2.4 Vector Spaces

Groups (군)

Groups(군)은 **구조화된 연산의 법칙성**을 제공하기 때문에 컴퓨터 공학에서 매우 중요한 역할을 합니다.

암호학(cryptography), 코딩 이론(coding theory), 컴퓨터 그래픽스(Grphics) 등에 사용됩니다.

정의

집합 $\mathcal{G}$와 연산 $\otimes : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ 가 $\mathcal{G}$에 대해 정의되어 있을 때, 아래의 조건을 만족하는 $G := (\mathcal{G}, \otimes)$ 를 Group(군)이라고 부릅니다.

1. *Closure* of $\mathcal{G}$ under $\otimes$: $\forall x, y \in \mathcal{G} : x \otimes y \in \mathcal{G}$
2. *Associativity:* $\forall x, y, z \in \mathcal{G} : (x \otimes y) \otimes z = x \otimes (y \otimes z)$
3. *Neutral element:* $\exists e \in \mathcal{G}\, \forall x \in \mathcal{G} : x \otimes e = x$ and $e \otimes x = x$
4. *Inverse element:* $\forall x \in \mathcal{G}\, \exists y \in \mathcal{G} : x \otimes y = e$ and $y \otimes x = e$, where e is the neutral element. We often write x^{-1} to denote the inverse element of x.

- 4가지 성질(닫혀있음, 분배법칙, 항등원, 모든 원소에 대하여 역원 존재)을 만족할 때 그 집합을 연산 $\otimes$ 에 대하여 Group이라고 합니다.
- 또한 Group G의 임의의 x, y 에 대해서 $x \otimes y = y \otimes x$ 가 보장될 경우 (연산에 대하여 교환법칙 성립) 이를 **abelian group** 이라고 합니다.

예시 Groups

Example 2.10 (Groups)
다음은 group의 관한 몇 가지 예제들입니다.

- $(\mathbb{Z}, +)$ is a group.
- $(\mathbb{N}_0, +)$ is not a group. 항등원(0)은 존재하지만 역원이 존재하지 않음 ($\mathbb{N}_0 := \mathbb{N} \cup \{0\}$).
- $(\mathbb{Z}, \cdot)$ is not a group. 항등원(1)은 존재하지만 ± 1이 아닌 모든 z에 대해 역원이 존재하지 않음.
- $(\mathbb{R}, \cdot)$ is not a group. 0에 대한 역원이 존재하지 않음.
- $(\mathbb{R}\backslash\{0\})$ is Abelian.
- $(\mathbb{N}^n, +), (\mathbb{Z}^n, +), n \in \mathbb{N}$ are Abelian if + is defined componentwise, i.e.,

$$(x_1, \cdots, x_n) + (y_1, \cdots, y_n) = (x_1 + y_1, \cdots, x_n + y_n) \tag{2.61}$$

$(x_1, \cdots, x_n)^{-1} := (-x_1, \cdots, -x_n)$ 은 역원이고, 항등원은 $e = (0, \cdots, 0)$ 입니다.

- $(\mathbb{R}^{m \times n}, +)$, the set of $m \times n$ -matrices is Abelian (with componentwise addition as defined in (2.61))
- $(\mathbb{R}^{n \times n}, \cdot)$, 즉, $n \times n$ 행렬 집합에 대해서 살펴보도록 하겠습니다.
  - Matrix multiplication 정의에 의해 Closure과 associativity 성질을 만족합니다.
  - 항등행렬(Identity matrix) $\boldsymbol{I}_n$ 은 주어진 조건에서 행렬 곱($\cdot$)에 대해 항등원입니다.
  - 만약 역행렬이 존재한다면, 즉, $\boldsymbol{A}$ 가 regular 라면, $\boldsymbol{A}^{-1}$ 은 행렬 $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ 의 역행렬입니다. 따라서, $(\mathbb{R}^{n \times n}, \cdot)$ 은 group이며, *general limear group* 이라고 부릅니다.

https://junstar92.github.io/mml-study-note/2022/07/02/ch2-4.html

역행렬이 존재하는 regular matrices $A \in \mathbb{R}^{n \times n}$ 의 집합은 group입니다.

이를 general linear group(일반 선형군)이라고 부르면 $GL(n, \mathbb{R})$로 표기합니다. (not Abelian)

Vector Spaces (벡터 공간)

Group에서는 하나의 연산에 대해서만 정의합니다.

