Fachhochschule Nordwestschweiz
Hochschule für Technik

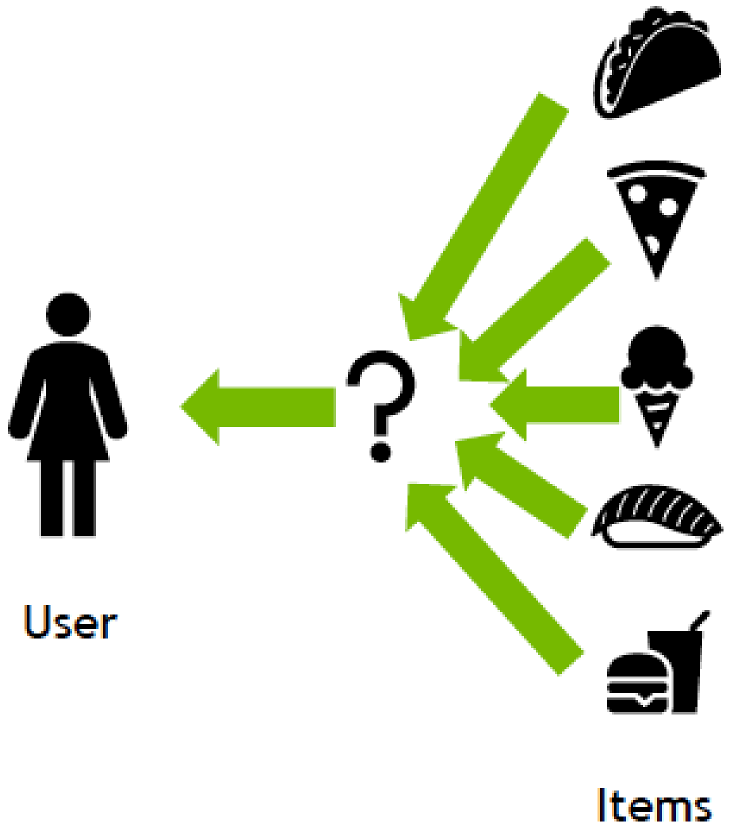
Anwendungsbeispiel 3: **Recommender-Systeme**

Anwendungen der linearen Algebra

Studiengang Data Science

Cédric Huwyler

5. Mai 2023



Anwendungsbeispiel nächste Stunde: Movie Recommender

simplemovierecommender.pythonanywhere.com

Simple Movie Recommender

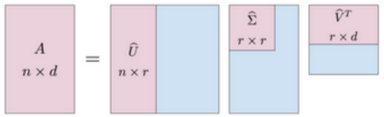
Based on the small Movie Lens dataset with movies until 2018 and cosine similarity.

find movies

Top 20 recommendations for Hackers (1995)

Cover	Title	Similarity	Genres
	Swordfish (2001)	0.4526	Action | Crime | Drama
	Snake Eyes (1998)	0.4485	Action | Crime | Mystery | Thriller
	Dude, Where's My Car? (2000)	0.4028	Comedy | Sci-Fi
	Short Circuit 2 (1988)	0.3953	Comedy | Sci-Fi
	Antitrust (2001)	0.3888	Crime | Drama | Thriller
	Not Another Teen Movie (2001)	0.3815	Comedy
	Lethal Weapon 4 (1998)	0.3800	Action | Comedy | Crime | Thriller
	Demolition Man (1993)	0.3796	Action | Adventure | Sci-Fi

Mini-Challenge



Anwendungen der linearen Algebra FS 2023

Fachexperten: Roger Burkhardt, Cédric Huwyler

Mini-Challenge 3 zum Thema *Recommender-Systeme*

In der dritten Mini-Challenge befassen wir uns mit Recommender-Systemen. Diese schlagen uns aufgrund unserer bisherigen Konsumpräferenzen neue Produkte vor, die wir auch mögen könnten. Recommender-Systemen begegnen wir in unserem digitalen Alltag typischerweise mehrmals täglich:

Aufgrund unserer bisherigen Historie (und, wie wir sehen werden, derjenigen von anderen Usern) schlägt uns zum Beispiel Youtube neue Videos, Netflix neue Filme und Serien, Spotify und Soundcloud neue Musik, Digitec und Amazon neue Produkte, Facebook und LinkedIn neue Freunde, NZZ und 20 Minuten neue Artikel, Google Play und Apple Store neue Apps, Google Maps und Trip Advisor neue Restaurants, die SBB App mögliche heutige Reisedestinationen, Partnerschaftsplattformen neue Partner vor, etc.

