IS 733 Lesson 11

Recommender Systems

Slides based on those from Data Mining by I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, Data Mining: Concepts and Techniques by Han et al., and Vandana Janeja, James Foulds

Announcements

- Homework 4 is due, do submit it on Blackboard by tonight

The _______ approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its genre.

Content filtering

Collaborative filtering

Matrix factorization

Neighborhood based

______ models map both users and items to a joint latent space, such that user-item interactions are modeled as inner products in that space.

Content filtering

Collaborative filtering

Matrix factorization

Neighborhood based

Which is NOT a popular learning algorithm for matrix factorization collaborative filtering models?

Stochastic gradient descent

Alternating Least Squares

Nearest neighbor search

Usually, ____ feedback comprises a sparse matrix, while ____ feedback is typically represented by a densely filled matrix.

Implicit, explicit

Explicit, implicit

Learning outcomes

By the end of the lesson, you should be able to:

- **Compare and contrast** the main **content-based filtering** and **collaborative filtering methods**
- **Explain** the intuition behind **neighborhood-based collaborative filtering** algorithms
- **Outline** the steps of the popular training algorithms for **matrix factorization collaborative filtering** methods: **stochastic gradient descent**, and **alternating least squares**, and **discuss** their advantages/disadvantages
- **Apply** these methods in real-world scenarios, making sensible choices of methods

Recommender Systems

- Match users with products/services they may enjoy
- We interact with recommender systems every day

- Recommendations for:
  - Movies, tv, and videos
  - Music
  - News articles
  - Physical products
  - Restaurants

  …

Digital Marketplaces and the Long Tail

- *"The main problem, if that's the word, is that we live in the physical world and, until recently, most of our entertainment media did, too. But that world puts dramatic limitations on our entertainment."*
  - Chris Anderson, 2006
- Unlike in physical stores, in digital marketplaces
  - we have **unlimited "shelf space"**
  - Customers **not** confined to one **geographic location**

- Products that have ***niche appeal*** can be profitable – the "**long tail**"

