

Data Mining

Empfehlungssysteme

Dr. Hanna Köpcke
Wintersemester 2020

Abteilung Datenbanken, Universität Leipzig
<http://dbs.uni-leipzig.de>

Übersicht

Hochdimensionale Daten

Clustering

Dimensions-
reduktion

Empfehlungs-
systeme

Assoziations-
regeln

Locality Sensitive
Hashing

Supervised ML

Graphdaten

Community
Detection

PageRank

Web Spam

Datenströme

Windowing

Filtern

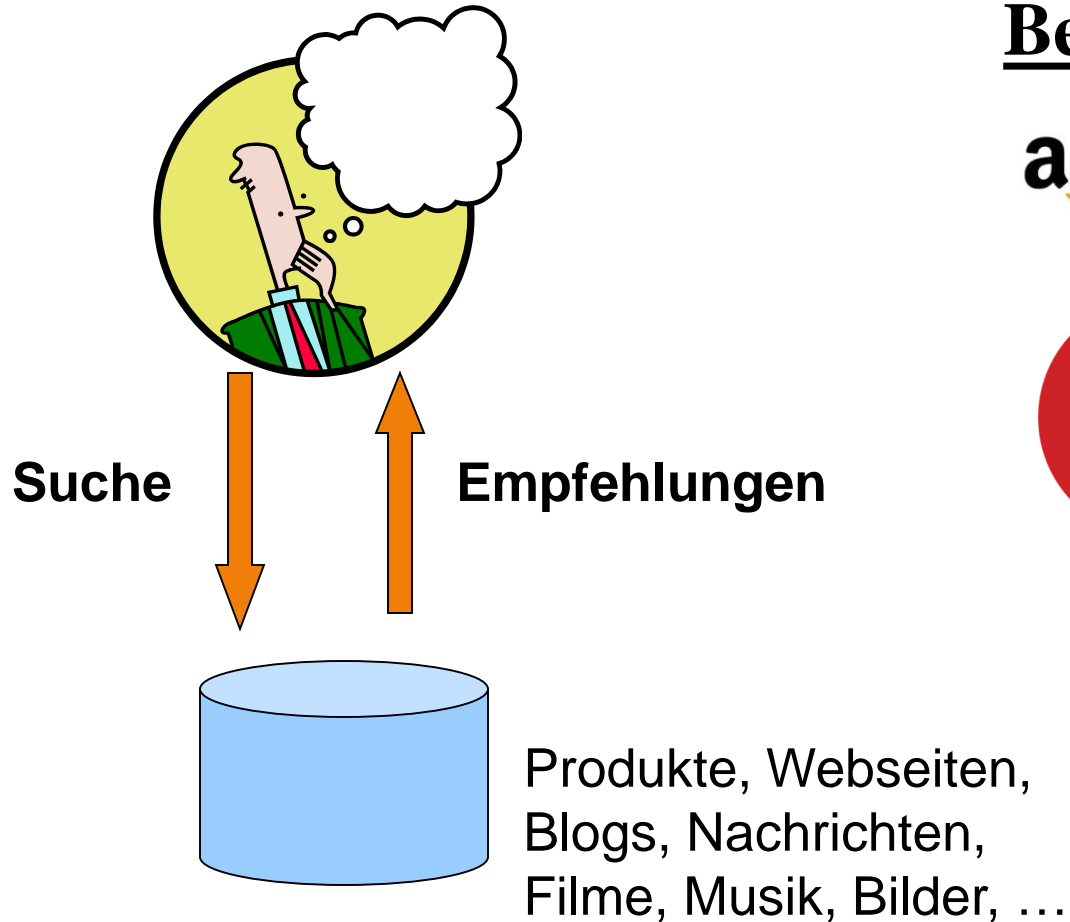
Momente

Web Advertising

Inhaltsverzeichnis

- **Einführung**
- **Inhaltsbasierte Analyse**
- **Kollaboratives Filtern**
- **Latentes Variablenmodell**

Empfehlungen



Beispiele:

amazon



NETFLIX

Google
News

last.fm
the social music revolution



Empfehlungen

- Unzählige Informationen im WWW benötigen Filter
- Arten der Empfehlung
 - Redaktion: Liste der beliebtesten/wichtigsten Produkte
 - Globale Aggregate: Meist gekauften Produkte, Neuesten Filme, ...
 - **Auf einzelnen Nutzer zugeschnitten**
- **Nutzenmatrix**
 - Menge von Nutzern
 - Menge von Objekten
 - Nutzenwerte in Zellen
 - Beispiel: 1-10 Sterne

	Avatar	LotR	Matrix	Pirates
Alice	10		2	
Bob		5		3
Carol	1		5	
David				4

Fragestellung

- **Schätzung der unbekannten Bewertungen**
 - Hauptsächlich möchte man die hohen Bewertungen wissen
 - Weniger interessant, welche Objekte nicht gemocht werden
- Drei Herangehensweisen (u.a.):
 1. Inhaltsbasiert Analyse
 2. Kollaboratives Filtern
 3. Latentes Variablenmodell