Manche solche Systeme nutzen wir aktiv und bewusst, die Präsenz anderer bemerken wir vielleicht gar nicht. Recommender-Systeme haben typischerweise entweder für dich einen Nutzen (Vereinfachung des Entdeckungsprozesses), für die Firma (Cross-Selling / weiteren Konsum begünstigen) oder für beide. Wem ein Recommender-System nützt ist oft kritisch zu hinterfragen - in vielen Fällen gewichtet die Firma ihre Interessen höher als deine. Zum Beispiel hat Youtube den Anreiz, dir jeweils viele kurze Videos ohne grossen Inhalt zu präsentieren, um dich länger auf der Plattform zu halten und mehr Werbung konsumieren zu lassen. Anstatt Videos vorgeschlagen zu bekommen, die dir im Leben tatsächlich einen Mehrwert bieten, wirst du etwas übertrieben gesagt ein von banalen Videos abhängiger Konsument.

Übung
Überlege dir, welche Recommender-Systeme du täglich nutzt. Welche davon sind klar als Recommender-Systeme erkennbar, welche vielleicht weniger? Hinterfrage auch kritisch, welche dieser Systeme eher dir und welche eher der Firma einen Mehrwert bieten. Welchen Mehrwert möchtest du aus dem Recommender-System ziehen und an welchem Mehrwert mag die Firma interessiert sein?

Musterlösung:

Zu dieser Minichallenge existiert (bewusst) keine Musterlösung. Du kannst sie aber gerne zur Durchsicht und Kommentierung uns entweder in einer der Kontaktstunden vorstellen oder zuschicken (cedric.huwyler@fhnw.ch).

Libraries:

Es reicht in dieser Übung aus, die folgenden Python-Libraries zu benutzen:

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Recommender-Systeme

Mehr dazu bei ..

Lernziele dieser Lektion

- Du weisst, was ein **Recommender-System** ist und in welchen Anwendungsbereichen es grob eingesetzt wird.

- Du hast verstanden, was **Collaborative Filtering** ist und welche Rolle die **Interaktionsmatrix** dabei spielt.

- Du kennst die **Cosine-Similarity** als **Ähnlichkeitsmass** für Vektoren und kannst diese für zwei beliebige Vektoren berechnen.

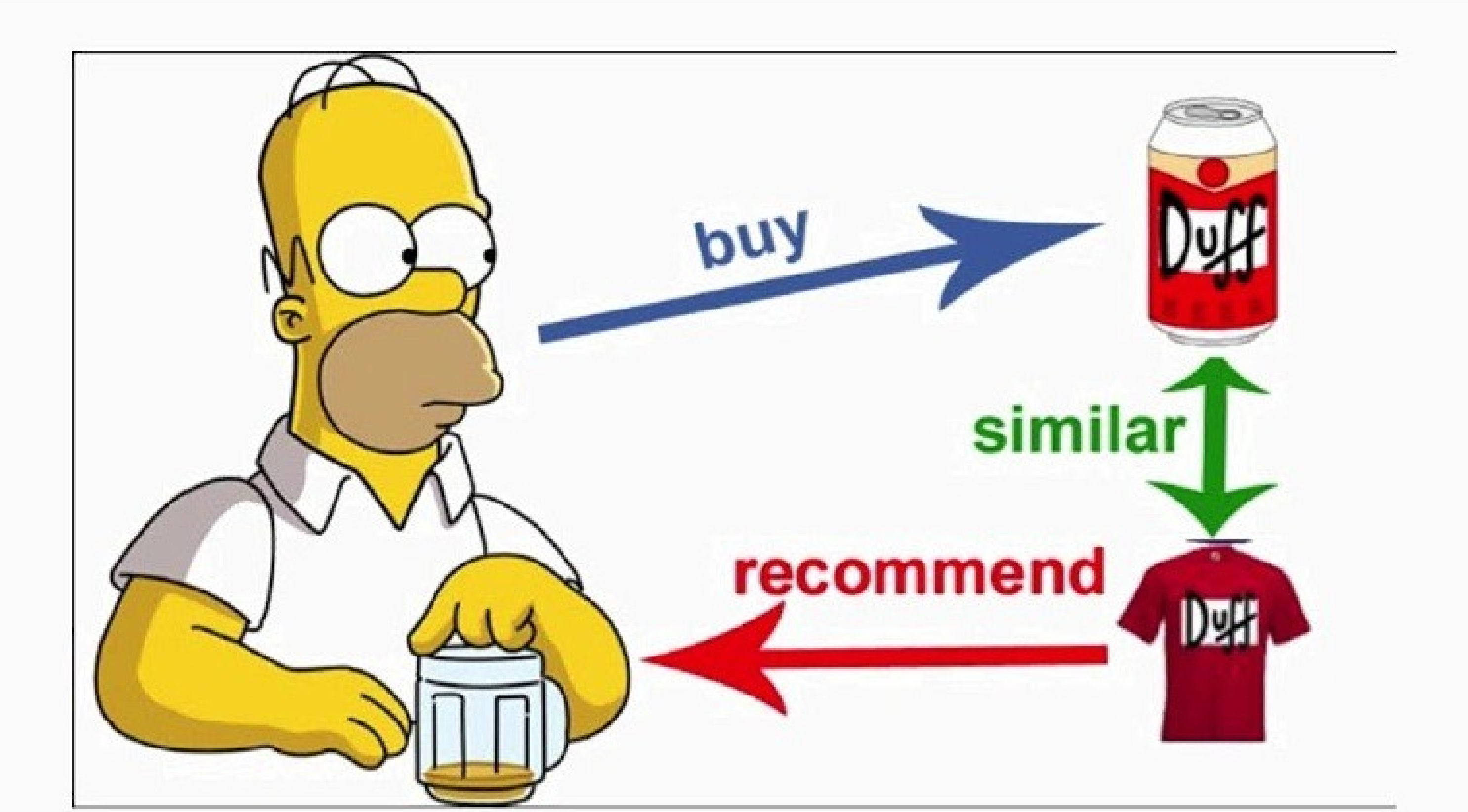
- Du verstehst, was eine **Ähnlichkeitsmatrix** ist, insbesondere ihre Eigenschaften und wie sie **effizient** berechnet wird.

