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

100 Hours a Minute

What do I care about?

Information overload

↓

Browsing is "history"

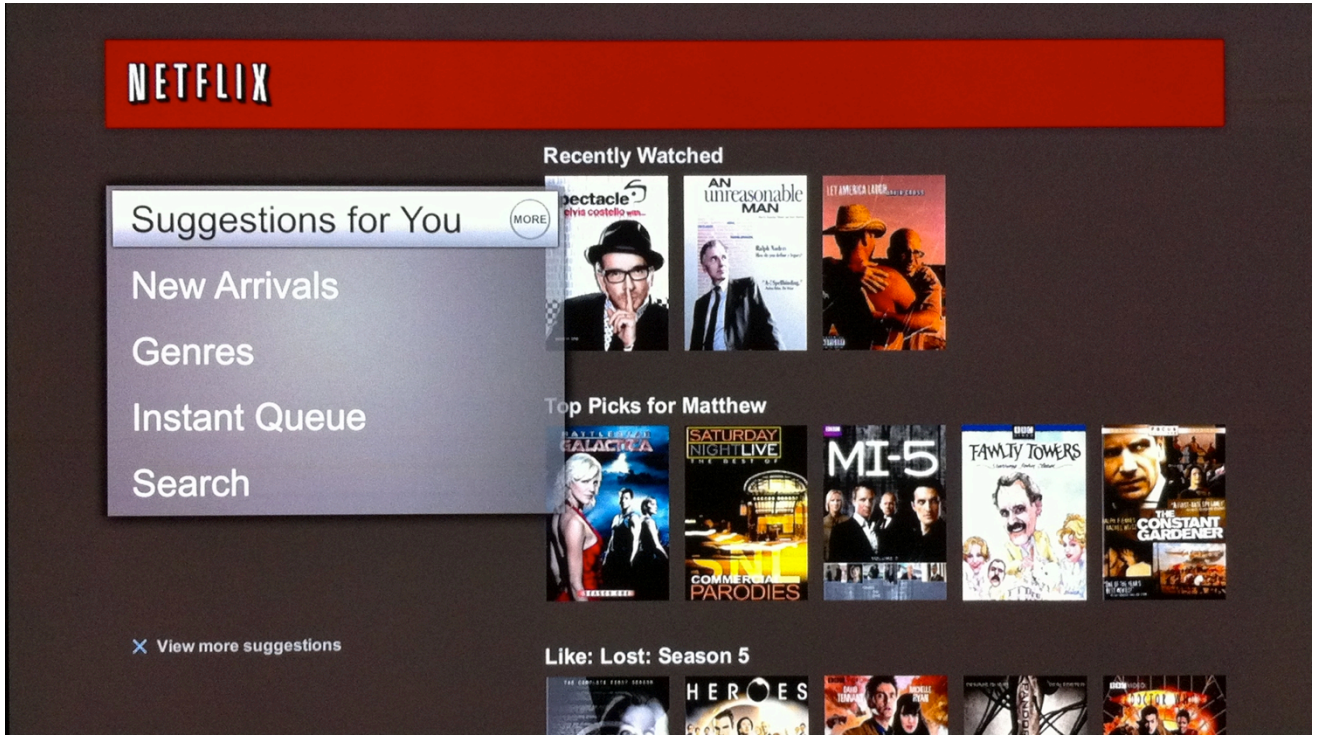
– Need new ways to discover content

