이 장의 목적

짧은 보충 강의:

- 비정방 행렬(nonsquare matrices)의 기하학적 의미
- 서로 다른 차원 사이의 변환
- 지금까지 배운 개념의 자연스러운 확장

왜 중요한가?

- 행렬 곱셈을 더 깊이 이해
- 선형 시스템을 일반화
- 내적(dot product)의 기초 (다음 장)

지금까지 배운 것

정방 행렬만 다룸:

- 2×2: 2D → 2D
- 3×3: 3D → 3D
- n×n: nD → nD

같은 차원 내 변환:

```
입력 차원 = 출력 차원
```

차원 간 변환의 필요성

질문: "서로 다른 차원 사이의 변환은 어떻게 표현하나?"

예:

- 2D → 3D: 평면을 3차원 공간에 임베딩
- 3D → 2D: 3차원을 평면에 투영
- 3D → 1D: 공간을 직선에 투영

여전히 선형:

- 직선은 직선으로
- 격자선 평행 유지
- 원점 고정

행렬 크기 해석

m×n 행렬:

```
- n개 열: 입력 차원 (n차원 벡터)
- m개 행: 출력 차원 (m차원 벡터)
```

읽는 법:

```
m×n 행렬 = nD → mD 변환
```

예:

- 3×2: 2D → 3D
- 2×3: 3D → 2D
- 1×3: 3D → 1D
- 4×2: 2D → 4D

2D → 3D 변환

예제: 3×2 행렬

```
A = [2   0]  (3개 행, 2개 열)
    [-1  1]
    [-2  1]
```

의미:

- **2개 열**: 입력 공간이 2D (î, ĵ)
- **3개 행**: 출력이 3D (각 좌표가 3개)

변환:

```
î = [1, 0] → [2, -1, -2]  (첫 번째 열)
ĵ = [0, 1] → [0, 1, 1]    (두 번째 열)
```

기하학적 의미:

입력 공간: 2D 평면 (xy평면)

출력 공간: 3D 공간

변환 과정:

1. 2D 평면의 기저벡터 î, ĵ
2. 각각이 3D 공간의 어디로 가는지 추적
3. 2D 평면 전체가 3D 공간의 평면으로 매핑

열공간:

```
Col(A) = span([2, -1, -2], [0, 1, 1])
```

- 3D 공간을 지나는 2D 평면
- 원점을 지남
- 이 평면이 출력의 전부

시각화:

```
입력 (2D):          출력 (3D):
    ĵ                   z
    ↑                   ↑
    |                  /|
    └──→ î            / |
                     /  |
                    ────→ y
                     x
```

2D 격자 → 3D 평면에 "붙음"

Full Rank 개념

이 경우:

```
rank(A) = 2
```

왜 Full Rank?

- 입력 공간의 차원 = 2
- 열공간의 차원 = 2
- 정보 손실 없음

일반적 정의:

```
Full rank: rank = min(m, n)
```

- 3×2 행렬: full rank는 2
- 2×3 행렬: full rank는 2
- 5×7 행렬: full rank는 5

행렬-벡터 곱셈

2D 벡터를 3D로:

```
[2   0] [x]   [2x + 0y]     [2]     [0]
[-1  1] [y] = [-x + y ] = x[-1] + y[1]
[-2  1]       [-2x + y]     [-2]    [1]
```

의미:

- x, y = 2D 좌표
- 결과 = 3D 좌표
- 2D 평면의 점 → 3D 평면의 점

예:

```
[2   0] [3]   [6 ]
[-1  1] [1] = [-2]
```

```
[-2  1]       [-5]
```

2D의 점 (3, 1) → 3D의 점 (6, -2, -5)

3D → 2D 변환

예제: 2×3 행렬

```
B = [3  1  4]  (2개 행, 3개 열)
    [1  5  9]
```

의미:

- **3개 열**: 입력이 3D (î, ĵ, k̂)
- **2개 행**: 출력이 2D

변환:

```
î = [1, 0, 0] → [3, 1]  (첫 번째 열)
ĵ = [0, 1, 0] → [1, 5]  (두 번째 열)
k̂ = [0, 0, 1] → [4, 9]  (세 번째 열)
```

기하학적 의미:

입력: 3D 공간

출력: 2D 평면

변환 과정:

