Recommendation systems[1]

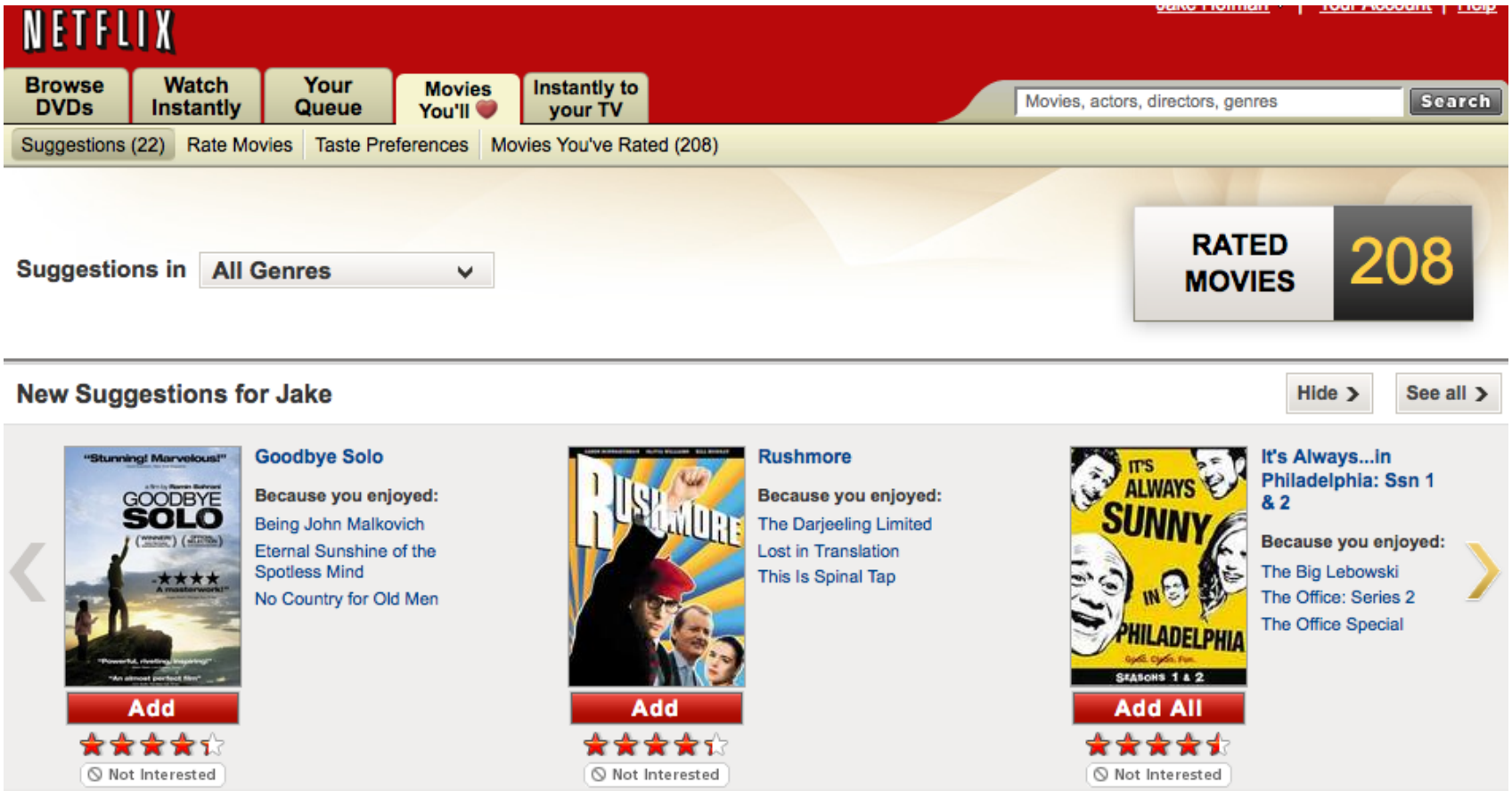
Jake Hofman

Yahoo! Research

April 4, 2011

[1] For large data sets

Personalized recommendations



Personalized recommendations

Not Interested

Our best guess for Jake: **5 stars**

Average of 4,275,920 ratings: **3.8 stars**

The Big Lebowski

1998 R 117 minutes

Slacker Jeff "The Dude" Lebowski (Jeff Bridges) gets involved in a gargantuan mess of events when he's mistaken for another man named Lebowski, whose wife has been kidnapped and is being held for ransom. All the while, Dude's friend, Walter (John Goodman), stirs the pot. Brothers Joel Coen and Ethan Coen write and direct this cult comedy classic that also stars Steve Buscemi, Philip Seymour Hoffman, Julianne Moore and John Turturro.