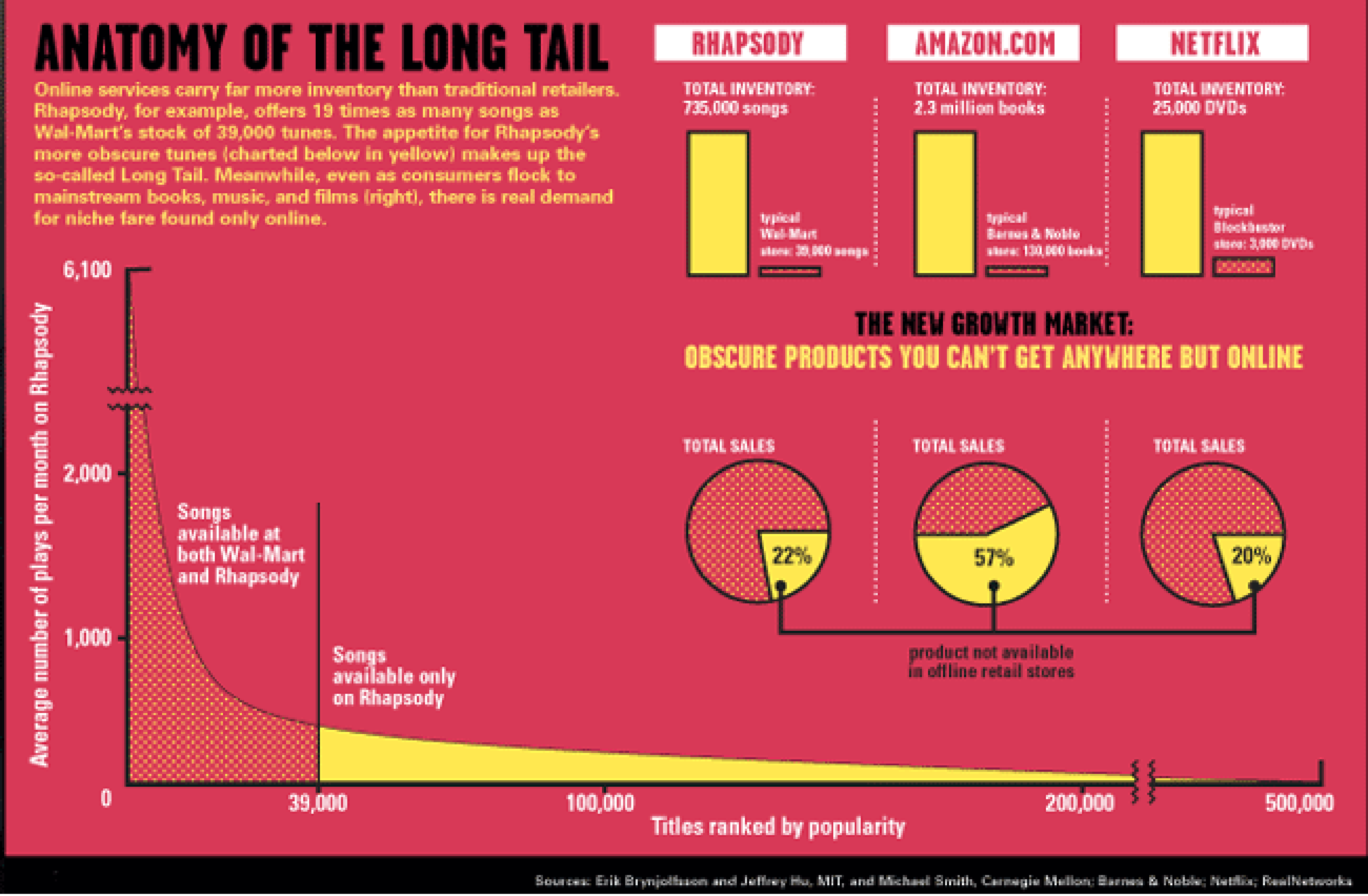


Figure due to Chris Anderson, Wired Magazine, https://www.wired.com/2004/10/tail/

Personalized Recommendation

- **Personalization**: target recommendation to a specific user's personal tastes

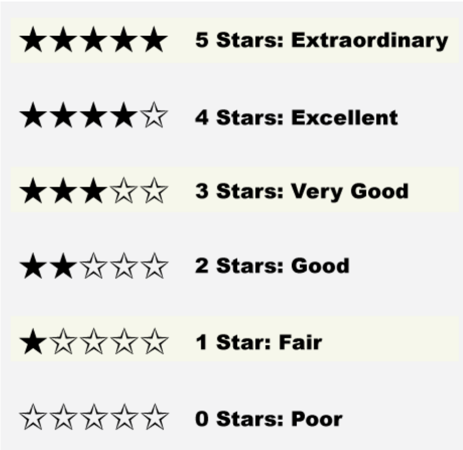
- Non-personalized methods
  - most popular item lists
  - editorial hand-curated lists
  - most recent

  Have some value, but do not address the long tail

Recommendation: Problem Setup

- *U* = set of users
- *S* = set of items (e.g., products, movies,...)
- Want to learn a utility function
  $f{:}U{\times}S \rightarrow R$
  - R referring to as **ratings**.
  - Often 1-5 stars, or binary

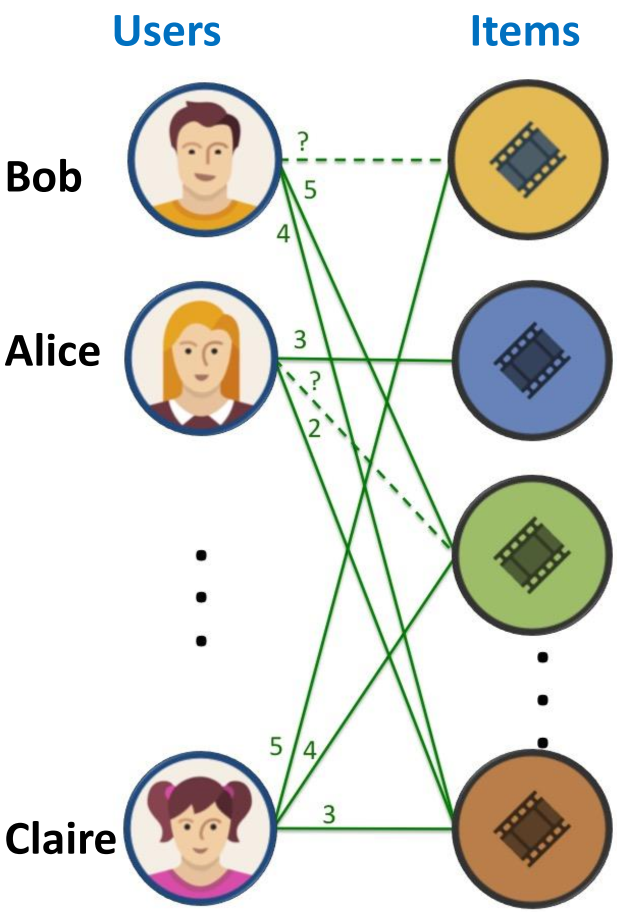
- May have observed some ratings, user behavior, content features,....



Ratings Matrix

	Titanic	Pulp Fiction	The Notebook	The Lion King
Bob	?	?	5	4
Alice	?	3	?	2
Claire	5	?	4	3

Ratings



Can view as a **bipartite graph**

Recommendation: Problem Setup

- We want to make *useful recommendations to users*

- **Proxy task**: predict the ratings that they will make on items that they have not yet rated. We can use these ratings for our final system

- **Challenges:**
  - **Cold start** – When new user or item enters system, we have little to no information
  - **Serendipity** – Want users to discover items that they like which are different to those they already know
  - **Scalability** to large datasets

serendipity [ser-*uh* n-**dip**-i-tee]

EXAMPLES | WORD ORIGIN

SEE MORE SYNONYMS FOR *serendipity* ON THESAURUS.COM

noun

1 an aptitude for making desirable discoveries by accident.

