

Recommending Products

Emily Fox & Carlos Guestrin
Machine Learning Specialization
University of Washington

Where we see recommender systems

Personalization is transforming our experience of the world

Information overload



100 Hours a Minute

What do I care about?



Browsing is “history”

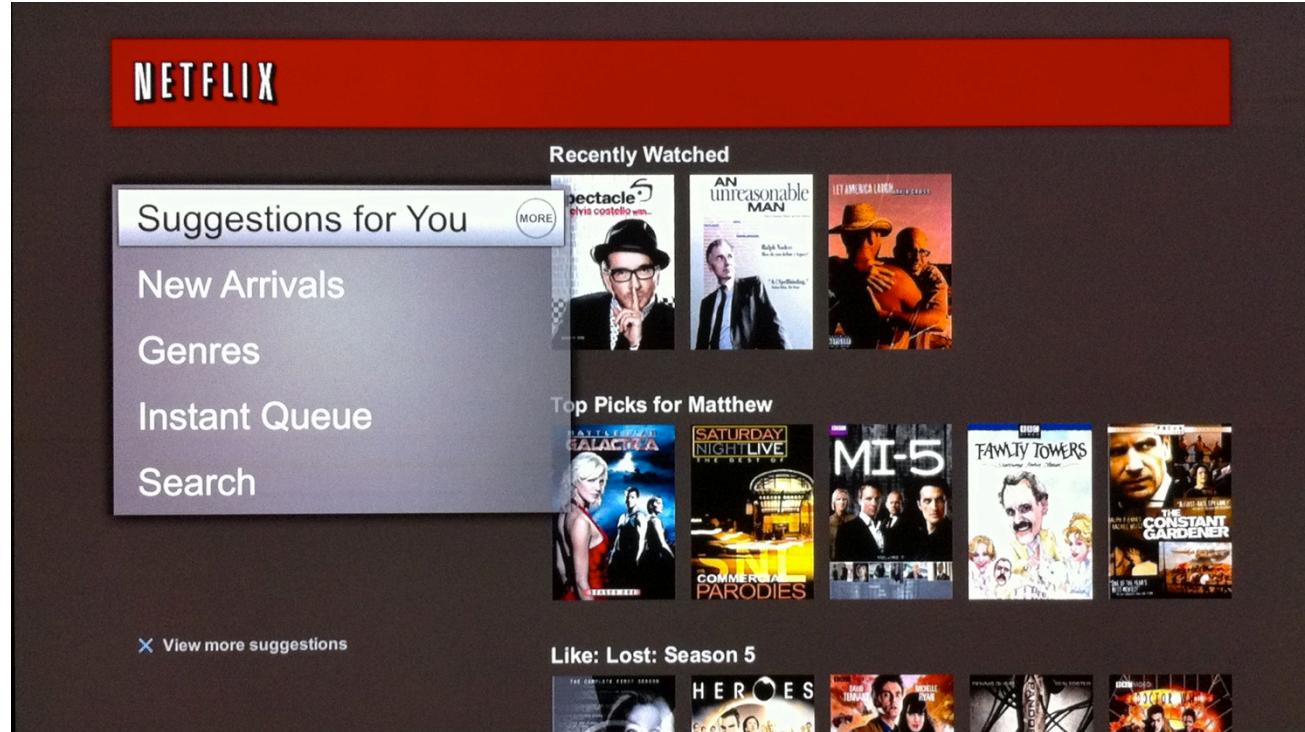
- Need new ways to discover content

Personalization: Connects *users & items*

viewers

videos

Movie recommendations



Connect users with movies
they may want to watch

Product recommendations

amazon.com®

Help | Close window

Recommended for You

 **High Performance Web Sites:
Essential Knowledge for
Front-End Engineers**
by Steve Souders (Author)
Our Price: \$19.79
Used & new from \$16.24

Add to Cart Add to Wish List

Because you purchased...

Programming Collective Intelligence: Building Smart Web 2.0 Applications (Paperback)
by Toby Segaran (Author)

Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to see all recommendations

 **Even Faster Web Sites:**
Perform... (Paperback) by Steve Souders
★★★★★ (7) \$23.10
Fix this recommendation

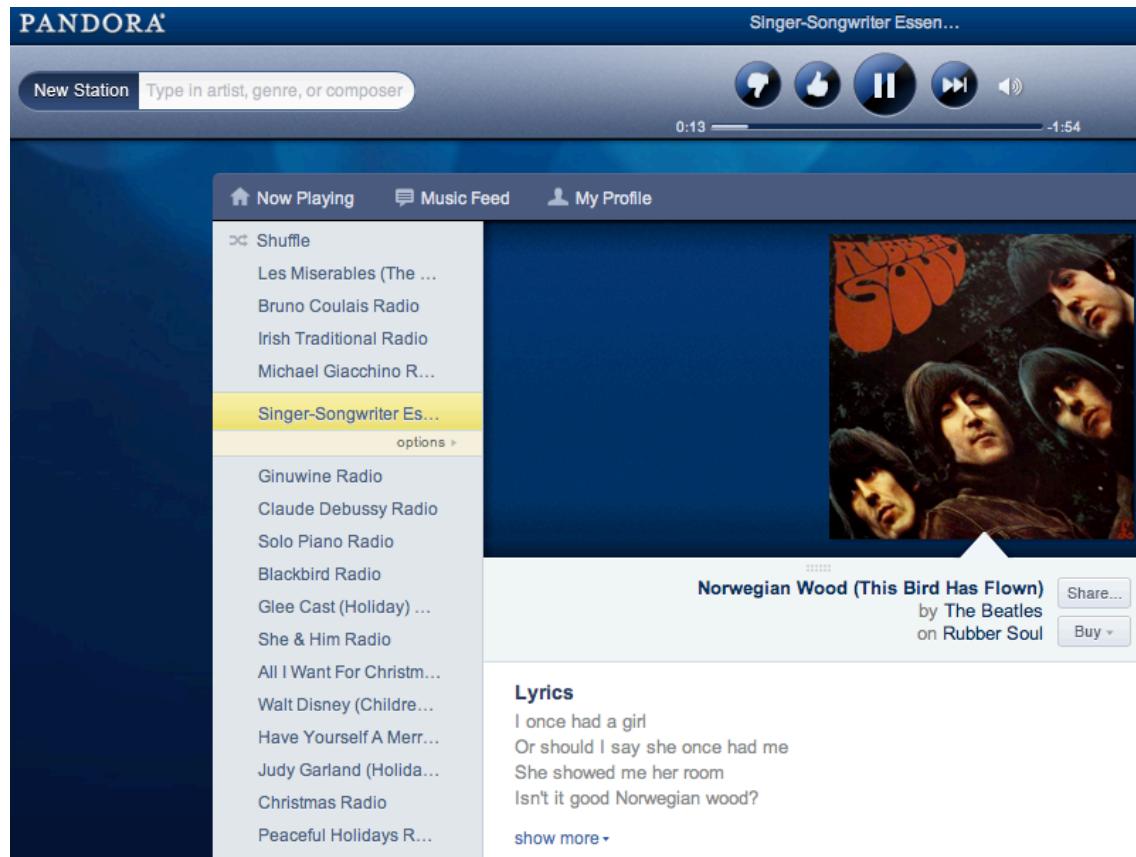
 **Simply JavaScript** (Paperback)
by Kevin Yank
★★★★★ (19) \$26.37
Fix this recommendation

 **The Art & Science of Java** (Paperback)
by Robert Sedgewick, Philippe Flajolet
★★★★★ (3) \$26.37
Fix this recommendation

Any Category Algorithms Boxed Sets Business & Culture Java
Art Networking Networks, Protocols & APIs New
SQL

Recommendations combine
global & session interests

Music recommendations



Recommendations form
coherent & diverse sequence

Friend recommendations



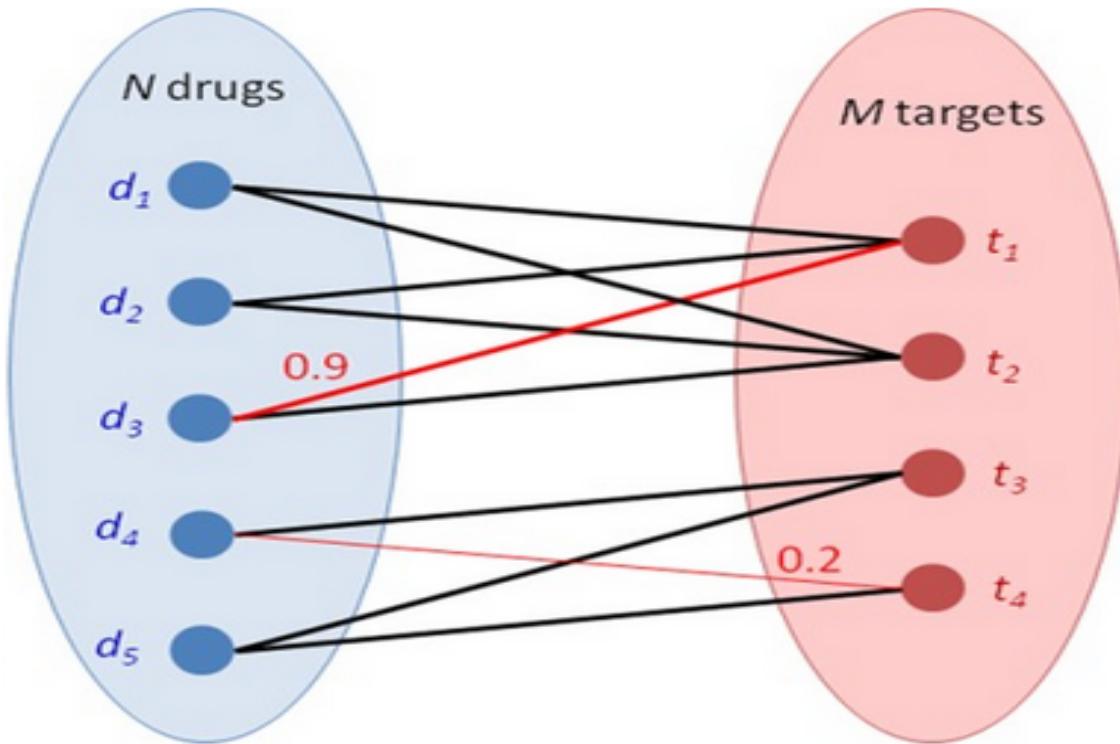
facebook



Users and “items”
are of the same “type”

Drug-target interactions

Cobanoglu et al. '13



What drug should we
“repurpose” for some disease?

