Zusammenfassung Linear Algebra

Joel von Rotz & Andreas Ming (Feedback)

10.06.2022

Table of contents

Vektor

Spezielle Vektoren

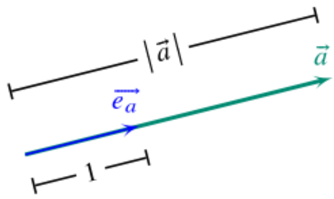
- Nullvektor $\vec{0} \to |\vec{0}| = 0$
- Gegen-, Kehr, Inverser Vektor von $\vec{a}$ ist $-\vec{a}$
- Einheitsvektor $\vec{e}$ mit em Betrag $|\vec{e}| = 1$

Kollinearität

Zwei Vektoren $\vec{a}$ & $\vec{b}$ nennt man *kollinear* oder *linear abhängig*, falls

$$\vec{a} = s \cdot \vec{b} \quad \text{mit } s \in \mathbb{R}$$

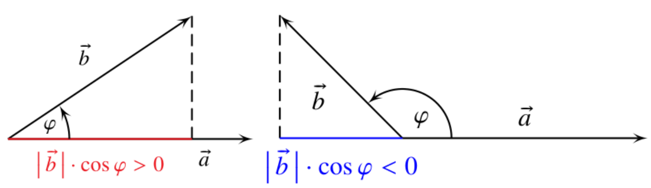
Normierung



$$\vec{e_a} = \frac{\vec{a}}{|\vec{a}|}$$

$$|\vec{e_a}| = 1$$

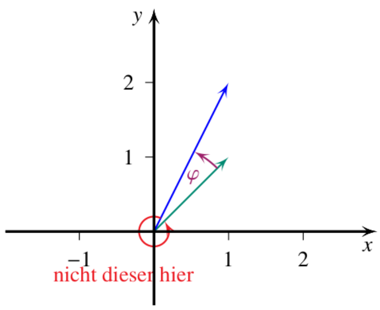
Skalarprodukt



Das Skalarprodukt entspricht der Multiplikation der Projektion $\vec{b_a}$ auf $\vec{a}$ mit $\vec{a}$

$$\vec{a} \bullet \vec{b} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \bullet \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = a_1 \cdot b_1 + \cdots + a_n \cdot b_n = \sum_{i=1}^{n} a_i \cdot b_i$$

Winkel & Orthogonalität



Beim Berechnen des Winkels zwischen zwei Vektoren

$$\varphi = \arccos \frac{x \bullet y}{\|x\| \cdot \|y\|}$$

Es gilt:

- $\vec{a} \bullet \vec{b} > 0 \quad \text{wenn} \quad \varphi < \frac{\pi}{2}$
- $\vec{a} \bullet \vec{b} < 0 \quad \text{wenn} \quad \varphi > \frac{\pi}{2}$

! Definition Orthogonalität

Sind zwei Vektoren *orthogonal*/*senkrecht* zueinander, ergibt das Skalarprodukt

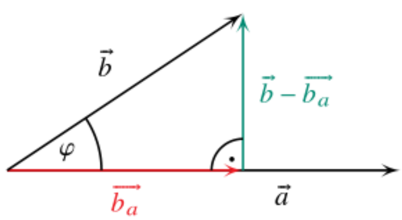
$$\vec{a} \bullet \vec{b} = 0 \quad \text{und} \quad \varphi = \frac{\pi}{2}$$

i Richtungswinkel in $\mathbb{R}^3$

$$\cos\alpha = \frac{a_x}{a} \;\&\; \cos\beta = \frac{a_y}{a} \;\&\; \cos\gamma = \frac{a_z}{a}$$

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1$$

Orthogonale Projektion

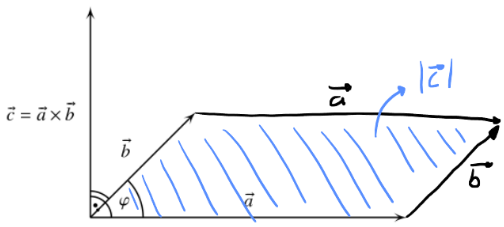


$$\vec{b_a} = \frac{\vec{a} \bullet \vec{b}}{|\vec{a}|^2} \cdot \vec{a}$$

Es gilt folgendes:

- $\vec{b_a} = 0 \quad \text{wenn} \quad \vec{b} \perp \vec{a}$
- $\vec{b_a} = \vec{b} \quad \text{wenn} \quad \varphi = 0$

Vektorprodukt / Kreuzprodukt



Tipp: Rechte Handregel → Mittelfinger $\vec{c}$, Zeigefinger $\vec{b}$ & Daumen $\vec{a}$

$$\vec{c} = \vec{a} \times \vec{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_y \cdot b_z - a_z \cdot b_y \\ a_z \cdot b_x - a_x \cdot b_z \\ a_x \cdot b_y - a_y \cdot b_x \end{bmatrix}$$

Der Betrag des Vektors $\vec{c}$ entspricht der Fläche, welche $\vec{a}$ & $\vec{b}$ aufspannen.

$$|\vec{c}| = |\vec{a} \times \vec{b}| = |\vec{a}| \cdot |\vec{b}| \cdot \sin(\varphi)$$

Nicht kommutativ & assoziativ

$$(\vec{a} \times \vec{b}) \times \vec{c} \neq \vec{a} \times (\vec{b} \times \vec{c})$$

$$\vec{a} \times \vec{b} \neq \vec{b} \times \vec{a}$$

Lineare Gleichungssystem

Good to know!

Ein lineares Gleichungssystem heisst **konsistent**, wenn es eine oder mehrere Lösungen hat, ansonsten wird es **inkonsistent** genannt (im Fall "zu viele" Lösungen).

Gaussche Eliminationsverfahren

Mit dem Gausschen Eliminationsverfahren dürfen Zeilen einer Matrix vertauscht, multipliziert oder eine Zeile zu einer anderen Zeile addiert werden.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{II} - 3\text{I} \Rightarrow \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$$

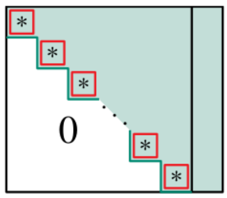
$$\begin{gathered} a_{11} \cdot x_1 + \dots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + \dots + a_{2n} \cdot x_n = b_2 \\ \dots \\ a_{m1} \cdot x_1 + \dots + a_{mn} \cdot x_n = b_m \end{gathered}$$

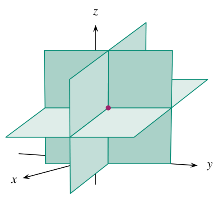
$$\Rightarrow \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \text{ und } \left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right]$$

Die Matrizen heissen *Koeffizientenmatrix*, bzw. *erweitere Koeffizientenmatrix* des Systems.

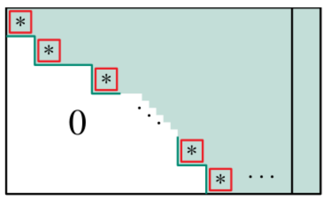
Allgemeine Lösungen

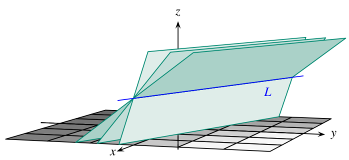
Eindeutige Lösung





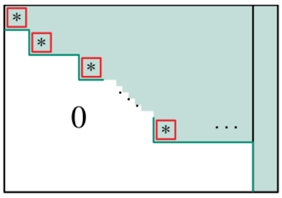
Unendliche Lösungen

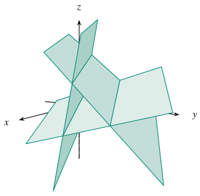




"Zu viele" Gleichungen

Im Fall "Zu viele" Gleichungen, gibt es schlicht zu viele Werte und daher gilt das LGS als **inkonsistent**.