- Du verstehst, wie die Cosine-Similarity benutzt werden kann, um **Empfehlungen** zu machen.

Wofür werden Recommender eingesetzt?

- Auf **Online-Plattformen** (E-Commerce, Medien, Entertainment, Soziale Netze, ..)
  - Up- und Cross-Selling
  - Alternativen zu nicht-verfügbaren Produkten anbieten
  - Erhöhung der Loyalität der bestehenden Benutzer:innen
- **Personalisierte Angebote** für bestehende Kund:innen. Zum Beispiel:
  - Empfehlung von Finanzprodukten im Banking- und Versicherungsbereich
  - Marketing-Kampagnen wie Migros Cumulus – welche Gutscheine werden ausgestellt?
- Charakterisierung von Kund:innen für **Segmentierungen**
  - Fallbeispiel *Saviva*: Ähnlichkeitsgruppen für Bestellkunden
- **Personalisierte Behandlung** in der Medizin, usw.

Inhalts- und kollaborationsbasierte Recommender

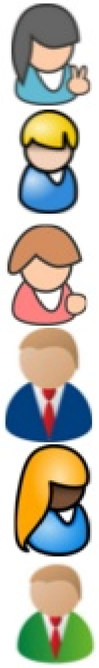


Bildquelle: Ming Zhi Sha

Inhaltsbasiert
Empfehlung aufgrund von ähnlichen Produkteigenschaften

Kollaborationsbasiert
Empfehlung, weil andere Benutzer:innen dieses Produkt zusätzlich oft auch noch konsumiert haben

Item-Based Collaborative Filtering



	1	5			
4	3			5	
5	5	1	3	4	
2				3	5
	3	1			4
	4		4		5

Interaktionsmatrix X:

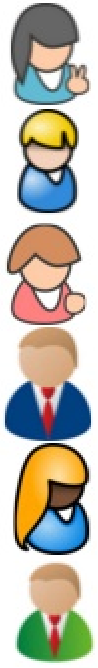
Datengrundlage für Collaborative Filtering

Item-Based Collaborative Filtering (IBCF):

- **Axiom:** Zwei Filme sind ähnlich, wenn sie von möglichst vielen Benutzer:innen ähnlich bewertet wurden.
- **Konsequenz:** Mag Person A den Film X, so empfehle ihr auch den zu X ähnlichsten Film Y.

Ähnlichkeitsmasse zwischen Vektoren

“Zwei Filme sind ähnlich, wenn sie von möglichst vielen Benutzer:innen ähnlich bewertet wurden.”