Cast: Jeff Bridges, John Goodman, Philip Seymour Hoffman, Steve Buscemi, Julianne Moore, Tara Reid, Peter Stormare, David Huddleston, Philip Moon, Mark Pellegrino, Flea, Torsten Voges, Jimmie Dale Gilmore, Jack Kehler, John Turturro, James G. Hoosier, Richard Gant, Christian Clemenson, David Thewlis, Peter Siragusa, Sam Elliott, Ben Gazzara, Jon Polito, Asia Carrera, Paris Themmen

Director: Joel Coen

Genres: Comedy, Cult Comedies, Universal Studios Home Entertainment, Blu-ray

This movie is: Quirky, Witty

Format: DVD and streaming (Blu-ray availability date unknown) (HD available)

Play

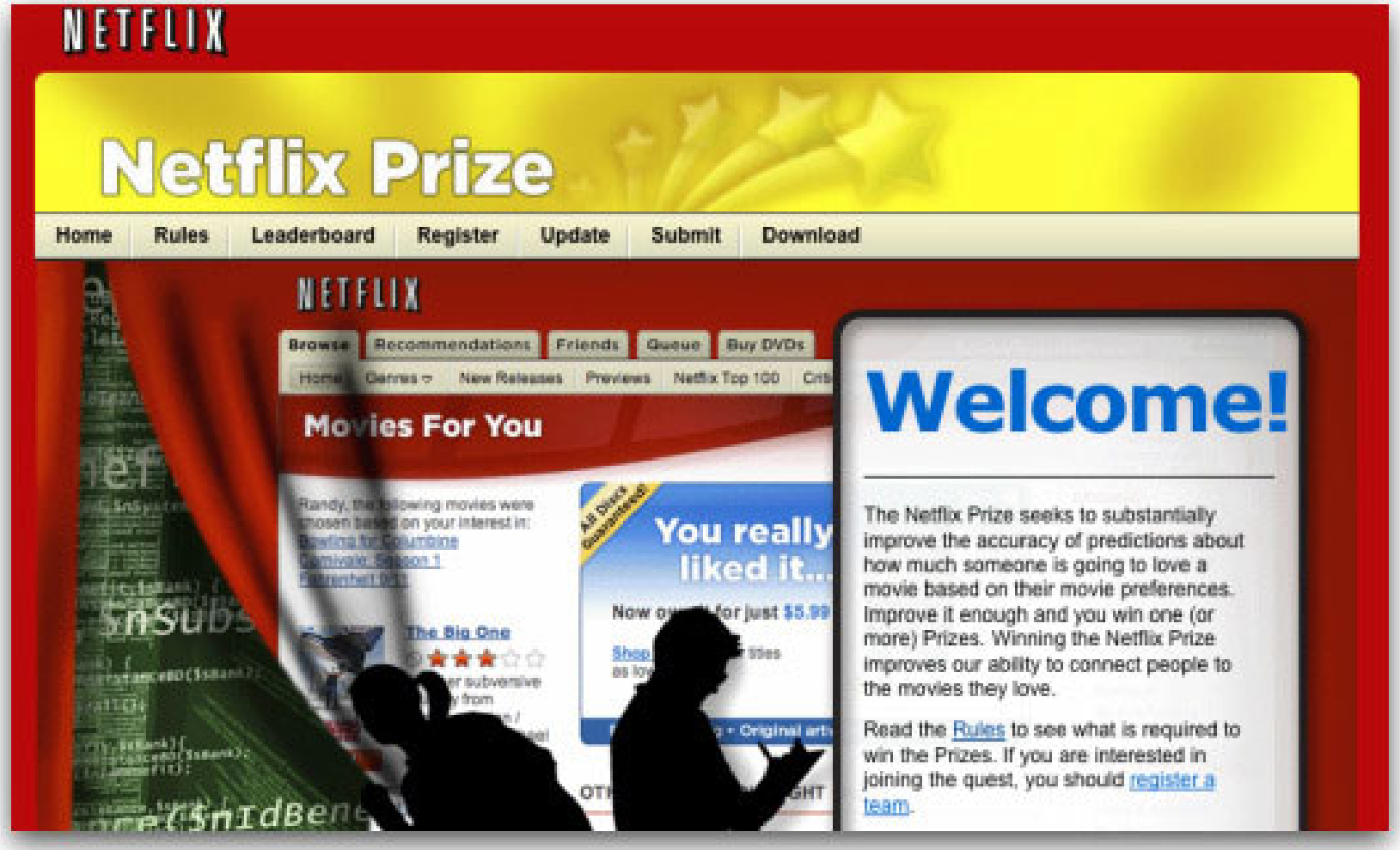
Add to Instant Queue

Add to DVD Queue

Play Trailer

Recommended based on your interest in: Fargo, O Brother, Where Art Thou? and No Country for Old Men

http://netflixprize.com

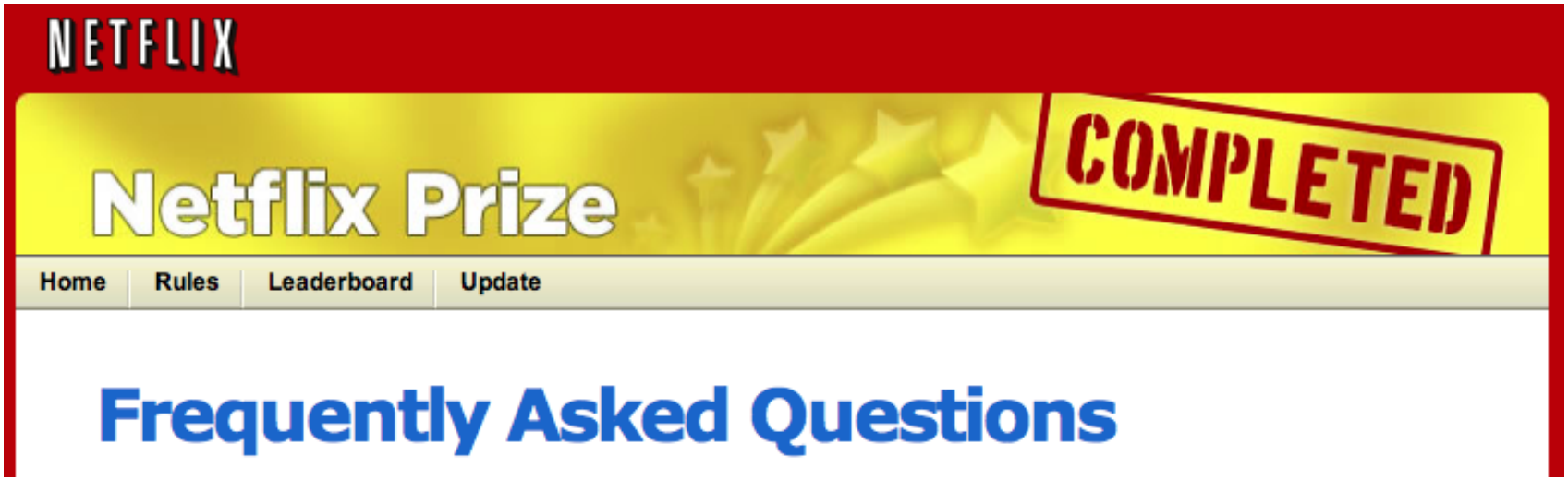


http://netflixprize.com/rules

Terms and Conditions in a Nutshell

- Contest begins October 2, 2006 and continues through at least October 2, 2011.
- Contest is open to anyone, anywhere (except certain countries listed below).
- You have to register to enter.
- Once you register and agree to these Rules, you'll have access to the Contest training data and qualifying test sets.
- To qualify for the $1,000,000 Grand Prize, the accuracy of your submitted predictions on the qualifying set must be at least 10% better than the accuracy Cinematch can achieve on the same training data set at the start of the Contest.
- To qualify for a year's $50,000 Progress Prize the accuracy of any of your submitted predictions that year must be less than or equal to the accuracy value established by the judges the preceding year.
- To win and take home either prize, your qualifying submissions must have the largest accuracy improvement verified by the Contest judges, you must share your method with (and non-exclusively license it to) Netflix, and you must describe to the world how you did it and why it works.

http://netflixprize.com/faq



How does Cinematch do it?