Building a recommender system

Solution 0: Popularity

Simplest approach: Popularity

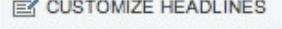
- What are people viewing now?
 - Rank by global popularity
- Limitation:
 - No personalization

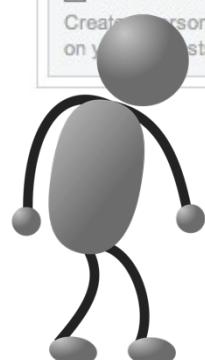
MOST POPULAR

E-MAILED BLOGGED SEARCHED

1. Really?: The Claim: Lack of Sleep Increases the Risk of Catching a Cold.
2. Magazine Preview: Coming Out in Middle School
3. Yes, We Speak Cupcake
4. Gossamer Silk, From Spiders Spun
5. Tie to Pets Has Germ Jumping to and Fro
6. Maureen Dowd: Where the Wild Thing Is
7. Maureen Dowd: Blue Is the New Black
8. The Holy Grail of the Unconscious
9. For Opening Night at the Metropolitan, a New Sound: Booing
10. Economic Scene: Medical Malpractice System Breeds More Waste

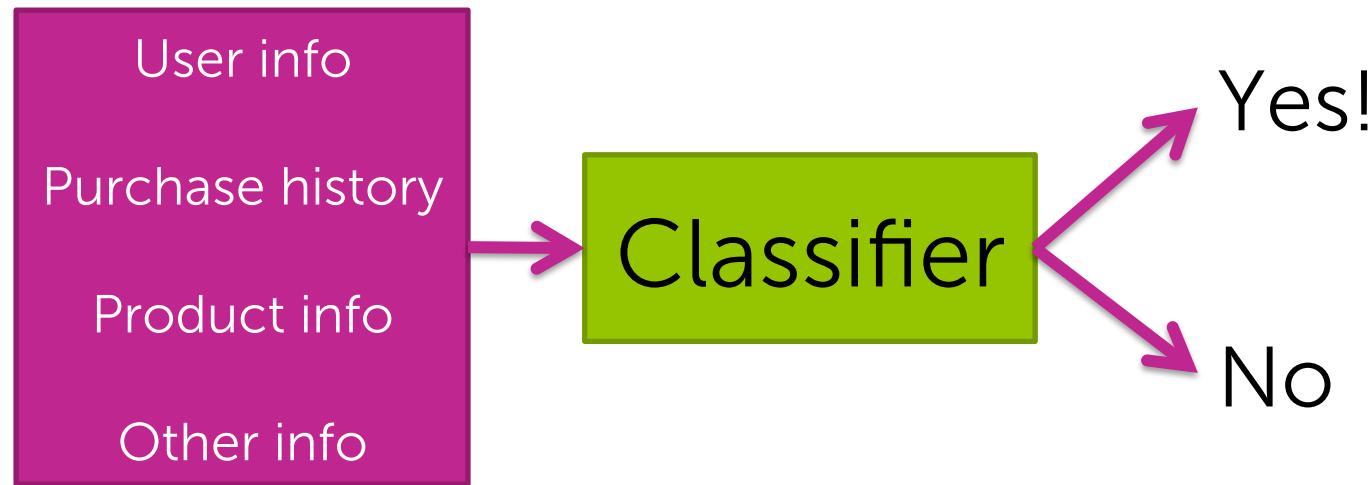
[Go to Complete List »](#)

 CUSTOMIZE HEADLINES
Create a personalized list of headlines based on your interests. [Get Started »](#)



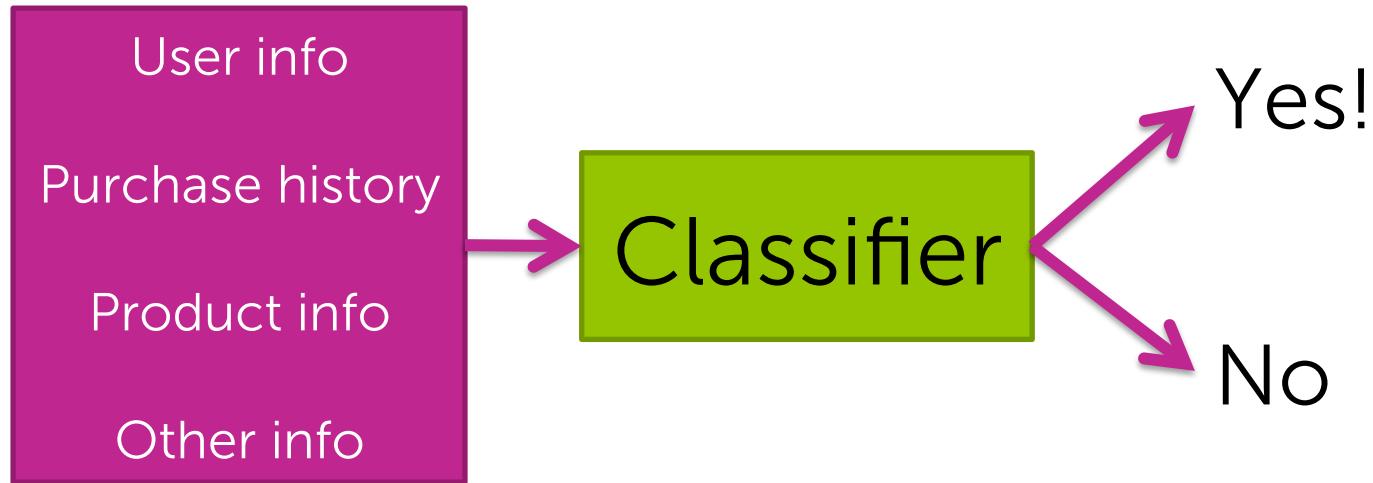
Solution 1: Classification model

What's the probability I'll buy this product?