Personalization: Connects *users & items*

viewers

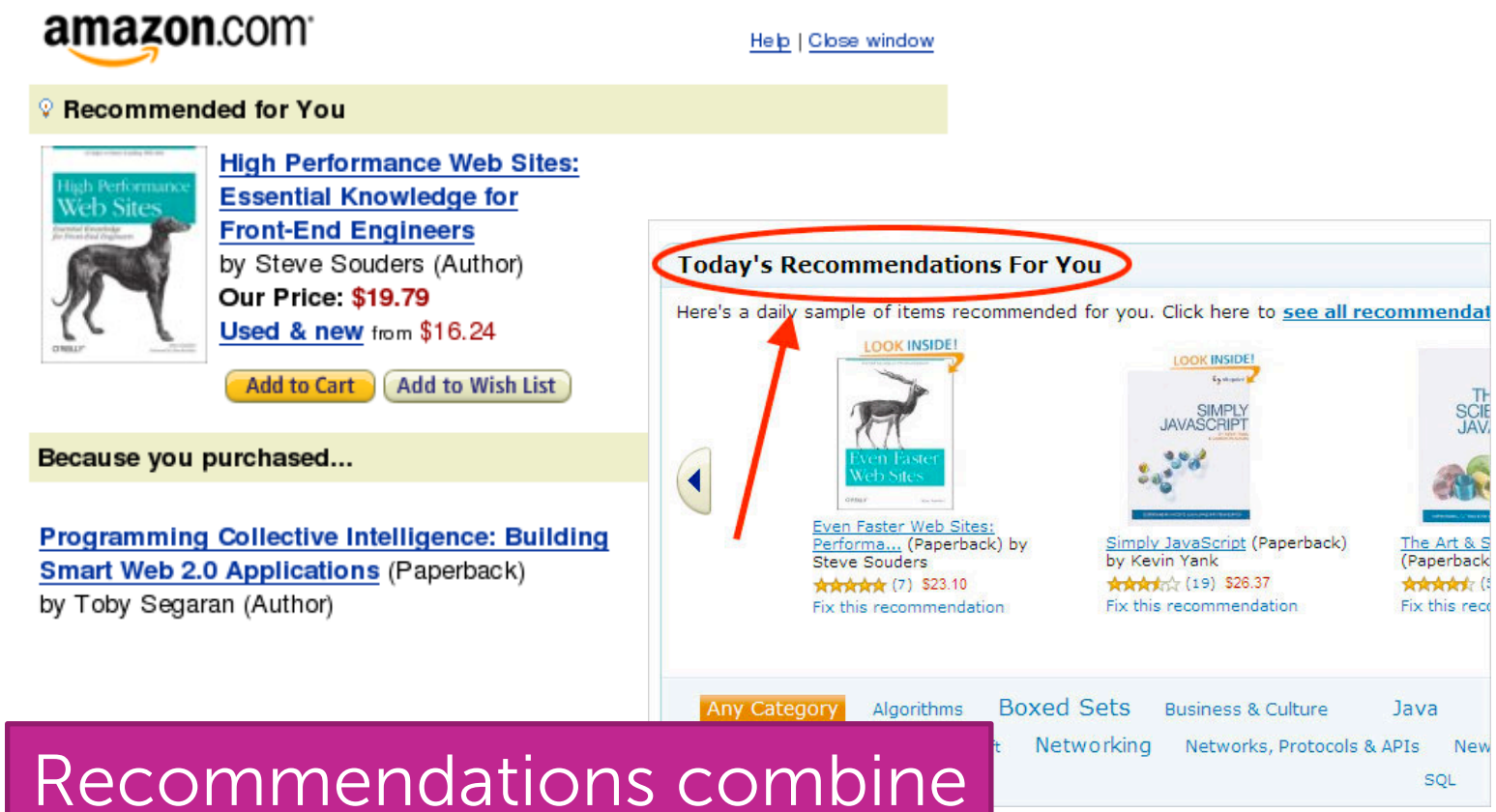
videos

Movie recommendations



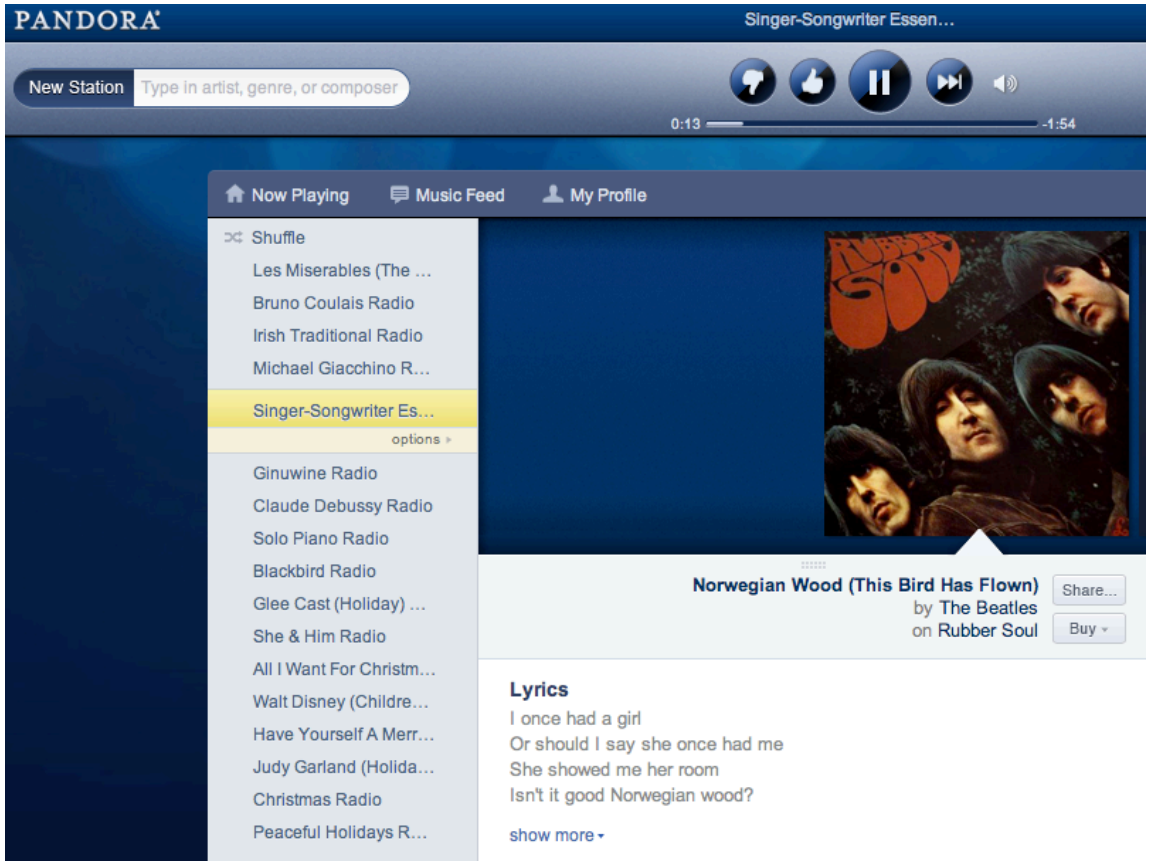
Connect users with movies they may want to watch

