Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

MATRIX FACTORIZATION MIT STOCHASTIC GRADIENT DESCENT ALS LERNALGORITHMUS

Fachseminar "Machine Learning"

Letztes Update: 26. Januar 2016

Tim Piechotka

Studienbereich Informatik
Hochschule RheinMain

EINLEITUNG

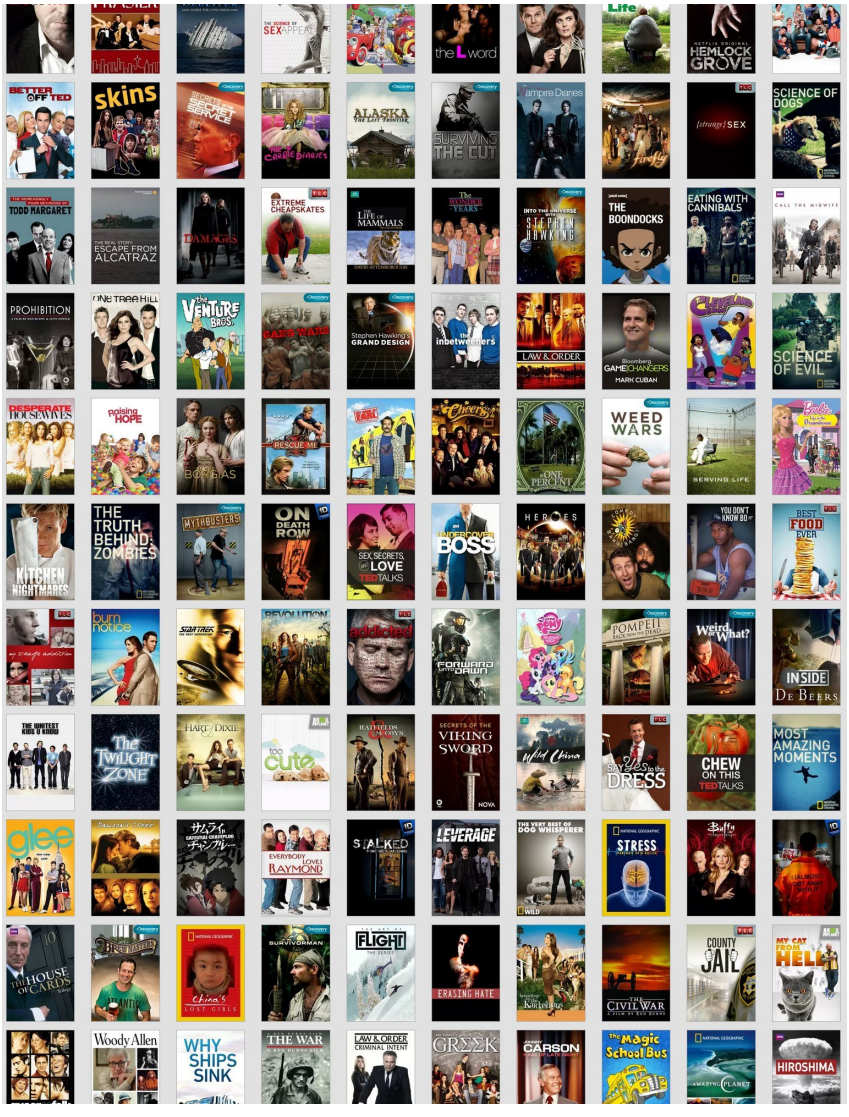
WAS SIND RECOMMENDER SYSTEMS?

Definition nach Wikipedia

Ein Empfehlungsdienst (englisch Recommender System) ist ein Softwaresystem welches das Ziel hat eine Vorhersage zu treffen, die quantifiziert wie stark das Interesse eines Benutzers an einem Objekt ist.

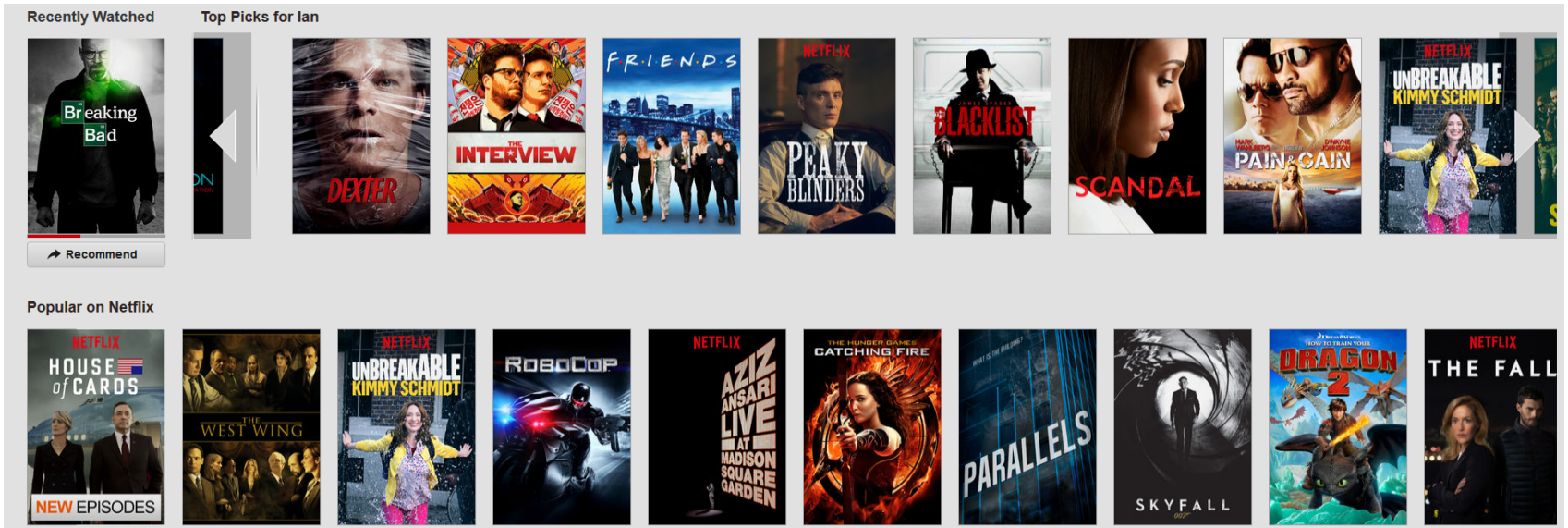
- Im Idealfall werden dem Benutzer nur Gegenstände empfohlen, welche ihm gefallen
- der Benutzer hat keinen Kontakt zu Items, welche ihm missfallen

WOFÜR WERDEN SIE BENÖTIGT?