Das System kann keine, genau eine oder unendliche viele Lösungen haben. Diese kann meistens anhand unwahren Aussagen, wie z.B. $t_2 \cdot 0 = 1$, erkannt werden. Da egal für was z.B. t_2 eingesetzt wird, kann das Resultat nie 1 werden.

! Important

Wichtig zu bemerken ist, dass folgende Lösungsmöglichkeit

$$\left[\begin{array}{cc|c} \cdots & \cdots & 1 \\ \cdots & 1 & 0 \end{array}\right]$$

nicht auf eine unwahre Aussage hindeutet. In diesem Fall wäre $\underline{t_2 = 0}$.

Matrix

i Rechenregel

- $A + B \ = \ B + A$
- $(A + B) + C \ = \ A + (B + C)$
- $(AB)C \ = \ A(BC)$
- $(A + B)C \ = \ AC + BC$
- $AB \neq BA$

Merkregel

Zeilen **z**uerst, **S**palten **s**päter

$$a_{ZS} \Rightarrow \begin{array}{c} \uparrow \\ Zeile \\ \downarrow \end{array} \begin{array}{c} \longleftarrow Spalte \longrightarrow \\ \begin{bmatrix} a_{11} & \cdots & a_{1S} \\ \vdots & & \vdots \\ a_{Z1} & \cdots & a_{ZS} \end{bmatrix} \end{array}$$

Homogenes Gleichungssystem

$$Ax = 0$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Inhomogenes Gleichungssystem

$$Ax = b$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Transponieren von Matrizen

Alle Matrizen können transponiert werden, egal ob quadratisch oder nicht!

$$A \in \mathbb{R}^{m \times n} \Rightarrow A^T \in \mathbb{R}^{n \times m}$$

$$\begin{bmatrix} 1 & a \\ 2 & b \\ 3 & c \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 3 \\ a & b & c \end{bmatrix}$$

Transponieren von symmetrischer Matrix

Symmetrische Matrizen (welche nur quadtratische sein können!) erhalten als Resultat des Transponieren sich selbst.

$$A^T = A$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}$$

i Rechenregel

$$(A + B)^T = A^T + B^T$$

Matrixmultiplikation

Rechenregel

- $(AB)C = A(BC)$
- $(A+B)C = AC+BC$
- $A(B+C) = AB+BC$
- $r(A)B = A(rB)$
- $I_m \cdot A = A \cdot I_n$ für $A \in \mathbb{R}^(m \times n)$

Merkregel

$$A \in \mathbb{R}^{m\times n},\ B \in \mathbb{R}^{n\times p}$$

$$A \cdot B \Rightarrow (m \times n) \cdot (n \times p) = (m \times p)$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

Achtung!

Matrixmultiplikation ist **nicht** kommutativ.

$$AB \neq BA$$

Allgemeine Lösungsmengen von linearen Gleichungssystemen

Homogene LGS

Bei einem homogenen Gleichungssystem ist die rechte Seite Null:

$$A \cdot x = 0$$

wobei $A \in \mathbb{R}^{m\times n}$ und $x \in \mathbb{R}^n$ der Vektor der Unbekannten ist.

Die allgemeine Lösung des Homogenen Systems:

$$x = x_h = t_1 \cdot v_1 + \cdots + t_n \cdot v_n \quad \text{mit} \quad t_1, \ldots, t_n \in \mathbb{R}$$

Ein homogenes LGS hat stets die triviale Lösung $x = 0 \in \mathbb{R}^n$.

Inhomogene LGS

Beim inhomogenen LGS wird nun erlaubt, dass $b \neq 0$ ist, also nicht mehr 0.

$$A \cdot x = A \cdot (x_p + x_h) = b$$

Die allgemeine Lösung des Inhomogenen Systems:

$$x = x_p + t_1 \cdot v_1 + \cdots + t_n \cdot v_n \quad \text{mit} \quad t_1, \ldots, t_n \in \mathbb{R}$$

x_p wird als *partikuläre* Lösung des inhomogenen Systems genannt.

Vorgehen

1. Freiwählbare Parameter bestimmen
2. Partikuläre Lösung bestimmen
    1. irgendwelche Werte für die Parameter nehmen und Unbekannte Werte ausrechnen
3. Homogene Lösung bestimmen
4. Partikuläre & Homogene Lösung zusammenführen.

Inverse Matrix A^{-1}

Eine **quadratische** Matrix $A \in \mathbb{R}^{n\times n}$ heisst regulär, invertierbar, nicht-singulär, wenn es eine Matrix A^{-1} gibt, sodass

$$AA^{-1} = I_n$$

Ansonsten heisst die Matrix A singulär!

Note

Eine Matrix A ist invertierbar, wenn $\det A \neq 0$. $\det A = 0$ wäre ähnlich wie eine "Divison durch 0".

> Vorgehen Berechnung inverse Matrix z.B. $A \cdot X = I_2$
>
> 1. Die gegebene Matrix $A \in \mathbb{R}^n$ links neben die Einheitsmatrix I_n schreiben:
> $$[A \mid I_n]$$
> 2. Durch Zeilenoperationen die Linke Seite A in Treppenform überführen
> 3. Wenn links eine Spalte <u>kein</u> Pivot-Element besitzt, ist A **nicht invertierbar**
> 4. Die linke Seite hat im anderen Fall eine obere Dreiecksgestalt, mit allen Einträgen auf der Haupteinträge **ungleich Null**.
> 5. Weitere Zeilenoperationen machen und die linke Seite in die Einheitsmatrix überführen.
> $$\to [I_n \mid \tilde{A}]$$
> 6. Matrix auf rechter Seite entspricht der inversen Matrix $A^{-1} = \tilde{A}$.

> Beispiel
>
> $$\left[\begin{array}{cc|cc} 3 & 4 & 1 & 0 \\ 6 & 2 & 0 & 1 \end{array}\right] \Rightarrow \left[\begin{array}{cc|cc} 1 & 0 & -\frac{1}{9} & \frac{2}{9} \\ 0 & 1 & \frac{1}{3} & -\frac{1}{6} \end{array}\right]$$
>
> Die Matrix A^{-1} entspricht:
>
> $$A^{-1} = \begin{bmatrix} -\frac{1}{9} & \frac{2}{9} \\ \frac{1}{3} & -\frac{1}{6} \end{bmatrix}$$

Eigenschaft

Die Inverse Matrix dient als "Matrix-Division" und kann verwendet werden um zum Beispiel gewisse Berechnungen rückgängig zu machen. Dies ist aber nur möglich, wenn die Matrix regulär ist.

$$AX = B \quad \Rightarrow \quad X = A^{-1}B$$

> Rechenregel für die Inverse
>
> - $(A^{-1})^{-1} = A$
> - $(sA)^{-1} = s^{-1}A^{-1}$
> - $(AB)^{-1} = B^{-1}A^{-1}$ (Tauschung beachten!)
> - $(A^T)^{-1} = (A^{-1})^T$

$$\left[\det(A) = (-1)^r\, a_{11} \cdot \ldots \cdot a_{nn}\right]$$

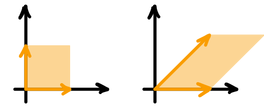
$[r = s + z]$ Spalten- & Zeilentauschung

Inverse einer $(2 \times 2)-$**Matrix**

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$A^{-1} = \frac{1}{\det A}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Determinante det



Das zentrale Resultate über Determinanten ist das Kriterium für die Regularität. Mit der Determinante kann auch die Änderung einer Fläche/Volumen/etc., zum Beispiel ob sich ein Bild vergrössert, verkleinert oder geflipped wird.

Wird eine **quadtratische** Matrix in eine obere oder untere Dreiecksform gebracht, gilt folgende Determinantenrechnung.