2 good fortune; luck:
the serendipity of getting the first job she applied for.

https://www.dictionary.com/browse/serendipity

Content-Based Filtering

- Try to recommend similar items to what the user has liked in the past
- Extract features based on content of the items (**item profiles**)
- Build feature vectors for users based on content features (**user profiles**)
- Recommend items that are similar to user profile

Item Profiles

- Set of features extracted for item, e.g. genre, actors, directors, year, textual features,…

	Horror	Comedy	…	Robin Williams	…	1993	…
Mrs Doubtfire	0	1		1		1	

Building User Profiles

- Represent users in the same feature space as items
- Typically aggregate content features of rated/purchased items, e.g., weighted average

	Horror	**Comedy**	**…**	**Robin Williams**	**…**	**1993**	**…**
Alice	10	0.1		3		0.1	

- **Implicit feedback** can be useful, e.g. items *browsed* but not purchased, *time spent* on the page…

Recommendation with Content-Based Filtering

- Compute *similarities* between users and items' profiles. **Cosine similarity** is often used

$$f(\mathbf{u}, \mathbf{s}) = \frac{\mathbf{u} \cdot \mathbf{s}}{||\mathbf{u}|| \, ||\mathbf{s}||}$$

- Recommend items whose *vectors are most similar*

- Alternatively, *classification algorithms* could be used

Which challenge is the biggest weakness for content-based filtering methods?

Cold start problem for items

Serendipity

Scalability of the computation

Strengths and Weaknesses of CB Filtering

- **Strengths:** No "**cold start**" problem for *items*
  - **Cold start problem** = challenge of recommending with little data, for users or items

- **Weaknesses:**
  - Cold start problem for *users*, **may overfit**
    - Arguably, user cold start is more important than for items
  - Content features for items may be financially **expensive**
  - Feature construction is **domain specific**
  - Not good for **serendipity** – suggestions are similar to *items the user already likes*, so may not help them to stumble upon new types of items

Collaborative Filtering

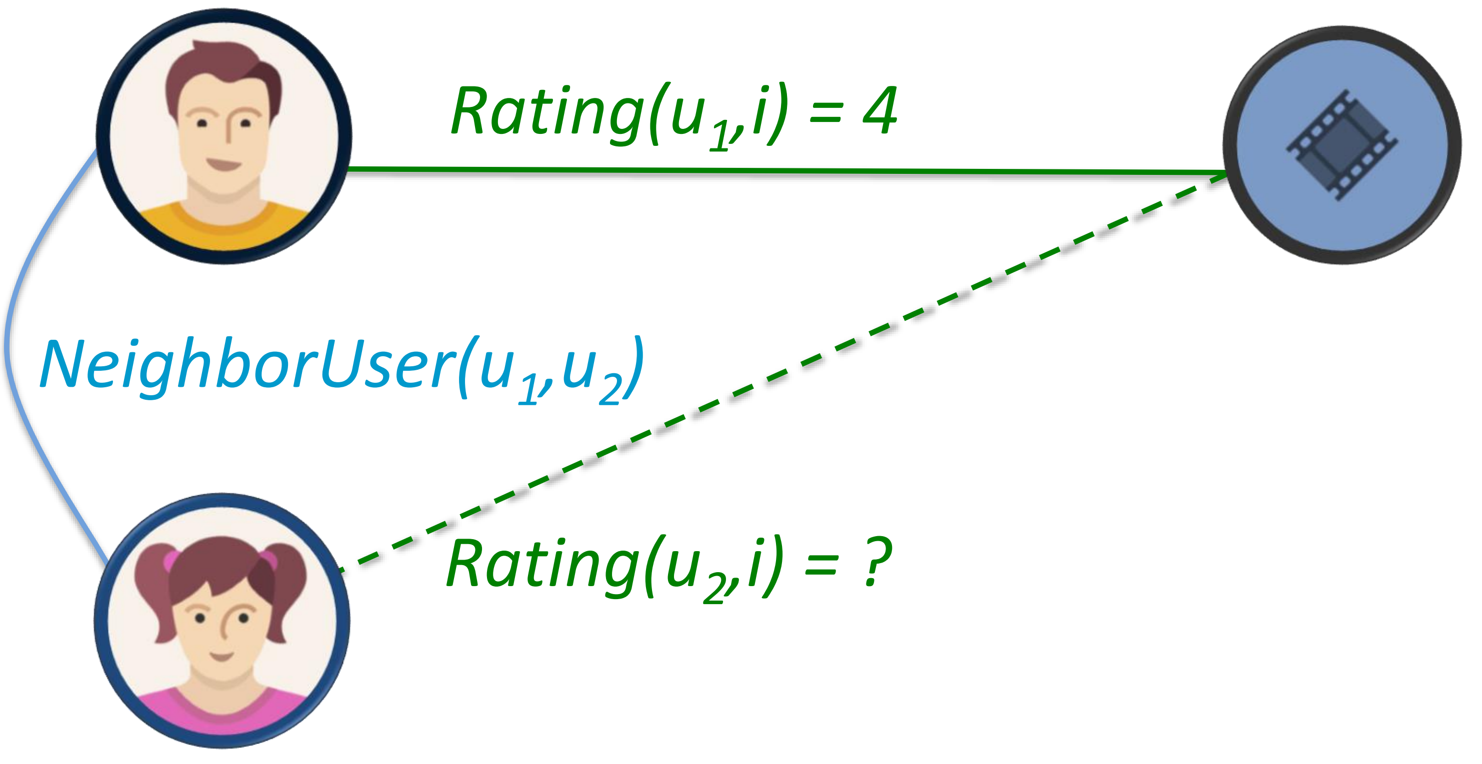
- Makes use of o**ther users**, or **other items**, to predict ratings "**collaboratively**"