- 3D의 세 기저벡터가 2D로
- 3D 공간 전체가 2D 평면으로 "투영"

열공간:

```
Col(B) = span([3, 1], [1, 5], [4, 9])
```

하지만 2D에서는:

```
실제 span = 2D 전체 (세 벡터가 2D를 span)
```

Full Rank?

```
rank(B) = 2 (full rank)
```

입력 3D이지만 출력은 최대 2D

3D → 1D 변환

예제: 1×3 행렬

```
C = [1  2  3]  (1개 행, 3개 열)
```

의미:

- **3개 열**: 입력이 3D
- **1개 행**: 출력이 1D (수직선)

변환:

```
î = [1, 0, 0] → 1
ĵ = [0, 1, 0] → 2
k̂ = [0, 0, 1] → 3
```

기하학적 의미:

3D 공간 → 수직선 (1D)

행렬-벡터 곱셈:

```
[1  2  3] [x]
          [y] = x + 2y + 3z
          [z]
```

결과는 **숫자** (스칼라)

시각화:

```
3D:                 1D:
  z                 ━━━━━→
  ↑                 (수직선)
 /|
/ |
──→ y
x

3D 점 → 1D 점(숫자)
```

예:

```
[1  2  3] [1]
          [0] = 1 + 0 + 6 = 7
          [2]
```

3D 점 (1, 0, 2) → 1D 점 7

특별한 경우: 1×n 행렬

1×2 행렬:

```
D = [3  -1]
```

의미: 2D → 1D (수직선)

변환:

```
[3  −1] [x]
        [y] = 3x − y
```

기하학적:

- 2D 평면의 각 점 → 숫자
- 평면 → 직선으로 투영

연결: 이것이 바로 **내적(dot product)**!

```
[3, −1] · [x, y] = 3x − y
```

1×n 행렬 = n차원 벡터와의 내적 (다음 장에서 상세히)

행렬 곱셈의 확장

차원 호환성:

```
(m×n) × (n×p) = (m×p)
```

필수 조건:

- 왼쪽 행렬의 열 수 = 오른쪽 행렬의 행 수

예:

```
(3×2) × (2×4) = (3×4) ✓
(2×3) × (3×1) = (2×1) ✓
(3×2) × (3×4) = ??? ✗
```

기하학적 의미:

예: (3×2) × (2×4) = (3×4)

```
Step 1: 4D → 2D (2×4 행렬)
Step 2: 2D → 3D (3×2 행렬)
합성: 4D → 3D (3×4 행렬)
```

중간 차원(2D)이 일치해야 합성 가능!

2D → 1D 변환의 시각화

설정:

```
[a  b] [x]
       [y] = ax + by
```

시각화 방법:

입력 공간: 2D 평면에 균등한 점들

변환:

```
각 점 (x, y) → 숫자 (ax + by)
```

출력 공간: 수직선에 점들

관찰:

- 2D 직선 → 1D 점
- 평행선 → 균등 간격 유지
- 원점 → 0

예:

```
[1  0] [x]
       [y] = x
```

- 세로선(x = c) → 점 c
- 평면이 수직선에 "접힘"

계수(Rank)의 재해석

비정방 행렬의 경우:

```
rank = 출력 공간의 차원
```

예:

- 3×2 행렬, rank = 2: 3D 공간의 2D 평면
- 2×3 행렬, rank = 2: 2D 평면 전체
- 1×3 행렬, rank = 1: 수직선

Full Rank:

```
rank = min(행 수, 열 수)
```

- 정보 손실이 최소
- 입력 차원을 최대한 보존

응용 예시

1. 컴퓨터 그래픽스:

3D → 2D 투영:

- 3D 모델을 2D 화면에

- 2×3 행렬로 표현
- 카메라 투영 행렬

2. 데이터 과학:

차원 축소:

- 고차원 데이터 → 저차원
- 예: PCA (주성분 분석)
- m×n 행렬 (m < n)

3. 로보틱스:

센서 데이터:

- 3D 공간의 측정
- 1D 거리 센서
- 1×3 행렬로 투영

4. 물리학:

투영 연산자:

- 3D 벡터를 평면에 투영
- 3×3 행렬이지만 rank = 2
- 한 차원 "제거"