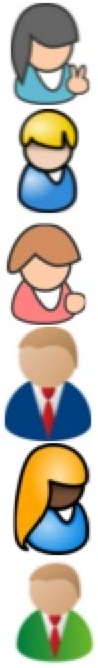


$X =$

	1	5			
4	3			5	
5	5	1	3	4	
2				3	5
	3	1			4
	4		4		5

Ähnlichkeitsmasse zwischen Vektoren

“Zwei Filme sind ähnlich, wenn sie von möglichst vielen Benutzer:innen ähnlich bewertet wurden.”

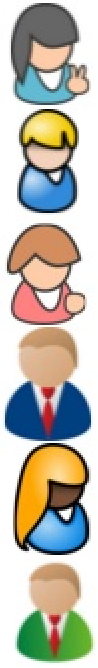


$X =$

	1	5			
4	3			5	
5	5	1	3	4	
2				3	5
	3	1			4
	4		4		5

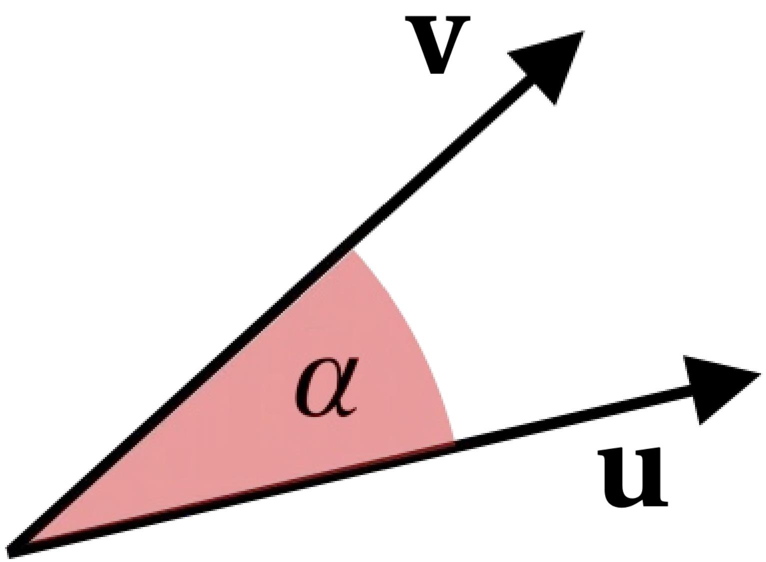
Ähnlichkeitsmasse zwischen Vektoren

“Zwei Filme sind ähnlich, wenn sie von möglichst vielen Benutzer:innen ähnlich bewertet wurden.”



$$X = \begin{bmatrix} 0 & 1 & 5 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 5 & 0 \\ 5 & 5 & 1 & 3 & 4 & 0 \\ 2 & 0 & 0 & 0 & 3 & 5 \\ 0 & 3 & 1 & 0 & 0 & 4 \\ 0 & 4 & 0 & 4 & 0 & 5 \end{bmatrix}$$