Straightforward statistical linear models with a lot of data conditioning. But a real-world system is much more than an algorithm, and Cinematch does a lot more than just optimize for RMSE. After all, we have a website to support. In production we have to worry about system scaling and performance, and we have additional sources to data we can use to guide our recommendations. But, as mentioned in the Rules and just to be perfectly clear, for the purposes of the Prize the RMSE values we report here do not use any of this extra data.

Netflix prize: results

The final standing of the Leaderboard at that time showed that two teams met the minimum requirements for the Grand Prize. "The Ensemble" with a 10.10% improvement over Cinematch on the Qualifying set (an RMSE of 0.8553), and "BellKor's Pragmatic Chaos" with a 10.09% improvement over Cinematch on the Qualifying set (an RMSE of 0.8554).[21] The Grand Prize winner was to be the one with the better performance on the Test set.

On September 18, 2009, Netflix announced team "BellKor's Pragmatic Chaos" as the prize winner, and the prize was awarded to the team in a ceremony on September 21, 2009.[22] "The Ensemble" team had in fact succeeded to match BellKor's result, but since BellKor submitted their results 20 minutes earlier, the rules award the prize to them.[23]

http://en.wikipedia.org/wiki/Netflix_Prize

Netflix prize: results

Add an asymmetric frequency feature $\mathbf{y}^{(3)}_{j,f_{ut}}$: **SBRAMF-UTB-UTF-MTF-ATF-MFF-AFF**

$$\widehat{r_{uit}} = \mu_i + \mu_u + \mu_{u,t} + \mu_{i,\text{bin}(t)} + \left(\mathbf{p}^{(1)}_i + \mathbf{p}^{(2)}_{i,\text{bin}(t)} + \mathbf{p}^{(3)}_{i,f_{ut}}\right)^T \left(\mathbf{q}^{(1)}_u + \mathbf{q}^{(2)}_{u,t} + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \left(\mathbf{y}^{(1)}_j + \mathbf{y}^{(2)}_{j,\text{bin}(t)} + \mathbf{y}^{(3)}_{j,f_{ut}}\right)\right) \tag{34}$$

Model extension (+)	epoch time	#epochs	probeRMSE, $k = 50$ features
SBRMF - SVD with biases	17[s]	69	0.9054
SBRAMF - asymmetric part	50[s]	30	0.8974
+UTB - user time bias	61[s]	50	0.8919
+UTF - user time feature	62[s]	38	0.8911
+MTF - movie time feature	74[s]	37	0.8908
+ATF - asymmetric time feature	74[s]	44	0.8905
+MFF - movie frequency feature	149[s]	46	0.8900
+AFF - asymmetric frequency feature	206[s]	45	0.8886 (0.8846 with $k = 1000$)

See [TJB09] and [Kor09] for more gory details.

Recommendation systems

High-level approaches:

- Content-based methods
  (e.g., $w_{\text{genre: thrillers}} = +2.3$, $w_{\text{director: coen brothers}} = +1.7$)
- Collaborative methods
  (e.g., "Users who liked this also liked")

Netflix prize: data

(userid, movieid, rating, date)

6	15124	5	9	3	2004
6	4506	3	9	3	2004
6	5069	3	9	3	2004
6	5924	5	9	3	2004
6	7040	3	9	3	2004
6	2095	4	9	3	2004
6	8301	3	9	3	2004
6	17295	5	9	3	2004
6	17195	3	9	3	2004
6	13007	4	9	3	2004
6	16912	3	9	3	2004
6	14240	5	9	3	2004
6	705	3	9	3	2004

Netflix prize: data

(movieid, year, title)

1	60034316	2003	Dinosaur Planet
2	70014695	2004	Isle of Man TT 2004 Review
3	60010225	1997	Character
4	60033124	1994	Paula Abdul's Get Up & Dance
5	70017841	2004	The Rise and Fall of ECW
6	60031092	1997	Sick
7	19826698	1992	8 Man
8	70011191	2004	What the #$*! Do We Know!?
9	70002378	1991	Class of Nuke 'Em High 2
10	60024682	2001	Fighter