즉, G의 원소에만 적용하는 연산을 inner operation이라고 합니다. → $+$

그러면 G 외의 집합의 원소에 대해서도 적용할 수 있는 연산을 outer operation이라고 합니다. → $\cdot$ (스칼라배)

해당 연산은 inner/outer products(내적/외적)과는 관련이 없습니다.

정의

$$+ : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$$
$$\cdot : \mathbb{R} \times \mathcal{V} \to \mathcal{V}$$

실수인 벡터 공간 $V = (V, +, \cdot)$은 위 두 연산을 포함하는 집합 $\mathcal{V}$ 입니다.

특성

1. $(\mathcal{V}, +)$ is an Abelian group
2. Distributivity:
   1. $\forall \lambda \in \mathbb{R}, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{V} : \lambda \cdot (\boldsymbol{x} + \boldsymbol{y}) = \lambda \cdot \boldsymbol{x} + \lambda \cdot \boldsymbol{y}$
   2. $\forall \lambda, \psi \in \mathbb{R}, \boldsymbol{x} \in \mathcal{V} : (\lambda + \psi) \cdot \boldsymbol{x} = \lambda \cdot \boldsymbol{x} + \psi \cdot \boldsymbol{x}$
3. Associativity (outer operation): $\forall \lambda, \psi \in \mathbb{R}, \boldsymbol{x} \in \mathcal{V} : \lambda \cdot (\psi \cdot \boldsymbol{x}) = (\lambda \psi) \cdot \boldsymbol{x}$
4. Neutral element with respect to the outer operation: $\forall \boldsymbol{x} \in \mathcal{V} : 1 \cdot \boldsymbol{x} = \boldsymbol{x}$

- 이 집합에 속해있는 원소 x를 **벡터(vector)**라고 합니다.

예시

Example 2.11 (Vector Spaces)
Let us have a look at some important examples:

- $\mathcal{V} = \mathbb{R}^n, n \in \mathbb{N}$ is a vector space with operations defined as follows:
  - Addition: $\boldsymbol{x} + \boldsymbol{y} = (x_1, \ldots, x_n) + (y_1, \ldots, y_n) = (x_1 + y_1, \ldots, x_n + y_n)$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$
  - Multiplication by scalars: $\lambda \boldsymbol{x} = \lambda(x_1, \ldots, x_n) = (\lambda x_1, \ldots, \lambda x_n)$ for all $\lambda \in \mathbb{R}, \boldsymbol{x} \in \mathbb{R}^n$
- $\mathcal{V} = \mathbb{R}^{m \times n}, m, n \in \mathbb{N}$ is a vector space with
  - Addition: $\boldsymbol{A} + \boldsymbol{B} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$ is defined elementwise for all $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{V}$
  - Multiplication by scalars: $\lambda \boldsymbol{A} = \begin{bmatrix} \lambda a_{11} & \cdots & \lambda a_{1n} \\ \vdots & & \vdots \\ \lambda a_{m1} & \cdots & \lambda a_{mn} \end{bmatrix}$ as defined in Section 2.2. Remember that $\mathbb{R}^{m \times n}$ is equivalent to $\mathbb{R}^{mn}$.
- $\mathcal{V} = \mathbb{C}$, with the standard definition of addition of complex numbers.

- n 차원의 실수 tuple 집합은, 성분끼리 더하고 , 실수배를 각 성분마다 해주는 vector space를 형성
- $m \times n$ 행렬에서는 각 성분의 덧셈, 그리고 각 성분에 스칼라배를 하는 vector space를 형성

- 복소수 또한 덧셈과 스칼라배로 vector space를 형성

Vector Subspaces (벡터 부분공간)

직관적으로 벡터 부분공간은 원래의 벡터 공간에 포함되어 있습니다.

벡터 공간의 정의된 연산을 부분공간에 적용시켜보면 그 연산의 결과가 절대로 벡터의 부분공간을 벗어나지 않게 됩니다.

예를 들어 벡터 공간을 정수, 스칼라 또한 정수를 가정하면 **짝수의 집합**은 부분공간이 됩니다.

짝수의 집합에서 덧셈과 스칼라배 연산은 계속 **부분공간**(짝수의 집합)에 머무르게 됩니다.

이는 벡터의 부분공간은 '**closed(닫혀)**'있습니다.

벡터의 부분공간은 머신 러닝에서 매우 중요한 아이디어가 됩니다.