- **Pros:**
 - Personalized:
Considers user info & purchase history
 - Features can capture context:
Time of the day, what I just saw,...
 - Even handles limited user history: Age of user, ...

Limitations of classification approach



- Features may not be available
- Often doesn't perform as well as **collaborative filtering** methods (next)

**Solution 2: People who bought this
also bought...**

Co-occurrence matrix

- People who bought *diapers* also bought *baby wipes*
- **Matrix C:**
store # users who bought both items $i \& j$
 - (# items x # items) matrix
 - **Symmetric:** # purchasing $i \& j$ same as # for $j \& i$ ($C_{ij} = C_{ji}$)

Making recommendations using co-occurrences

- User purchased *diapers*



1. Look at *diapers* row of matrix
2. Recommend other items with largest counts
 - *baby wipes, milk, baby food,...*

Co-occurrence matrix must be normalized

- What if there are very popular items?
 - Popular baby item:
Pampers Swaddlers diapers
 - For any baby item (e.g., $i = \text{Sophie giraffe}$)
large count C_{ij} for $j = \text{Pampers Swaddlers}$
- Result:
 - Drowns out other effects
 - Recommend based on popularity



Normalize co-occurrences: Similarity matrix

- Jaccard similarity: normalizes by popularity
 - Who purchased i and j divided by who purchased i or j
- Many other similarity metrics possible, e.g., cosine similarity

Limitations

- Only current page matters, no history
 - Recommend similar items to the one you bought
- What if you purchased many items?
 - Want recommendations based on purchase history

(Weighted) Average of purchased items

- User  bought items $\{\text{diapers}, \text{milk}\}$
 - Compute user-specific score for each item j in inventory by combining similarities:

$$\begin{aligned} \text{Score}(\text{User}, \text{baby wipes}) &= \\ &\frac{1}{2} (S_{\text{baby wipes}, \text{diapers}} + S_{\text{baby wipes}, \text{milk}}) \end{aligned}$$

- Could also weight recent purchases more
- Sort $\text{Score}(\text{User}; j)$ and find item j with highest similarity

Limitations

- Does **not** utilize:
 - context (e.g., time of day)
 - user features (e.g., age)
 - product features (e.g., baby vs. electronics)
- Cold start problem
 - What if a new user or product arrives?