Von watson.ch

- Recommender Systems schaffen Abhilfe
- durch ein gutes Recommender System wird die Bindung von Nutzern zum Anbieter stärker



Von blog.manugarri.com

NETFLIX PRIZE COMPETITION

- 2006 - 2009
- Ziel: 10% besser als Cinematch
- 100 Millionen Bewertungen zu 17.000 Filmen von 500.000 Usern
- hat sehr große Wellen geschlagen

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20

Von http://www.netflixprize.com/leaderboard

CONTENT FILTERING

- Jeder User und jedes Item besitzen ein Profil
- Diese Profile werden miteinander verglichen
- Profile hängen beispielsweise von Genre oder auch Schauspielern ab
- Je mehr Produkte der Benutzer bewertet, desto genauer wird sein Profil
- Anfänglich häufig Interviews

COLLABORATIVE FILTERING

- Auf Basis von bereits abgegebenen Bewertungen eines Users werden neue Bewertungen zu anderen Produkten vorausgesagt
- Weit verbreitet
- Einem Benutzer gefällt mit hoher Wahrscheinlichkeit ein Produkt, welches auch schon einem anderen Nutzer mit ähnlichen Geschmack gefallen hat
- Tripel (Benutzer, Gegenstand, Bewertung)
- Die Menge aller Tripel ist unsere Bewertungsmatrix

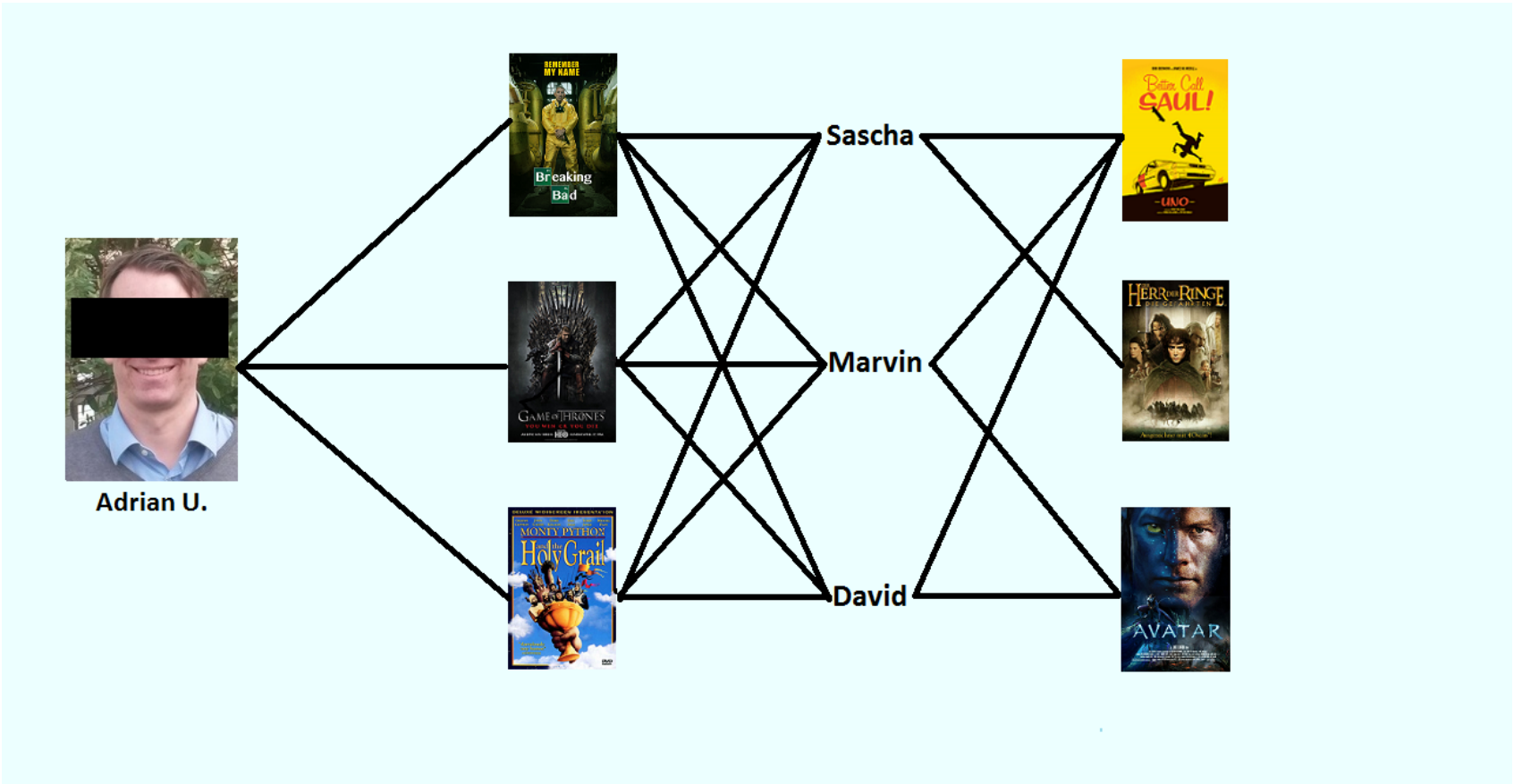
	Breaking Bad	Game of Thrones	Monty Python
Adrian	1	5	5
Marvin	2	4	-
David	5	-	5