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

$$\det U = (u_{11} \cdot u_{22} \cdot \ldots \cdot u_{nn})$$

$\det(A) = 0 \to$ Nullspace (Fläche/Volumen ist gleich 0)

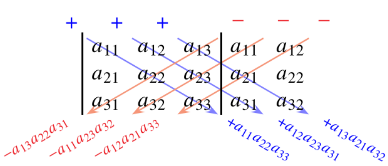
Eigenschaften

> Rechenregel
>
> Für quadratische Matrizen A und B und alle Skalare $s \in \mathbb{R}$ gilt
>
> - $\det A^T = \det A$
> - $\det(sA) = s^n \det A$ (n = Anzahl Zeilen)
> - $\det(AB) = \det(A) \cdot \det(B)$
>
> **Good to know**: Ein Produkt AB ist dann regulär wenn A und B regulär sind.

Determinante einer $(2 \times 2)-$**Matrix**

$$\det A = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = a \cdot d - b \cdot c$$

Determinante einer $(3 \times 3)-$**Matrix**



$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}$$

Euklidische Vektorräume

Lineare Unterräume

Definition linearer Unterraum U

Damit U als Unterraum gilt, müssen folgende Bedingungen gelten.

1. $0 \in U$
2. $v, w \in U \Rightarrow v + w \in U$
3. $v \in U, s \in \mathbb{R} \Rightarrow s \cdot v \in U$

Aufgespannte Unterräume

Unterräume können aufgespannt werden, das heisst die Basen, welche ein Unterraum definieren, werden zusätzlich parametrisiert. Dieser Unterraum definiert dadurch einen gewissen Wertebereich.

$$U = \text{span}\left(\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}\right) = \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot t_1 + \begin{bmatrix}0\\1\\0\end{bmatrix} \cdot t_2$$

Möchte zum Beispiel geprüft werden, ob Vektor $\vec{v}$ in U definiert ist, gilt:

$$U = \text{span}(v_1, v_2) = \vec{v} \Rightarrow \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot t_1 + \begin{bmatrix}0\\1\\0\end{bmatrix} \cdot t_2 = \begin{bmatrix}v_1\\v_2\\v_3\end{bmatrix}$$

Important

Die Vektoren, welche U aufspannen werden *Basen* genannt!

Vorgehen Test, ob $v \in U \subseteq \mathbb{R}^n$ ist

1. Lösen des linearen Gleichungssystems

$$\left[\begin{array}{ccc|c} v_1 & \cdots & v_k & v \end{array}\right]$$

2. Ist das System **konsistent**, so gilt $v \in U$, sonst $v \notin U$.

Geometrische, implizite & explizite Darstellung

Implizite Darstellung

Implizite Darstellung eines Unterraums wird der Unterraum in die Treppenform gebracht und dann mit den Parametern ausgeschrieben. Folgende Matrix in Treppenform

$$\left[\begin{array}{ccc|c} -1 & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

wird umgeschrieben in

$$-x_1 + \frac{1}{2}x_2 + x_3 = 0$$

Explizite Darstellung

Die explizite Darstellung gibt den Unterraum in Vektorform und Parametern an. Diese Darstellung wird auch **Parameterdarstellung** genannt.

$$U = \text{span}\left(\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix} \cdot t_1 + \begin{bmatrix}0\\1\end{bmatrix} \cdot t_2$$

Geometrische

Die geometrische Darstellung ist, wie der Name bereits beschreibt, die Visualisierung des Unterraums in einem vorstellbaren Bereich $(\mathbb{R}^1, \mathbb{R}^2, \mathbb{R}^3)$

Ein Beispiel wäre die Orthogonale Projektion auf der ersten Seite.

Lineare Unabhängigkeit

Die lineare Unabhängigkeit bezieht sich auf die Vektoren eines Unterraums. *Linear abhängige* Vektoren können durch andere im Unterraum enthaltene Vektoren "aufgebaut" werden.

Good to know!

Wird eine Ebene in $\mathbb{R}^3$ aufgespannt von drei Vektoren v_1, v_2, v_3, wird ein Vektor linear **abhängig** von den anderen sein, da die Ebene in dieser Dimension nur zwei Vektoren benötigt.

Die Lineare Unabhängigkeit einer Matrix kann mit den Ermitteln der Treppenform der Matrix geprüft werden.

$$\left(s_1 f_1(x) + s_2 f_2(x) + s_3 f(x) = 0\right)$$

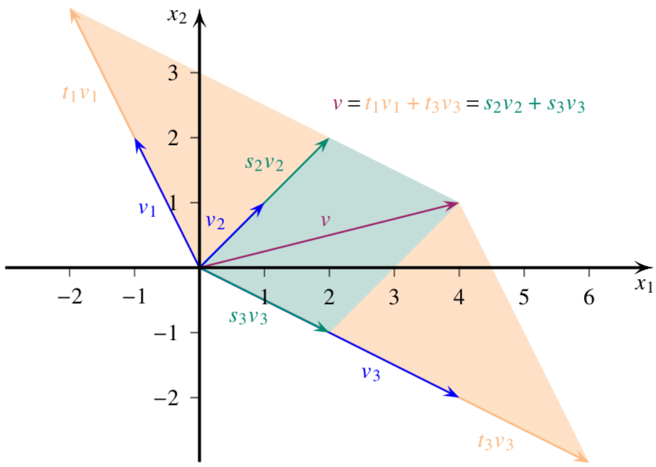
$$s_1 = s_2 = \cdots = 0$$

Vorgehen

Test der Vektoren $v_1, \dots, v_k \in \mathbb{R}^n$ auf lineare Unabhängigkeit

1. Ist $k > n$ so sind $v_1, \dots, v_k$ linear abhängig
2. Im anderen Fall, werden die Vektoren in eine $(n \times k)$-Matrix geschrieben und in die **Treppenform** gebracht.
3. Ist in einer Spalte kein Pivot vorhanden, so sind die Vektoren linear abhänging.
4. Haben alle Spalten einen Pivot, sind die Vektoren linear unabhängig.

Basis und Koordinaten



Definition *Koordinaten*

Es sei $v_1, \dots, v_k \in U$ eine Basis von U. Dann kann jeder Vektor $v \in U$ *eindeutig* als Linearkombination

$$v = s_1 \cdot v_1 + \dots + s_k \cdot v_k$$

geschrieben werden. Die Koeffizienten $s_1, \dots, s_k$ heissen *Koordinaten* von v **bzgl. der Basis** $v_1, \dots, v_k$. Die Koordinaten werden zu einem Koordinatenvektor zusammengefasst:

$$\begin{bmatrix} s_1 \\ \vdots \\ s_k \end{bmatrix} \in \mathbb{R}^k$$

Definition *Basis*

Die Vektoren $v_1, \dots, v_k$ eines linearen Unterraums U heissen eine *Basis von U*, falls sie

1. den ganzen Unterraum U erzeugen $\to U = \text{span}(v_1, \dots, v_k)$ und
2. linear unabhängig sind.

Dimension dim

Eigenschaften der Dimension

Es sei U ein Unterraum von $\mathbb{R}^n$ der Dimension d und $v_1, \dots, v_k \in U$.

1. Wenn $v_1, \dots, v_k$ linear unabhängig sind $\Rightarrow k \leq d$
2. Wenn $V = \text{span}(v_1, \dots, v_k)$ ist $\Rightarrow k \geq d$
3. Für $k = d$ gilt: $v_1, \dots, v_k$ ist Basis

- $v_1, \dots, v_k$ sind linear unabhängig
- $V = \text{span}(v_1, \dots, v_k)$

Basiswahl

Durch das bestimmen der Lösungsmenge einer Matrix können die Basen der entsprechenden Lösungsmenge ermittelt werden.