Solution 3: Discovering hidden structure by matrix factorization

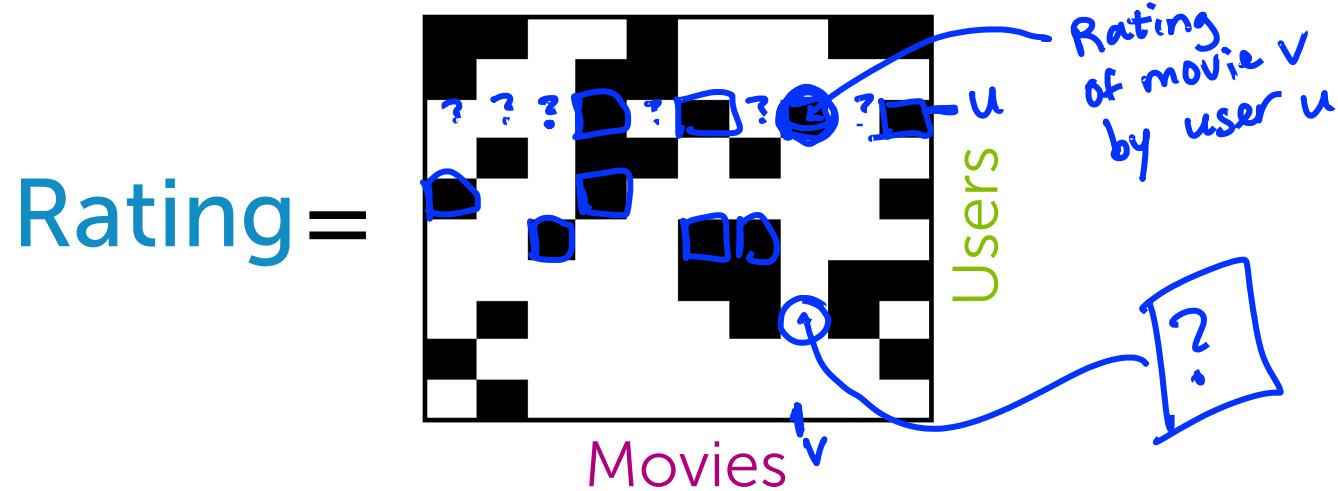
Movie recommendation

- Users watch movies and rate them

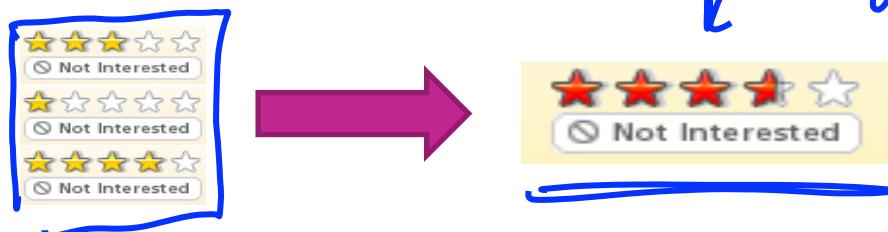
| User | Movie | Rating |
|--------|---------|--------|
| User 1 | Movie 1 | ★★★★★ |
| User 1 | Movie 2 | ★★★★★ |
| User 1 | Movie 3 | ★★★★★ |
| User 1 | Movie 4 | ★★★★★ |
| User 1 | Movie 5 | ★★★★★ |
| User 2 | Movie 1 | ★★★★★ |
| User 2 | Movie 2 | ★★★★★ |
| User 2 | Movie 3 | ★★★★★ |
| User 2 | Movie 4 | ★★★★★ |
| User 2 | Movie 5 | ★★★★★ |
| User 3 | Movie 1 | ★★★★★ |
| User 3 | Movie 2 | ★★★★★ |
| User 3 | Movie 3 | ★★★★★ |
| User 3 | Movie 4 | ★★★★★ |
| User 3 | Movie 5 | ★★★★★ |
| User 4 | Movie 1 | ★★★★★ |
| User 4 | Movie 2 | ★★★★★ |
| User 4 | Movie 3 | ★★★★★ |
| User 4 | Movie 4 | ★★★★★ |
| User 4 | Movie 5 | ★★★★★ |
| User 5 | Movie 1 | ★★★★★ |
| User 5 | Movie 2 | ★★★★★ |
| User 5 | Movie 3 | ★★★★★ |
| User 5 | Movie 4 | ★★★★★ |
| User 5 | Movie 5 | ★★★★★ |
| User 6 | Movie 1 | ★★★★★ |
| User 6 | Movie 2 | ★★★★★ |
| User 6 | Movie 3 | ★★★★★ |
| User 6 | Movie 4 | ★★★★★ |
| User 6 | Movie 5 | ★★★★★ |
| User 7 | Movie 1 | ★★★★★ |
| User 7 | Movie 2 | ★★★★★ |
| User 7 | Movie 3 | ★★★★★ |
| User 7 | Movie 4 | ★★★★★ |
| User 7 | Movie 5 | ★★★★★ |
| User 8 | Movie 1 | ★★★★★ |
| User 8 | Movie 2 | ★★★★★ |
| User 8 | Movie 3 | ★★★★★ |
| User 8 | Movie 4 | ★★★★★ |
| User 8 | Movie 5 | ★★★★★ |
| User 9 | Movie 1 | ★★★★★ |
| User 9 | Movie 2 | ★★★★★ |
| User 9 | Movie 3 | ★★★★★ |
| User 9 | Movie 4 | ★★★★★ |
| User 9 | Movie 5 | ★★★★★ |

Each user only watches a few of the available movies

Matrix completion problem



- **Data:** Users score some movies
 - $\text{Rating}(u, v)$ known for black cells
 - $\text{Rating}(u, v)$ unknown for white cells
- **Goal:** Filling missing data?



Suppose we had d topics for each user and movie

- Describe movie v  with topics R_v
 - How much is it **action, romance, drama,...**