Product recommendations



Recommendations combine global & session interests

Music recommendations



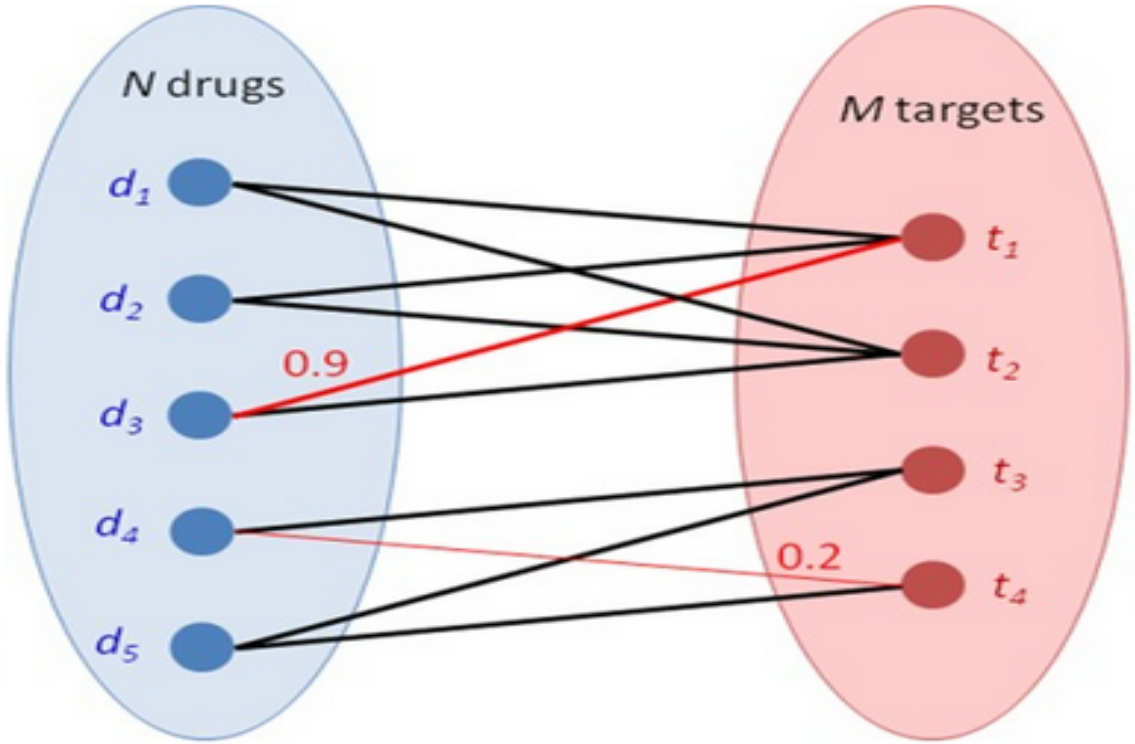
Recommendations form coherent & diverse sequence

Friend recommendations

Users and "items"
are of the same "type"

Drug-target interactions

Cobanoglu et al. '13



What drug should we "repurpose" for some disease?