- Predictions based on *other ratings*

- **Neighborhood-based** methods
  - Find similar users or similar items, predict that the target rating will be similar
- **Matrix factorization** / **latent factor** methods
  - **Factorize the ratings matrix**, find **latent vector** representations for users and items based on the factorization

Neighborhood-Based CF Methods

- User-user neighborhood-based CF



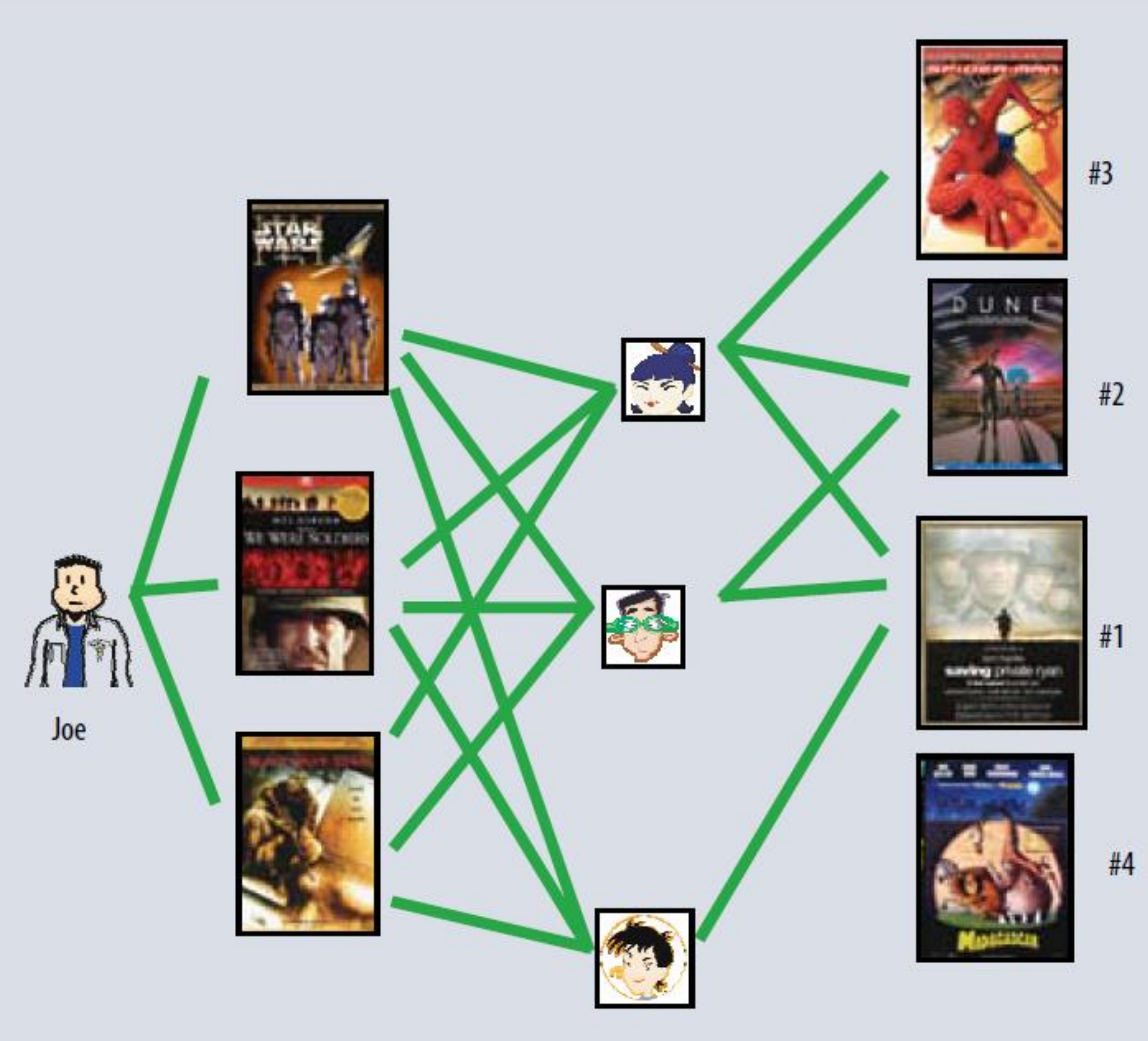


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first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

