The Art of Relevance and Recommendations

@cchio

slides+code: https://git.io/vXRwv

…i.e. how to hit the ground running with recommender systems

What do I do?

- Stanford B.S./M.S. Computer Science
- Research Engineer at Shape Security
- Organizer, ‘Data Mining for Cyber Security’ meetup group in Silicon Valley
- Authoring ‘Machine Learning and Security’ (O'Reilly, mid-late 2017)
- Presented independent research in Adversarial ML at dozens of security/ML conferences over last two years

- Bought Justin Bieber
- Bought Selena Gomez

- Performs a search for Justin Bieber
- Recommender system suggests Selena Gomez

facebook amazon Spotify

last.fm YouTube yelp

Google NETFLIX Quora

LinkedIn ebay bing

twitter

stackoverflow reddit airbnb

Ways to serve recommendations (in desc. order of naivety)

1. Hand-crafted; “our picks”
2. Global aggregates; top 10 among all users, most recent items
3. Individualized recommendations based on inferred/learned user preferences

MovieLens dataset

1. Activity from MovieLens, a movie recommendation service
2. 100004 ratings, 1296 tags, 9125 movies
3. 671 users, all selected users rated at least 20 movies

How to guess what a user might like?

Explicit

Implicit

How to guess what a user might like?

users

movies	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	

- unknown rating
- rating between 1 to 5

Jure Leskovec , Stanford CS246

Real Problems

1. Cold-start for new users/items
2. Not leveraging on crowd wisdom
3. Sparse ratings
4. Popularity bias/Long-tail problem
5. Computational efficiency issues

Types of Recommender Systems

1. Content based filtering
2. Collaborative filtering
3. Latent-factor based
4. Multi-armed bandits

Content based filtering

1. Recommendations made based solely on the characteristics of the item
2. Build a profile of a user based on his past likes
3. System recommends items similar to what a user has liked in the past

Content based filtering

solves

1. New items can be recommended
2. Obscure items will not be neglected
3. Can be efficiently pre-computed

but

- Finding good features is hard/laborious
- Cannot recommend to new users without a profile

Collaborative filtering

User-based

1. Get the active user's item ratings (explicit + implicit)
2. Identify other users that made similar ratings on similar items
3. Recognize items that these similar users liked
4. Generate a prediction for the active user

Collaborative filtering

Item-based

1. Look at items the active user rated
2. Compute the similarity of these items to other items, based on how other users rated all of them
3. Generate a prediction for the active user

Collaborative filtering

---

solves

1. Leverages heavily on crowd wisdom
2. No feature engineering required

but

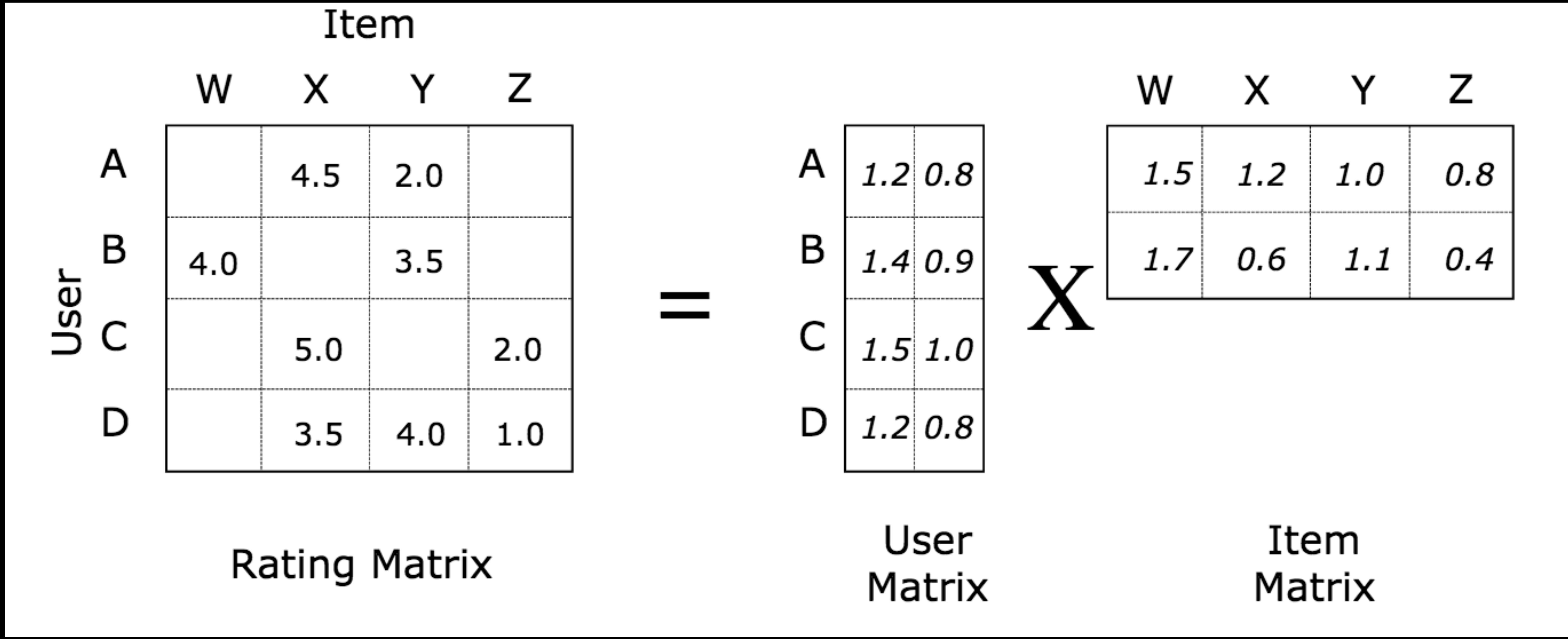
- Cold-start
- Sparse ratings
- Cannot recommend obscure items
- Hard to pre-compute, ratings constantly being updated

Latent-factor based

1. Develop a model of user-item relations that tries to uncover latent/hidden relationships or preferences
2. Use ML algorithms to model similarities

Latent-factor based

Low-rank matrix factorization



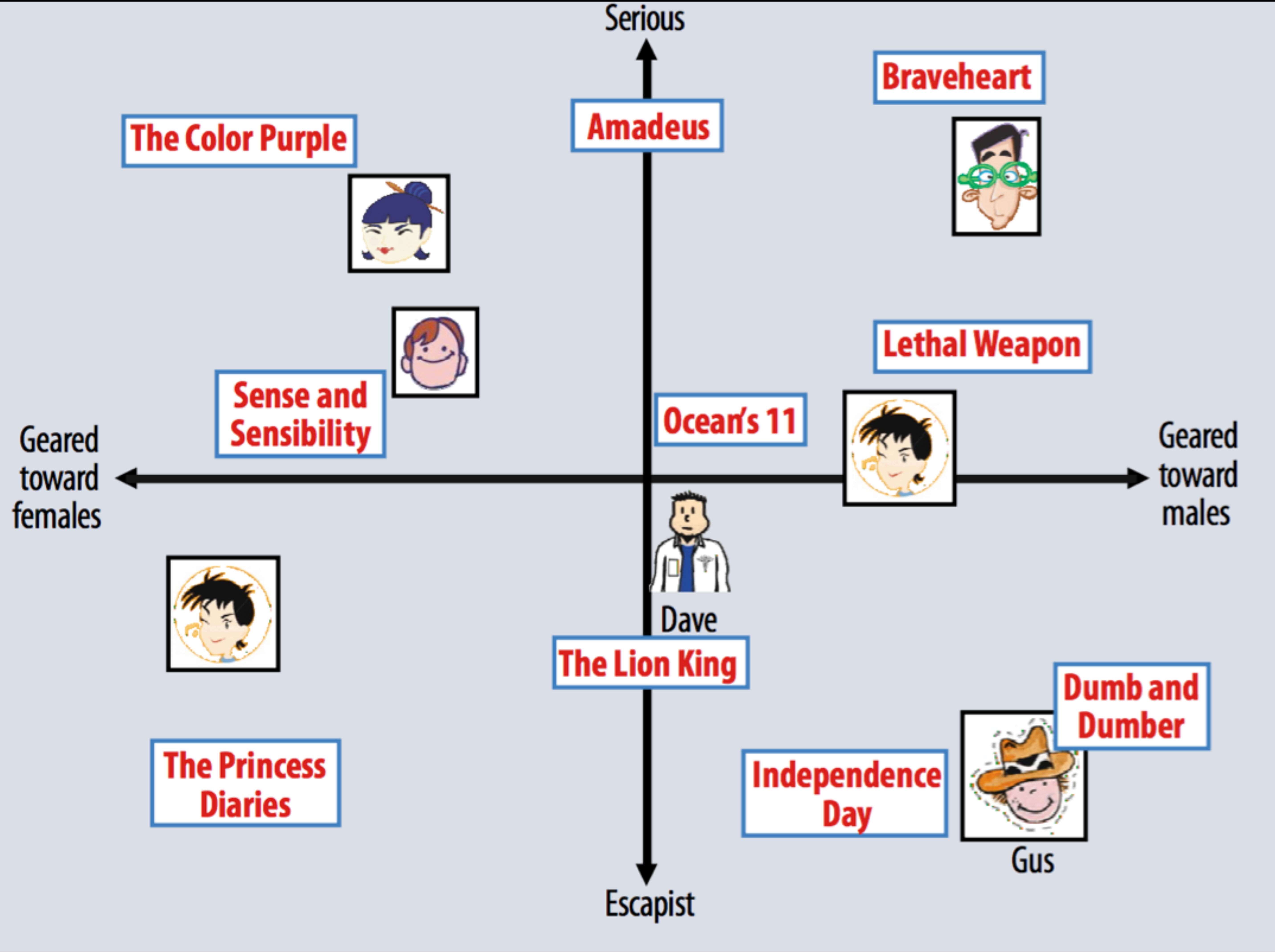


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

Latent-factor based

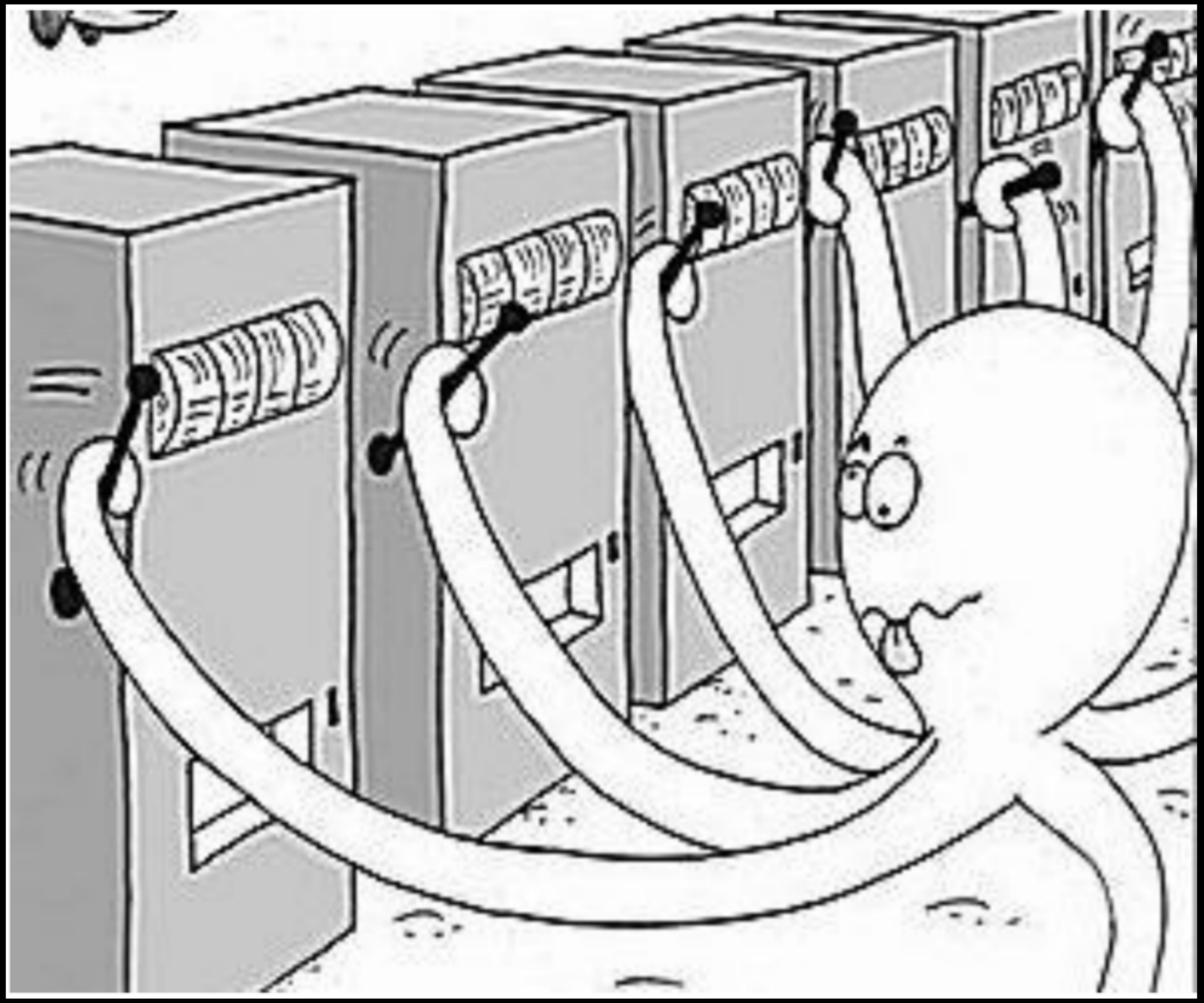
solves

1. Fixes sparse ratings
2. Similarity comparisons more efficient

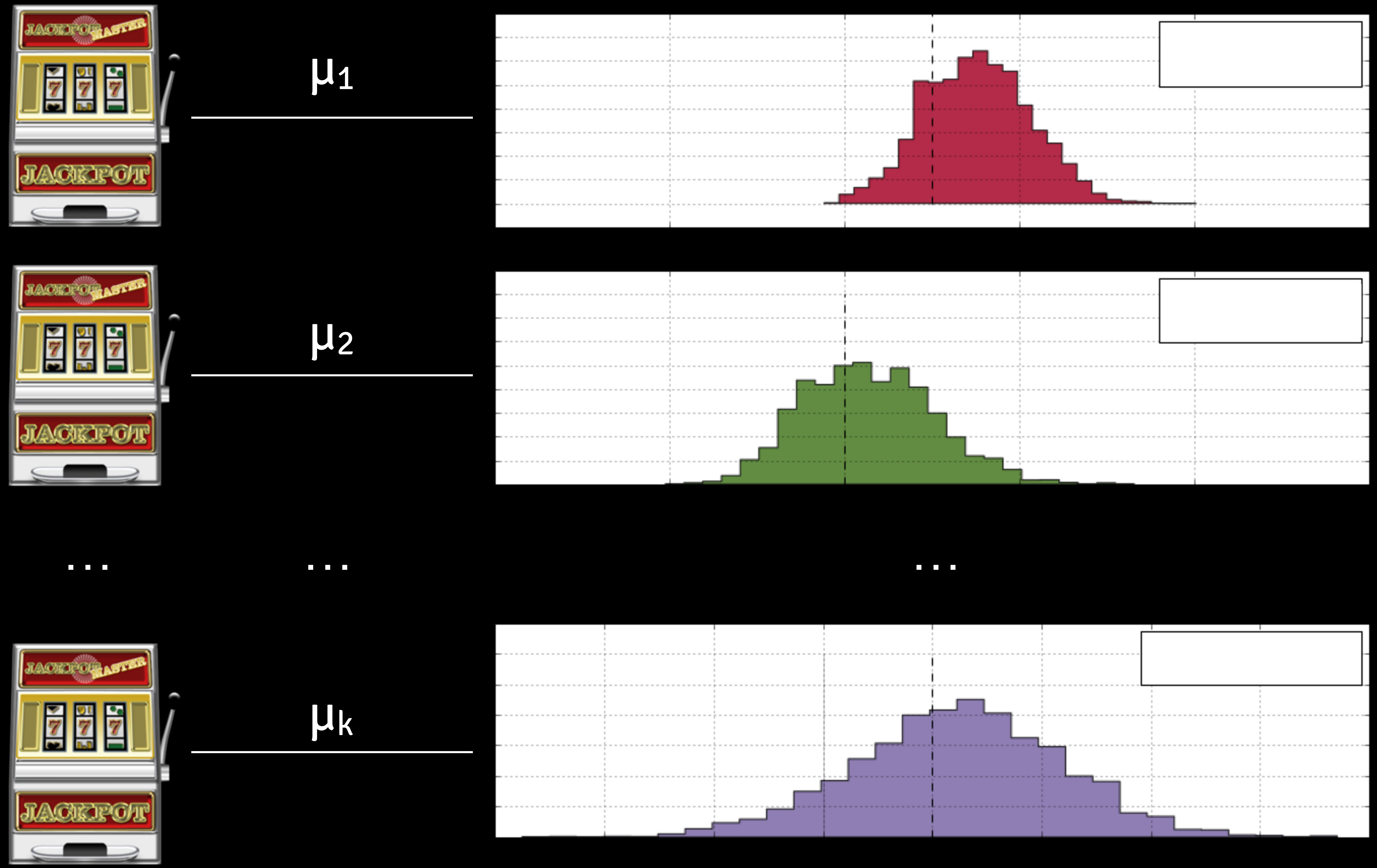
but

- Still not enough recommending new/unique items to users (long-tail)

The multi-armed bandit problem



1. Each arm x
   A. Wins (output = 1) with a fixed unknown probability μ_x
   B. Loses (output = 0) with a fixed unknown probability $1-\mu_x$
2. All you have is posterior probability
3. How to pull arms to maximize total reward?


μ1
μ2
...
μk

Naive solution

1. Pull each arm k times (e.g. 100)
2. Record the mean reward of each arm
3. Pull the arm with the highest mean reward for eternity

Thompson Sampling works real well

1. Maintain a probability function for each arm, based on data collected over time
2. Draw a sample from every arm's probability function
3. Pull the arm that gives the largest drawn sample
4. Repeat for eternity

How does this apply to content A/B testing?

k arms → k unique content versions
goal → maximize user retention, CTR

How does this apply to online advertising?

---

k arms → k distinct page visitors
goal → maximize ad engagement

How does this apply to clinical trials?

k arms → k distinct treatments
goal → minimize patient loss

How does this apply to recommender systems?

k arms → k distinct items to recommend
goal → maximize user interest

…you're running an experiment each time you pull an arm

How does this apply to recommender systems?

k arms → k distinct items to recommend

goal → maximize user interest

…you're running an experiment each time you pull an arm

Evaluation

Practical evaluation concerns

$$\sqrt{\sum_{xi}\left(r_{xi} - r^*_{xi}\right)^2}$$

RMSE unreasonably penalizes a method that does well for predicting high ratings but not low ratings

vs.

- Precision @ Top-5
- Receiver Operating Characteristic (ROC) Area Under Curve (AUC)

Practical evaluation concerns

Prediction diversity, context, order
are often as important as accuracy

Practical evaluation concerns

Cost function - what happens when you make a bad recommendation?

Complexity/Efficiency

- Expensive step is item/customer similarity matching
- Often too expensive to do at runtime
  - Precompute, good for item-item collab filtering

- Dimensionality reduction
- Locality sensitive hashing
- Clustering

Questions?

Takeaways

1. Implementing collaborative filtering is much easier than you think

2. Use matrix factorization to fix sparse ratings

3. Exploitation vs. Exploration
   → using multi-armed bandit algorithms

Practical tips

1. Implicit signals > explicit signals
2. Optimize for model explainability instead of just accuracy
3. (More data + simple algorithm) > (less data + complex algorithm)

slides+code: https://git.io/vXRwv

@cchio