ARTEN DES INPUTS

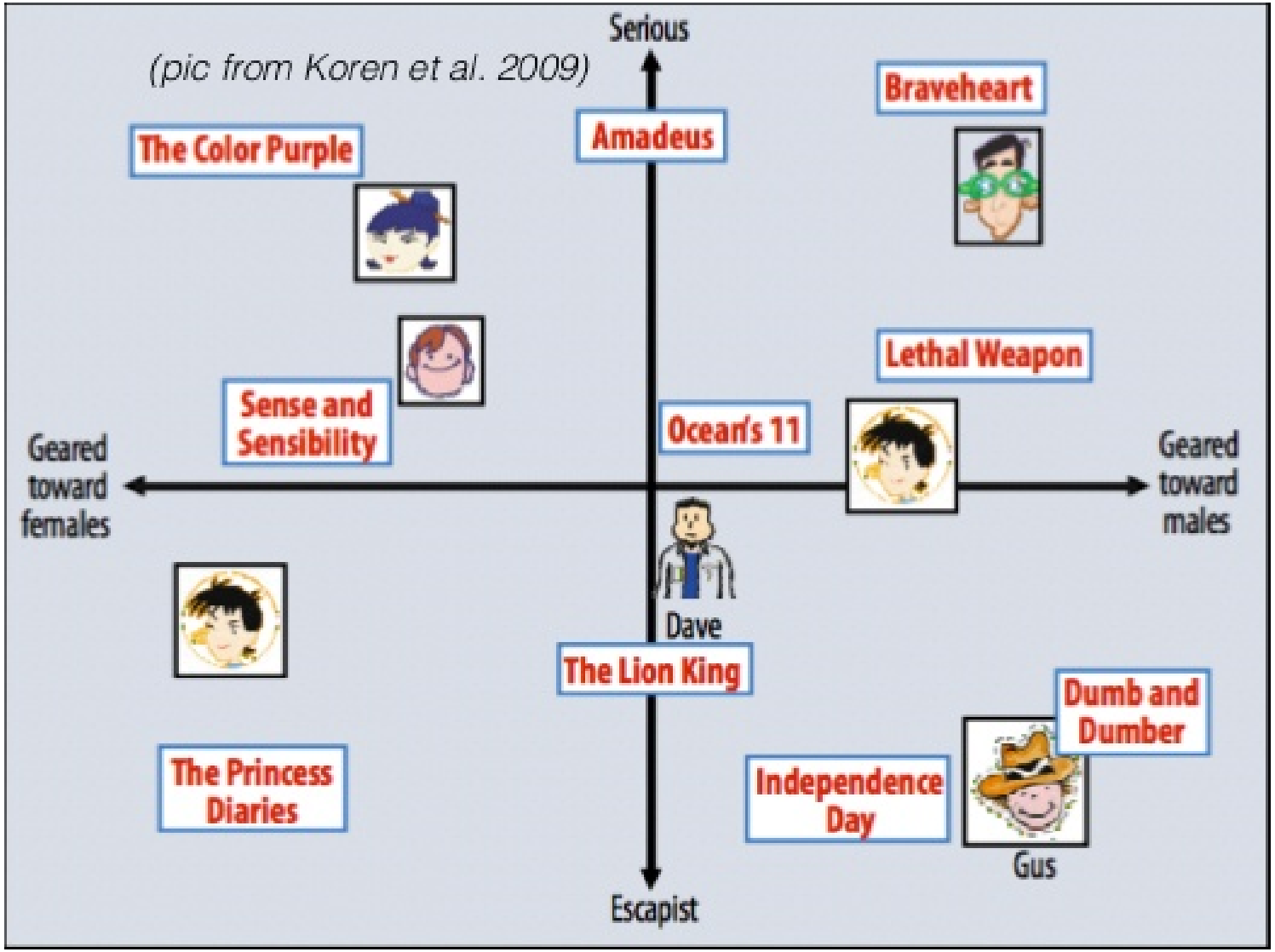
	Breaking Bad	Game of Thrones	Monty Python
Adrian	1	5	5
Marvin	2	4	-
David	5	-	5

- Explizites Feedback:
  - direktes Feedback des Users
  - qualitativ hochwertiges Feedback
  - spärlich gefüllte Matrix (sparse Matrix), da die meisten Benutzer nur zu wenigen Gegenständen eine Bewertung abgegeben haben
- Implizites Feedback:
  - Feedback, welches sich aus dem Verhalten des Users erschließen lässt
  - ungenauere oder auch falsche Daten
  - dicht gefüllte Matrix
  - negatives Feedback lässt sich kaum aus dem Verhalten des Benutzers ableiten

NEIGHBORHOOD METHODE



LATENT FACTOR



MATRIX FACTORIZATION

EINLEITUNG

- Die Grundidee ist es zu einer Matrix zwei andere Matrizen zu finden, welche miteinander multipliziert wieder die Ursprungsmatrix ergeben
- Matrix Factorization kann die Position eines Users oder eines Gegenstandes in dem Raum des Latent Factor Models bestimmen
- Das Ziel ist es durch MF jede Zelle der Bewertungsmatrix zu füllen
- Alle bekannten beziehungsweise benötigten Informationen stehen in der Bewertungsmatrix

MATHEMATIK DES GRUNDMODELLS

- Die Bewertungsmatrix R versuchen wir nun durch $P \times Q^T$ zu approximieren
- Aus der $|U| \times |I|$ Matrix R werden nun die $|U| \times L$ Matrix P und die $|I| \times L$ Matrix Q.
- L gibt die Anzahl unserer Faktoren des Latent Factor Models an

$$R \approx P \times Q^T = \hat{R}$$

BEISPIEL

$$R = \begin{pmatrix} 5 & 2 & 0 & 1 \\ 2 & 0 & 0 & 3 \\ 2 & 0 & 5 & 2 \\ 0 & 4 & 5 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} 2.24 & 0.74 \\ 0.79 & 1.59 \\ 1.33 & 0.93 \\ 1.12 & 1.63 \end{pmatrix} \quad Q = \begin{pmatrix} 1.61 & 0.73 \\ 0.58 & 1.63 \\ 2.49 & 1.44 \\ 0.35 & 0.96 \end{pmatrix}$$

$$\hat{R} = \begin{pmatrix} 4.14 & 2.50 & 6.65 & 1.50 \\ 2.43 & 3.06 & 4.26 & 1.82 \\ 2.83 & 2.30 & 4.67 & 1.37 \\ 2.98 & 3.30 & 5.12 & 1.96 \end{pmatrix}$$

BEISPIEL

$$R = \begin{pmatrix} 5 & 2 & 0 & 1 \\ 2 & 0 & 0 & 3 \\ 2 & 0 & 5 & 2 \\ 0 & 4 & 5 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} 2.24 & 0.74 \\ 0.79 & 1.59 \\ 1.33 & 0.93 \\ 1.12 & 1.63 \end{pmatrix} \quad Q = \begin{pmatrix} 1.61 & 0.73 \\ 0.58 & 1.63 \\ 2.49 & 1.44 \\ 0.35 & 0.96 \end{pmatrix}$$

$$\hat{R} = \begin{pmatrix} 4.14 & 2.50 & 6.65 & 1.50 \\ 2.43 & 3.06 & 4.26 & 1.82 \\ 2.83 & 2.30 & 4.67 & 1.37 \\ 2.98 & 3.30 & 5.12 & 1.96 \end{pmatrix}$$