Vorgehen

1. Aufschreiben der $(n \times m)$-Matrix mit den Spaltenvektoren

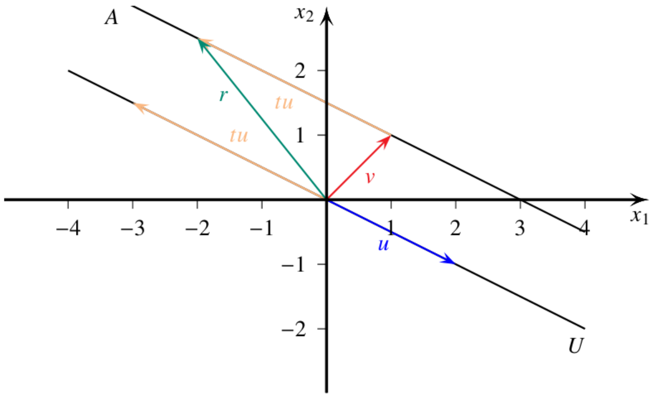
$$A = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$$

2. Überführen von A in eine Treppenform via Zeilenumformung
3. Wenn $j_1, \dots, j_p$ die Spalten mit Pivot-Element sind, dann bilden die Vektoren

$$v_{j_1}, \dots, v_{j_p} \quad \text{eine Basis von } U$$

Wichtig! Es werden die **ursprünglichen** Vektoren, nicht die der Treppenform, genommen.

Affine Unterräume



Definition

Ein *affiner Unterraum* des $\mathbb{R}^n$ ist eine Menge der Form

$$A = v + U = \{v + t_1 v_1 + \cdots + t_k v_k \mid t_1 + \cdots + t_k \in \mathbb{R}\}$$

mit einem festen **Verschiebungsvektor** $v \in \mathbb{R}^n$.
Die *Dimension* von A ist die gleiche, wie von U:

$$\dim A = \dim U$$

! Inhomogenes LGS

Es sei A eine $(m \times n)$-Matrix und $b \in \mathbb{R}^m$.

$$Ax = b$$

Die Lösungsmenge ist ein affiner Unterraum von $\mathbb{R}^n$, welcher um einen *Verschiebungsvektor* $v \in \mathbb{R}^n$ verschoben wurde. Dies heisst, dass der affine Unterraum **nicht** durch 0 geht, sondern durch den Punkt von v.
Die *Parameterdarstellung* von A ist

$$A = \{v + t_1 v_1 + \cdots + t_n v_n \mid t_1, \ldots, t_n \in R\}$$

Lineare Abbildung

Eine Abbildung ist in anderen Worte eine Funktion mit Matrizen. Diese Abbildungen können zum Beispiel Vektoren auf einer Ebene rotieren, vergrössern oder verkleinern.

Definition Lineare Abbildung

Eine gültige lineare Abbildung muss folgende Bedingungen erfüllen:

- $f(a \cdot x) = a \cdot f(x)$ wobei $a \in \mathbb{R}$ & $x \in \mathbb{R}^n$
- $f(x + y) = f(x) + f(y)$ wobei $x, y \in \mathbb{R}^n$

Eine lineare Abbildung $L : \mathbb{R}^n \to \mathbb{R}^m$ bildet den Nullvektor auf den Nullvektor ab:

$$L(0) = 0$$

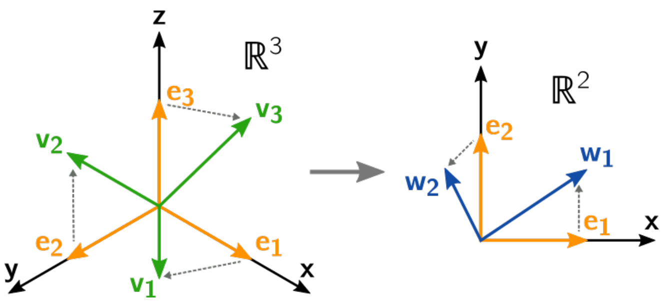
Definition Abbildung durch Matrix

Die Matrix einer Abbildung wird auch "Transformationsmatrix" genannt, da diese nichts anderes macht als gewisse Operationen auszuführen (z.B. Rotation, Streckung, etc.)
Ist A eine $(m \times n)$-Matrix, so definiert die Vorschrift

$$L_A : \ \mathbb{R}^n \to \mathbb{R}^m, \quad x \mapsto Ax$$

eine Abbildung.

Die Matrix einer linearen Abbildung



Matrix bezüglich den Standardbasis-Vektoren

$$a_1 = L(e_1) \qquad a_2 = L(e_2)$$

$$L(x) = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Ax$$

Vorgehen Matrix bzgl. anderen Basen

Bestimmung der Matrix der linearen Abbildung $L : \mathbb{R}^n \to \mathbb{R}^m$ bzgl. den Basen $v_1, \ldots, v_n \in \mathbb{R}^n$ und $w_1, \ldots, w_m \in \mathbb{R}^m$

1. Berechnen der Bilder der Basisvektoren $L(v_1), \ldots, L(v_n)$ in $\mathbb{R}^m$
2. Entwickeln der Bilder nach Basis $w_1, \ldots, w_m$ von $\mathbb{R}^m$

$$L(v_x) = a_{1x}\ w_1 + \cdots + a_{mx}\ w_m \quad \text{für} \quad 1 \leq x \leq n$$

3. Zusammenfassen der Koeffizienten zu Spalten und bilden der Matrix mit allen diesen Spalten (Reihenfolge beachten)

$$L(v_1) = a_{1x} w_1 + \cdots + a_{mx} w_m$$

$$a_x = \begin{bmatrix} a_{1x} \\ \vdots \\ a_{mx} \end{bmatrix} \to A = \begin{bmatrix} a_1 & \cdots & a_x \end{bmatrix}$$

$a_1, \ldots, a_x$ sind Koordinatenvektoren

Inverse Abbildung

Mit inversen Abbildung können Operationen rückgängig gemacht werden (z.B. 90° Drehung und als invers wäre es eine -90°). Damit die Abbildung $L : \mathbb{R}^n \to \mathbb{R}^n$ invertierbar ist, gelten folgende Vorraussetzungen:

- $L(v) = w \quad \Leftrightarrow \quad v = L^{-1} = M(w)$
- L invertierbar $\Longleftrightarrow$ A regulär ($\det A \neq 0$)

für alle $v, w \in \mathbb{R}^n$. M ist dabei die *zu L inverse Abbildung* $\to M = L^{-1}$.

Affine Abbildung

Eine Abbildung $F : \mathbb{R}^n \to \mathbb{R}^m$ heisst *affin*, wenn es eine lineare Abbildung $L : \mathbb{R}^n \to \mathbb{R}^m$ gibt und ein Verschiebungsvektor $w \in \mathbb{R}^m$.

$$F(v) = L(v) + w \quad \text{für alle} \quad v \in \mathbb{R}^n$$

Vorgehen

1. Verschieben des Zentrums in den Ursprung
2. Anwendung der linearen Abbildung bzgl. des Ursprungs
3. Verschieben des Ursprungs zurück zum Zentrum

Zum Beispiel "affine" Drehung. r_z ist der Ortsvektor des verschobenen Zentrums Z.

$$F(x) = D(x - r_z) + r_z = D(x) - D(r_z) + r_z$$

Nützliche Abbildungen $\mathbb{R}^2$

Siehe Lineare Abbildung für die Eigenschaften der Abbildungen ($f(x \cdot a) = a \cdot f(x)$, etc.).

Drehung um Winkel α

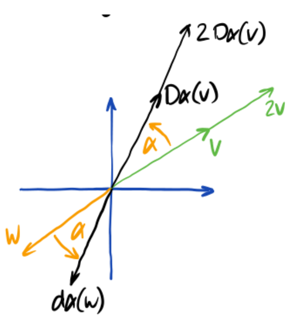


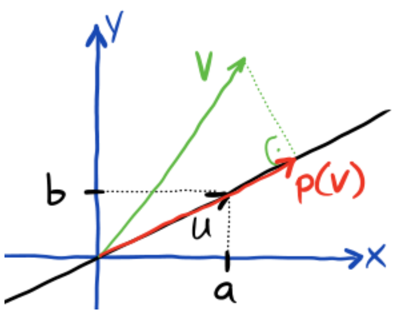
Abbildung D_α bzgl. Standardbasen

$$D_\alpha(x) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot x$$

Die inverse Abbildung wäre mit $-\alpha$

Projektion auf eine Gerade (durch Ursprung)

Bei der Projektion kann keine inverse Abbildungen ermittelt werden.