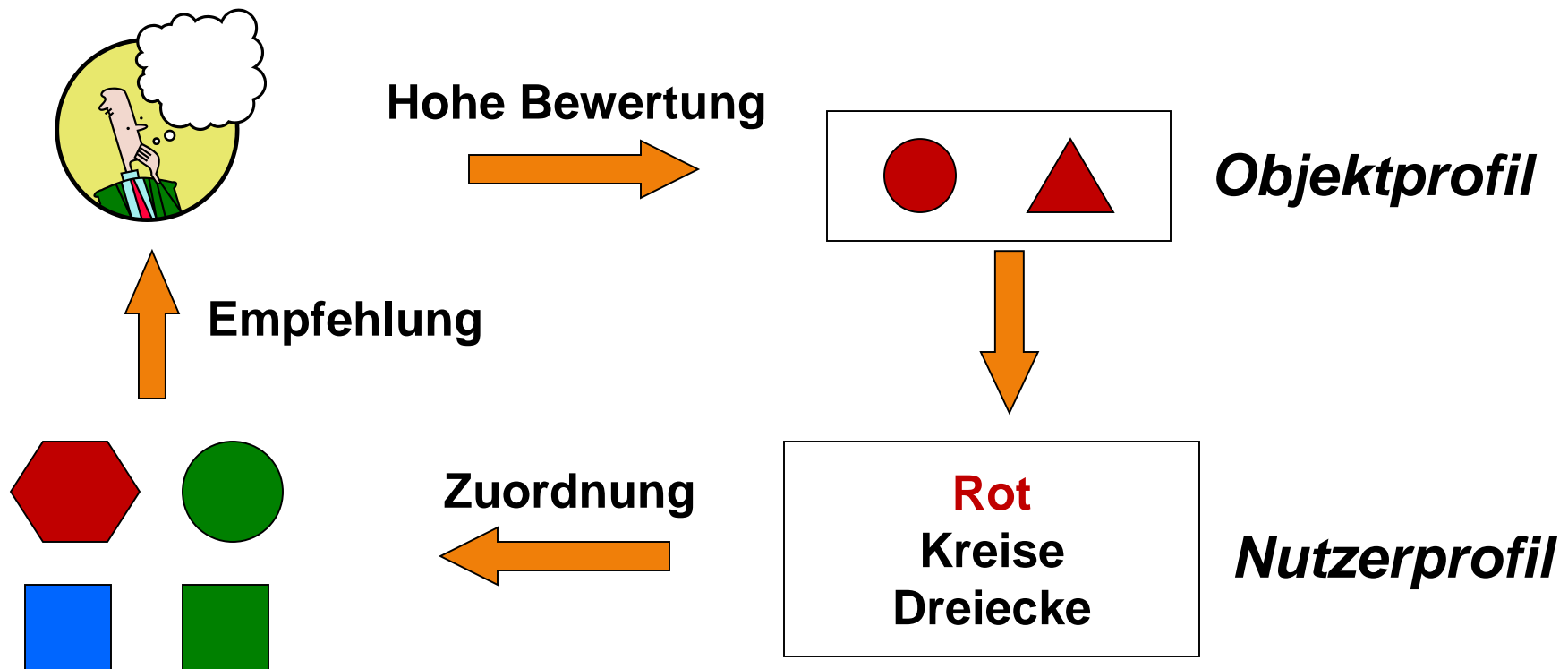
	Avatar	LotR	Matrix	Pirates
Alice	10		2	
Bob		5		3
Carol	1		5	
David				4

Inhaltsverzeichnis

- Einführung
- **Inhaltsbasierte Analyse**
- Kollaboratives Filtern
- Latentes Variablenmodell

Inhaltsbasierte Analyse

- Idee: Empfehlung von Objekten für einen Nutzer, wenn *ähnliche* Objekte von dem Nutzer positiv bewertet wurden
- Beispiele:
 - Empfehle Filme mit gleichen Schauspielern, Regisseur, Genre, ...
 - Empfehle Webseiten mit ähnlichen Themen/Wörtern



Inhaltsbasierte Analyse

- **Objektprofil:** Menge von Merkmalen
- z.B. Filme mit Schauspielern, Regisseur, Genre, ...

	Cameron	Wachowski	UK	Action	Fantasy
Avatar	1	0	1	1	1
Matrix	0	1	1	1	0

Nutzenmatrix

	Avatar	Matrix
Alice	10	2
Carol	1	5

Zentrierung

	Avatar	Matrix
Alice	4	-4
Carol	-2	2

- **Nutzerprofil:**

- Gewichteter Durchschnitt der bewerteten Objektprofile
- Gewichtet nach (zentrierter) Bewertung aus Nutzenmatrix

	Cameron	Wachowski	UK	Action	Fantasy
Alice	4	-4	0	0	4
Carol	-2	2	0	0	-2

Inhaltsbasierte Analyse

- Unbewerteter Film

	Cameron	Wachowski	UK	Action	Fantasy
Titanic	1	0	1	0	0

- Ähnlichkeit zwischen Objekt und Nutzer, z.B. über Kosinus-Ähnlichkeit

$$\cos(\mathbf{x}, \mathbf{i}) = \frac{\mathbf{x} \cdot \mathbf{i}}{||\mathbf{x}|| \cdot ||\mathbf{i}||} \in [-1, 1]$$