BEISPIEL

$$R = \begin{pmatrix} 5 & 2 & 0 & 1 \\ 2 & 0 & 0 & 3 \\ 2 & 0 & 5 & 2 \\ 0 & 4 & 5 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} 2.24 & 0.74 \\ 0.79 & 1.59 \\ 1.33 & 0.93 \\ 1.12 & 1.63 \end{pmatrix} \quad Q = \begin{pmatrix} 1.61 & 0.73 \\ 0.58 & 1.63 \\ 2.49 & 1.44 \\ 0.35 & 0.96 \end{pmatrix}$$

$$\hat{R} = \begin{pmatrix} 4.14 & 2.50 & 6.65 & 1.50 \\ 2.43 & 3.06 & 4.26 & 1.82 \\ 2.83 & 2.30 & 4.67 & 1.37 \\ 2.98 & 3.30 & 5.12 & 1.96 \end{pmatrix}$$

MATHEMATIK DES GRUNDMODELLS

Rating

$$r_{ui} = p_u^T q_i$$

- Das Rating ist eine vom User u zu einen Item i vorausgesagte Bewertung
- Sind die Einstellungen zu einem Faktor verschieden, so wird das Ergebnis des Skalarprodukts und somit das Rating kleiner

BEISPIEL

Kleiner Ausschnitt

$$r_{ui} = \begin{pmatrix} .. \\ 1 \\ 0 \\ -1 \\ .. \end{pmatrix} \cdot \begin{pmatrix} .. \\ -1 \\ 1 \\ -1 \\ .. \end{pmatrix}$$

- Zwei unterschiedliche Vorzeichen $\rightarrow$ Verschlechtert das Rating
- Min. eine der Zahlen ist neutral(0)$\rightarrow$ Beeinflusst das Rating nicht direkt
- Zwei gleiche Vorzeichen $\rightarrow$ Verbessert das Rating
- Größere Zahlen beeinflussen das Rating stärker

REGULARIZED SQUARED ERROR (RSE)

- P und Q sind uns jedoch nicht bekannt
- Wir versuchen eine Matrix P und eine Matrix Q zu finden, welche die bekannten Ratings möglichst genau approximieren
- Wir berechnen also den Fehler zwischen den bekannten und den errechneten Werten

$$\min_{P,Q} \sum_{(u,i)} (r_{ui} - p_u q_i)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

- Dieses Optimierungsproblem kann mit Stochastic Gradient Descent gelöst werden

STOCHASTIC GRADIENT DESCENT

GRADIENT DESCENT

- Verfahren um Optimierungsprobleme zu lösen
- Ein Gradient einer Funktion f wird durch den Nablaoperator ∇f dargestellt und ist der Vektor der partiellen Ableitungen von f

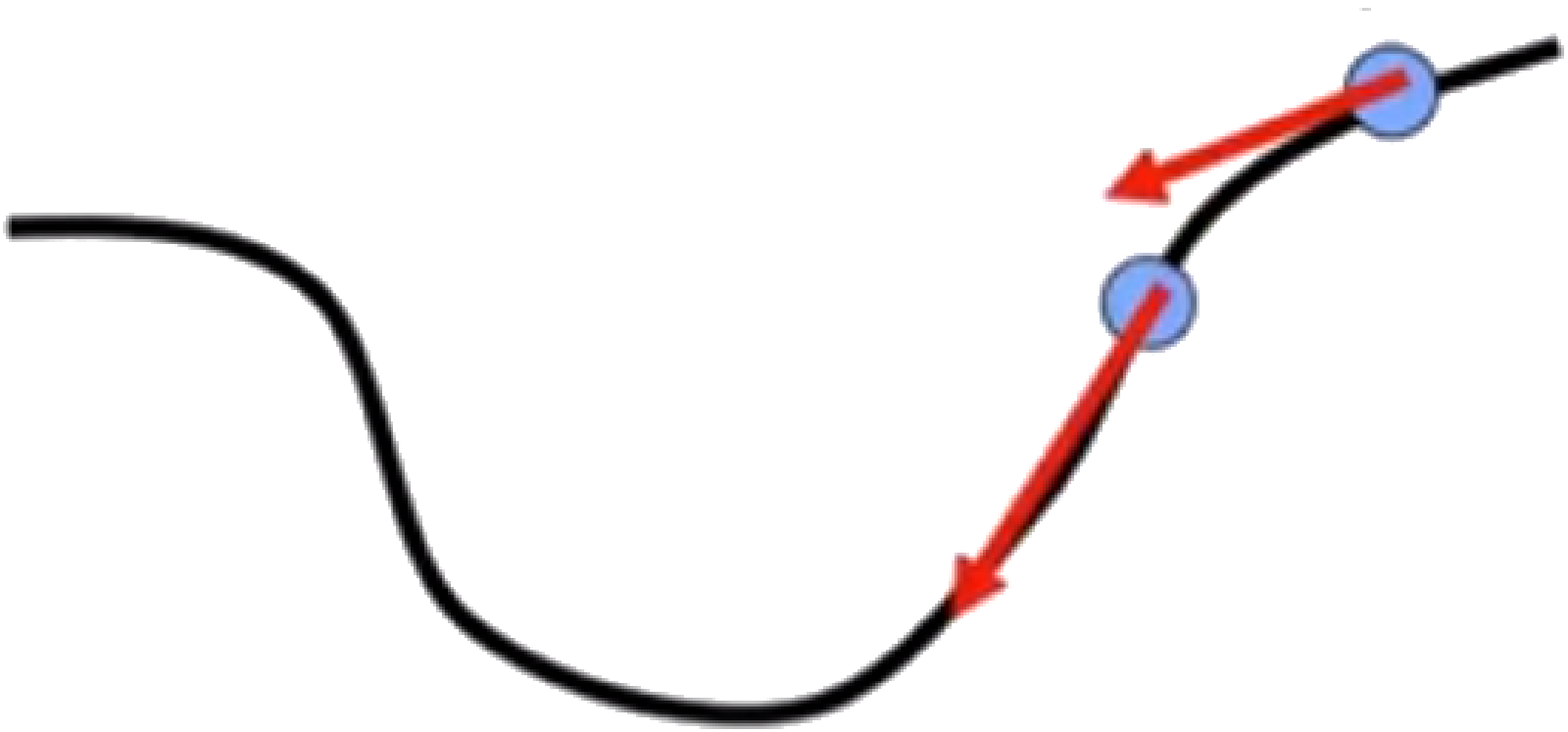
$$x_{k+1} = x_k - \gamma \cdot \nabla f(x_k)$$

