | |
| 3 | 외적의 표기는 이고 내적의 표기는 로 표기가 다르다. 이는 외적은 [ a ]이고, 내적은 [ b ]임을 의미하며 교재 중반부에 이해하면 된다.  **※** 기본적으로 아무런 처리가 없다면 **열벡터임을 가정**한다(기억). | a.  b. |
| 4 | **< 브로드캐스팅과 외적 >**   * 브로드캐스팅 : 덧셈, 곱셈, 나눗셈과 같은 **산술 연산**을 벡터로 확장한 일반적인 [ a ]. * 외적 : 두 벡터를 곱하는 특수한 [ b ]. | a. 코딩 연산  b. 수학적 기법 |
| 5 | NumPy를 이용해 각각 열과 행방향인 두 벡터를 [ a ] 또는 [ b ] 함수에 입력하여 외적을 계산할 수 있다. | a. np.outer()  b. np.dot() |

**1.5.3 교차곱과 삼중곱**

기하학과 물리학에서 사용되지만 기술 관련 응용 분야에서 자주 등장하지는 않기에 교재에서 다루지는 않는다.

**1.6 직교벡터 분해**

|  |  |  |
| --- | --- | --- |
| 1 | 벡터 또는 행렬을 [ a ]한다는 것은 여러 **간단한 조각**들로 나뉘어짐을 의미한다.  example)  42.01 → 42 + 0.01  42 → (소인수 분해, Prime factorization) | a. 분해 |
| 2 | **< 직교 투영법, Orthogonal projection >**    **① 목표**  두 벡터 a와 b가 있음을 가정할 때 벡터 a에서 벡터 b의 머리와 최대한 **가까운 점**을 찾는다.  **② 최적화 문제로 변환**  투영 거리가 최소가 되도록 b를 a에 투영  → 벡터 a에 b를 투영한 점은 벡터 a의 크기를 줄인 **a** (는 스칼라)  → 스칼라 를 찾는 것이 최종 목표  **③ 핵심은 벡터의 뺄셈**  벡터 b의 머리와 벡터 a가 **직각**으로 만나도록 그리면 벡터 b와가장 가까운 벡터 a 위의 점 **a**를 찾을 수 있다.  → 벡터의 뺄셈은 두 벡터의 머리 사이의 직선 = [ a ].  → 벡터 a와 벡터 b-a는 직교.  → 벡터 a와 벡터 b-a의 내적은 [ b ].  위를 수식으로 풀어 쓰면 다음과 같다.  **④ 결론**  **직교 투영법(Orthogonal projection)**은 점을 [ d ]로 선에 투영하는 공식. 이는 선형 모델을 푸는 데 잘 알려진 최소제곱식, 통계학, 머신러닝 등의 많은 응용분양의 기초가 된다. | a. (b-a)  b. 0  c.  d. 최소 거리 |
| 3 | **< 직교 투영법 → 직교 벡터 분해 >**    t : 목표벡터,  r : 기준벡터,  : 기준벡터 r과 평행 성분,  : 기준벡터 r과 수직 성분  **① 목적**  목표 벡터(t)를 두 개의 다른 벡터(와 )로 분해  **② 직교 벡터 성질**   1. 두 벡터의 합은 **목표벡터**이다.   [ a ]   1. 하나의 벡터는 기준벡터와 [ b ]이고, 다른 하나의 벡터는 기준벡터와 [ c ]이다.   **③ 직교 투영법(Orthogonal projection)을 통한 계산.**  직교 투영법을 적용하면 다음과 같다.  **※** 직교 투영법의 결과는 **스칼라**이지만 직교 분해의 결과는 **크기를 조정한 벡터**이다.  **④ 직교 벡터의 성질을 통한 계산.**  두 벡터의 합은 목표벡터이다.  →  →  →  **⑤ 증명 방법**  수직 성분과 기준벡터의 내적이 [ f ]임을 증명.  →  → | a.  b. 직교()  c. 평행()  d.  e.  f. |
| 5 | **< 요약 >**   * 하나의 수학적 대상을 다른 수학적 대상들로 **분해**한다. * 분해의 세부적인 내용은 **제약 조건**(현제는 기준벡터와 직교 & 평행)에 따라 달라진다. * 다른 제약 조건의 경우(**분석 목표가 다름**) 동일 벡터라도 다른 방식으로 분해한다. | |

**1.7 정리**

* 아무리 복잡한 계산이라도 선형대수학에서는 결국 간단한 연산으로 이루어져 있으며 대부분은 기하학적 직관으로 이해할 수 있다.
* 이번 장에서 배운 내용들은 책의 나머지 부분과 응용 선형대수학자로서 앞으로 많은 도움이 된다.

**\*\* 요점정리 \*\***

|  |  |  |
| --- | --- | --- |
| 1 | **< 요점정리 >**   * 벡터는 열 또는 행에 숫자를 [ a ]한 것이다. 벡터의 원소 수를 [ b ]이라고 하며, 벡터는 차원과 동일한 수의 [ c ]을 가진 기하학적 공간에서 **하나의 선**으로 나타낼 수 있다. * 덧셈, 뺄셈, 아다마르곱과 같은 벡터 산술 연산은 [ d ]별로 계산한다. * 내적은 차원이 같은 두 벡터 간의 [ e ]를 인코딩한 단일 숫자로, **원소별로 곱하고 합**해서 구한다. * 두 벡터가 직교하면 내적은 [ f ]이며 기하학적으로 벡터가 [ g ]으로 만나는 것을 의미한다. * 직교벡터 분해는 하나의 벡터를 기준벡터와 [ h ]하는 벡터, [ i ]한 벡터로 나누는 것이다. 분해 공식은 기하학적으로 도출될 수 있지만, 공식이 내포한 개념인 ‘[ j ]’이라는 문구를 기억해야 한다. | a. 나열  b. 차원(dimensionality)  c. 축  d. 원소  e. 관계  f. 0  g. 직각  h. 직교  i. 평행  j. 크기에 대한 매핑 |