	Cameron	Wachowski	UK	Action	Fantasy
Alice	4	-4	0	0	4
Carol	-2	2	0	0	-2

- Beispiel

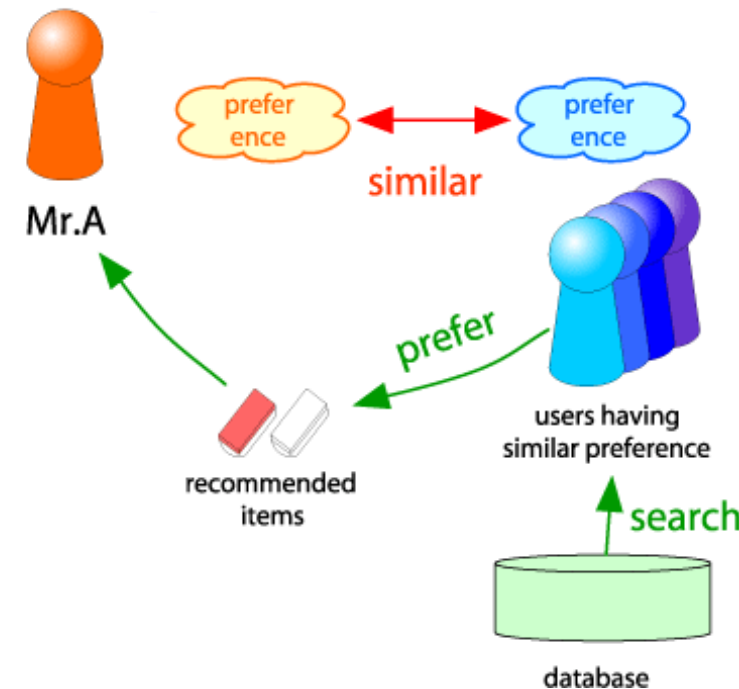
- Alice: $\cos(\mathbf{x}, \mathbf{i}) = \frac{4}{9.8} = 0.41$
- Carol: $\cos(\mathbf{x}, \mathbf{i}) = \frac{-2}{4.9} = -0.41$

Inhaltsverzeichnis

- Einführung
- Inhaltsbasierte Analyse
- **Kollaboratives Filtern**
- Latentes Variablenmodell

Kollaboratives Filtern

- KF für Nutzer
 - Suche nach einer Menge N von Nutzern mit ähnlichen Präferenzen (**Nutzenmatrix**)
 - Schätzung der unbekannten Bewertungen über die Bewertungen der Nutzer aus N
- Alternative: KF für Objekte
 - Suche nach einer Menge N von Objekten mit ähnlichen Bewertungen
 - Schätzen der unbekannten Bewertungen über die Bewertungen der Objekte aus N
- Vorteil: Objekte sind oft einfacher klassifizierbar als Menschen
 - Ein Musikalbum ist entweder Rock oder Klassik
 - Menschen können beide Musikrichtungen mögen
 - Es ist leicht möglich, dass zwei Menschen Rock mögen und gleichzeitig eine andere Musikrichtung, für die sich der jeweils andere gar nicht begeistert



Kollaboratives Filtern für Objekte

- Sei r_x der Vektor mit den Bewertungen des Nutzers X
- Keine Bewertung r_{xi} für Objekt i
- Sei $N(i; x)$ die Menge der k ähnlichsten Objekte, welche von X bewertet wurden
- Schätzung:

$$\hat{r}_{xi} = \frac{1}{k} \sum_{j \in N(i; x)} r_{xj}$$

- Gewichtet nach Ähnlichkeit:

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} s_{ij}}$$

s_{ij} = (Kosinus-)Ähnlichkeit zwischen Objekt i und j

Kollaboratives Filtern für Objekte

Nutzer

Filme


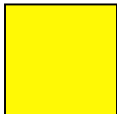
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Unbekannt

Bewertung

Kollaboratives Filtern für Objekte

		Nutzer												s_{1j}
		1	2	3	4	5	6	7	8	9	10	11	12	
Filme	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	3	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	6	1		3		3			2			4		<u>0.59</u>

 Unbekannt
  Bewertung

Kollaboratives Filtern für Objekte

		Nutzer												s_{1j}
		1	2	3	4	5	6	7	8	9	10	11	12	
Filme	1	1		3		2.6	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	3	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	6	1		3		3			2			4		<u>0.59</u>

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum_{j \in N(i;x)} s_{ij}} = \frac{0.41 \cdot 2 + 0.59 \cdot 3}{0.41 + 0.59} = 2.6$$

Vergleich: Vorteile

Kollaboratives Filtern	Inhaltsbasierte Analyse
Keine Auswahl von Merkmalen notwendig (insb. bei Bildern)	Benötigt keine Daten anderer Nutzer
Trennung verschiedener Interessen eines Nutzers möglich (KF für Objekte)	Empfehlungen für Nutzer mit einzigartigem Geschmack möglich
	Empfehlungen von neuen/unpopulären Objekten möglich
	Erklärung für Empfehlung möglich: Auflisten der Merkmale mit höchstem Gewicht