- γ ist die Learningrate
- x_k ist der Startpunkt eines jeden Schritts

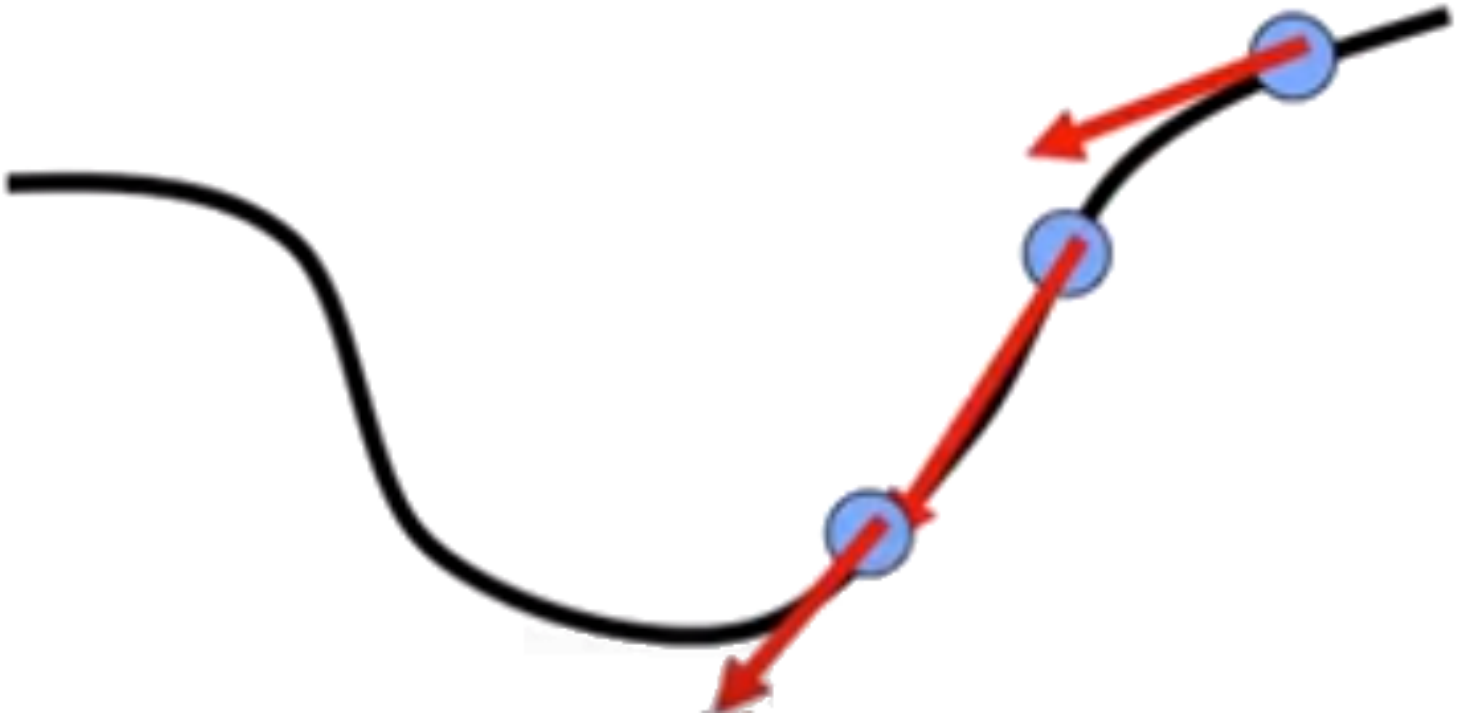
GRADIENT DESCENT

GRADIENT DESCENT

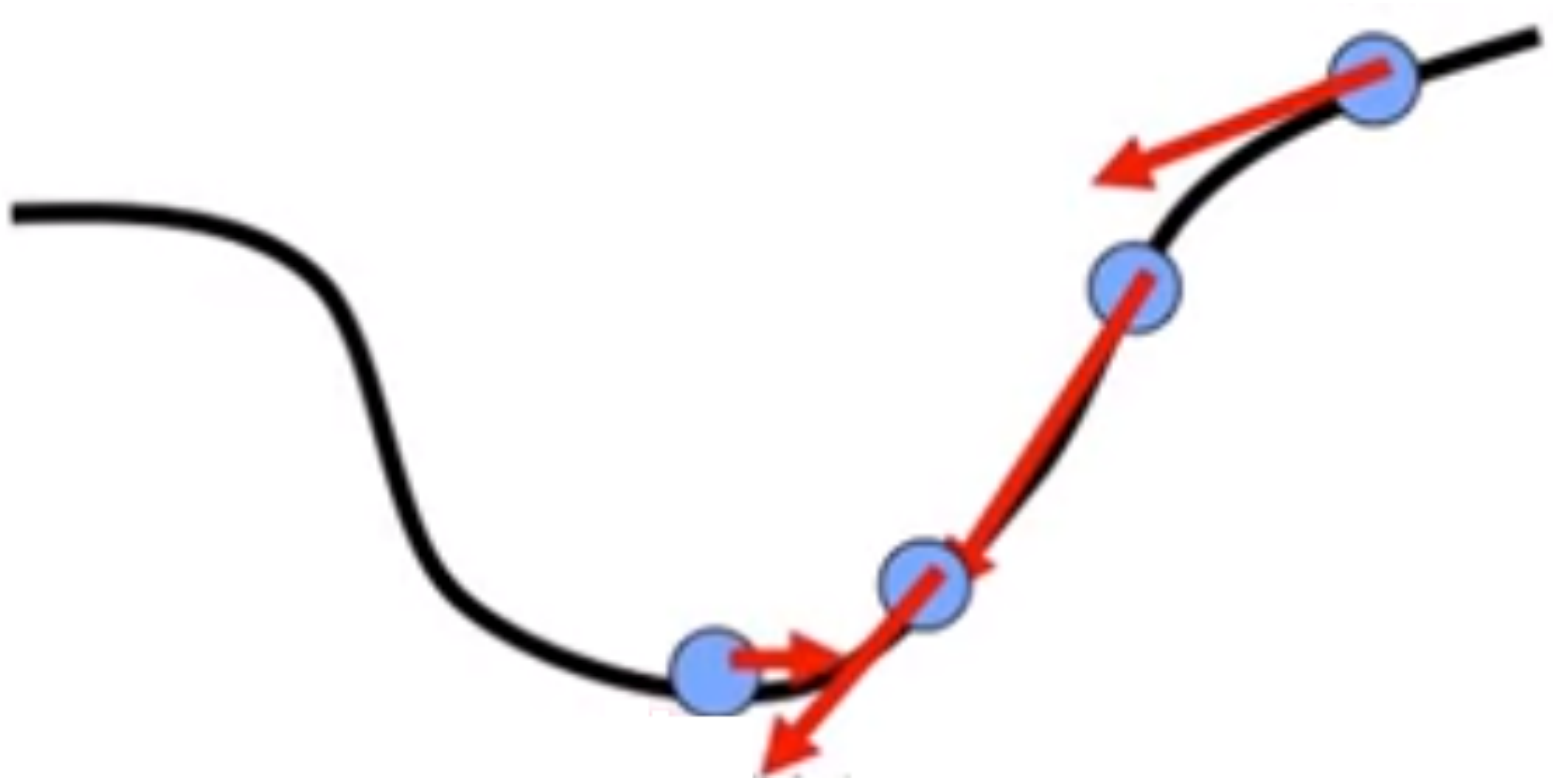
GRADIENT DESCENT



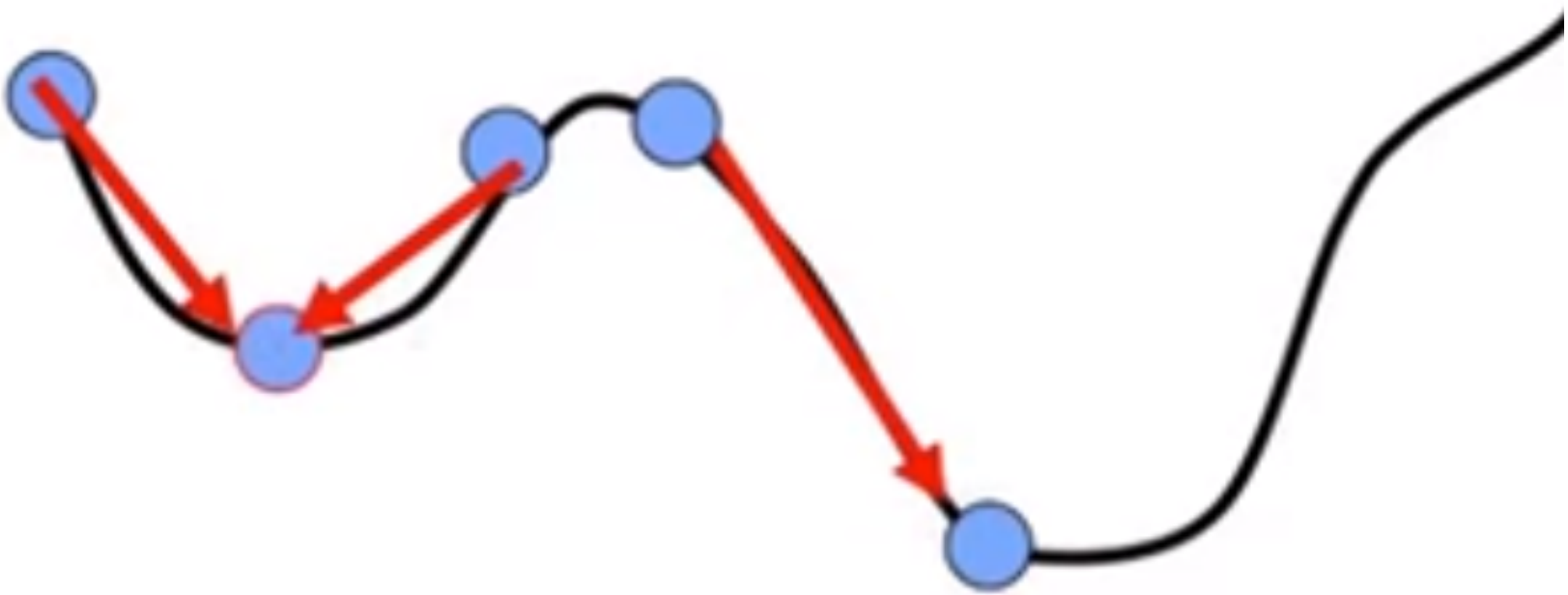
GRADIENT DESCENT



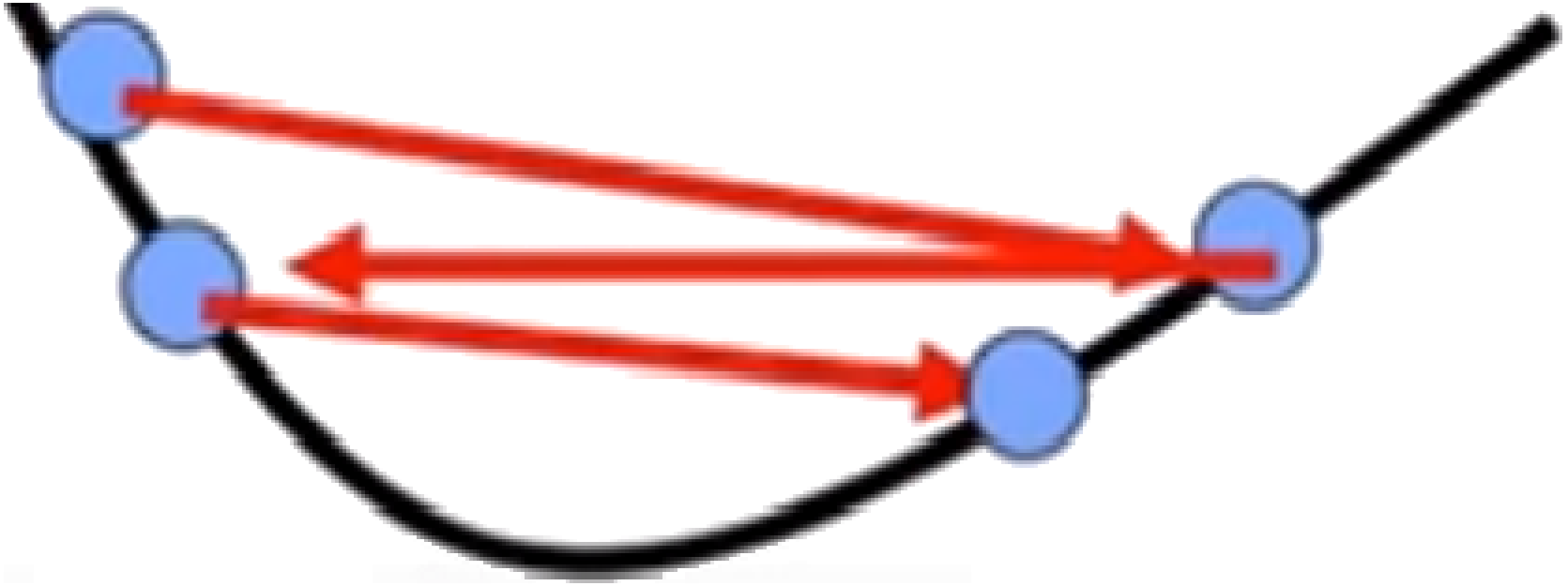
GRADIENT DESCENT



ABHÄNGIGKEIT DES STARTPUNKTS



WAHL DER LERNRATE



WAHL DER LERNRATE

STOCHASTIC GRADIENT DESCENT

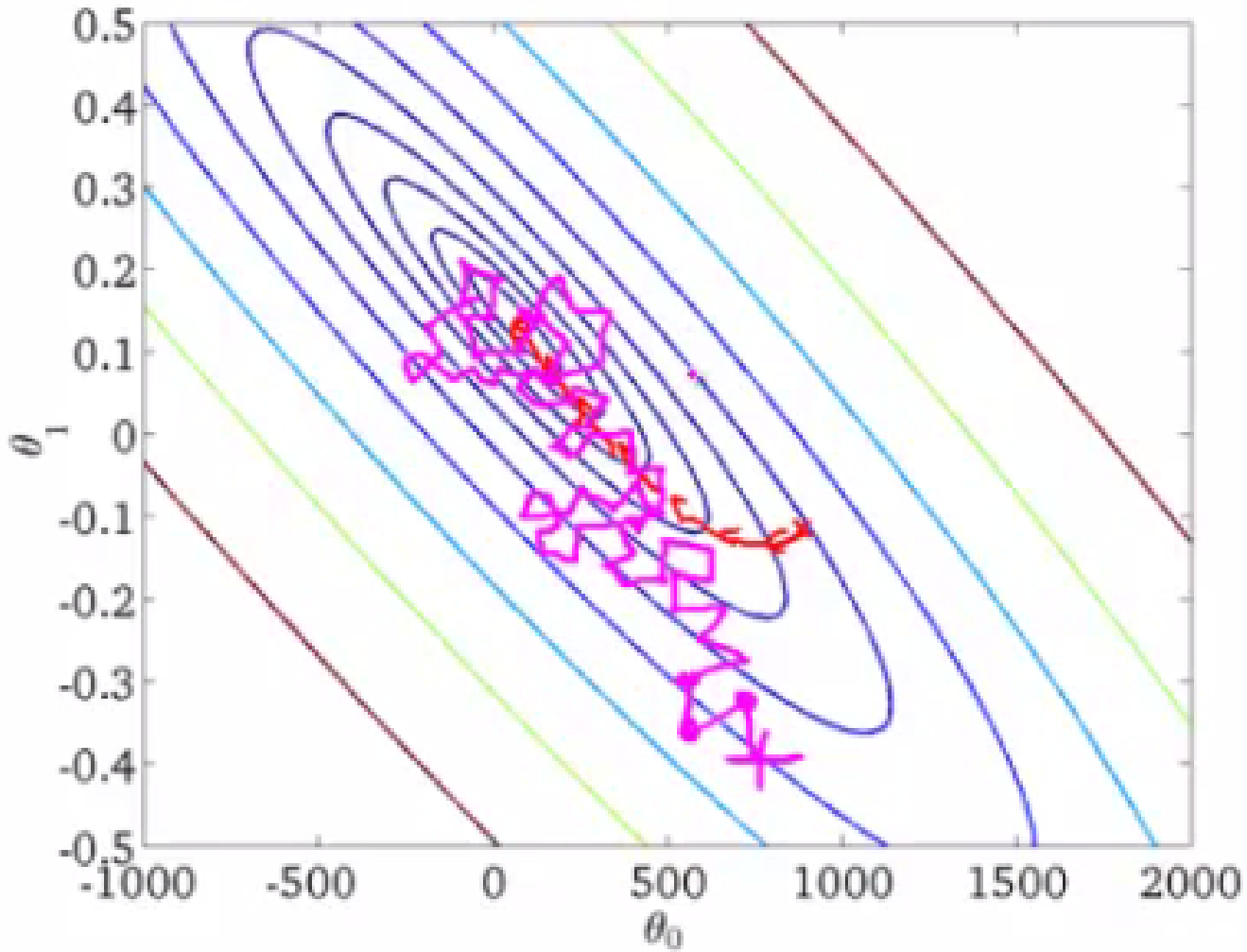
- Baut auf Gradient Descent auf
- Funktioniert nur auf Funktionen, welche in Summenform formuliert sind

