Linear Algebra 1

Industrial AI Lab.

Prof. Seungchul Lee

Some materials from linear algebra review by Prof. Zico Kolter from CMU

Linear Equations

- Set of linear equations (two equations, two unknowns)

$$\begin{aligned} 4x_1 - 5x_2 &= -13 \\ -2x_1 + 3x_2 &= 9 \end{aligned}$$

Linear Equations

- Solving linear equations
  - Two linear equations

$$\begin{aligned} 4x_1 - 5x_2 &= -13 \\ -2x_1 + 3x_2 &= 9 \end{aligned}$$

  - In a vector form, $Ax = b$, with

$$A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}$$

Linear Equations

- Solving linear equations
  - Two linear equations

$$4x_1 - 5x_2 = -13$$
$$-2x_1 + 3x_2 = 9$$

  - In a vector form, $Ax = b$, with

$$A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}$$

  - Solution using inverse

$$Ax = b$$
$$A^{-1}Ax = A^{-1}b$$
$$x = A^{-1}b$$

Linear Equations

- Solving linear equations
  - Two linear equations

$$\begin{aligned} 4x_1 - 5x_2 &= -13 \\ -2x_1 + 3x_2 &= 9 \end{aligned}$$

  - In a vector form, $Ax = b$, with

$$A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}$$

  - Solution using inverse

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

  - Don’t worry here about how to compute matrix inverse
  - We will use a numpy to compute

Linear Equations in Python

$$\begin{aligned} 4x_1 - 5x_2 &= -13 \\ -2x_1 + 3x_2 &= 9 \end{aligned}$$

$$A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}$$

```
import numpy as np
```

```
A = np.array([[4, -5],
              [-2, 3]])
b = np.array([[-13],
              [9]])

x = np.linalg.inv(A).dot(b)
print(x)
```

```
[[ 3.]
 [ 5.]]
```

```
A = np.asmatrix(A)
b = np.asmatrix(b)

x = A.I*b
print(x)
```

```
[[ 3.]
 [ 5.]]
```

System of Linear Equations

- Consider a system of linear equations

$$\begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ y_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ &\vdots \\ y_m &= a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{aligned}$$

- Can be written in a matrix form as $y = Ax$, where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Elements of a Matrix

- Can write a matrix in terms of its columns

$$A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}$$

- Careful, a_i here corresponds to an entire vector $a_i \in \mathbb{R}^m$
- Similarly, can write a matrix in terms of rows

$$A = \begin{bmatrix} - & b_1^T & - \\ - & b_2^T & - \\ & \vdots & \\ - & b_m^T & - \end{bmatrix}$$

- $b_i \in \mathbb{R}^n$

Vector-Vector Products

- Inner product: $x, y \in \mathbb{R}^n$

$$x^T y = \sum_{i=1}^{n} x_i \, y_i \qquad \in \mathbb{R}$$

```
x = np.array([[1],
              [1]])
y = np.array([[2],
              [3]])

print(x.T.dot(y))
```

```
[[5]]
```

```
x = np.asmatrix(x)
y = np.asmatrix(y)

print(x.T*y)
```

```
[[5]]
```

Matrix-Vector Products

- $A \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n \Leftrightarrow Ax \in \mathbb{R}^m$

- Writing A by rows, each entry of Ax is an *inner product* between x and a row of A

$$A = \begin{bmatrix} - & b_1^T & - \\ - & b_2^T & - \\ & \vdots & \\ - & b_m^T & - \end{bmatrix}, \qquad Ax \in \mathbb{R}^m = \begin{bmatrix} b_1^T x \\ b_2^T x \\ \vdots \\ b_m^T x \end{bmatrix}$$

Matrix-Vector Products

- $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n \Longleftrightarrow Ax \in \mathbb{R}^m$

- Writing A by columns, Ax is a *linear combination* of the columns of A, with coefficients given by x

$$A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}, \qquad Ax \in \mathbb{R}^m = \sum_{i=1}^{n} a_i x_i$$

Symmetric Matrices

- Symmetric matrix:

$$A \in \mathbb{R}^{n \times n} \text{ with } A = A^T$$

- Arise naturally in many settings

- For $A \in \mathbb{R}^{m \times n}$,

$$A^T A \in \mathbb{R}^{n \times n} \text{ is symmetric}$$

Norms (Strength or Distance in Linear Space)

- A vector norm is any function $f: \mathbb{R}^n \Longrightarrow \mathbb{R}$ with

1. $f(x) \geq 0$ and $f(x) = 0 \iff x = 0$
2. $f(ax) = |a| f(x)$ for $a \in \mathbb{R}$
3. $f(x + y) \leq f(x) + f(y)$

- l_2 norm

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- l_1 norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

- $\|x\|$ measures length of vector (from origin)

Norms in Python

```
x = np.array([[4],
              [3]])

np.linalg.norm(x, 2)
```

```
5.0
```

```
np.linalg.norm(x, 1)
```

```
7.0
```

Orthogonality

- Two vectors $x, y \in \mathbb{R}^n$ are *orthogonal* if

$$x^T y = 0$$

- They are *orthonormal* if

$$x^T y = 0 \quad \text{and} \quad \|x\|_2 = \|y\|_2 = 1$$

Angle between Vectors

- For any $x, y \in \mathbb{R}^n$,

$$|x^T y| \leq \|x\| \, \|y\|$$

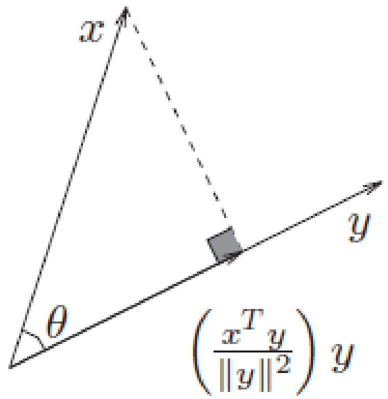
- (unsigned) angle between vectors in $\mathbb{R}^n$ defined as

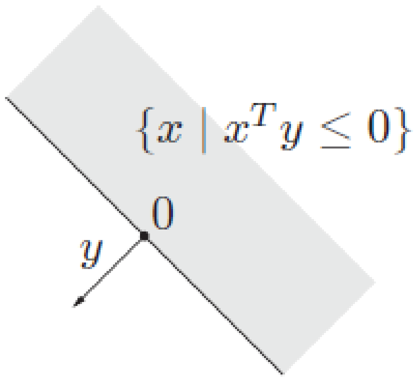
$$\theta = \angle(x, y) = \cos^{-1} \frac{x^T y}{\|x\| \|y\|}$$

$$\text{thus } x^T y = \|x\| \|y\| \cos \theta$$

Angle between Vectors

$$\theta = \angle(x, y) = \cos^{-1} \frac{x^T y}{\|x\| \|y\|}$$





- $\{x | x^T y \leq 0\}$ defines a half space with outward normal vector y, and boundary passing through 0