- Allgemeines Problem: Nutzenmatrix ist oft **spärlich besetzt**
- Lösung (siehe auch Übungsaufgabe):
 - Clusteranalyse auf Objekte und Zusammenfassen der Objekte eines Clusters (Mittelwert über Bewertungen)
 - Anschließend Clusteranalyse auf Nutzer und Zusammenfassen der Nutzer eines Clusters (Mittelwert über gemittelte Bewertungen)
 - Wiederholung des Prozesses bis Matrix ausreichend besetzt

Netflix Prize

- Trainingsdaten
 - 100 Million Bewertungen (1-5 Sterne)
 - ca. 480 000 zufällig ausgewählte Nutzer
 - ca. 17 770 Filme
 - Zeitraum: 2000-2005
- Testdaten
 - Menge R : die letzten Bewertungen der ausgewählten Nutzer (2.8 Millionen)
 - Evaluation über **Root Mean Squared Error (RMSE)**:
$$\sqrt{\frac{1}{|R|} \sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$
 - System von Netflix: CineMatch
 - RMSE von CineMatch: 0.9514 (durchschnittliche Fehler: ein Stern)
- Wettbewerb: \$1 Million für das erste Team dessen Algorithmus eine Verbesserung um 10% (RMSE von 0.8572 oder weniger) erreicht

Netflix Prize: RMSE



KF mit Bias

- KF für Objekte:

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Bessere Ergebnisse durch Berücksichtigung „globaler Effekte“:
 - Globaler Durchschnitt aller Bewertungen μ
 - Nutzerbias: $b_x = \frac{1}{n_x} \sum_i r_{xi} - \mu$ (n_x ist Anzahl der Bewertungen von x)
 - Filmbias: $b_i = \frac{1}{n_i} \sum_x r_{xi} - \mu$ (n_i ist Anzahl der Bewertungen für i)
 - Baseline-Schätzer für r_{xi} : $b_{xi} = \mu + b_x + b_i$

- **KF mit Bias:**

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

KF mit Bias und gelernten Gewichten

- Problem bei Verwendung von s_{ij} : willkürlich festgelegtes Maß (z.B. Kosinus-Ähnlichkeit)
- Anpassung über gewichtete Summe:

$$\widehat{r_{xi}} = b_{xi} + \sum_k w_{ik}(r_{xk} - b_{xk})$$

- Die Gewichte w_{ik} werden gelernt: **Minimierung des RMSE bei Anwendung auf Trainingsdaten**

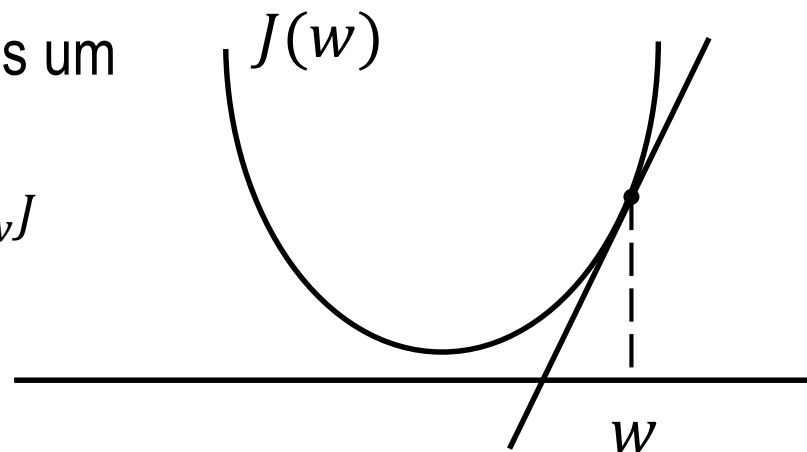
KF mit Bias und gelernten Gewichten

- Finde Gewichte $\mathbf{w} = (w_{ij})_{ij}$, die folgenden Ausdruck minimieren:

$$J(\mathbf{w}) = \sum_{x,i} \left(\left[b_{xi} + \sum_k w_{ik} (r_{xk} - b_{xk}) \right] - r_{xi} \right)^2$$