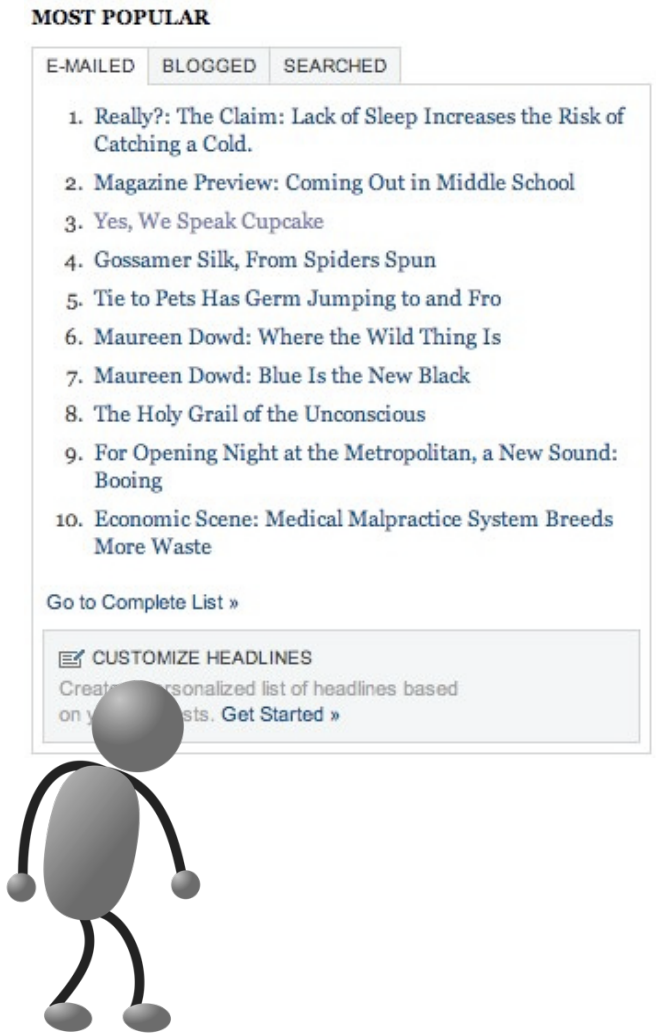
Building a recommender system

Solution 0: Popularity

Simplest approach: Popularity

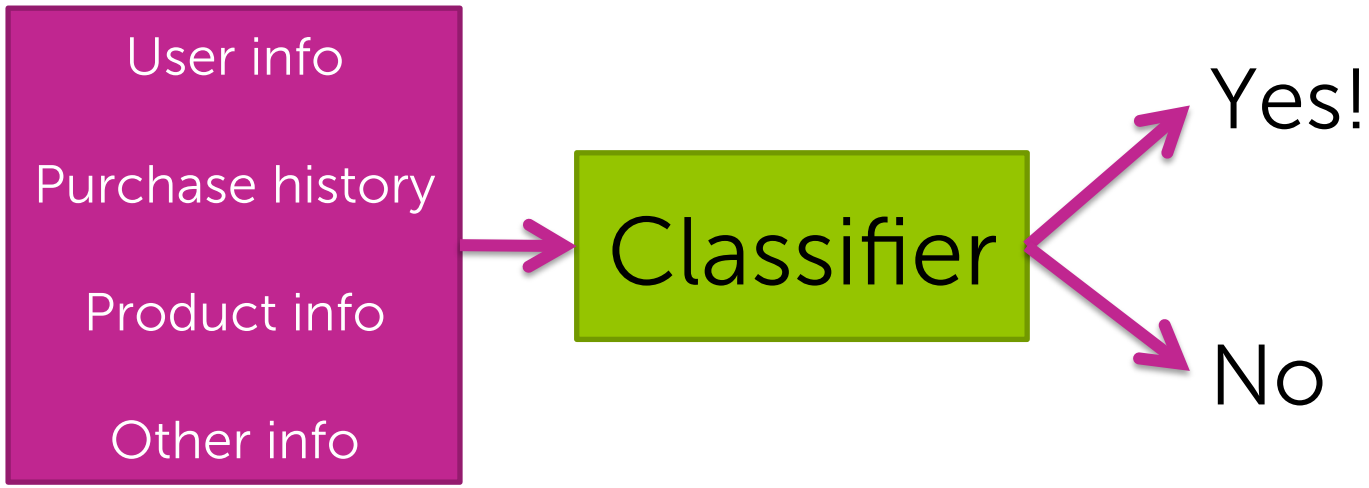
- What are people viewing now?
  - Rank by global popularity

- **Limitation:**
  - No personalization



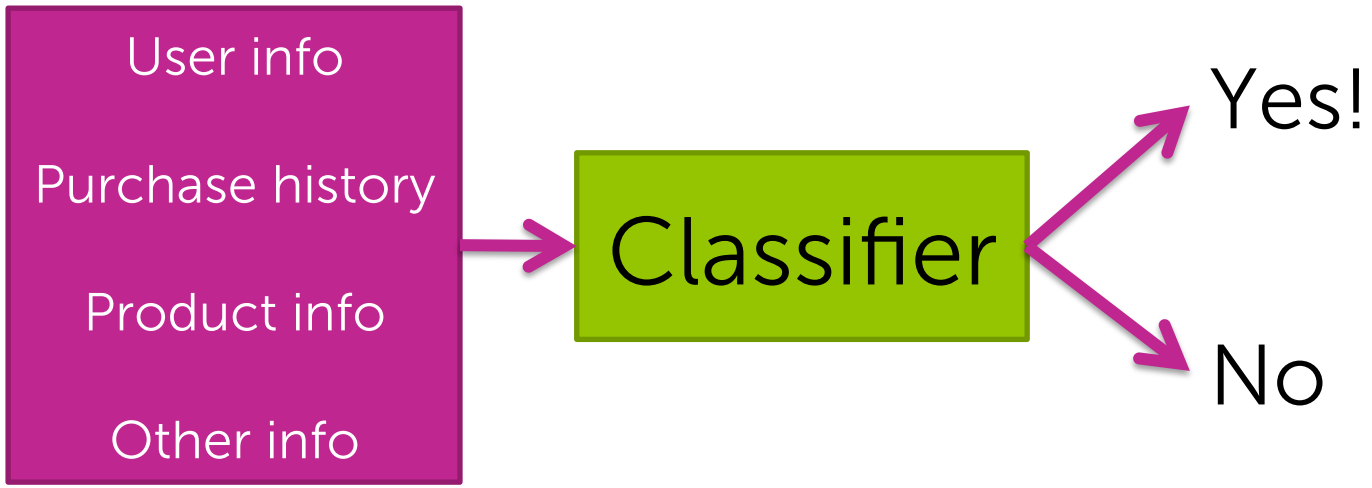
Solution 1: Classification model

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 for collaborative filtering

- 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-occurences

- 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***=*Sophie giraffe* ) large count C_{ij} for ***j***=*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 {*diapers, milk*}
  - Compute user-specific score for each item *j* in inventory by combining similarities:

    Score( *, baby wipes) =*

    $$\tfrac{1}{2}\,(S_{\text{baby wipes, diapers}} + S_{\text{baby wipes, milk}})$$

  - Could also weight recent purchases more

- Sort *Score(*  *, j)* and find item *j* with highest similarity

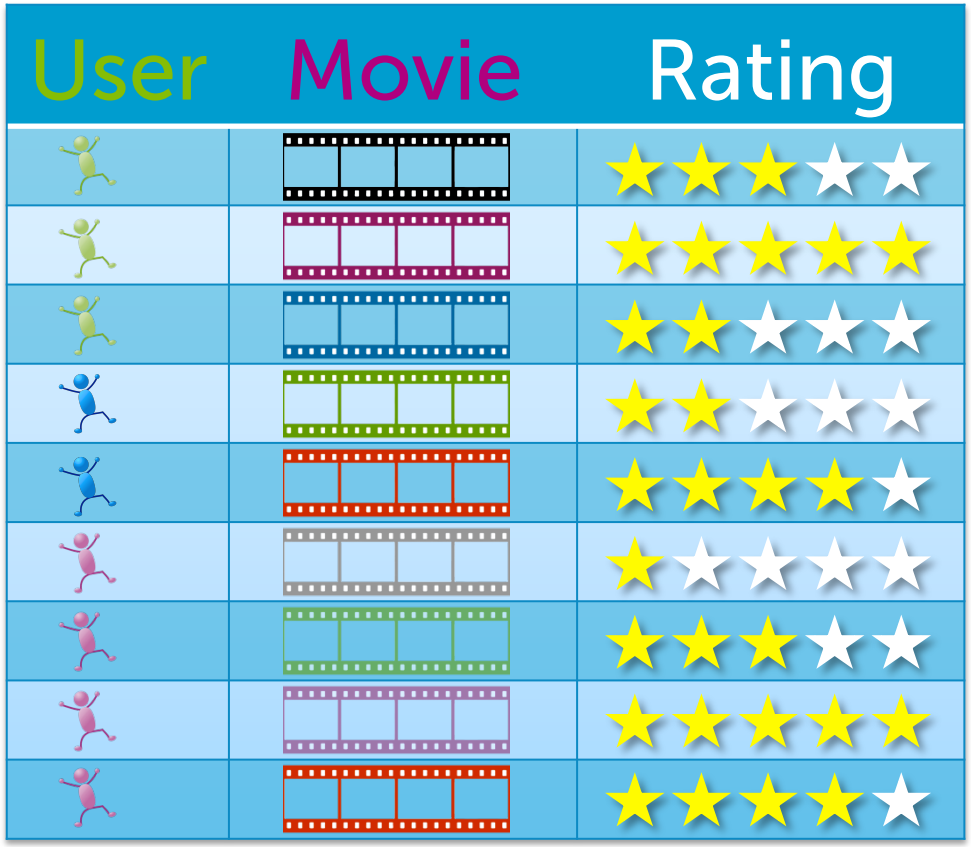
Limitations

- 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

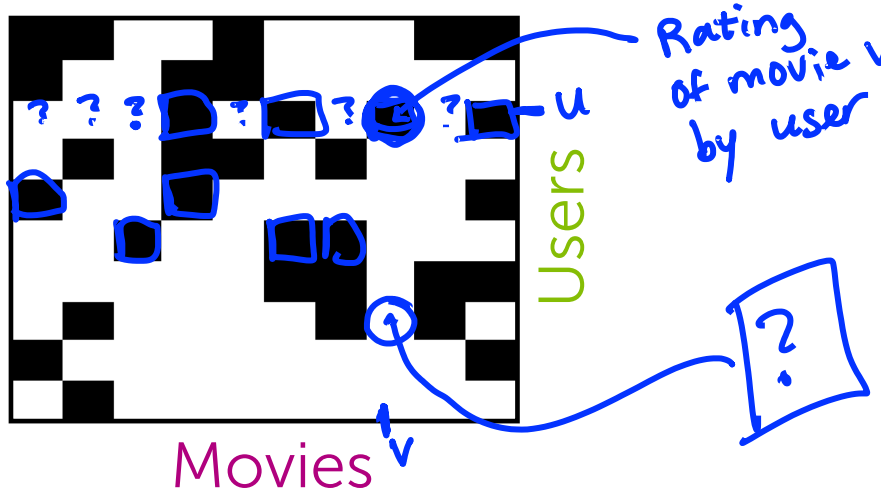
- Users watch movies and rate them



Each user only watches a few of the available movies

Matrix completion problem

Rating =

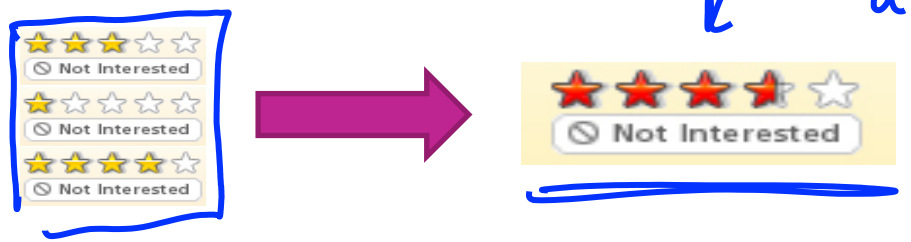


Matrix is sparse since given users have watched only a few movies.