Figure due to Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8)

Neighborhood-Based CF Methods

- User-user neighborhood-based CF
  - **Find K-nearest neighbor users** according to ratings
    - Represent **users** by their **rows of the ratings matrix**
    - Only consider *users who have rated the item s* in question
  - Aggregate their ratings for an item to predict rating

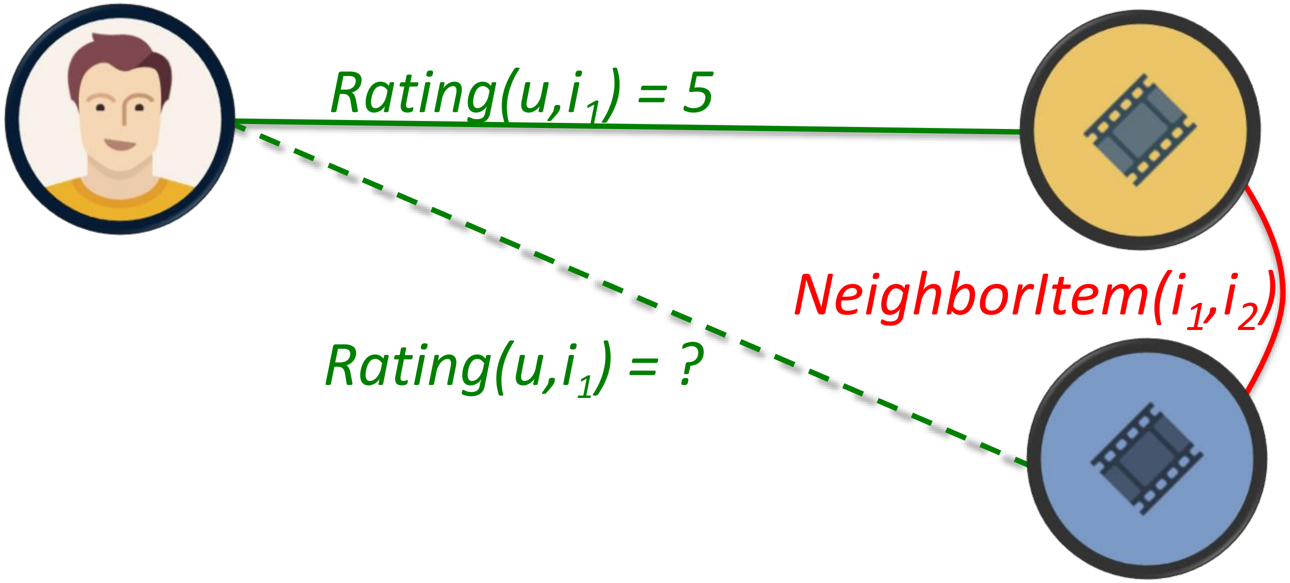
$$f(u,s) = \frac{1}{\sum_{u' \in neighbors(u)} \mathrm{sim}(\mathbf{u},\mathbf{u}')} \sum_{u' \in neighbors(u)} \mathrm{sim}(\mathbf{u},\mathbf{u}') f(u',s)$$

$$sim(\mathbf{u},\mathbf{u}') = \frac{\mathbf{u} \cdot \mathbf{u}'}{\|\mathbf{u}\| \|\mathbf{u}'\|}$$