행렬 표기법 해독

행렬을 볼 때:

3×5 행렬:

```
"5차원 벡터를 입력으로 받아
 3차원 벡터를 출력"
```

7×2 행렬:

```
"2차원 벡터를 입력으로 받아
 7차원 벡터를 출력"
```

1×n 행렬:

```
"n차원 벡터를 입력으로 받아
 숫자(스칼라)를 출력"
```

다음 장 예고: 내적

1×n 행렬의 특별함:

```
[a₁  a₂  ...  aₙ] [x₁]
                   [x₂]  = a₁x₁ + a₂x₂ + ... + aₙxₙ
                   [⋮ ]
                   [xₙ]
```

이것은:

```
[a₁, a₂, ..., aₙ] · [x₁, x₂, ..., xₙ]
```

내적 = 1×n 행렬 변환

다음 장에서 이 관계를 깊이 탐구!

핵심 직관 정리

1. 행렬 크기 = 입출력 차원

```
m×n 행렬: nD → mD
```

2. 열 = 입력 기저벡터의 도착지

- n개 열 = n개 입력 기저
- 각 열 = m차원 좌표

3. 차원 간 변환도 선형

- 직선 → 직선
- 격자선 평행
- 원점 고정

4. Full rank = 입력 차원 보존

```
rank = min(m, n)
```

5. 행렬 곱셈 = 변환 합성

- 차원 호환성 중요
- (m×n)(n×p) = (m×p)

---

2. 퀴즈

문제 1

m×n 행렬의 기하학적 의미를 설명하시오. 특히 m(행의 개수)과 n(열의 개수)이 각각 무엇을 나타내는지, 그리고 왜 그런지 서술하시오.

문제 2

다음 행렬을 기하학적으로 해석하시오:

```
A = [2  -1   3]
    [1   0   4]
```

입력과 출력의 차원, 각 기저벡터가 어디로 매핑되는지, 그리고 열공간이 무엇인지 설명하시오.

문제 3

3×2 행렬이 2D → 3D 변환을 나타낸다고 할 때: (a) 이 변환이 full rank인 경우 열공간은 무엇인가? (b) 왜 이 경우를 "full rank"라고 부르는가? (c) rank가 1이라면 어떤 의미인가?

문제 4

1×n 행렬(1개 행, n개 열)의 특별한 의미를 설명하시오. 이러한 행렬과 벡터의 곱셈이 무엇을 나타내는지, 그리고 이것이 다음 장에서 배울 내적(dot product)과 어떤 관계가 있는지 서술하시오.

문제 5

행렬 곱셈 (m×n) × (p×q)가 정의되기 위한 조건을 차원 간 변환의 합성 관점에서 설명하시오. 특히 n = p여야 하는 이유를 기하학적으로 서술하고, 결과 행렬의 크기가 m×q인 이유를 설명하시오.

---

3. 퀴즈 답

문제 1 답안

m×n 행렬의 기하학적 의미:

일반적 형태:

```
m×n 행렬 = nD → mD 변환
```

의미:

- **n차원 벡터를 입력**으로 받음
- **m차원 벡터를 출력**으로 내보냄

---

n(열의 개수)의 의미:

1. 입력 공간의 차원:

```
n개의 열 = n개의 기저벡터
```

이유:

- n차원 공간은 n개의 기저벡터 필요
- $\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_n$

- 각 기저벡터가 어디로 가는지를 기록

2. 각 열 = 변환된 기저벡터:

n차원 입력 벡터:

```
x = [x₁]    = x₁î₁ + x₂î₂ + ... + xₙîₙ
    [x₂]
    [⋮ ]
    [xₙ]
```

변환 후:

```
Ax = x₁(변환된 î₁) + x₂(변환된 î₂) + ... + xₙ(변환된 îₙ)
   = x₁(1열) + x₂(2열) + ... + xₙ(n열)
```

따라서:

- n개 열 = n개 입력 기저벡터의 도착지
- 입력 차원 = n

---

m(행의 개수)의 의미:

1. 출력 공간의 차원:

```
m개의 행 = m차원 좌표
```

이유:

- 각 변환된 기저벡터는 m차원 공간에 존재
- m개 좌표로 표현

2. 각 행 = 출력 좌표의 한 성분:

변환된 기저벡터:

```
변환된 î₁ = [a₁₁]  ← m개 좌표
            [a₂₁]
            [⋮  ]
            [aₘ₁]
```

따라서:

- m개 행 = m차원 출력 좌표
- 출력 차원 = m

구체적 예:

3×2 행렬:

```
A = [a  b]  (3행, 2열)
    [c  d]
    [e  f]
```

해석:

- **2열:** 입력은 2D (î, ĵ)
- **3행:** 출력은 3D

변환:

```
2D 평면 --A--> 3D 공간
```

기저벡터:

```
î = [1, 0] → [a, c, e]  (3D)
ĵ = [0, 1] → [b, d, f]  (3D)
```

행렬-벡터 곱:

```
[a  b] [x]   [ax + by]     [a]     [b]
[c  d] [y] = [cx + dy] = x[c] + y[d]
[e  f]       [ex + fy]     [e]     [f]
```

2D 입력 (x, y) → 3D 출력

일반적 원리:

행렬의 구조:

```
[← m차원 →]
  a₁₁  a₁₂  ...  a₁ₙ  ↑
  a₂₁  a₂₂  ...  a₂ₙ  |
   ⋮     ⋮    ⋱    ⋮   n개 열
  aₘ₁  aₘ₂  ...  aₘₙ  ↓
```

각 열:

- n차원 입력 기저벡터의 변환
- m차원 좌표로 표현

행렬 전체:

- n차원 → m차원 변환 완전 설명

결론: 행렬의 크기 m×n은 변환의 입출력 차원을 직접적으로 나타내며, 이는 선형변환의 본질적인 특성입니다.

문제 2 답안

주어진 행렬:

```
A = [2  -1   3]  (2행, 3열)
    [1   0   4]
```

---

(1) 입력과 출력의 차원:

행렬 크기 분석:

- **3개 열:** 입력이 3차원
- **2개 행:** 출력이 2차원

변환:

```
3D → 2D
```

입력: 3차원 벡터 [x, y, z] 출력: 2차원 벡터

---

(2) 각 기저벡터의 매핑:

3차원 표준 기저벡터:

```
î = [1, 0, 0]
ĵ = [0, 1, 0]
k̂ = [0, 0, 1]
```

변환 후 위치:

î의 변환 (첫 번째 열):

```
î = [1, 0, 0] → [2, 1]
```

ĵ의 변환 (두 번째 열):

```
ĵ = [0, 1, 0] → [-1, 0]
```

k̂의 변환 (세 번째 열):

```
k̂ = [0, 0, 1] → [3, 4]
```

---

(3) 기하학적 의미:

3D 공간의 기저벡터들:

- x축 단위벡터 → 2D의 [2, 1]
- y축 단위벡터 → 2D의 [-1, 0]
- z축 단위벡터 → 2D의 [3, 4]

변환 과정:

- 3D 공간 전체가 2D 평면으로 "투영"
- 3D의 격자선이 2D 평면에 그려짐
- 차원 축소

---

(4) 행렬-벡터 곱셈:

```
A [x]   [2  −1   3] [x]   [2x − y + 3z]
  [y] = [1   0   4] [y] = [x + 4z     ]
  [z]                [z]
```

선형결합 형태:

```
결과 = x[2] + y[−1] + z[3]
        [1]      [0]      [4]

     = x(변환된 î) + y(변환된 ĵ) + z(변환된 k̂)
```

---

(5) 열공간 (Column Space):

정의:

```
Col(A) = span([2, 1], [−1, 0], [3, 4])
```

분석:

세 벡터 확인:

- [2, 1]
- [-1, 0]
- [3, 4]

선형독립성:

2차원에서 최대 2개 벡터만 선형독립 가능.

확인해보자:

```
[3, 4]가 [2, 1]과 [-1, 0]의 선형결합인가?

a[2, 1] + b[-1, 0] = [3, 4]
[2a - b, a] = [3, 4]

a = 4
2a - b = 3 → 8 - b = 3 → b = 5

확인: 4[2, 1] + 5[-1, 0] = [8, 4] + [-5, 0] = [3, 4] ✓
```

따라서 세 번째 열은 앞 두 열의 선형결합!