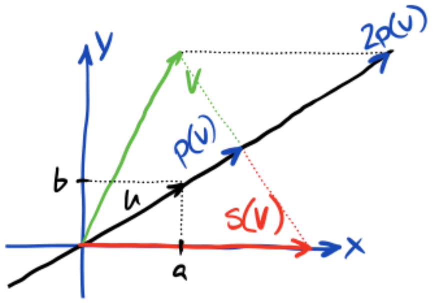
$$P(\vec{v}) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}|^2} \cdot \vec{u}$$

Orthogonale Zerlegung bezüglich einer Gerade

$$v = p + w$$

mit $w \perp p$ (Abbildung oben ist w die gestrichelte Linie)

Spiegelung an einer Gerade



$$S(\vec{v}) = 2 \cdot P(\vec{v}) - \vec{v}$$

Kern ker & Bild im

Kern (engl. Kernel) (null space)

$$\ker L = \{v \in \mathbb{R}^n \mid L(v) = 0\}$$

$$\ker A \to \text{Lösungsraum von } Ax = 0$$

→ Treppenform auflösen und die parametrisierten Vektoren/Lösungsmenge entsprechen dem Kern.

Beispiel

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \Rightarrow \left[\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Aus diesem Gleichungssystem entsteht folgende Lösungsmenge

$$\begin{cases} x_1 + x_3 + x_4 = 0 \\ x_2 - x_4 = 0 \end{cases} \Rightarrow x = t_1 \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + t_2 \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Der Kern dieses Beispiels ist somit:

$$\ker A = \operatorname{span}\left(\begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right)$$

Bild (engl. Image)

$$\operatorname{im}(L) = \{L(v) \mid v \in \mathbb{R}^n\}$$

- Sind $a_1, \ldots, a_n$ die Spalten von A, so ist

$$\operatorname{im}(A) = \operatorname{span}(a_1, \ldots, a_n)$$

Es gibt aber den Fall, dass die Matrix A linear abhängige Vektoren besitzt, welche minimiert werden können und dadurch ein kleineres Bild erhält.

→ Entspricht allen ursprünglichen Vektoren, welche in der Treppenform ein Pivot besitzen (linear unabhängigen)!

Beispiel

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Im Vergleich zum Kern, wird jetzt nicht die Lösungsmenge genommen, sondern die **ursprünglichen** Vektoren, welche in der Treppenform ein Pivot besitzen. Dazu muss die Lösungsmatrix noch ein bisschen weiter verarbeitet werden, damit die Pivots bestimmt werden können

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Das Bild dieses Beispiels ist somit:

$$\operatorname{im}(A) = \operatorname{span}\left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right)$$

! Kern und Bild sind Vektorräume!

Ist $L : \mathbb{R}^n \to \mathbb{R}^m$ eine lineare Abbildung, so ist $ker(L)$ ein linearer Unterraum von $\mathbb{R}^n$ und $im(L)$ ein linearer Unterraum von $\mathbb{R}^m$.
Das Bild $im(L)$ wird auch *Spaltenraum* von A genannt.

Rang rk & Defekt def

Rang (engl. rank)

$$\operatorname{rk}(A) = \dim \operatorname{im} A$$

→ Anzahl Spalten **mit** Pivots zählen

Number of Dimensions in the output

Defekt (engl. defect)

$$\text{def}(A) = \dim \ker(A)$$

$\rightarrow$ Anzahl Spalten **ohne** Pivots zählen

Dimensionssatz

Für jede Matrix A gilt

$$\text{rk}(A) + \text{def}(A) = n \quad ; \quad A \in R^n$$

wobei n die Anzahl Spalten von A ist.

Eigenwerte & -vektoren

Definition

1. $L : \mathbb{R}^n \rightarrow \mathbb{R}^n$, Skalar $\lambda \in \mathbb{R}$ & Vektor $v \in \mathbb{R}^n$

$$v \neq 0 \quad \text{und} \quad L(v) = \lambda \cdot v$$

$\rightarrow$ v heisst *Eigenvektor von L* und λ heisst *Eigenwert von L*

2. Die *Eigenwerte* und *-vektoren* einer $(n \times n)$-Matrix A sind die Eigenwerte und vektoren der linearen Abbildung

$$L(v) = \lambda v \quad \Leftrightarrow \quad Ax = \lambda x$$

Wichtig!

- Eigenvektoren gibt es **nur** für **quadratische** Matrizen.
- $v \neq 0$, da ansonsten λ jeder Wert annehmen könnte.
- Zu jedem Eigenwert gibt es **unendlich viele** Eigenvektoren

Eigenraum

Die Menge aller Eigenvektoren zusammen mit dem Nullvektor

$$U_\lambda = \{v \in \mathbb{R}^n \mid L(v) = \lambda v\}$$

ist ein linearer Unterraum von V. Er heisst *Eigenraum* von L zum Eigenwert λ.

Bestimmung von Eigenwerten & -vektoren

Definition Eigenraum

Der Eigenraum entspricht der Lösungsmenge/Kern der folgenden Gleichung.

$$U_\lambda = \ker(A - \lambda I_n)$$

Definition Charakteristische Polynom von A

Für eine quadratische Matrix A heisst das Polynom

$$p_A(\lambda) = det(A - \lambda I_n)$$

das *charakteristische Polynom* von A.
Damit können die Eigenwerte einer **quadratischen** Matrix berechnet werden (die Nullstellen des Polynoms p_A entspricht den Eigenwerten λ).

Vorgehen

1. Berechnung des charakteristischen Polynoms

$$p_A(\lambda) = det(A - \lambda I_n)$$

2. Bestimmung der Nullstellen $\lambda_1, \ldots, \lambda_r$ von $p_a(\lambda)$. Dies sind die <u>Eigenwerte von A</u>.

3. Für jeden Eigenwert λ_i Bestimmung des Eigenraums U_{λ_i} als Lösungsmenge des Gleichungssystem

$$(A - \lambda_i I_n) \cdot v = 0$$

$v \in U_{\lambda_i}$ ist der Eigenvektor (oder mehrere).

Diagonalisierbarkeit

Definition

Eine quadratische $(n \times n)$-Matrix A heisst *diagonalisierbar*, wenn es eine Basis von $\mathbb{R}^n$ aus Eigenvektoren von A gibt.

$$v_1, \ldots, v_n \in \mathbb{R}^n \text{ Basis mit } Av_i = \lambda_i v_i \quad \text{für} \quad 1 \leq i \leq n$$

Ist eine diagonalisierbar, kann die **Diagonalmatrix** D zusammengestellt werden:

$$D = \text{diag}(\lambda_1, \ldots, \lambda_n) = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

Eigenwertzerlegung

Eine $(n \times n)$-Matrix A ist genau dann diagonalisierbar, wenn es eine reguläre Matrix S gibt, so dass

$$S^{-1}AS = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$$

Ist in diesem Fall $v_1, \ldots, v_n$ eine Basis aus **Eigenvektoren** zu den Eigenwerten $\lambda_1, \ldots, \lambda_n$ von A, so gilt

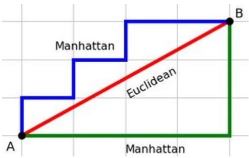
$$S = [v_1 \ \ldots \ v_n]$$

Normen & Metriken

Normen

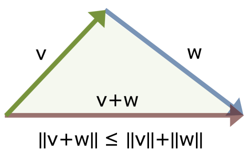
Normen sind in anderen Worten der Betrag von Vektoren. Je nach Norm ist der Betrag anders, wobei drei Normen betrachtet wird: Manhatten, Euklidisch & Maximum.

1. **Manhatten-Norm** (um die Ecken gehen)



$\rightarrow \|v\|_1 = |v_1| + \cdots + |v_n|$

2. **Euklidische Norm** (direkter Weg)



$\rightarrow \|v\|_2 = \|v\| = \sqrt{v_1^2 + \cdots + v_n^2}$

3. **Maximum-Norm** (maximaler Betrag)

$\rightarrow \|v\|_\infty = \max(|v_1|, \cdots, |v_n|)$

Diese Norm gibt den Betrag eines Vektors v_n zurück mit dem grössten Betrag.