예를 들어, ch10 에서는 벡터의 부분공간을 사용해서 차원 축소(dimensionality reduction)을 어떻게 사용하는지 보여줍니다.

정의

> **Definition 2.10** (Vector Subspace). Let $V = (\mathcal{V}, +, \cdot)$ be a vector space and $\mathcal{U} \subseteq \mathcal{V}$, $\mathcal{U} \neq \emptyset$. Then $U = (\mathcal{U}, +, \cdot)$ is called *vector subspace* of V (or *linear subspace*) if U is a vector space with the vector space operations $+$ and $\cdot$ restricted to $\mathcal{U} \times \mathcal{U}$ and $\mathbb{R} \times \mathcal{U}$. We write $U \subseteq V$ to denote a subspace U of V.

- $V = (\mathcal{V}, +, \cdot)$가 벡터 공간이고, $\mathcal{U} \subseteq \mathcal{V}$, $\mathcal{U} \neq \emptyset$ 라고 할 때, $U = (\mathcal{U}, +, \cdot)$ 을 V의 벡터 부분공간이라고 합니다.
- V의 부분공간 U를 $U \subseteq V$ 로 표기합니다.

> 1. $\mathcal{U} \neq \emptyset$, in particular: $\mathbf{0} \in \mathcal{U}$
> 2. Closure of U:
>   a. With respect to the outer operation: $\forall \lambda \in \mathbb{R}\, \forall \boldsymbol{x} \in \mathcal{U} : \lambda \boldsymbol{x} \in \mathcal{U}$.
>   b. With respect to the inner operation: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{U} : \boldsymbol{x} + \boldsymbol{y} \in \mathcal{U}$.

만약 $U \subseteq V$ 일 경우, U는 V의 많은 성질(properties)를 그대로 받게 됩니다.

예를 들어 abelian group의 성질, 분배법칙, 결합법칙, 항등원등을 모두 포함합니다.

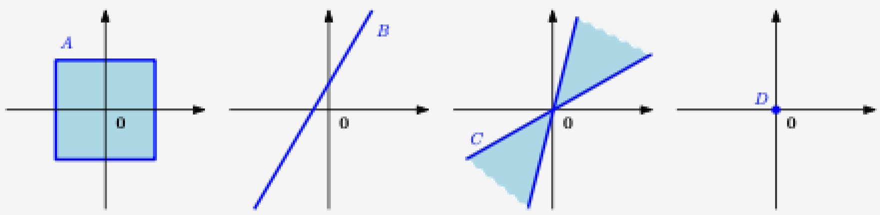
또한 벡터의 부분공간임을 확인하기 위해서는 위 두 가지를 확인하면 됩니다.

1. U는 공집합이 아니며, 0은 U의 원소이다.
2. U의 Closure
    a. 벡터의 덧셈, 스칼라배에서도 닫혀있다.

예시

Example 2.12 (Vector Subspaces)
Let us have a look at some examples:

- For every vector space V, the trivial subspaces are V itself and $\{\mathbf{0}\}$.
- Only example D in Figure 2.6 is a subspace of $\mathbb{R}^2$ (with the usual inner/outer operations). In A and C, the closure property is violated; B does not contain $\mathbf{0}$.
- The solution set of a homogeneous system of linear equations $\boldsymbol{Ax} = \mathbf{0}$ with n unknowns $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$ is a subspace of $\mathbb{R}^n$.
- The solution of an inhomogeneous system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, $\boldsymbol{b} \neq \mathbf{0}$ is not a subspace of $\mathbb{R}^n$.
- The intersection of arbitrarily many subspaces is a subspace itself.



- 벡터 공간 V에 대해서 V 자신도 부분공간이 됩니다. 또한 $\{0\}$ 도 부분공간이 됩니다. → **Closure**
- A, B, C, D 그림 중에서 D만 벡터의 부분공간이 됩니다.
    - (A, C → not closed) | (B → 0포함 X)
- $Ax = 0$ 을 만족하는 해의 집합은 R^n 의 부분공간이 됩니다.
- $Ax = b \ (b \neq 0)$ 의 해들은 부분공간이 되지 않습니다.
- 부분 공간들의 임의의 교집합에 의한 결과도 부분공간이 됩니다.

출처

- Mathmatics for Machine Learning (https://github.com/mml-book/mml-book.github.io)
- https://blog.naver.com/walk_along/222158878677
- https://junstar92.github.io/mml-study-note/2022/07/02/ch2-4.html