  - Other aggregation functions possible

Neighborhood-Based CF Methods

- Item-item neighborhood-based CF



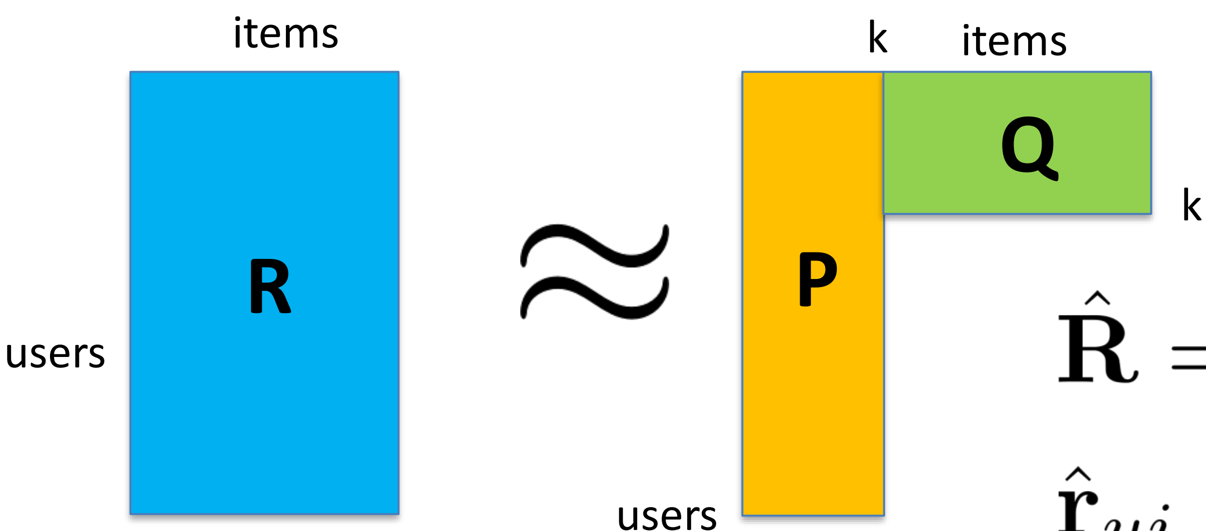
Neighborhood-Based CF Methods

- Item-item neighborhood-based CF
  - Find **K-nearest neighbor items** according to ratings
    - represent **items** by their **columns of the ratings matrix**
    - Only consider *items rated by user u in question*
  - Aggregate their ratings for an item to predict rating

$$f(u,s) = \frac{1}{\sum_{s' \in neighbors(s)} \mathrm{sim}(\mathbf{s},\mathbf{s}')} \sum_{s' \in neighbors(s)} \mathrm{sim}(\mathbf{s},\mathbf{s}') f(u,s') \qquad \mathrm{sim}(\mathbf{s},\mathbf{s}') = \frac{\mathbf{s} \cdot \mathbf{s}'}{\|\mathbf{s}\| \|\mathbf{s}'\|}$$

Matrix Factorization CF (Latent Factor Models)

- Use a **low-rank factorization model** to impute the unobserved ratings
- This represents users and items with vector-valued representations: "**latent factors**"



$$\hat{\mathbf{R}} = \mathbf{PQ}$$

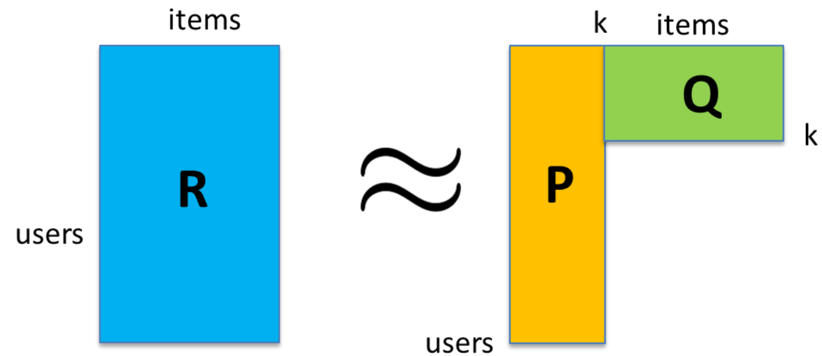
$$\hat{\mathbf{r}}_{ui} = \mathbf{p}_u \cdot \mathbf{q}_i$$