$$f(x) = \sum_{i=0}^{n} g_i(x)$$

- Es wird in jedem Iterationsschritt nur ein zufälliges Sample i aus der Trainingsmenge betrachtet

$$x_{k+1} = x_k - \gamma \cdot \nabla g_i(x_k)$$

GRADIENT DESCENT VS. STOCHASTIC GRADIENT DESCENT



Von http://www.holehouse.org/

ANWENDUNG VON STOCHASTIC GRADIENT DESCENT

- Erinnern wir uns an unser Minimierungsproblem zurück

$$\min_{P,Q} \sum_{(u,i)} (r_{ui} - p_u q_i)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

- Wenden wir Stochastic Gradient Descent an, erhalten wir die Updateformeln für jeden Iterationsschritt:

$$q_i^{k+1} = q_i^k + \gamma \cdot (p_u \cdot (r_{ui} - q_i p_u) - \lambda \cdot q_i^k)$$

$$p_u^{k+1} = p_u^k + \gamma \cdot (q_i \cdot (r_{ui} - q_i p_u) - \lambda \cdot p_u^k)$$

FAZIT

FAZIT

- Immer mehr Dienste setzen auf Recommender Systems
- 75% der Benutzer von Netflix gucken Filme, welche ihnen empfohlen wurden
- Hat sich im Rahmen der NPC als überlegen herausgestellt
- Matrix Factorization funktioniert relativ zuverlässig bei ausreichend großer Bewertungsmatrix
- Sie lässt sich einfach erweitern