실제 열공간:

```
Col(A) = span([2, 1], [-1, 0])
```

차원 확인:

[2, 1]과 [-1, 0]이 선형독립인가?

```
a[2, 1] + b[-1, 0] = [0, 0]
[2a - b, a] = [0, 0]

a = 0
2a - b = 0 → b = 0
```

유일해 (0, 0) ⟶ 선형독립 ✓

결론:

```
Col(A) = 2D 평면 전체
rank(A) = 2
```

---

(6) Full Rank:

```
rank = 2 = min(2, 3) ✓
```

의미:

- 출력 공간(2D)을 완전히 span
- 2D 평면의 모든 점에 도달 가능
- 입력 3D이지만 출력 2D를 모두 활용

시각화:

```
입력 (3D):               출력 (2D):
     z                        y
     ↑                        ↑
```

```
    /|                      |
   / |                      |
  /  |                   ───┼───→ x
 /   |
 ────→ y                  (전체 평면)
  x

3D 공간 → 2D 평면 (전체)
```

결론: 이 행렬은 3D 공간을 2D 평면 전체로 투영하는 full rank 변환입니다.

문제 3 답안

3×2 행렬의 기하학적 분석:

일반 형태:

```
A = [a  b]  (3행, 2열)
    [c  d]
    [e  f]
```

변환: 2D → 3D

(a) Full rank인 경우의 열공간:

Full rank 조건:

```
rank(A) = min(3, 2) = 2
```

의미:

- 두 열벡터가 선형독립
- 입력 2D의 정보를 모두 보존

열벡터:

```
v₁ = [a, c, e]  (첫 번째 열)
v₂ = [b, d, f]  (두 번째 열)
```

Full rank이면: v_1과 v_2가 선형독립

- $v_2 \neq k \cdot v_1$ (어떤 스칼라 k에 대해서도)

열공간:

```
Col(A) = span(v₁, v₂)
```

기하학적 의미:

3D 공간에서:

- 두 선형독립 벡터의 span = **평면**
- 원점을 지나는 2차원 평면
- 3D 공간을 "잘라" 지나감

시각화:

```
        z
        ↑
       /|
      / |  Col(A)
     /  | (2D 평면)
    /   |  ↗
   /    | /
  ———→|/—— y
   x
```

결론: Full rank 3×2 행렬의 열공간은 **3D 공간을 지나는 2D 평면**입니다.

(b) 왜 "Full Rank"인가?

Rank의 정의:

```
rank = 열공간의 차원
```

3×2 행렬의 경우:

최대 가능 rank:

```
max rank = min(행 수, 열 수) = min(3, 2) = 2
```

이유:

1. **입력 차원 제약:**
   - 입력이 2D
   - 2개 기저벡터만
   - 최대 2차원 span 가능
2. **출력 차원 여유:**
   - 출력은 3D
   - 2차원 평면을 담기에 충분

Full rank = 최대 rank 달성:

```
rank(A) = 2 = max rank
```

의미:

- 입력 공간의 차원을 **완전히 보존**
- 정보 손실 없음
- 2D 정보가 모두 3D 평면에 담김

대조: Not full rank (rank = 1):

- 두 열벡터가 선형종속
- 예: $v_2 = 2v_1$
- 정보 손실 발생
- 2D → 1D (직선)로 압축

왜 "Full"이라는 표현?

- 입력 차원을 "완전히(fully)" 활용
- 가능한 최대치
- 정보 보존 최대화

(c) rank = 1인 경우의 의미:

조건:

```
rank(A) = 1 < min(3, 2) = 2
```

Rank deficient!

원인:

```
두 열벡터가 선형종속
v₂ = k·v₁ (어떤 k에 대해)
```

예:

```
A = [2   4]
    [1   2]
    [-1  -2]

v₂ = [4, 2, -2] = 2[2, 1, -1] = 2v₁
```

열공간:

```
Col(A) = span([2, 1, -1])
```

기하학적 의미:

3D 공간에서:

- 하나의 벡터의 span = **직선**

- 원점을 지나는 1차원 직선

변환 과정:

```
2D 평면 --A--> 3D 직선
```

시각화:

```
입력 (2D):              출력 (3D):
    ĵ                        z
    ↑                        ↑
    |                       /
    └──→ î                 /  Col(A)
                          /   (직선)
                        ────→ y
                         x
```

정보 손실:

- 2D → 1D 압축
- 한 차원 "소멸"
- 2D 평면의 무수히 많은 점들이 직선 위 같은 점으로

영공간:

```
Null(A) ≠ {0}
```

- 원점으로 가는 벡터들이 존재
- 직선에 "수직"인 방향
- 1차원 직선

계수-영성 정리:

```
rank + nullity = 2
1 + 1 = 2 ✓
```

응용:

- 투영(projection) 연산
- 정보 압축
- 차원 축소

결론: rank = 1인 3×2 행렬은 2D 평면을 3D 공간의 1D 직선으로 압축하며, 이는 정보 손실을 동반합니다.

문제 4 답안

1×n 행렬의 특별한 의미:

일반 형태:

```
A = [a₁  a₂  a₃  ...  aₙ]  (1행, n열)
```

---

(1) 기하학적 해석:

차원 분석:

- **n개 열:** 입력이 n차원
- **1개 행:** 출력이 1차원 (수직선, 스칼라)

변환:

```
nD → 1D (숫자)
```

의미:

- n차원 벡터를 받아서
- **숫자 하나**를 출력

---

(2) 행렬-벡터 곱셈:

```
[a₁  a₂  ...  aₙ] [x₁]
                   [x₂]  = a₁x₁ + a₂x₂ + ... + aₙxₙ
                   [⋮ ]
                   [xₙ]
```

결과:

- 벡터가 아닌 **스칼라**
- 각 성분의 가중합

---

(3) 기하학적 의미:

n차원 공간 → 수직선

변환 과정:

예 (2D → 1D):

```
[3  -1] [x]
        [y] = 3x - y
```

시각화:

입력 (2D 평면):

```
    y
    ↑
    |
    |
    └──→ x
```

출력 (1D 수직선):

```
━━━━━━━━━━━→
(숫자들)
```

각 2D 점 (x, y) → 숫자 (3x - y)

특성:

- 2D 직선 → 1D 점
- 평행선 → 균등 간격 유지
- 원점 → 0

---

(4) 내적(Dot Product)과의 관계:

핵심 관찰:

1×n 행렬과 n차원 벡터의 곱:

$$\begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = a_1x_1 + a_2x_2 + ... + a_nx_n$$

이것은 정확히 **내적**!

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = a_1x_1 + a_2x_2 + ... + a_nx_n$$

표기법 변환:

행 벡터 (1×n 행렬):

$$\begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix}$$

열 벡터 (n×1 벡터):

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

```
[ ⋮ ]
[aₙ]
```

관계:

```
행 벡터 × 열 벡터 = 내적
```

---

(5) 다음 장 예고:

내적의 기하학적 의미:

투영(Projection):

```
a · x = |a| |x| cos(θ)
```

- x를 a 방향으로 투영
- 1×n 행렬 = 특정 방향으로의 투영 연산

선형 함수로서의 내적:

모든 선형 함수 f: $\mathbb{R}^n \to \mathbb{R}$는:

```
f(x) = a · x (어떤 벡터 a에 대해)
```

즉, 1×n 행렬로 표현 가능!

쌍대성(Duality):

- n차원 벡터 ↔ 1×n 행렬
- 기하학적 객체 ↔ 선형 함수
- 다음 장의 핵심 주제

---

(6) 구체적 예:

2D 예:

```
[2  3] [x]
       [y] = 2x + 3y
```

기하학적:

- [2, 3] 방향으로의 "투영"
- 각 점의 "높이" 측정

3D 예:

```
[1  0  1] [x]
          [y]  = x + z
          [z]
```

기하학적:

- xz평면에 투영 후 거리
- y 좌표는 무시

결론:

1×n 행렬은:

1. **n차원 → 숫자** 변환
2. **내적 연산**과 동일
3. **선형 함수**를 행렬로 표현
4. **다음 장(내적과 쌍대성)**의 기초

이는 선형대수에서 가장 중요한 통찰 중 하나입니다!