Recommendation systems

High-level approaches:

- Content-based methods
  (e.g., $w_{\text{genre: thrillers}} = +2.3$, $w_{\text{director: coen brothers}} = +1.7$)
- Collaborative methods
  (e.g., "Users who liked this also liked")

Collaborative filtering

Memory-based
(e.g., k-nearest neighbors)

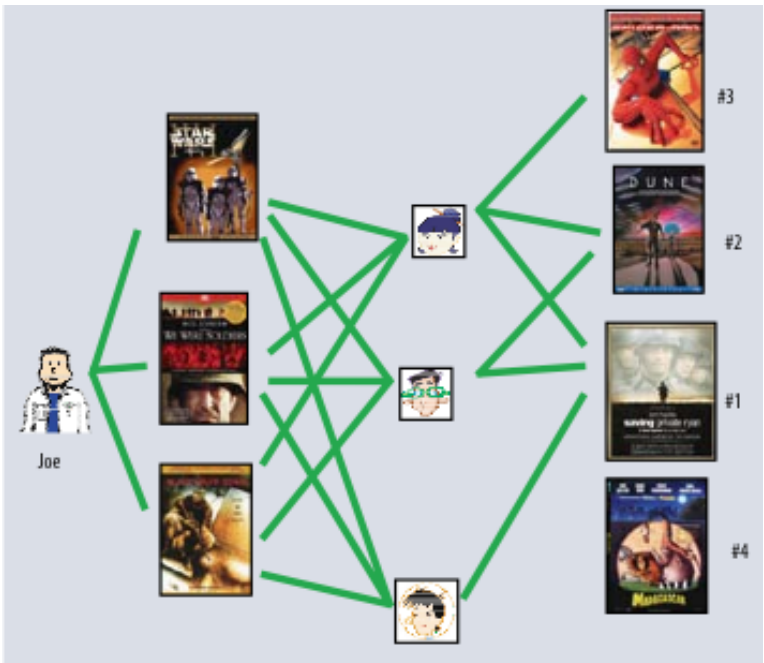


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

Model-based
(e.g., matrix factorization)

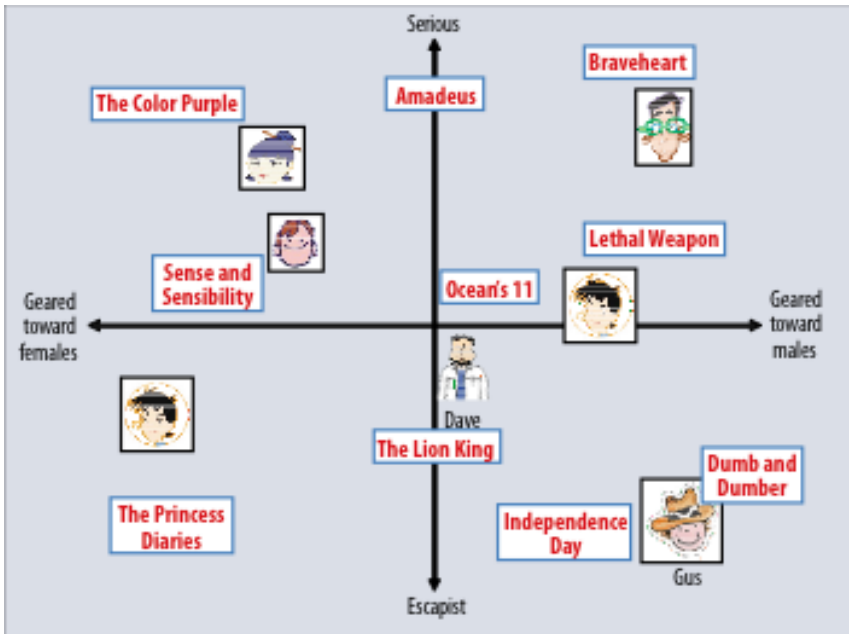


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

`http://research.yahoo.com/pub/2859`

Problem statement

- Given a set of past ratings R_{ui} that user u gave item i
  - Users may explicitly assign ratings, e.g., $R_{ui} \in [1, 5]$ is number of stars for movie rating
  - Or we may infer implicit ratings from user actions, e.g. $R_{ui} = 1$ if u purchased i; otherwise $R_{ui} =$?