- Gradient Descent:** Einfacher Algorithmus um lokales Minimum zu finden

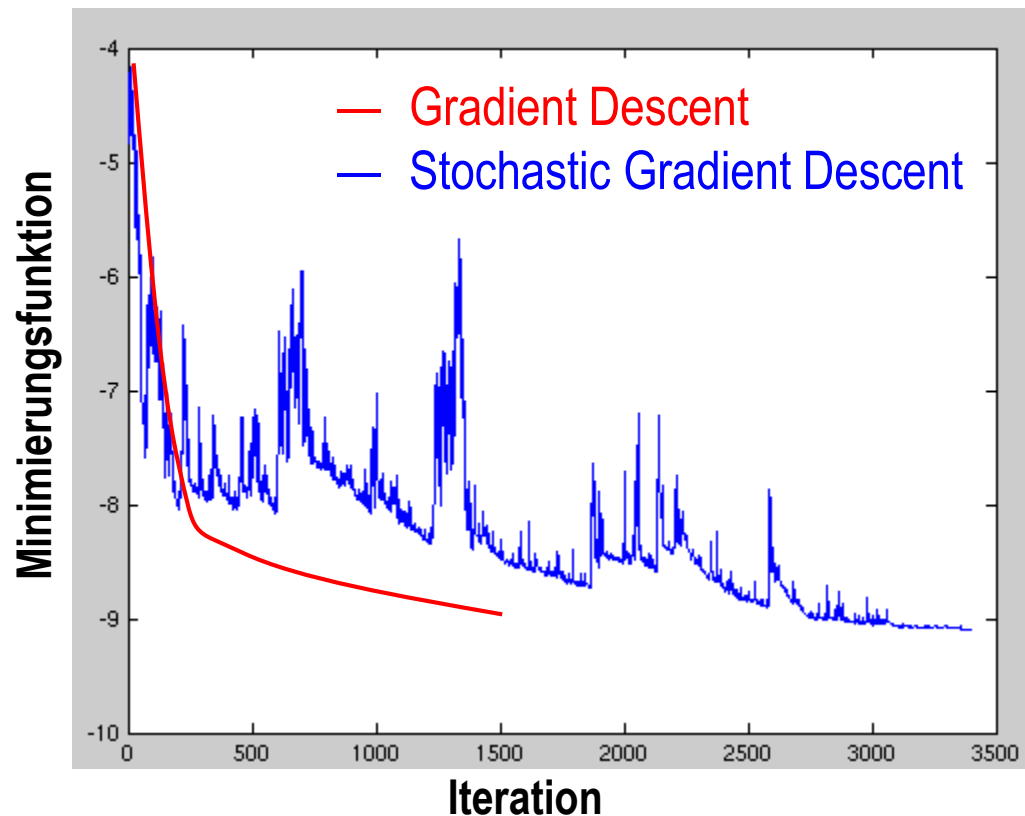
- Wiederhole bis Konvergenz: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J$
- Lernrate: η
- Gradient an der Stelle (i,j) :



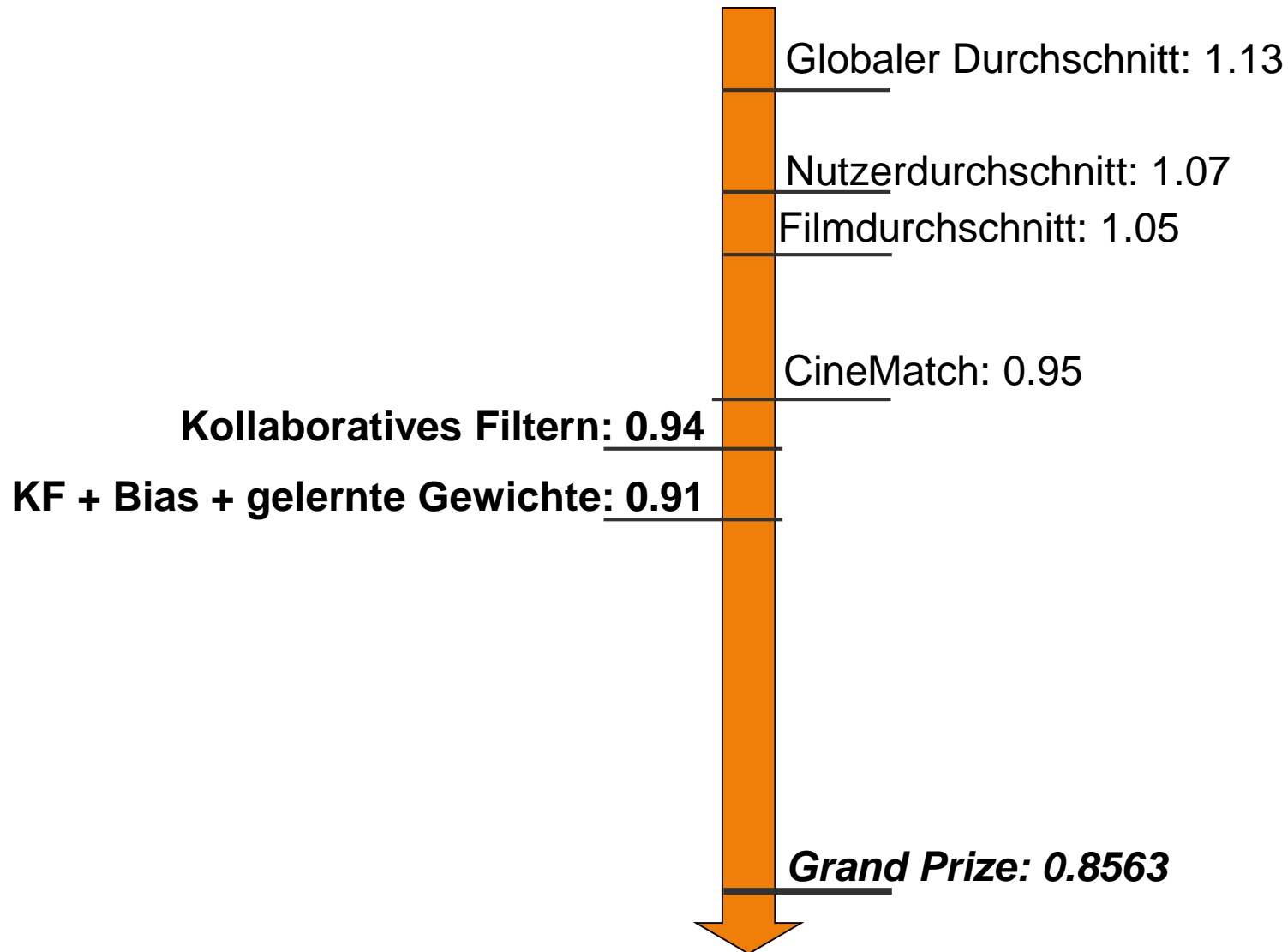
$$(\nabla_{\mathbf{w}} J)_{ij} = \frac{\partial J(\mathbf{w})}{\partial w_{ij}} = 2 \sum_x \left(\left[b_{xi} + \sum_k w_{ik} (r_{xk} - b_{xk}) \right] - r_{xi} \right) (r_{xj} - b_{xj})$$