$$R_v = [0.3 \quad 0.01 \quad 1.5 \quad \dots]$$

- Describe user u  with topics L_u

How much she likes **action, romance, drama,...**

$$L_u = [2.5 \quad 0 \quad 0.8 \quad \dots]$$

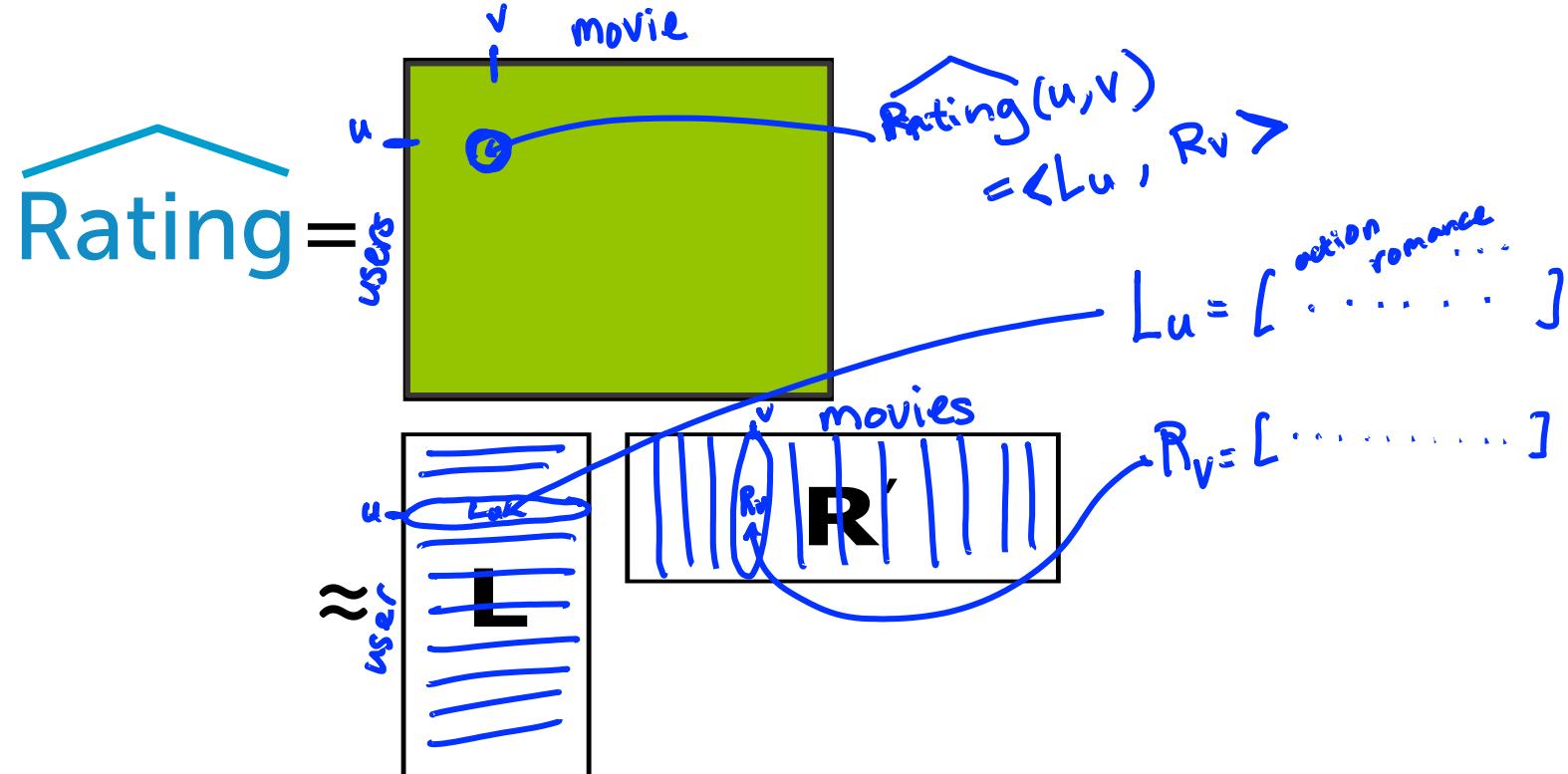
- **Rating(u, v)** is the product of the two vectors

$$R_v = [0.3 \quad 0.01 \quad 1.5 \quad \dots] \rightarrow 0.3 * 2.5 + 0 + 1.5 * 0.8 + \dots = 7.2$$

$$L_u = [2.5 \quad 0 \quad 0.8 \quad \dots] \rightarrow 0 + 0.01 * 3.5 + 1.5 * 0.01 + \dots = 0.8$$

- **Recommendations:** sort movies user hasn't watched by **Rating(u, v)**

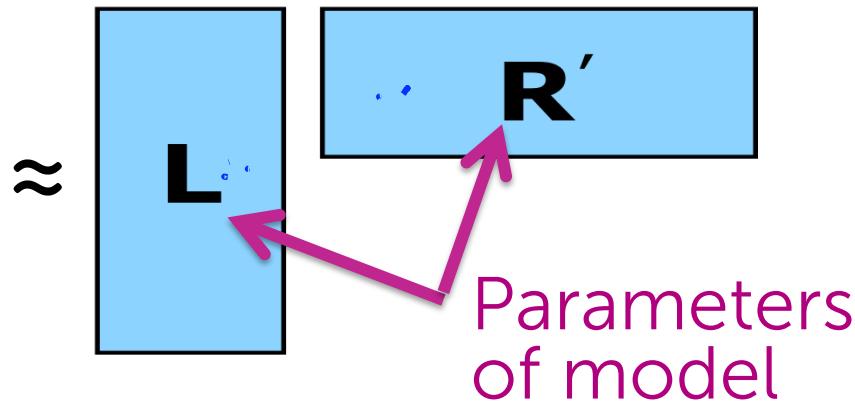
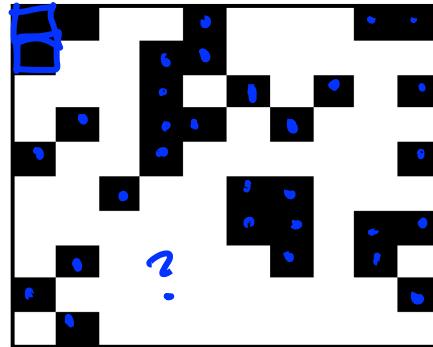
Predictions in matrix form



But we don't
know topics of
users and movies...