Problem statement

- Given a set of past ratings R_{ui} that user u gave item i
  - Users may explicitly assign ratings, e.g., $R_{ui} \in [1, 5]$ is number of stars for movie rating
  - Or we may infer implicit ratings from user actions, e.g. $R_{ui} = 1$ if u purchased i; otherwise $R_{ui} =$?
- Make recommendations of several forms
  - Predict unseen item ratings for a particular user
  - Suggest items for a particular user
  - Suggest items similar to a particular item
  - ...

Problem statement

- Given a set of past ratings R_{ui} that user u gave item i
  - Users may explicitly assign ratings, e.g., $R_{ui} \in [1, 5]$ is number of stars for movie rating
  - Or we may infer implicit ratings from user actions, e.g. $R_{ui} = 1$ if u purchased i; otherwise $R_{ui} =$?
- Make recommendations of several forms
  - Predict unseen item ratings for a particular user
  - Suggest items for a particular user
  - Suggest items similar to a particular item
  - ...
- Compare to natural baselines
  - Guess global average for item ratings
  - Suggest globally popular items

k-nearest neighbors

Key intuition:

Take a local popularity vote amongst "similar" users

k-nearest neighbors
User similarity

Quantify similarity as a function of users' past ratings, e.g.

- Fraction of items u and v have in common

$$S_{uv} = \frac{|R_u \cap R_v|}{|R_u \cup R_v|} = \frac{\sum_i R_{ui} R_{vi}}{\sum_i (R_{ui} + R_{vi} - R_{ui} R_{vi})} \quad (1)$$

Retain top-k most similar neighbors v for each user u

k-nearest neighbors
User similarity

Quantify similarity as a function of users' past ratings, e.g.

- Angle between rating vectors

$$S_{uv} = \frac{R_u \cdot R_v}{|R_u|\,|R_v|} = \frac{\sum_i R_{ui} R_{vi}}{\sqrt{\sum_i R_{ui}^2 \sum_j R_{vj}^2}} \tag{1}$$

Retain top-k most similar neighbors v for each user u

k-nearest neighbors
Predicted ratings

Predict unseen ratings $\hat{R}_{ui}$ as a weighted vote over u's neighbors' ratings for item i

$$\hat{R}_{ui} = \frac{\sum_v R_{vi} S_{uv}}{\sum_v S_{uv}} \tag{2}$$

k-nearest neighbors
Practical notes

We expect most users have nothing in common, so calculate similarities as:

for each item i:
  for all pairs of users u, v that have rated i:
    calculate S_{uv} (if not already calculated)

k-nearest neighbors
Practical notes

Alternatively, we can make recommendations using an item-based approach [LSY03]:

- Compute similarities S_{ij} between all pairs of items
- Predict ratings with a weighted vote $\hat{R}_{ui} = \sum_j R_{uj} S_{ij} / \sum_j S_{ij}$

k-nearest neighbors
Practical notes

Several (relatively) simple ways to scale:

- Sample a subset of ratings for each user (by, e.g., recency)
- Use MinHash to cluster users [DDGR07]
- Distribute calculations with MapReduce

Matrix factorization

Key intuition:

Model item attributes as belonging to a set of unobserved "topics and user preferences across these "topics"

Matrix factorization
Linear model

Start with a simple linear model:

$$\hat{R}_{ui} = \underbrace{b_0}_{\text{global average}} + \underbrace{b_u}_{\text{user bias}} + \underbrace{b_i}_{\text{item bias}} \tag{3}$$

Matrix factorization
Linear model

For example, we might predict that a harsh critic would score a popular movie as

$$\hat{R}_{ui} = \underbrace{3.6}_{\text{global average}} + \underbrace{-0.5}_{\text{user bias}} + \underbrace{0.8}_{\text{item bias}} \quad (3)$$

$$= 3.9 \quad (4)$$

Matrix factorization
Low-rank approximation

Add an interaction term:

$$\hat{R}_{ui} = \underbrace{b_0}_{\text{global average}} + \underbrace{b_u}_{\text{user bias}} + \underbrace{b_i}_{\text{item bias}} + \underbrace{W_{ui}}_{\text{user-item interaction}} \quad (5)$$

where $W_{ui} = P_u \cdot Q_i = \sum_k P_{uk} Q_{ik}$

- P_{uk} is user u's preference for topic k
- Q_{ik} is item i's association with topic k