- **Data:** Users score some movies

  Rating(u,v) known for black cells
  Rating(u,v) unknown for white cells

- **Goal:** Filling missing data?



Fill in all unknowns.

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

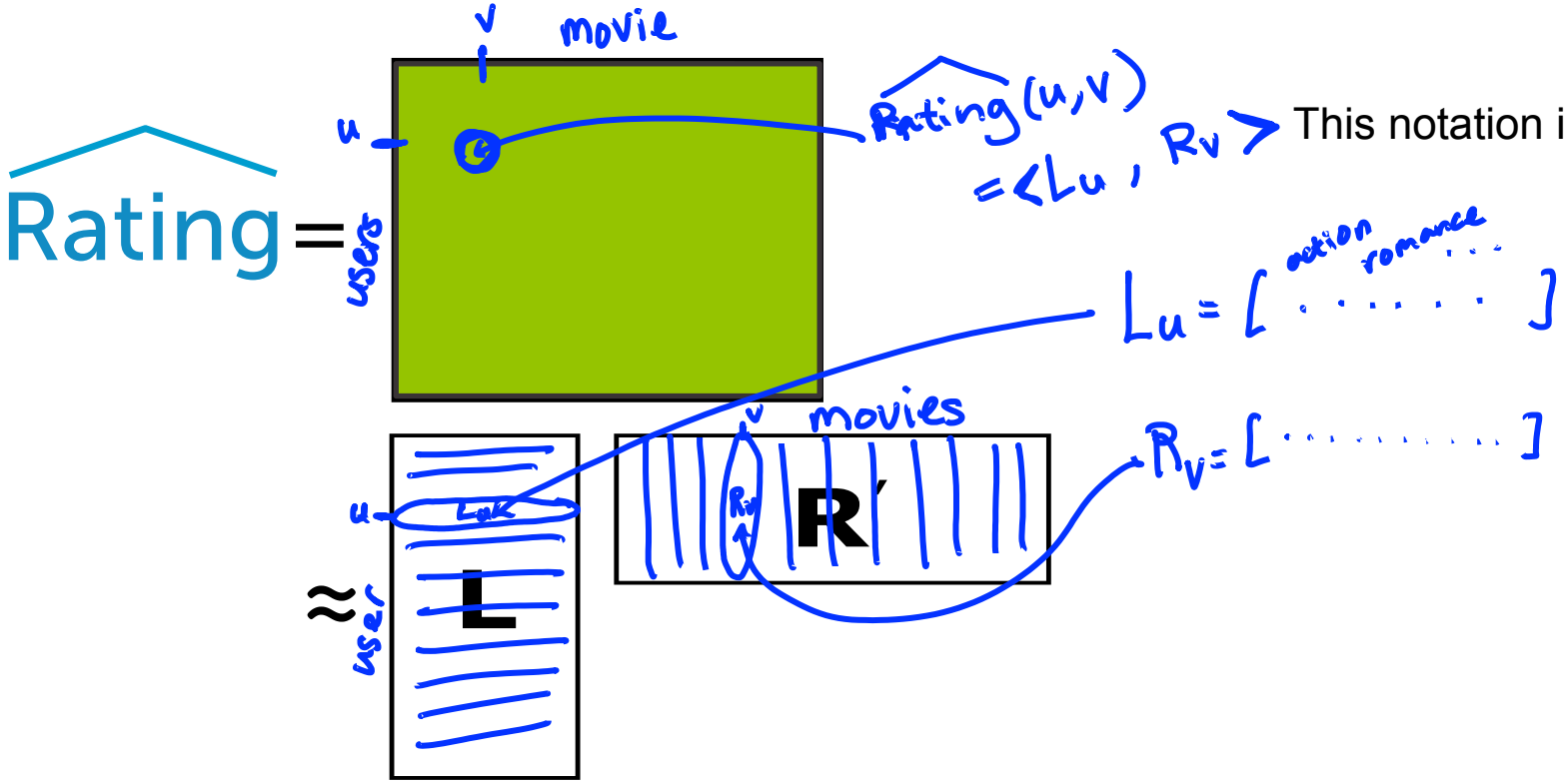
$R_v = [\ 0.3 \quad 0.01 \quad 1.5 \quad \dots \ ]$ → $0.3 * 2.5 + 0 + 1.5 * 0.8 + \dots = 7.2$ > 5

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

$L_{u'} = [\ 0 \quad 3.5 \quad 0.01 \quad \dots \ ]$ → $0 + 0.01 * 3.5 + 1.5 * 0.01 + \dots = 0.8$

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

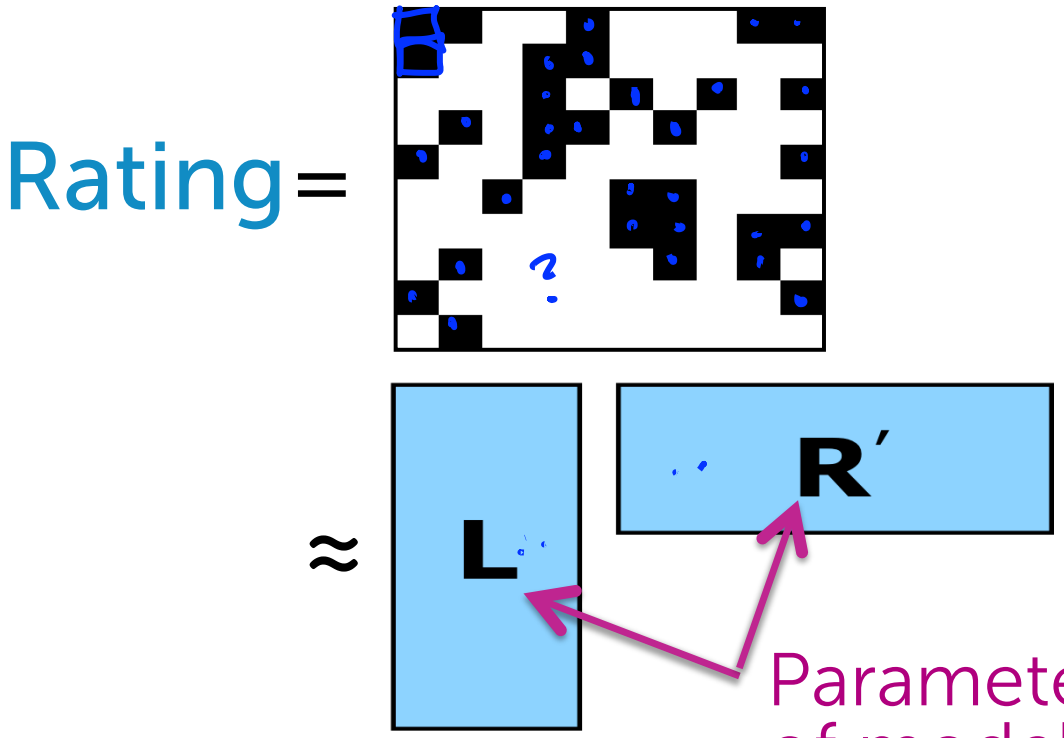
Predictions in matrix form



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

Matrix factorization model: take a matrix, and approximate it with factorization.

Discovering topics from data



Evaluating how well L and R fit observed ratings:

$$RSS(L,R) = \sum_{(u,v)} \left(\text{Rating}(u,v) - \langle L_u, R_v \rangle\right)^2$$

$\langle L_u, R_v \rangle$ ← predicted rating

+ [include all (u,v) pairs where Rating(u,v) are available]

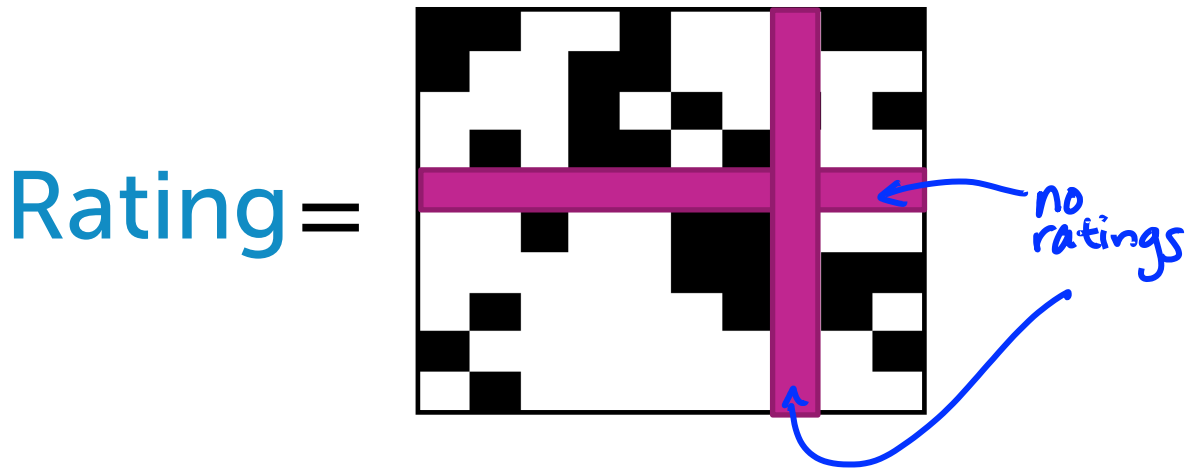
Many efficient algorithms for factorization

- 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

output is the set of estimated parameters

Limitations of matrix factorization

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



Bringing it all together: Featurized matrix factorization

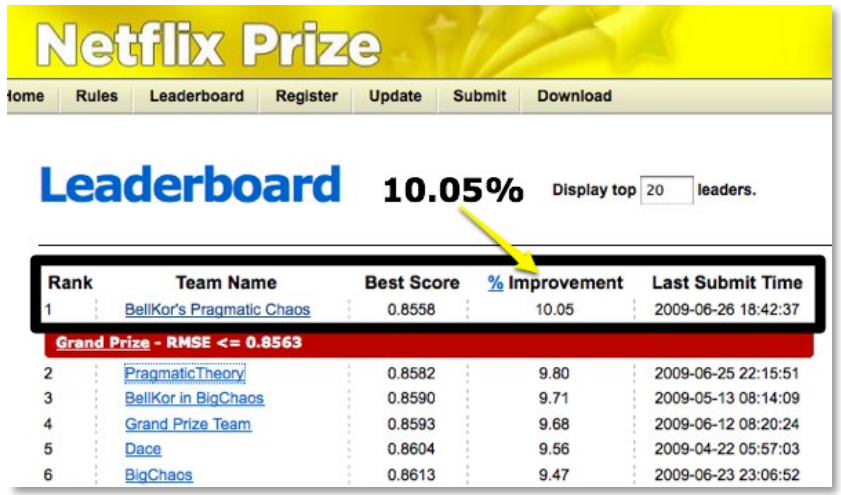
Combining features and discovered topics

- 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 performance metric for recommender systems

The world of all baby products

User likes subset of items

Goal is to discover products that the user likes from the puchases that they've made.

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?

Shown(colored) - recommended
Gray - not recommended
magenta box - liked

Recall

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

$= \frac{3}{5}$

How many recommended items were liked?

Green box - recommended

Precision

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

$= \frac{3}{11}$

Maximize recall: Recommend everything

Recall

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

$= 1 \leftarrow \frac{5}{5}$ ✓

Resulting precision?

Precision

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

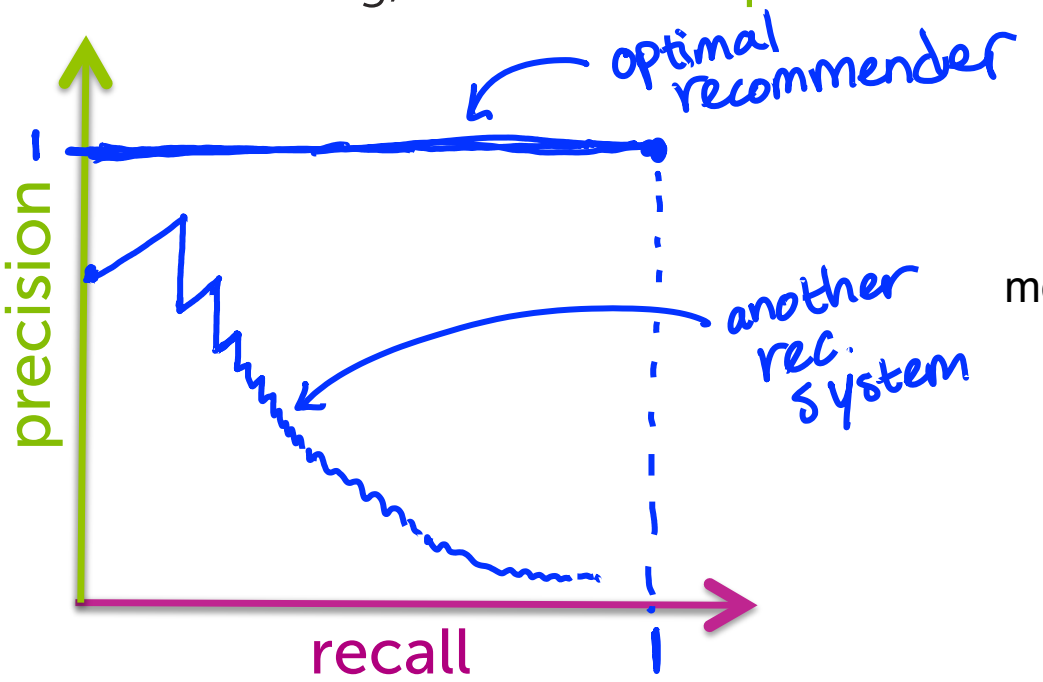
small, maybe very small

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

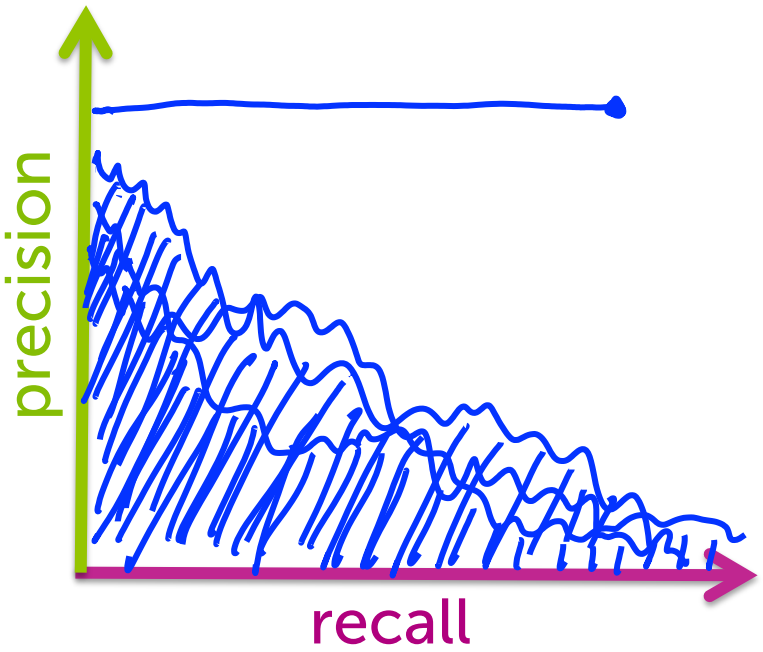


only recommend products I like => precision always 1
e.g. #like = 10, #recommend = 1,2,3,...,10

more realistic system

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