Metrik

Jede Norm definiert einen Begriff von *Distanz zwischen Punkten* wie folgt.

$$d(x, y) = \|x - y\|$$

Es gibt aber auch Distanzen, die nicht von einer Norm herkommen.

Die *Hamming-Distanz* $d_H(\cdots)$ zählt zwischen zwei Zeichenkette gleicher Länge die Anzahl der Positionen, an denen unterschiedliche Zeichen stehen.

$$d_H(x, y) = \text{Anzahl der } i \text{ so dass } x_i \neq y_i$$

Definition

Eine *Metrik* d auf einer Menge M ist eine Funktion, die jedem Paar von Elementen $x, y \in M$ eine relle Zahl $d(x, y) \in \mathbb{R}$ zuordnet, sodass die folgenden Eigenschaften erfüllt sind: Für alle $x, y, z \in \mathbb{R}^n$ gilt

1. $d(x, x) \geq 0$
2. $d(x, y) = 0 \quad \Leftrightarrow \quad x = y$
3. $d(x, y) \leq d(y, x)$ (Symmetrie)
4. $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung)

Orthonormalbasen

Definition *Orthonormal*

$$v_i \bullet v_j = \delta_{i,j} = \begin{cases} 1 & \text{falls} \quad i = j \\ 0 & \text{falls} \quad i \neq j \end{cases}$$

1. Eine Menge $\{v_1, \ldots, v_k\}$ von Vektoren heisst *orthogonal*, wenn

$$v_i \bullet v_j = 0 \quad \text{für alle} \quad 1 \leq i, j \leq k \quad \text{mit } i \neq j$$

2. Eine Menge $\{v_1, \ldots, v_k\}$ von Vektoren heisst *ortho***normal**, wenn

orthonormal $\rightarrow$ orthogonal

$$v_i \bullet v_j = 0 \quad \text{für alle} \quad 1 \leq i, j \leq k \quad \text{mit } i \neq j$$

ortho**normal** $\rightarrow$ normiert

$$\|v_i\| = 1 \quad \text{für alle} \quad 1 \leq i \leq k$$

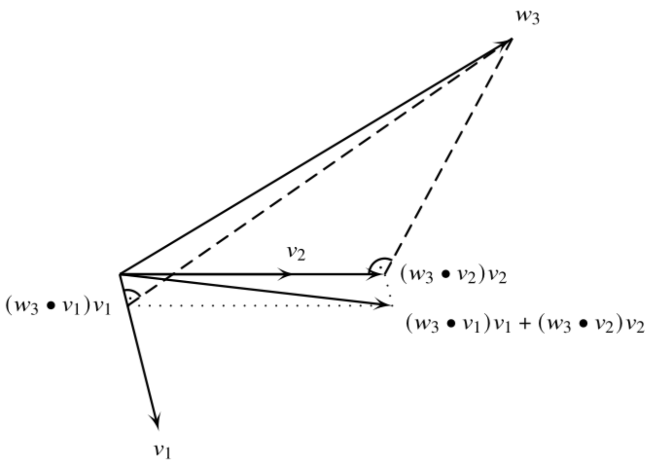
Orthonormalbasen spannen ein Unterraum auf, wobei alle Basen voneinander **orthogonal** sind und jeweils den euklidischen Betrag $\|v\|_2 = 1$ beträgt.

Definition Orthonormalbasis ONB

Eine *Orthonormalbasis* (ONB) eines linearen Unterraums $U \subseteq \mathbb{R}^n$ ist eine orthonormale Menge, die auch eine Basis von U ist.

Orthonormalbasen können mit dem Gram-Schmidt Verfahren aufgebaut werden.

Gram-Schmidt-Verfahren



Vorgehen

1. Normierung: $v_1 = w_1/\|w_1\|$
2. Für $i \in \{2, \ldots, n\}$ nimm an, dass $v_1, \ldots, v_{i-1}$ bereits konstruiert ist
3. Projektion des Vektors auf die normierten ONB-Vektoren

$$v_i' = w_i - ((w_i \bullet v_1)v_1 - \cdots - (w_i \bullet v_{i-1})v_{i-1})$$

4. Normierung: $v_i = v_i'/\|v_i'\|$
5. Dies wird für alle benötigten Vektoren im ONB wiederholt.

Die Funktion in $(\cdots)$ berechnet den Projektionsvektor $\vec{p}$. Durch die Projektion auf allen bereits normierten Vektoren, wird die Orthogonalität untereinander aufgestellt.

Spektralsatz

Orthogonale Matrizen

Die Matrix V besteht aus normierten Eigenvektoren! Bilden die Spalten der quadratischen Matrix $V \in \mathbb{R}^{n \times n}$ eine **Orthonormalbasis**, so gilt

$$V^T V = I_n$$

→ Eigenvektoren einzeln normieren!
Solche Matrizen werden *orthogonal* genannt (nicht *orthonormal*)

Aus S eine orthonormale Matrix V
↳ mit Gram Schmidt Verfahren

Spektralsatz

Matrix A ist eine symmetrische quadratische Matrix A und somit orthogonal diagonalisierbar.

$$V^T A V = D$$

Hierbei sind die Spalten von V eine Basis des $\mathbb{R}^n$ von Eigenvektoren von A.

Für eine quadratische Matrix A erhält man die *Hauptachsentransformation*

$$A = VDV^T$$

Berechnung der Hauptachsentransformation

Wenn Matrix $A \in \mathbb{R}^{n \times n}$ symmetrisch ist:

1. Bestimmung der Eigenwerte $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$
2. Für jeden Eigenwert λ_i Bestimmung einer Basis des Eigenraums U_λ
3. Für jeden Eigenraum Anwendung des Gram-Schmidt-Verfahrens auf die Basis, um eine ONB zu finden $\Rightarrow$ ONB $v_1, \ldots, v_n$ des $\mathbb{R}^n$
4. $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ und $V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}$
5. Hauptachsentransformation berechnen mit vorheriger Gleichung

Singulärwertzerlegung (engl. singular value decomposition (SVD))

Die Singulärwertzerlegung existiert für jede Matrix, nicht nur für quadratische.

Für jede $(n \times m)$-Matrix A ist die $(n \times n)$-Matrix S **symmetrisch**.

Daraus Eigenwerte berechnen

$S = A^T A$

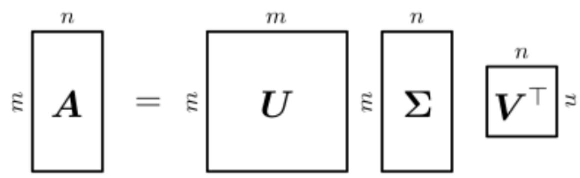
↳ Matrix D zusammensetzen
& sortieren

Auf S kann der Spektralsatz angewandt werden, und es gibt eine othogonale Matrix V mit

$$S = A^T A = VDV^T$$

→ Diagonalmatrix D beinhaltet die Eigenwerte

Singulärwertzerlegung



Sei A eine ($m \times n$). Dann existieren Matrizen

- **ONB** $U \in \mathbb{R}^{m \times m}$ mit Spaltenvektoren u_i
- **ONB** $V \in \mathbb{R}^{n \times n}$ mit Spaltenvektoren v_i
- $\Sigma \in \mathbb{R}^{m \times n}$ ist die *Singulärwertmatrix* von A mit Einträgen

$$\Sigma_{ij} = \begin{cases} \sigma \geq 0 & \text{für } i = j \\ 0 & \text{für } i \neq j \end{cases}$$

Die Einträge erfüllen $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{min(m,n)} \geq 0$ (Sortieren nicht vergessen!)

so dass

$$A = U \Sigma V^T$$

Diese Gleichung wird *Singulärwertzerlegung* (engl. singular value decomposition *SVD*) genannt.

Die Matrix Σ beinhaltet die Wurzelwerte der Eigenwerte.