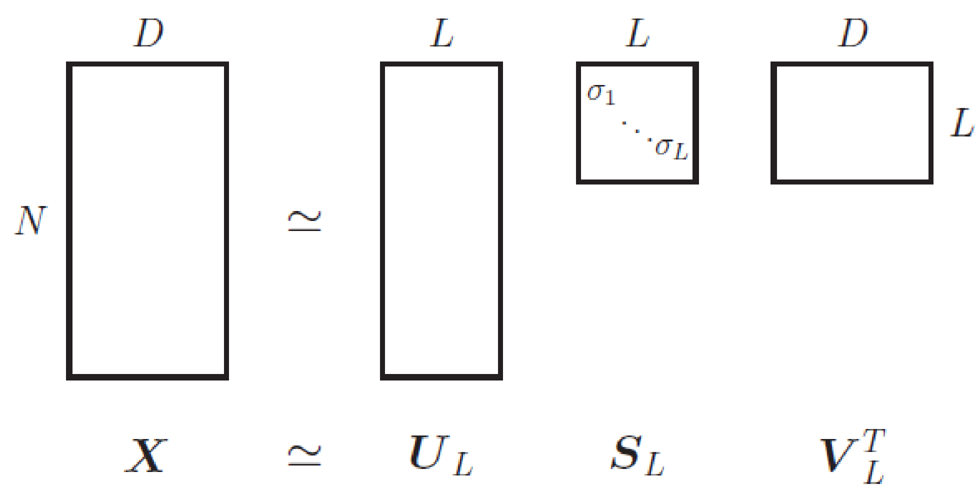
Connection to PCA

- Closely related to singular value decomposition (SVD) used for PCA

- Key difference: Most entries of **R** are missing

- Standard factorization algorithms won’t apply





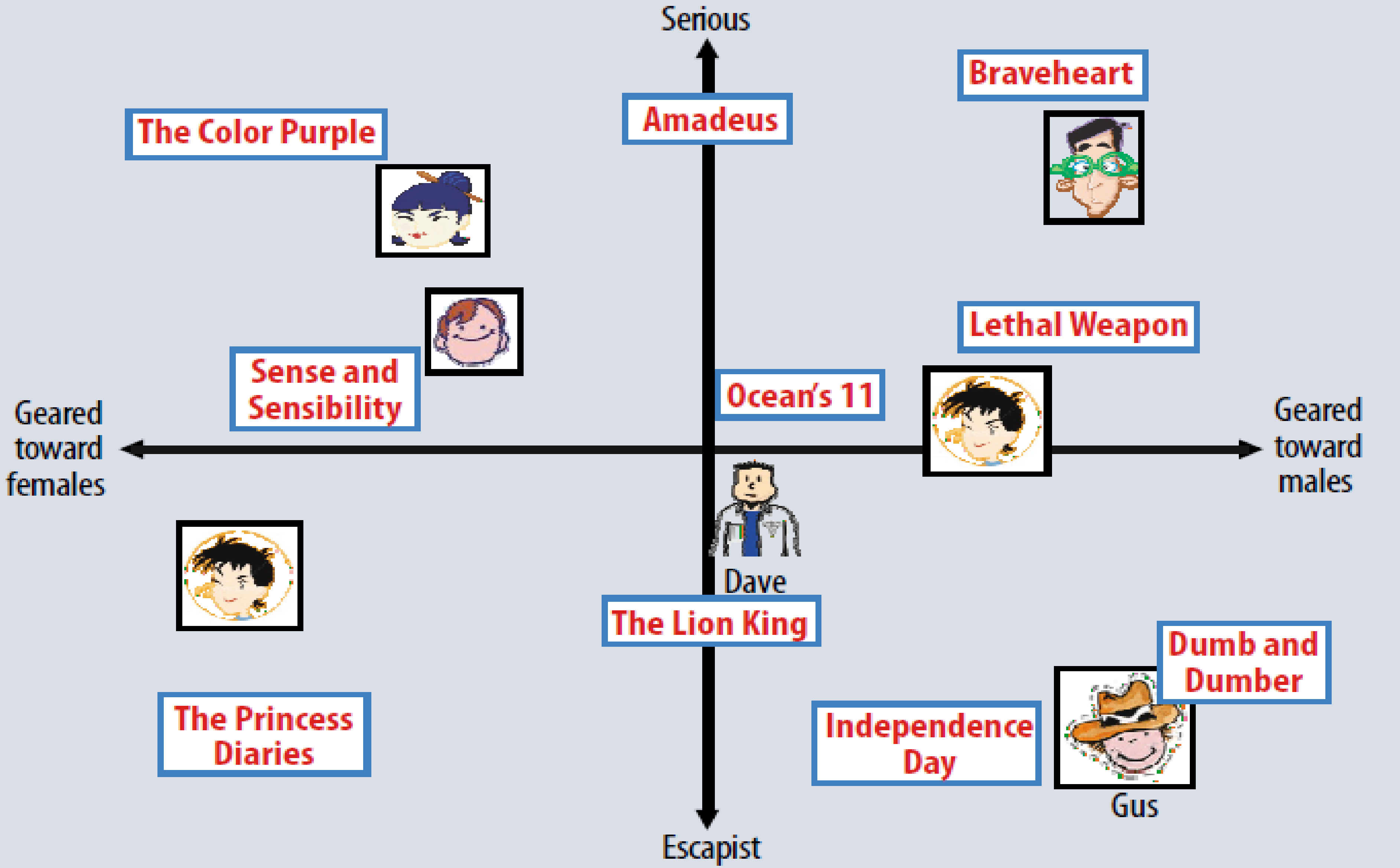


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Figure due to Koren, Y., Bell, R., & Volinsky, C. (2009).

Consider the following objective function for matrix factorization CF:

$$obj(\quad,\quad) = \sum_{(u,i)\in training}(r_{ui} - \;_u \cdot \;_i)^2$$

Should we minimize, or maximize this objective?

Minimize

Maximize

Matrix Factorization CF

- Objective function: regularized squared error

$$\text{obj}(\mathbf{Q}, \mathbf{P}) = \sum_{(u,i) \in \text{training}} (r_{ui} - \mathbf{p}_u \cdot \mathbf{q}_i)^2 + \lambda\Big(\sum_i \|\mathbf{q}_i\|^2 + \sum_u \|\mathbf{p}_u\|^2\Big)$$

Squared error

Regularization penalty

Controls trade-off between squared error and regularization penalty

Generalizes SVD to matrices with missing entries. SVD also minimizes squared error, but assumes all entries of matrix are known.

Learning algorithms: Stochastic Gradient Descent

- This approach was popularized for CF by a **blog post**!