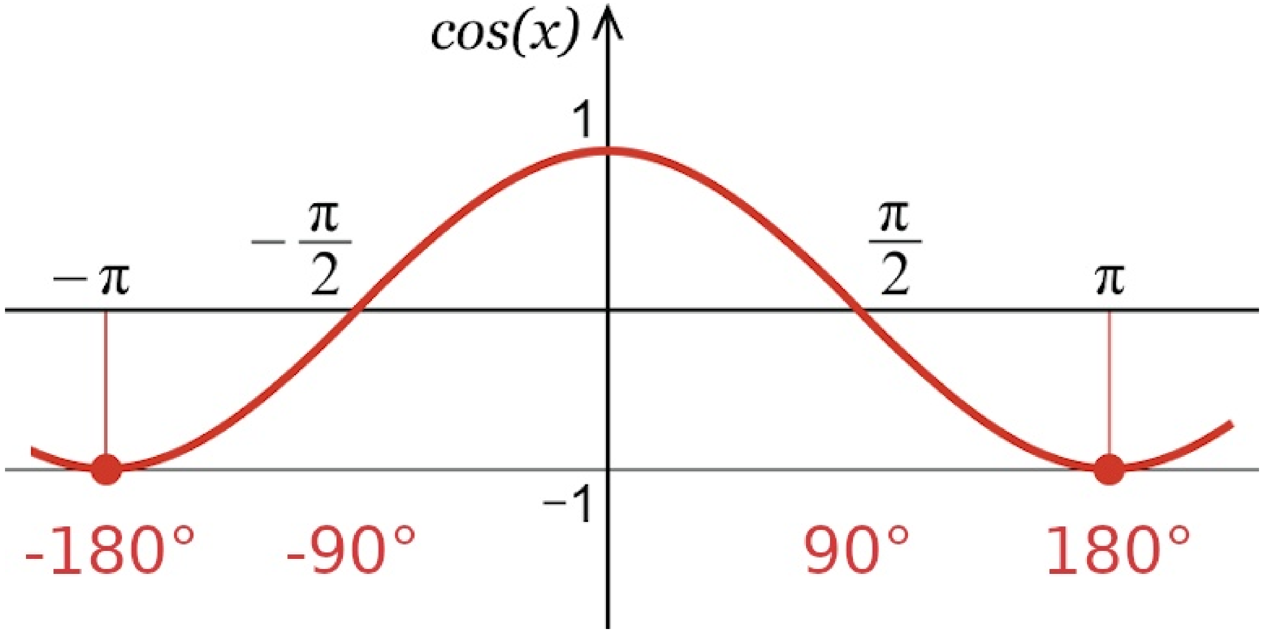
Repetition: Die Zwischenwinkelformel



$$\cos\alpha = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}|\,|\mathbf{v}|}$$

$$\mathbf{u}\cdot\mathbf{v} = |\mathbf{u}|\,|\mathbf{v}|\,\cos\alpha = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Repetition: Die Zwischenwinkelformel



$$\cos\alpha = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}|\,|\mathbf{v}|}$$

Spezialfall:

$$u_i, v_i \geq 0 \;\forall i$$

$$\Rightarrow \quad \alpha \in [-90^\circ, 90^\circ]$$

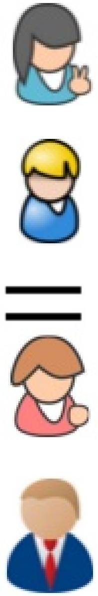
$$\Rightarrow \quad \underline{\cos\alpha \in [0,1]}$$

Die Cosine-Similarity

$$s(\mathbf{u},\mathbf{v}) = \cos\alpha = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}|\,|\mathbf{v}|}$$

- \+ Benötigt die Kosinusfunktion nicht
- \+ symmetrisch
- \+ Erzeugt automatisch Werte auf Skala zwischen 0 und 1
- \+ Sehr effizient zu berechnen
- \+ Berücksichtigt im Zähler nur Fälle, wo beide Bewertungen vorliegen
- – .. ist natürlich nur ein Modell für Ähnlichkeit!

Übung: Berechnung der Cosine-Similarity



$$X = \begin{array}{ccc} \begin{pmatrix} 0 & 1 & 5 \\ 4 & 3 & 0 \\ 5 & 5 & 1 \\ 2 & 0 & 0 \end{pmatrix} \\ \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \end{array}$$

Gegeben sei die folgende Interaktionsmatrix X.

Berechne alle Cosine-Similarites zwischen den drei Filmen.

$$s_{ij} = \cos\alpha = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|}$$

Welchen Film würdest du einer Person empfehlen, die den ersten Film mag?

Die Ähnlichkeitsmatrix S

$$s_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|}$$

$$S = \begin{pmatrix} 1 & 0.93 & 0.15 \\ 0.93 & 1 & 0.33 \\ 0.15 & 0.33 & 1 \end{pmatrix}$$

$$s_{12} = s_{21} \approx 0.93$$

$$s_{13} = s_{31} \approx 0.15$$

$$s_{23} = s_{32} \approx 0.33$$

- S ist quadratisch:
  Anzahl Zeilen und Spalten = Anzahl Produkte
- S ist symmetrisch
- Alle Elemente von S sind zwischen 0 und 1
- Diagonalelemente von S sind 1

Berechnung der Ähnlichkeitsmatrix mit Python

$$s_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|}$$

Gegeben sei die Interaktionsmatrix X:

Wie könnte nun die ganze Ähnlichkeitsmatrix S mit Python berechnet werden?

Welchen Ansatz würdest du wählen?