문제 5 답안

행렬 곱셈의 차원 호환성:

조건:

```
(m×n) × (p×q)가 정의되려면:
n = p
```

---

(1) 기하학적 설명:

행렬 곱셈 = 변환의 합성

A: m×n 행렬

```
n차원 → m차원 변환
```

B: p×q 행렬

```
q차원 → p차원 변환
```

합성 AB:

```
먼저 B 적용, 그 다음 A 적용
```

변환 순서:

```
q차원 --B--> p차원 --A--> m차원
```

차원 연결:

```
B의 출력 차원 = A의 입력 차원
즉, p = n
```

왜 필요한가?

B의 출력이 A의 입력으로 들어가야 하므로:

- B가 p차원 벡터를 출력
- A가 n차원 벡터를 입력으로 받음
- **p = n이어야 연결 가능!**

---

(2) 구체적 예:

가능한 경우:

예 1:

```
A: 3×2 (2D → 3D)
B: 2×4 (4D → 2D)

AB: (3×2) × (2×4)
        ↑     ↑
     같음! (n = p = 2)
```

합성:

```
4D --B--> 2D --A--> 3D
```

결과:

```
AB: 3×4 (4D → 3D)
```

예 2:

```
A: 2×3 (3D → 2D)
B: 3×5 (5D → 3D)

AB: (2×3) × (3×5)
        ↑     ↑
     같음! (n = p = 3)
```

합성:

```
5D --B--> 3D --A--> 2D
```

결과:

```
AB: 2×5 (5D → 2D)
```

불가능한 경우:

예:

```
A: 3×2 (2D → 3D)
B: 4×5 (5D → 4D)

AB: (3×2) × (4×5)
          ↑      ↑
      2 ≠ 4
```

문제:

```
5D --B--> 4D --A?--> ???
```

- B는 4차원 벡터를 출력
- A는 2차원 벡터를 입력으로 원함
- **차원 불일치!**

합성 불가능 → 행렬 곱셈 정의 안 됨

---

(3) 결과 행렬의 크기: m×q

왜 m×q인가?

합성 변환:

```
q차원 --B--> (중간) --A--> m차원
```

입력: q차원 **출력:** m차원

결과 행렬:

```
AB는 q차원 → m차원 변환
따라서 m×q 행렬
```

행과 열의 의미:

- **q개 열:** 입력 차원 (B의 입력)
- **m개 행:** 출력 차원 (A의 출력)

---

(4) 파이프 비유:

행렬 = 파이프의 한 부분

```
A: [n차원 입구] → [m차원 출구]
B: [q차원 입구] → [p차원 출구]
```

연결하려면:

```
B의 출구 크기 = A의 입구 크기
즉, p = n
```

연결 후:

```
AB: [q차원 입구] → [m차원 출구]
```

파이프가 맞아야 연결 가능!

---

(5) 일반적 규칙:

차원 확인:

```
(m × n) × (p × q)
      └─────┘
        ┬
       같아야 함
```

결과:

```
m × q
└─┘

┌────┐
(m×n)(n×q) = (m×q)
```

외곽 차원이 남음:

- 최종 입력: q (B의 입력)
- 최종 출력: m (A의 출력)

---

(6) 3개 이상 행렬의 곱:

예:

```
A: 2×3
B: 3×4
C: 4×5

ABC = (2×3)(3×4)(4×5)
```

순서대로 확인:

```
BC = (3×4)(4×5) = (3×5) ✓
A(BC) = (2×3)(3×5) = (2×5) ✓
```

또는:

```
AB = (2×3)(3×4) = (2×4) ✓
(AB)C = (2×4)(4×5) = (2×5) ✓
```

최종:

```
5D → 4D → 3D → 2D
```

---

(7) 실용적 의미:

컴퓨터 그래픽스:

```
모델링: 3D → 3D (회전)
뷰: 3D → 3D (카메라)
투영: 3D → 2D (화면)

합성: (2×3)(3×3)(3×3)
```

차원이 맞아야 합성 가능!

데이터 과학:

```
데이터: m×n (m개 샘플, n개 특징)
변환 1: n×p
변환 2: p×q

최종: m×q
```

결론:

행렬 곱셈의 차원 호환성(n = p)은:

1. **변환 합성의 기하학적 필연성**
2. **출력-입력 차원 일치**
3. **파이프 연결의 자연스러운 조건**

결과 크기(m×q)는:

1. **최종 입출력 차원**
2. **외곽 차원의 보존**

이는 선형변환의 합성이라는 기하학적 의미에서 자연스럽게 나옵니다!