Stochastic Gradient Descent

- Gradienten für die Elemente der Gewichtungsmatrix sind Summen über mehrere Datenpunkte: **Berechnungen kann sehr lange dauern**
- Schnellere Konvergenz wenn, für jede Iteration, nur ein (zufällig ausgewählter) Mini-Batch der Daten verwendet wird



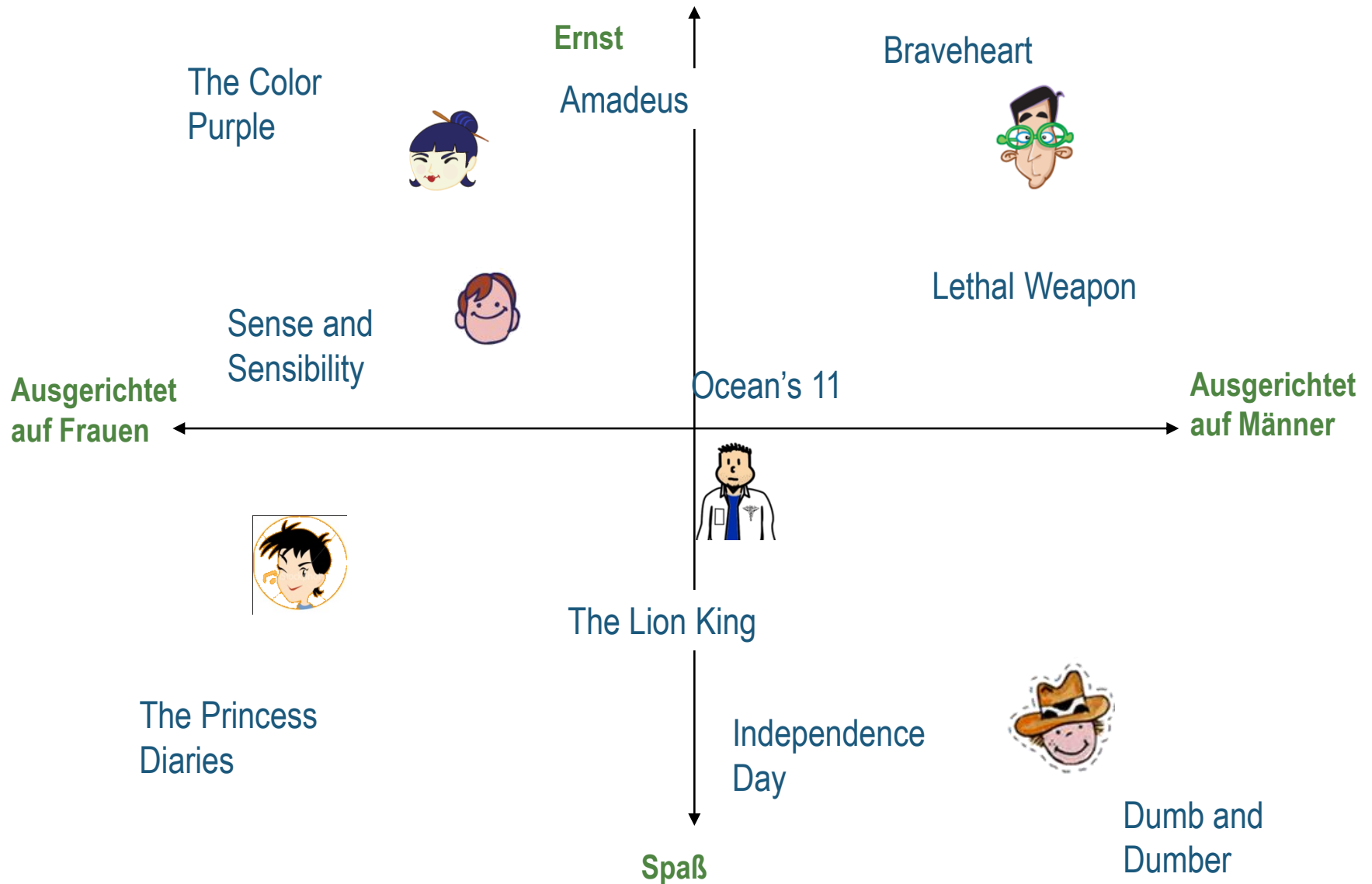
Netflix Prize: RMSE



Inhaltsverzeichnis

- Einführung
- Inhaltsbasierte Analyse
- Kollaboratives Filtern
- **Latentes Variablenmodell**