Berechnung der Ähnlichkeitsmatrix mit einem For-Loop

n = 9724

```
n = X.shape[1]
S = np.zeros( (n, n) )
for i in range(n):
    for j in range(n):
        x1 = X[:,i]
        x2 = X[:,j]
        S2[i,j] = np.dot( x1, x2 ) / (np.linalg.norm(x1)*np.linalg.norm(x2))
```

```
100% ████████████████████ 9724/9724 [06:58<00:00, 18.26it/s]
```

Berechnung der Ähnlichkeitsmatrix mit einem For-Loop

```
n = X.shape[1]
S = np.zeros(
for i in range
    for j in r
        x1 = X
        x2 = X
        S2[i,j                                   norm(x2))
```

100% , 18.26it/s]

Berechnung der A… em For-Loop

```
n = X.shape[1]
S = np.zeros(
for i in range
    for j in r
        x1 = X
        x2 = X
        S2[i,j                                          norm(x2))
```

100% … 18.26it/s]

Berechnung der Ähnlichkeitsmatrix mit Python

$$s_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|}$$

Gibt es einen Weg, die Ähnlichkeitsmatrix S **ausschliesslich über Matrizenmultiplikationen** zu berechnen?

+ Keine Zeitverluste über for-Loops
+ Arbeitslast kann beliebig auf CPU, mehreren CPUs oder GPUs verteilt werden

Vektorisierte Berechnung der Ähnlichkeitsmatrix

$$X = \begin{pmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{pmatrix}$$

$$X^T X = \begin{pmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \cdots & \\ — & \mathbf{x}_n & — \end{pmatrix} \begin{pmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T \mathbf{x}_1 & \mathbf{x}_1^T \mathbf{x}_2 & \dots & \mathbf{x}_1^T \mathbf{x}_n \\ \mathbf{x}_2^T \mathbf{x}_1 & \mathbf{x}_2^T \mathbf{x}_2 & \dots & \mathbf{x}_2^T \mathbf{x}_n \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_n^T \mathbf{x}_1 & \mathbf{x}_n^T \mathbf{x}_2 & \dots & \mathbf{x}_n^T \mathbf{x}_n \end{pmatrix}$$

Vektorisierte Berechnung der Ähnlichkeitsmatrix

$$s_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|} = \mathbf{x}_i \cdot \mathbf{x}_j$$

$$|\mathbf{x}_i| = |\mathbf{x}_j| = 1$$

s_{12}

$$S = \begin{pmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \boxed{\mathbf{x}_1^T\mathbf{x}_2} & \dots & \mathbf{x}_1^T\mathbf{x}_n \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \dots & \mathbf{x}_2^T\mathbf{x}_n \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_n^T\mathbf{x}_1 & \mathbf{x}_n^T\mathbf{x}_2 & \dots & \mathbf{x}_n^T\mathbf{x}_n \end{pmatrix}$$

Vektorisierte Berechnung der Ähnlichkeitsmatrix

Schritt 1: Interaktionsmatrix X normalisieren

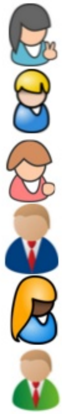
$$|\mathbf{x}_1| = |\mathbf{x}_2| = \cdots = |\mathbf{x}_n| = 1$$

Broadcasting!

Schritt 2: Ähnlichkeitsmatrix berechnen

$$S = X^T X$$

Mathematik eines einfachen Recommenders



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

$$\mathbf{x}_1$$

Ähnlichkeit zweier Produkt (Spalten)-Vektoren über Cosine-Similarity:

$$s_{ij} = \cos\alpha = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|}$$

Gegeben Produkt *i*, empfehle Produkt *j* mit maximaler Cosine-Similarity zu Produkt *i*.

Effiziente Berechnung der Ähnlichkeitsmatrix (**keine for-Loops benutzen**): 😿

$$S = X^T X$$ (mit vorgängiger Normierung der Spalten von *X*)

Lernziele dieser Lektion

- ✔ Du weisst, was ein **Recommender-System** ist und in welchen Anwendungsbereichen es grob eingesetzt wird.
- ✔ Du hast verstanden, was **Collaborative Filtering** ist und welche Rolle die **Interaktionsmatrix** dabei spielt.
- ✔ Du kennst die **Cosine-Similarity** als **Ähnlichkeitsmass** für Vektoren und kannst diese für zwei beliebige Vektoren berechnen.
- ✔ Du verstehst, was eine **Ähnlichkeitsmatrix** ist, insbesondere ihre Eigenschaften und wie sie **effizient** berechnet wird.
- ✔ Du verstehst, wie die Cosine-Similarity benutzt werden kann, um **Empfehlungen** zu machen.

Ausblick

In der nächsten Lektion wirst du die erworbenen Kenntnisse praktisch umsetzen und dir unter Anleitung selbst einen Film-Recommender bauen.

Natürlich existiert noch eine Vielzahl weiterer Ansätze, um Recommender-Systeme zu erstellen!