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ & \ddots & \sigma_n \\ \vdots & & 0 \\ & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 \\ 0 & \ddots & 0 \\ & \ddots & \sqrt{\lambda_n} \\ \vdots & & 0 \\ & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

$0 \quad 0 \quad 0$

Singulärwert-Gleichung

Mit der *Singulärwert-Gleichung* werden die Vektoren für die *Links-Singulärmatrix U* berechnet. Falls zu wenig σ_i für die Matrix, wird die Singulärwert-Gleichung als homogen betrachtet (Warum? keine Ahnung).

$$A \cdot v_j = \begin{cases} \sigma_j \cdot u_j & \text{für } j = 1, \ldots, r \\ 0 & \text{für } j = r+1, \ldots, n \end{cases}$$

$$\left[A^T u_j = 0 \right]$$

Wichtig! σ_j entspricht dem Singulär**wert**!

$$u_j = \frac{1}{\sigma_j} A \cdot v_j$$

Damit kann dann U zusammengesetzt werden:

$$U = [u_1 \quad \cdots \quad u_n]$$

Eigenwerte 2x2 Matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

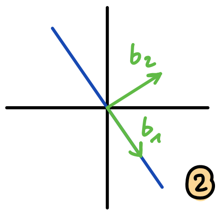
$$\left[m = \frac{a \cdot d}{2} \right]$$

$$\left[p = \det A = a \cdot d - b c \right]$$

$$\left[\lambda_{1,2} = m \pm \sqrt{m^2 - p} \right]$$

Small Stuff Spiegelung bzgl anderen Basen

Spiegelung bzgl. Standardbasen:



① Matrix S bzgl. b_1 b_2 bestimmen

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

② Koordinaten bestimmen für e_1, e_2

$$[b_1 \; b_2] \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} = [e_1 \; e_2] \longrightarrow \begin{bmatrix} -3 & 1 \\ 1 & 3 \end{bmatrix} X = [e_1 \; e_2]$$

$$X = \begin{bmatrix} -3 & 1 \\ 1 & 3 \end{bmatrix}^{-1} = \frac{1}{10} \begin{bmatrix} -3 & 1 \\ 1 & 3 \end{bmatrix}$$

$$e_1 = -\frac{3}{10} b_1 + \frac{1}{10} b_2 \; ; \; e_2 = \frac{1}{10} b_1 + \frac{3}{10} b_2$$

③ Matrix bestimmen

$$\underline{L(e_1)} = -\frac{3}{10} L(b_1) + \frac{1}{10} L(b_2) = -\frac{3}{10} b_1 - \frac{1}{10} b_2 = \underline{a_1}$$

$$\underline{L(e_2)} = \ldots = \ldots = \underline{a_2}$$

④ Matrix S bzgl. Standardbasen

$$S = [a_1 \; a_2] = \begin{bmatrix} \frac{4}{5} & -\frac{3}{5} \\ -\frac{3}{5} & -\frac{4}{5} \end{bmatrix}$$

Python

Basics

Listen

Eine Liste kann gemischte Datentpen beinhalten.

```
pizza = ['Pizza',1,2,3,'not Pasta']
len(pizza) # = 5
pizza[4] = 4
```

Andere Funktionen mit Listen.

```
pizza[-1]     # get last item
pizza + pizza # concat two lists
pizza[:3]     # get 3 first items
pizza[1:4]    # get item Nr.2 & 3
```

Linspace

Der Befehl `np.linspace(...)` erzeugt eine Liste, welche gleichmässige Abstände zu den Einträgen erzeugt.

```
x = np.linspace(start=1, stop=10, num=4)
array([ 1., 4., 7., 10.])
```

Arange

Der Befehl `np.arange(...)` erzeugt eine Liste, welche vom `start`-Wert in `step` Schritten bis **vor** dem `stop`-Wert geht.

arange(9) → (0,1,...,8)

```
y = np.arange(start=1, stop=7, step=.6)
array([1. , 1.6, 2.2, 2.8, 3.4, 4. , 4.6,
       5.2, 5.8, 6.4])
```

Ein- & Ausgabe

Eingabe

```
input("Flavour Text before Input: ")
```

Ausgabe

```
print("This prints out this text")
print(object)
```

Nützliche Befehle

- `cls` , `clear` → Konsole leeren
- `who` → Liste aller Variablen anzeigen
- `whos` → Liste aller Variablen mit ihren Werten anzeigen
- `del <variable>` → Variable löschen
- `%reset` → Alle Variablen löschen
- `help()` → Interaktive Hilfe

Linear Algebra Libraries

Für Linear Algebra können zwei Bibliotheken verwendet werden:

```
import numpy as np
from scipy import linalg
```

Genauigkeit & Kompatibilität

Generell sollten die Lineare-Algebra-Methoden von scipy verwendet werden. `scipy` beinhaltet alle Linear-Algebra von `numpy`.
→ `scipy` ist genauer als `numpy`

Format & Anzahl Elemente

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

`.shape` gibt das Format der Matrix `(<Zeilen>,<Spalten>)`, nach dem Ansatz von "Zeilen zuerst, Spalten Später".

```
B.shape # Output: (3, 2)
```

`.size` Gibt die Anzahl Elemente der Matrix aus.

```
B.size # Output: 6
```

Matrixtypen

```
A = np.array([[1, 2], [3, 4], [5, 6]])
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

```
B = np.zeros([2,3])
```

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
C = np.ones([3,2])
```

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

```
D = np.eye(3)
```

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
E = np.diag([1,2,3])
```

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Zugriff auf Untermatrizen

Ähnlich wie bei Listen, können bei Matrizen/Arrays einzelne Werte, Spalten, Zeilen oder Teilmatrizen entnommen werden.

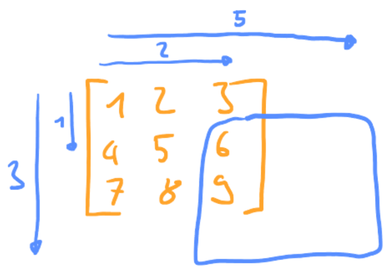
$$F = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
F[1,0] => 4
F[4,5] => ERROR - out of bounds
```

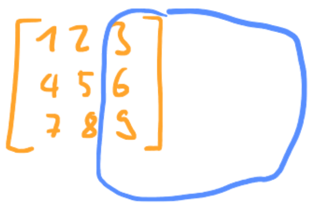
Anstatt einzelne Werte abzufragen, können auch

```
F[<row start>:<row end>,<col start>:<col end>]

IN:  F[1:3,2:5]
OUT: array([[6],
            [9]])
```



```
IN:  F[:3,2:]
OUT: array([[3],
            [6],
            [9]])
```



```
IN:  F[:2,:]
OUT: array([[1, 2, 3],
            [4, 5, 6]])
```

❗ Referenzen

```
G = F[1:3,2:4]
```

Lässt `G` als eine Referenz von `F` dienen. Jegliche Änderungen an `F` wird direkt an `G` übertragen.
Um eine Kopie einer Matrix zu machen, muss die Methode `numpy.copy` verwendet werden.

```
G = F.copy()
# oder wenn nur Untermatrix
G = F[1:3,2:4].copy()
```

Stitching

Die *Stitching*-Funktionen `hstack` & `vstack` können verwendet werden, um Matrizen über die horizontale oder vertikale Seite zusammenzusetzen.

```
np.hstack([A,C])
```

$$\begin{bmatrix} A & C \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 3 & 4 & 1 & 1 \\ 5 & 6 & 1 & 1 \end{bmatrix}$$

```
np.vstack([B,D])
```

$$\begin{bmatrix} B \\ D \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Grundoperationen

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Transponieren

```
A.T
```

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Elementweise Operationen +, -, *, /

```
A*2
```

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

```
A*np.eye(2)
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

Elementweise Potenz **

```
A**2
```

$$\begin{bmatrix} 1^2 & 2^2 \\ 3^2 & 4^2 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix}$$

```
A**A
```

$$\begin{bmatrix} 1^1 & 2^2 \\ 3^3 & 4^4 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 27 & 256 \end{bmatrix}$$

Matrixmultiplikation @

```
A@A
```

$$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

Potenz im Sinne Matrixmultiplikation

```
np.linalg.matrix_power(A,3)
```

$$AAA = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

Inverse

```
np.linalg.inv(A)
```

$$A^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

> **Caution**
>
> Diese Funktion gibt eine Fehlermeldungen aus, wenn Matrix `A` eine singuläre Matrix ist.

Determinante

```
det_A = np.linalg.det(A)
```

$$\det A = -2$$

> **Nicht Vergessen!**
>
> - $\det A = 0 \Rightarrow$ Nicht invertierbar!
> - $\det A \neq 0 \Rightarrow$ Invertierbar!

Eigenwerte und -vektoren

```
B = np.array([[1,1], [1,1]])
ev, S = np.linalg.eig(B)
```

```
# ev:
[2. 0.]