MITEINBEZIEHEN VON TENDENZEN

- Jeder User hat bei seinen Bewertungen Tendenzen $\rightarrow$ er bewertet Gegenstände in Vergleich zu anderen Benutzern besonders hoch oder niedrig
- Die Differenz zwischen dem Durchschnitt aller Bewertungen und dem Durchschnitt aller Bewertungen, welche ein User abgegeben hat, ist ein sogenannter Bias b_u
- Für jeden Gegenstand gibt es ebenfalls einen Bias b_i
- Mit diesem Wissen kann unsere Ratingsgleichung erweitert werden:

$$r_{ui} = \mu + b_u + b_i + p_u q_i$$

ZEITLICHE ABHÄNGIGKEIT

- Bisher: Alle abgegebenen Bewertungen des Users wirken sich gleich stark auf seine Empfehlungen aus
- Geschmäcker können sich jedoch entwickeln
- Lösung: Alle Dinge die sich über die Zeit ändern könnten, werden von Dieser abhängig gemacht

$$r_{ui}(t) = \mu + b_u(t) + b_i(t) + p_u(t)q_i$$

Es gibt mehrere Möglichkeiten diese Zeitabhängigkeiten umzusetzen:

1. Alte Bewertungen werden verworfen
2. Ältere Ratings werden weniger stark gewichtet

COLD START PROBLEM

- Einem Recommender System fällt es schwer einen neuen Benutzer im Raum einzuordnen, da dieser fast keine Bewertungen abgegeben hat
- Das wir von diesen Usern kein explizites Feedback haben, müssen wir auf implizites Feedback zurückgreifen
- In der Menge N(u) befinden sich alle Items, welche für den User demnach interessant wären
- In der Menge A(u) würden alle Gegenstände stehen, welche nach den angegebenen demografischen Daten zum User passen würden

COLD START PROBLEM

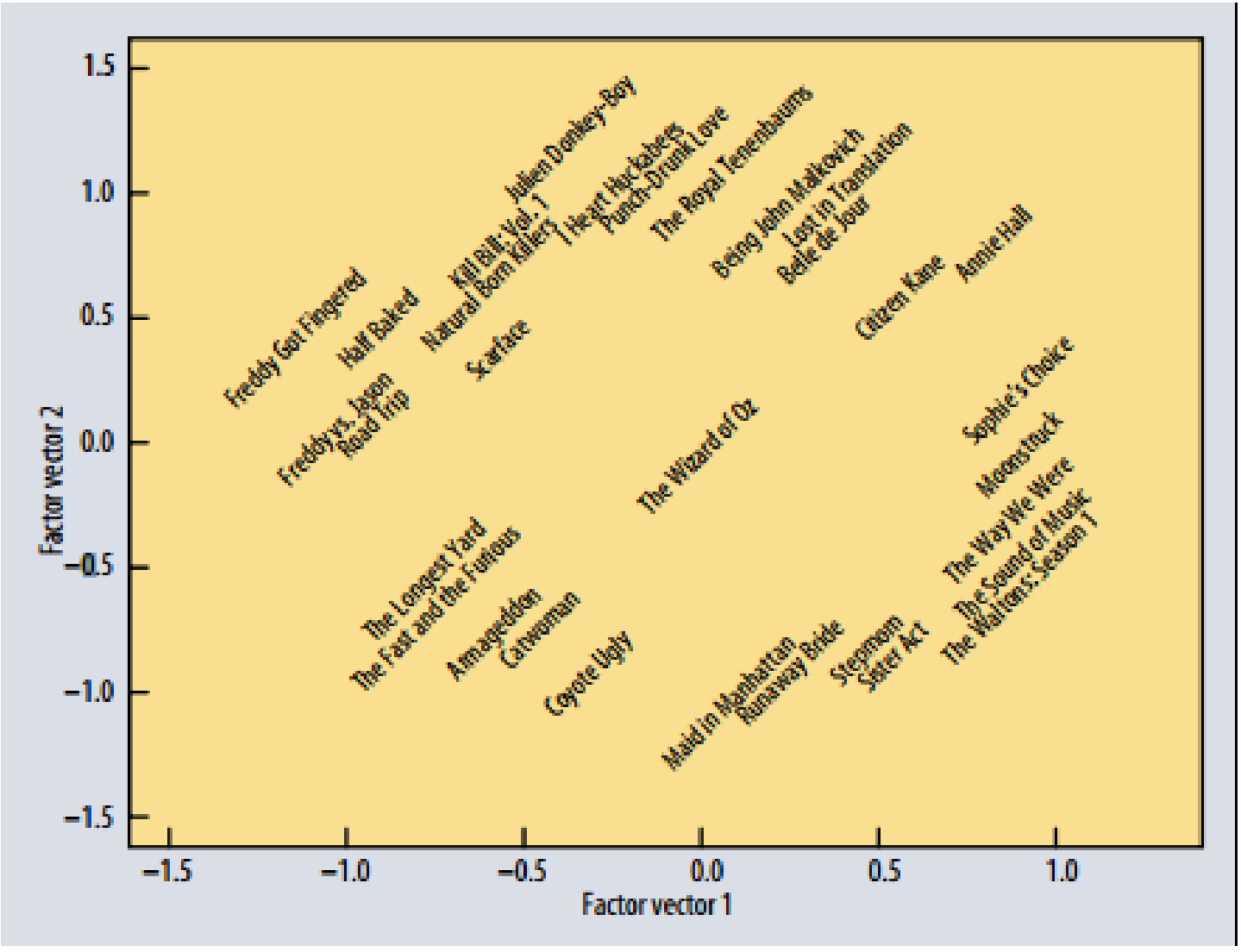
- Jeder Gegenstand dieser Menge hat bereit eine Position im Raum
- Mitteln wir nun alle Positionen, so erhalten wir die ungefähre Position des Users

$$|N(u)|^{-1} \sum_{i \in N(u)} x_i$$

- Damit können wir unsere Ratingsgleichung wieder anpassen

$$r_{ui} = b_{ui} + q_i \left[p_u + |N(u)|^{-1} \cdot \sum_{i \in N(u)} x_i + |A(u)|^{-1} \cdot \sum_{i \in A(u)} y_i \right]$$

BEISPIEL:ZWEI FAKTOREN



Von Yehuda Koren, Robert Bell and Chris Volinsky

Kurz in Python: Matrix Factorization mit GRADIENT DESCENT