Matrix factorization
Loss function

Measure quality of model fit with squared-loss:

$$\mathcal{L} = \sum_{(u,i)} \left(\hat{R}_{ui} - R_{ui} \right)^2 \tag{6}$$

$$= \sum_{(u,i)} \left([PQ]_{ui} - R_{ui} \right)^2 \tag{7}$$

Matrix factorization
Optimization

The loss is non-convex in (P, Q), so no global minimum exists

Instead we can optimize $\mathcal{L}$ iteratively, e.g.:

- Alternating least squares: update entire matrix P, holding Q fixed, and vice-versa
- Stochastic gradient descent: update individual rows P_u and Q_i for each observed R_{ui}

Matrix factorization
Alternating least squares

$\mathcal{L}$ is convex in P (Q fixed), so take partials and solve linear system for updates:

$$0 = \frac{\partial \mathcal{L}}{\partial P} \quad \rightarrow \quad \hat{P} = RQ\left[Q^T Q\right]^{-1} \tag{8}$$

$$0 = \frac{\partial \mathcal{L}}{\partial Q} \quad \rightarrow \quad \hat{Q} = R^T P\left[P^T P\right]^{-1} \tag{9}$$

Matrix factorization
Stochastic gradient descent

Alternatively, we can avoid inverting matrices by taking steps in the direction of the negative gradient for each observed rating:

$$P_u \leftarrow P_u - \eta \frac{\partial \mathcal{L}}{\partial P_u} = P_u + \left(R_{ui} - \hat{R}_{ui}\right) Q_i \tag{10}$$

$$Q_i \leftarrow Q_i - \eta \frac{\partial \mathcal{L}}{\partial Q_i} = Q_i + \left(R_{ui} - \hat{R}_{ui}\right) P_u \tag{11}$$

for some step-size η

Matrix factorization
Practical notes

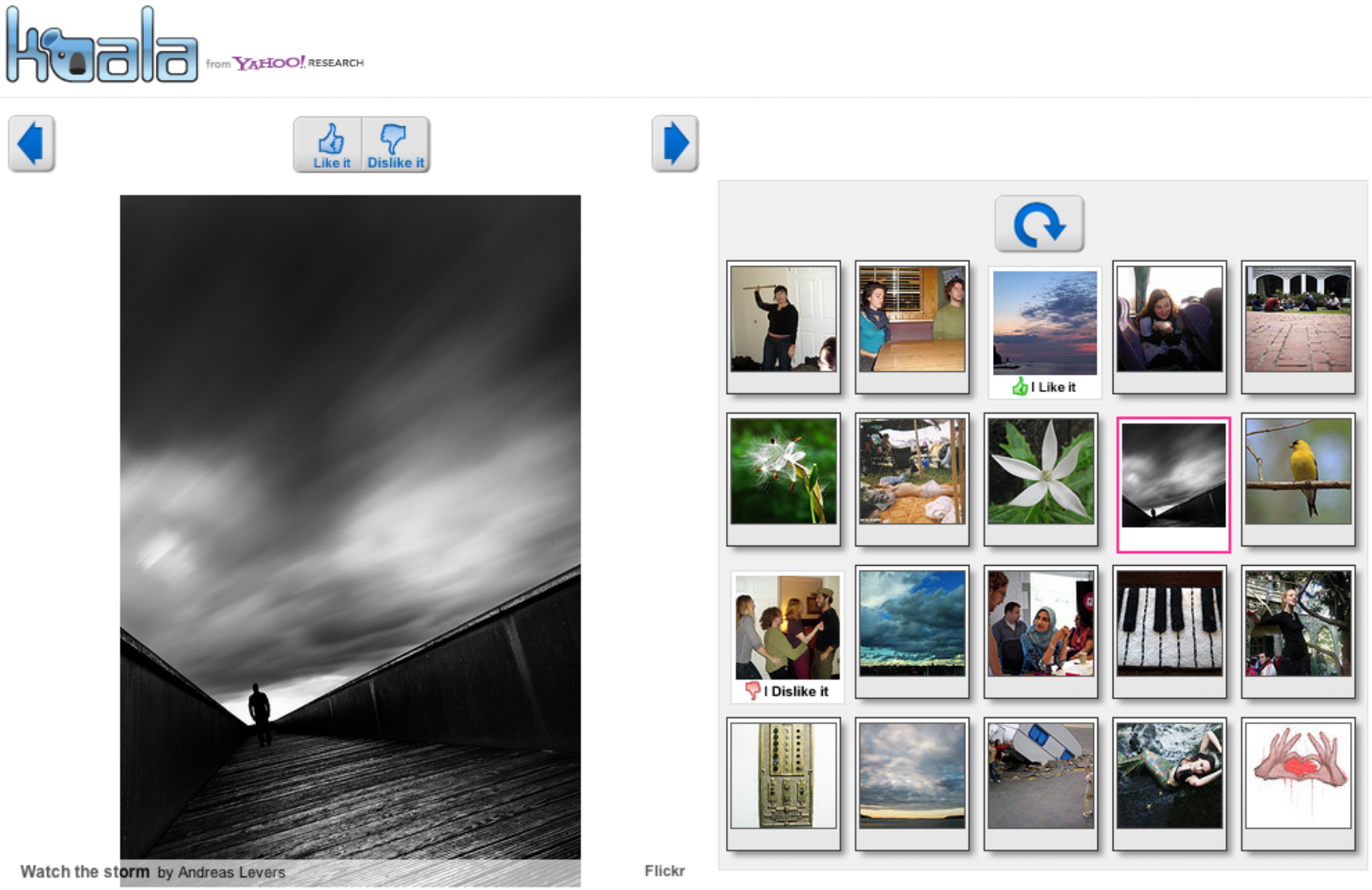
Several ways to scale:

- Distribute matrix operations with MapReduce [GHNS11]
- Parallelize stochastic gradient descent [ZWSL10]
- Expectation-maximization for pLSI with MapReduce [DDGR07]

Datasets

- Movielens
  `http://www.grouplens.org/node/12`
- Reddit
  `http://bit.ly/redditdata`
- CU "million songs"
  `http://labrosa.ee.columbia.edu/millionsong/`
- Yahoo Music KDDcup
  `http://kddcup.yahoo.com/`
- AudioScrobbler
  `http://bit.ly/audioscrobblerdata`
- Delicious
  `http://bit.ly/deliciousdata`
- ...

Photo recommendations



http://koala.sandbox.yahoo.com

Thanks. Questions?[2]

[2] hofman@yahoo-inc.com

References I


RM Bell, Y Koren, and C Volinsky.
The bellkor 2008 solution to the netflix prize.
Statistics Research Department at AT&T Research, 2008.


AS Das, M Datar, A Garg, and S Rajaram.
Google news personalization: scalable online collaborative filtering.
page 280, 2007.


R Gemulla, PJ Haas, E Nijkamp, and Y Sismanis.
Large-scale matrix factorization with distributed stochastic gradient descent.
2011.


JL Herlocker, JA Konstan, LG Terveen, and JT Riedl.
Evaluating collaborative filtering recommender systems.
ACM Transactions on Information Systems, 22(1):5–53, 2004.

References II


T Hofmann.
Latent semantic models for collaborative filtering.
ACM Transactions on Information Systems, 22(1):89–115, 2004.


Yehuda Koren, Robert Bell, and Chris Volinsky.
Matrix factorization techniques for recommender systems.
Computer, 42(8):30–37, 2009.


Yehuda Koren.
The bellkor solution to the netflix grand prize.
pages 1–10, Aug 2009.


G Linden, B Smith, and J York.
Amazon. com recommendations: Item-to-item collaborative filtering.
IEEE Internet computing, 7(1):76–80, 2003.

References III


A Toscher, M Jahrer, and RM Bell.
The bigchaos solution to the netflix grand prize.
2009.


M. Zinkevich, M. Weimer, A. Smola, and L. Li.
Parallelized stochastic gradient descent.
In Neural Information Processing Systems (NIPS), 2010.