2. 행렬

2.1 행렬의 곱셈

- 행렬 곱셈(matrix multiplication)은 **i번째 행벡터와 j번째 열벡터 사이의 내적을 성분으로 가지는 행렬**을 계산한다.
- 행렬곱은 X의 열 개수와 Y의 행 개수가 같아야 한다.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1\ell} \\ y_{21} & y_{22} & \cdots & y_{2\ell} \\ \vdots & \vdots & & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{m\ell} \end{bmatrix}$$

$$\mathbf{XY} = \left(\sum_k x_{ik} y_{kj} \right)$$

2.2 행렬의 내적

- 넘파이의 `np.inner`는 **i번째 행벡터와 j번째 행벡터 사이의 내적을 성분으로 가지는 행렬**을 계산한다.
- 수학에서 말하는 내적과는 다르다.
    - 수학에서는 보통 $tr(XY^T)$ 을 내적으로 계산한다.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1m} \\ y_{21} & y_{22} & \cdots & y_{2m} \\ \vdots & \vdots & & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nm} \end{bmatrix}$$

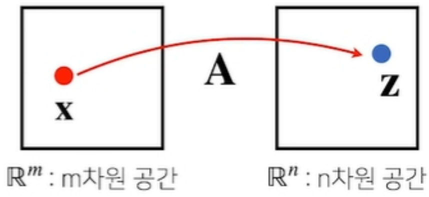
$$\mathbf{XY}^{\top} = \left(\sum_k x_{ik} y_{jk} \right)$$

2.3 연산자(operator)로서의 행렬

- 행렬은 벡터공간에서 사용되는 연산자(operator)로 이해한다.
- 행렬곱을 통해 벡터를 다른 차원의 공간으로 보낼 수 있다.
- 행렬곱을 통해 **패턴을 추출**할 수 있고 **데이터를 압축**할 수 있다.
- 모든 선형변환(linear transform)은 행렬곱으로 계산할 수 있다.

$$z_i = \sum_j a_{ij} x_j$$

$$\underset{\mathbf{z}}{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}} = \underset{\mathbf{A}}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}$$

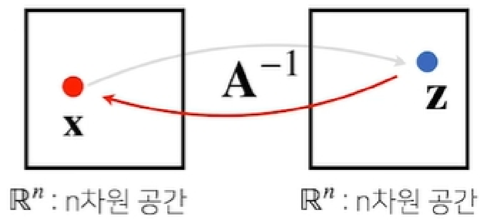


2.4 역행렬

- `np.linalg.inv()`
- 어떤 행렬 A 의 연산을 거꾸로 되돌리는 행렬을 역행렬(inverse matrix)이라고 부르고 A^{-1} 라 표기한다.
- 역행렬을 구하기 위한 조건 (2가지)
  - 행과 열 숫자가 같아야 한다. ($n = m$)
  - 행렬식(determinant)이 0이 아니어야 한다. ($det(A) \neq 0$)

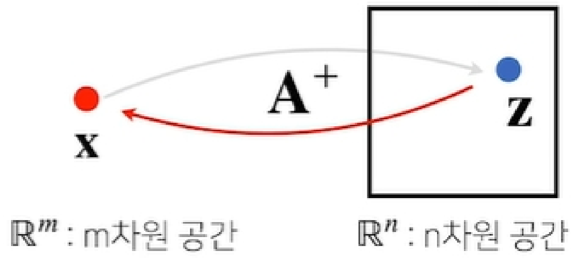
$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \underset{\text{항등행렬}}{\mathbf{I}}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$



2.5 유사역행렬

- `np.linalg.pinv()`
- 역행렬을 계산할 수 없다면 **유사역행렬(pseudo-inverse)** 또는 **무어-펜로즈(Moore-Penrose) 역행렬** A^+을 이용한다.
- $n \geq m$ 인 경우 (행의 갯수가 열의 갯수보다 많은 경우)
  - $A^+ = \left(A^T A\right)^{-1} A^T$
  - $A^+ A = I$ 가 성립
- $n \leq m$ 인 경우 (행의 갯수가 열의 갯수보다 적은 경우)
  - $A^+ = A^T \left(A A^T\right)^{-1}$
  - $A A^+ = I$ 만 성립



x
A+
z
ℝᵐ : m차원 공간
ℝⁿ : n차원 공간