Matrix factorization model: Discovering topics from data

Rating =



$$RSS(L, R) =$$

$$(Rating_{u,v} - \langle L_u, R_v \rangle)^2$$

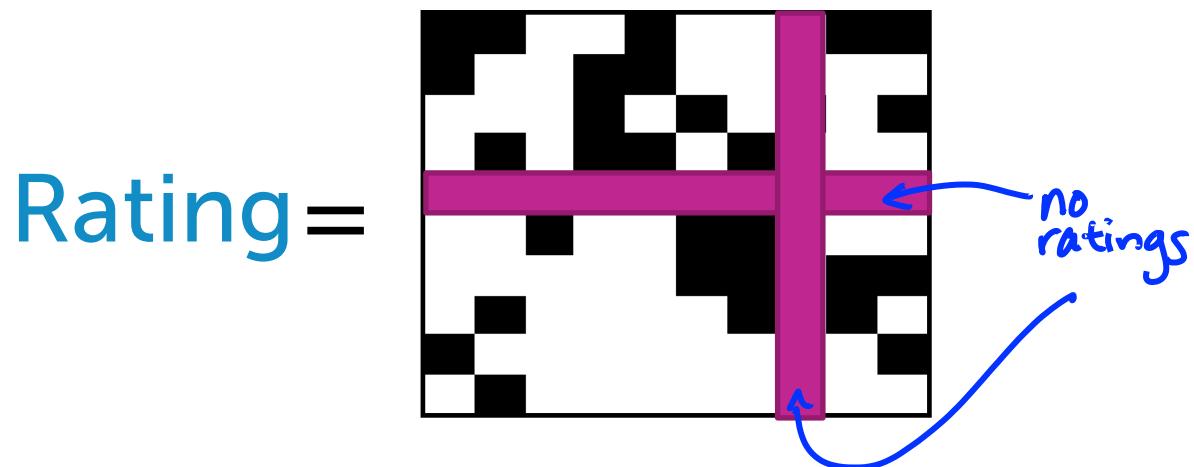
+ [include all (u, v) pairs where

$Rating_{u,v}$ are available]

- Only use observed values to estimate "topic" vectors \hat{L}_u and \hat{R}_v
- Use estimated \hat{L}_u and \hat{R}_v for recommendations

Limitations of matrix factorization

- Cold-start problem
 - This model still cannot handle a new user or movie



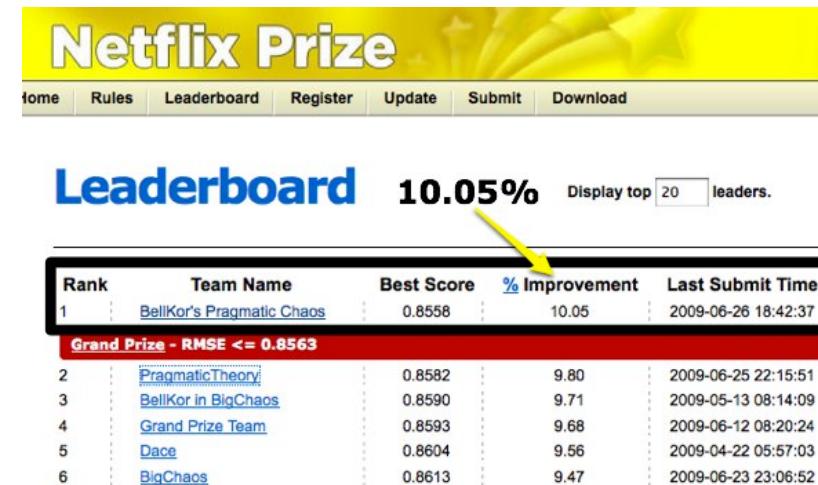
Bringing it all together: Featurized matrix factorization

Combining features and discovered topics

- Features capture **context**
 - *Time of day, what I just saw, user info, past purchases,...*
- Discovered topics from matrix factorization capture **groups of users** who behave similarly
 - *Women from Seattle who teach and have a baby*
- **Combine** to mitigate cold-start problem
 - Ratings for a new user from **features** only
 - As more information about user is discovered, matrix factorization **topics** become more relevant

Blending models

- Squeezing last bit of accuracy by blending models
- Netflix Prize 2006-2009
 - 100M ratings
 - 17,770 movies
 - 480,189 users
 - Predict 3 million ratings to highest accuracy
 - **Winning team blended over 100 models**



A screenshot of the Netflix Prize Leaderboard. The top banner says "Netflix Prize". Below it is a navigation bar with links: Home, Rules, Leaderboard, Register, Update, Submit, Download. The main title is "Leaderboard" with a subtitle "10.05%" and a note "Display top 20 leaders." A yellow arrow points to the "% Improvement" column in the table below. The table has columns: Rank, Team Name, Best Score, % Improvement, and Last Submit Time. The top row shows the winning team: BellKor's Pragmatic Chaos with a Best Score of 0.8558 and a % Improvement of 10.05, submitted on 2009-06-26 18:42:37. A red banner at the bottom says "Grand Prize - RMSE <= 0.8563".

| Rank | Team Name | Best Score | % Improvement | Last Submit Time |
|--|---|------------|---------------|---------------------|
| 1 | BellKor's Pragmatic Chaos | 0.8558 | 10.05 | 2009-06-26 18:42:37 |
| Grand Prize - RMSE <= 0.8563 | | | | |
| 2 | PragmaticTheory | 0.8582 | 9.80 | 2009-06-25 22:15:51 |
| 3 | BellKor in BigChaos | 0.8590 | 9.71 | 2009-05-13 08:14:09 |
| 4 | Grand Prize Team | 0.8593 | 9.68 | 2009-06-12 08:20:24 |
| 5 | Dace | 0.8604 | 9.56 | 2009-04-22 05:57:03 |
| 6 | BigChaos | 0.8613 | 9.47 | 2009-06-23 23:06:52 |

A performance metric for recommender systems

The world of all baby products



User likes subset of items



Why not use classification accuracy?

- Classification accuracy =
fraction of items correctly classified
(liked vs. *not liked*)
- Here, not interested in what a person
does not like
- Rather, how quickly can we discover the
relatively few *liked* items?
 - (Partially) an imbalanced class problem

How many liked items were recommended?



Recall

$$\frac{\text{\# liked \& shown}}{\text{\# liked}}$$

$$= \frac{3}{5}$$

How many recommended items were liked?