Latentes Variablenmodell



Latentes Variablenmodell

- Reduktion der Dimensionen auf wenige Faktoren
- Zerlegung: $R \approx Q \cdot P^T$ für nicht-leere Zellen von R

Nutzer

Filme

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

\approx

Faktoren

Filme

Q

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-.2
-1	.7	.3

Nutzer

Faktoren

P^T

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Latentes Variablenmodell

- Schätzung der Werte der leeren Felder über Faktoren

Nutzer

Filme

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

≈

$$\hat{r}_{xi} = \sum_f q_{if} \cdot p_{xf}$$

R

Faktoren

Filme

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

Q

Nutzer

Faktoren

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

P^T

Latentes Variablenmodell

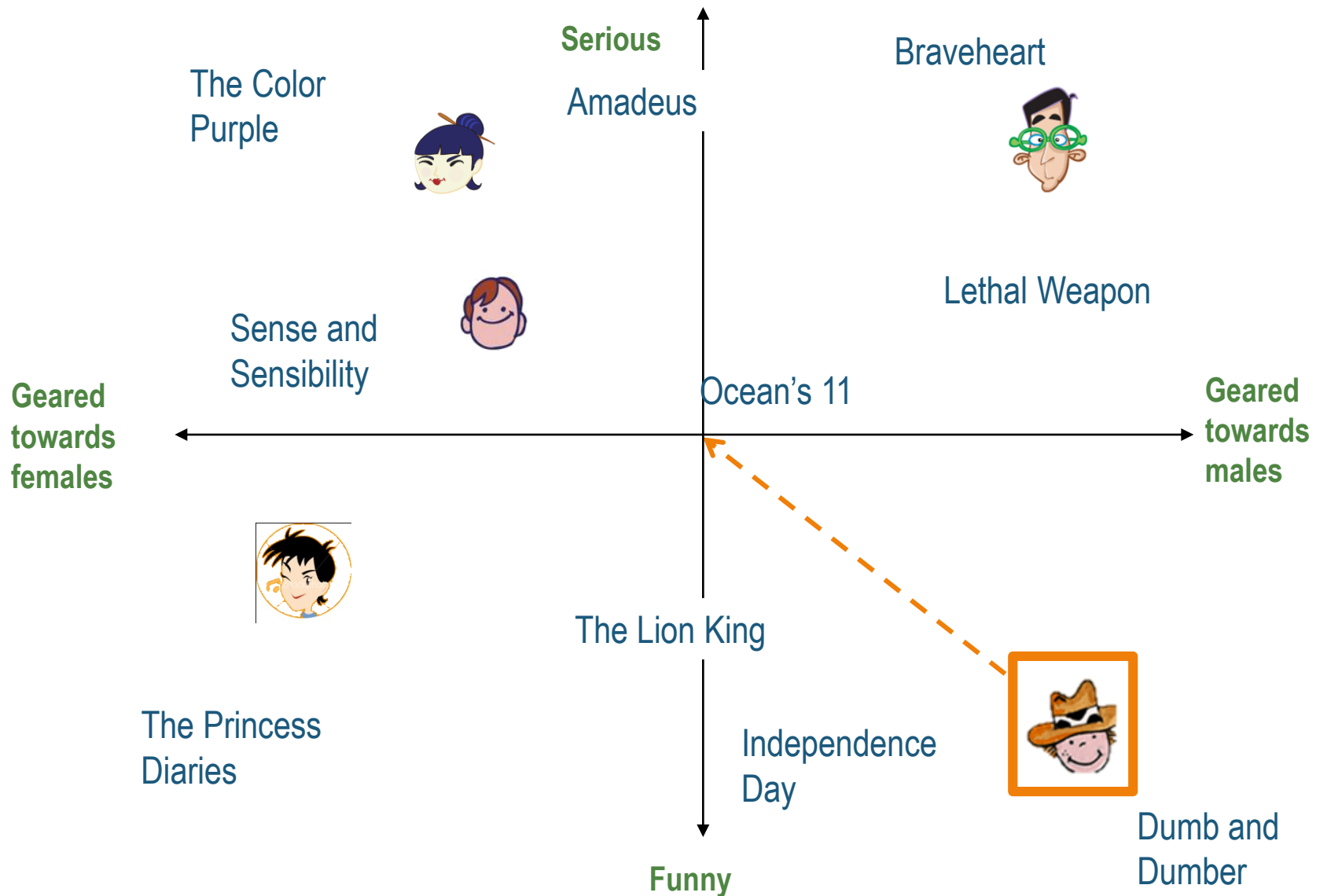
- Ziel: Finden zweier Matrizen Q und P, so dass folgender Ausdruck für alle vorhandenen Felder minimiert wird

$$\sum_{i,x} (r_{xi} - q_i \cdot p_x)^2$$

- Notation: q_i bzw. p_x bezeichnet die i-te Zeile von Q bzw. x-te Zeile von P
- Bestes Ergebnis für Trainingsdaten durch hohe Anzahl an Faktoren (Spalten von Q und P) → Gefahr: **Overfitting**
 - „Auswendig lernen“ der Daten (inkl. zufälliger Fehler)
 - Keine Generalisierung auf unbekannte Daten möglich → Hohe Fehlerrate bei Testdaten
- Regularisierung:

$$\min_{P,Q} \sum_{i,x} (r_{xi} - q_i \cdot p_x)^2 + \lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2$$