- By Simon Funk (a pseudonym)
- This blog post is still *recommended reading*

- Author was 3rd on Netflix Prize leaderboard.
  He made a big impact by sharing his methods

Monday, December 11, 2006

Netflix Update: Try This at Home

[Followup to this]

Ok, so here's where I tell all about how I (now we) got to be tied for third place on the netflix prize. And I don't mean a sordid tale of computing in the jungle, but rather the actual math and methods. So yes, after reading this post, you too should be able to rank in the top ten or so.

Ur... yesterday's top ten anyway.

http://sifter.org/~simon/journal/20061211.html

Learning algorithms: Stochastic Gradient Descent

- Compute the **gradient** of the error with respect to **one rating**
  - take a step downhill (opposite direction of the gradient).
  - Loop over all ratings in the training set, and repeat.

- Prediction error: $$e_{ui} \stackrel{def}{=} r_{ui} - q_i^T p_u$$

- SGD updates: $$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

SGD: Pros and Cons

- **Pros:**
  - Simple to implement
  - Fast execution time

- **Cons:**
  - Does not exploit parallelism
  - Slower for *dense matrices*, e.g. implicit feedback (a randomized strategy could mitigate this)

Learning Algorithms: Alternating Least-Squares

- Until converged:
  - Fix **P**, solve for **Q**
  - Fix **Q**, solve for **P**
- These are **least squares** problems similar to linear regression. Can be solved in closed form (requires a matrix inverse for each value)

- Each iteration can be **performed in parallel**. Useful when *target matrix is dense*, e.g. with implicit feedback

Think-Pair-Share: Which Algorithm Would You Use? SGD or ALS?

1. You are building a recommender system for Amazon.com which recommends products leveraging not only ratings, but also which products were viewed or mouse-overed

2. You aim to recommend high-end jewelery items based on feedback after purchases. You only have one server machine

Including Bias Terms

- Some items are more popular than others
- Some users have more stingy or lenient ratings than others
- We can include bias terms in the model, and modify the MF objective function:

Predicted ratings: $$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

Obj = $$\sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

Including Implicit Feedback

- Ratings are **explicit feedback** – the user explicitly told us their preference. We may not persuade them to give us many of these

- **Implicit feedback** may be easier to obtain
  - User browsed a certain item, spent x minutes there
  - Mouse movements, clicked "continue reading"
  - Purchases…
  - Usually represented as a dense binary matrix

Including Implicit Feedback

- Modify ratings model:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T [p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a]$$

Scaling factor

Items user showed preference for

Demographic information

Item factors for implicit feedback

Temporal Dynamics

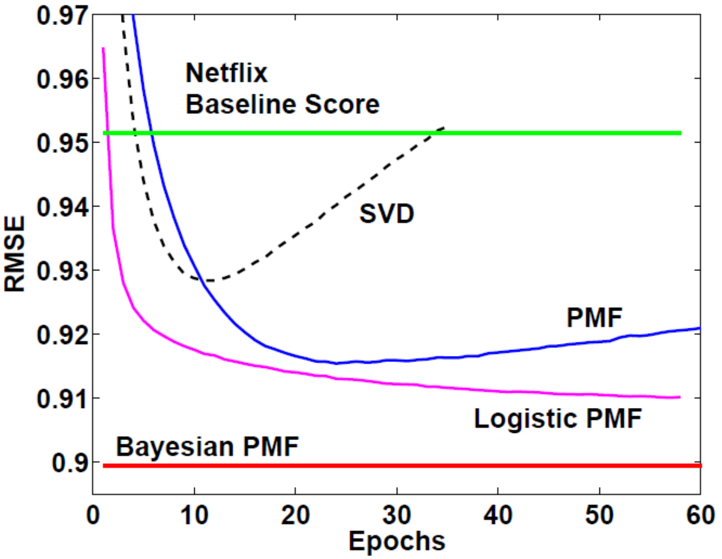
- Item popularity, user biases, user factors (taste profiles) change over time

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

Value at timestep t

Probabilistic Matrix Factorization

- Gaussian probability model on the ratings
  - Leads to the familiar squared error terms
- Gaussian prior probabilities on parameters
- Train via statistical inference (MAP estimation)



- Bayesian probabilistic matrix factorization manages uncertainty due to cold start, has excellent performance

Mnih, Andriy, and Ruslan R. Salakhutdinov. "Probabilistic matrix factorization." *Advances in neural information processing systems*. 2008.

Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

Think-Pair-Share: Recommending Textbooks on Amazon

- Suppose you work for Amazon, and are designing a recommender system specifically for **textbooks**. You will serve all of their customers who are interested in purchasing such books (**millions of users** and **hundreds of thousands of items**), and will have access to all of their computational resources and data.
  - What **methods** and **algorithms** will you use?
  - How will you **scale up** to this scenario?
  - You have access to **ratings**, **content**, **temporal information**, and many kinds of **implicit feedback**. How will you leverage these?