Precision

$\frac{\text{\# liked \& shown}}{\text{\# shown}}$

$$= \frac{3}{11}$$

Maximize recall: Recommend everything



Recall
 $\frac{\text{\# liked \& shown}}{\text{\# liked}}$

$$= | \leftarrow \frac{5}{5} \checkmark$$

Resulting precision?



Precision

$\frac{\text{\# liked \& shown}}{\text{\# shown}}$

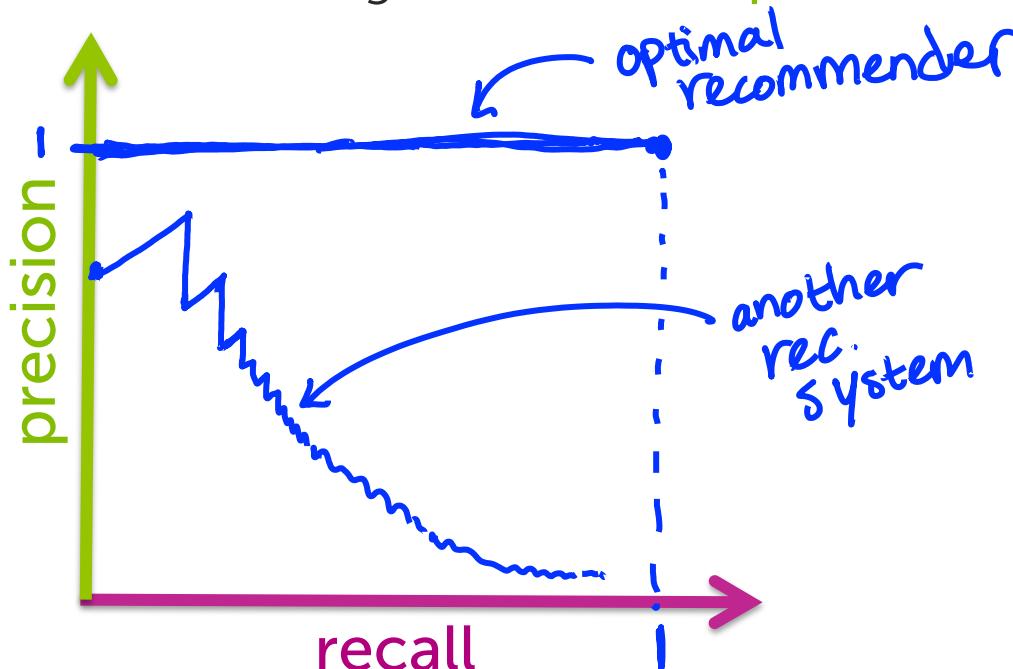
small,
maybe very
small

Optimal recommender



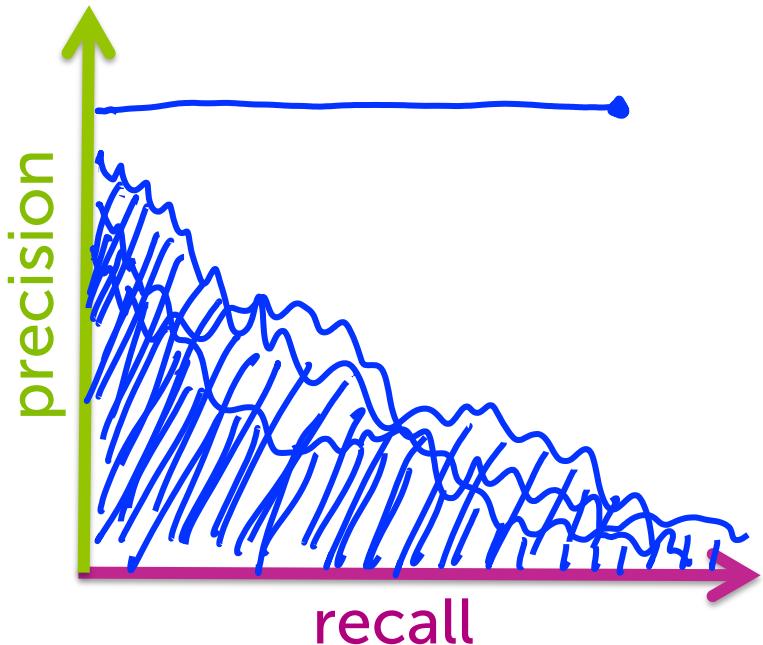
Precision-recall curve

- **Input:** A specific recommender system
- **Output:** Algorithm-specific precision-recall curve
- To draw curve, vary threshold on # items recommended
 - For each setting, calculate the **precision** and **recall**



Which Algorithm is Best?

- For a given **precision**, want **recall** as large as possible (or vice versa)
- One metric: largest **area under the curve (AUC)** 
- Another: set desired recall and maximize precision **(precision at k)**



Summary of recommender systems

What you can do now...

- Describe the goal of a recommender system
- Provide examples of applications where recommender systems are useful
- Implement a co-occurrence based recommender system
- Describe the input (observations, number of “topics”) and output (“topic” vectors, predicted values) of a matrix factorization model
- Exploit estimated “topic” vectors (algorithms to come...) to make recommendations
- Describe the cold-start problem and ways to handle it (e.g., incorporating features)
- Analyze performance of various recommender systems in terms of precision and recall
- Use AUC or precision-at-k to select amongst candidate algorithms