# S:
[[ 0.70710678 -0.70710678]
 [ 0.70710678 0.70710678]]
```

> **Vorsicht!**
>
> Ist die Matrix nicht diagonalisierbar, so liefert Python linear abhängige Eigenvektoren zurück.
>
> ```
> IN:  C = np.array([[1,1], [-1,3]])
> IN:  np.linalg.eig(C)
> OUT: (array([2.00000002, 1.99999998]),
>       array([[ 0.70710677, -0.70710679],
>              [ 0.70710679, -0.70710677]]))
> ```

Normen, Skalarprodukt und das Gram-Schmidt-Verfahren

Norm

```
np.linalg.norm(<array>,<norm>)
```

- `1` → Manhatten Norm $\|v\|_1$
- `2` → euklidische Norm $\|v\|_2$
- `inf` oder `np.inf` → Maximumnorm $\|v\|_\infty$

Skalarprodukt

~~`np.dot(<vector>,<vector>)`~~

Nur folgendes verwenden

```
<vector>.T@<vector>
```

> **!** Important
>
> Nach Polonis Worten, funktioniert `np.dot(...)` **nicht**.

Was der folgenden Gleichung entspricht

$$x \bullet y = x^T y = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}$$

Berechnet das Skalarprodukt der angegebenen Vektoren.

Gram-Schmidt-Verfahren

```
np.linalg.qr(<matrix>)

u = np.array([[1], [2], [0]])
v = np.array([[8], [1], [-6]])
w = np.array([[0], [0], [1]])

A = np.hstack([u, v, w])

Q,R = np.linalg.qr(A)
```

- `Q` → Orthonormale Vektoren $v_1, \ldots, v_n$
- `R` → Koordinaten der ursprünglichen Vektoren $w_1, \ldots, w_n$

Singulärwertzerlegung

```
U, sigma, V_T = np.linalg.svd(A)
```

> **!** Important
>
> `sigma` beinhaltet nur die Singulärwerte, nicht die Matrix!
> `sigma` $= \{\sigma_1, \ldots, \sigma_n\}$

Lineare Gleichungssysteme

> **i** Mini Repetition
>
> $$Ax = b \quad \Leftrightarrow \quad x = A^{-1}b$$
>
> Wenn x unbekannt ist, A zuerst prüfen ob regulär mit `det(A)`, ansonsten nicht möglich, bzw. unendlich viele Lösungen möglich.
> **Matrixmultiplikationen** werden mit dem Befehl `@` ($\rightarrow A \cdot B$) anstatt `*` ($\rightarrow A \circ B$) gemacht!

Die Funktion `np.linalg.solve(A,b)` löst das Gleichungssystem $Ax = b$ nach x auf.

```
np.linalg.solve(A,b)
```

> Caution
>
> Diese Funktion gibt Fehlermeldungen aus, wenn...
>
> - Matrix `A` eine singuläre Matrix ist
> - Matrix `A` nicht quadratisch ist

Dieses Gleichungssystem kann auch mit der inversen Matrix berechnet werden:

```
x = np.linalg.inv(A)@b
```

Fall: Matrix ist singulär

Wenn die Determinante einer Matrix (z.B. A) $\det A = 0$ ist, dann kann `np.linalg.solve(...)` nicht gebraucht werden. Es wird die Funktion `np.linalg.lstsq()` bzw. `scipy.linalg.lstsq()` verwendet (`lstsq` $\rightarrow$ least-squares solution)

> **! Important**
>
> Die Lösung von `np.linalg.lstsq()` bzw. `scipy.linalg.lstsq()` immer überprüfen, da es möglich sein kann, dass die gegebene Lösung eigentlich gar keine Lösungist.
> Im Fall einer singulären Koeffizientenmatrix kann das Gleichungssystem entweder gar keine oder unendlich viele Lösungen haben.

> **Note**
>
> Hat ein lineares Gleichungssystem $Ax = b$ unendlich viele Lösungen, so wird durch `scipy.linalg.lstsq()` der euklidische Abstand von x zum Ursprung minimiert.

> **Note**
>
> Hat ein lineares Gleichungssystem $Ax = b$ keine Lösung, so wird durch `scipy.linalg.lstsq()` der euklidische Abstand von Ax zu b minimiert.

Kern

Der Kern einer Matrix wird mit dem Befehl

```
np.linalg.null_space(A)
# bzw.
linalg.null_space(A)
```

berechnet.

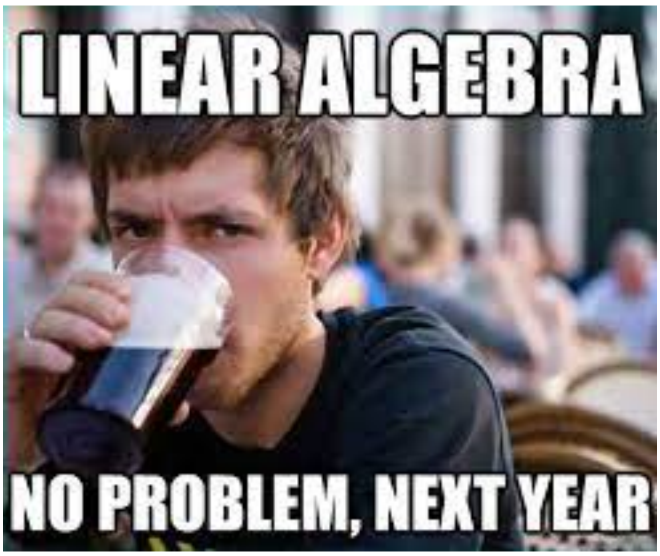
Rang

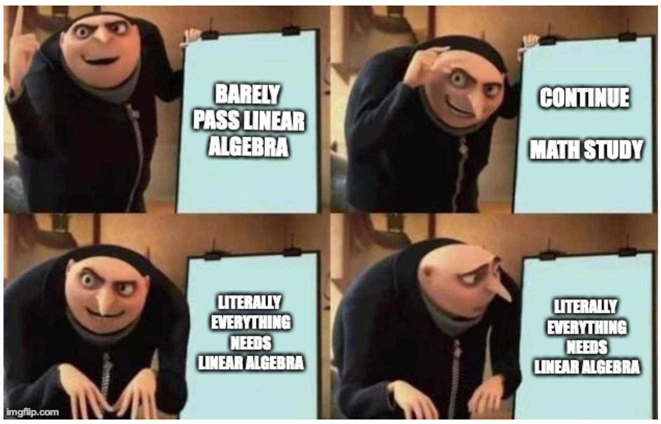
> **Note**
>
> Ein lineares Gleichungssystem $Ax = b$ ist genau dann konsistent, wenn
>
> $$\text{rk } A = \text{rk} \begin{bmatrix} A & b \end{bmatrix}$$

Der Rang wird mit dem Befehl

```
np.linalg.matrix_rank(A)
```

berechnet (`scipy` besitzt diese Funktion **nicht**)





Python
- Broadcasting -> 2 · [..., ..., ...]
- np.sum(<array>)