Effekt der Regularisierung



Berechnung

- Über (stochastic) Gradient Descent
- Ziel:

$$\min_{P,Q} \sum_{i,x} (r_{xi} - q_i \cdot p_x)^2 + \lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2$$

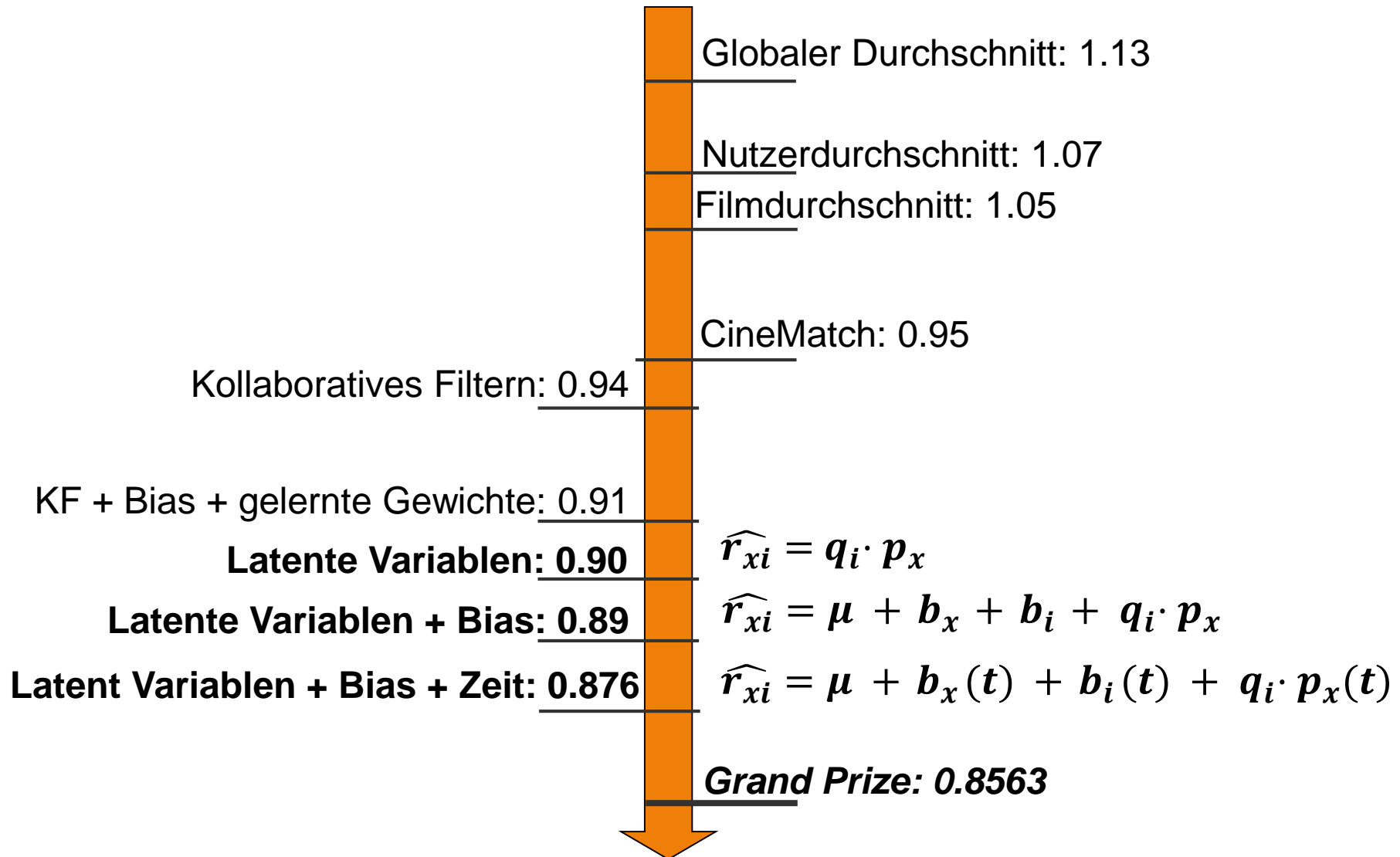
- Gradient Descent:
 - $P \leftarrow P - \eta \cdot \nabla P$, wobei $\nabla P = [\nabla p_{xf}]$ und

$$\nabla p_{xf} = \sum_i -2(r_{xi} - q_i \cdot p_x)q_{if} + 2\lambda_1 p_{xf}$$

- $Q \leftarrow Q - \eta \cdot \nabla Q$, wobei $\nabla Q = [\nabla q_{if}]$ und

$$\nabla q_{if} = \sum_x -2(r_{xi} - q_i \cdot p_x)p_{xf} + 2\lambda_2 q_{if}$$

Netflix Prize: Bewertungen



Netflix Prize

COMPLETED

[Home](#) [Rules](#) [Leaderboard](#) [Update](#) [Download](#)

Leaderboard

 Showing Test Score. [Click here to show quiz score](#)

 Display top leaders.

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
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos

13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50

Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell