Intro to Data Science:

Recommendation Systems

Alice Li Engineer, Yipit

Agenda

I. Content-based filtering

II. Collaborative filtering

III. A simple matrix factorization model

IV. The Netflix prize

Excercise: Recsys in python

Recommendation systems

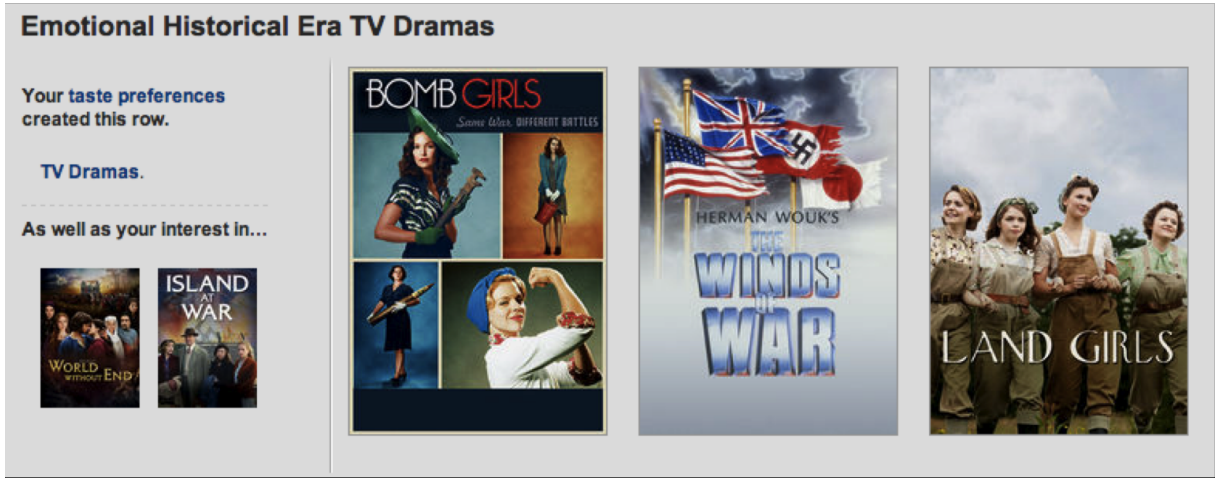
- The purpose of a recommendation system is to predict a rating that a user will give an item that they have not yet rated.
- This rating is produced by analyzing either item characteristics or other user/item ratings (or both) to provide personalized recommendations to users.

Two general approaches

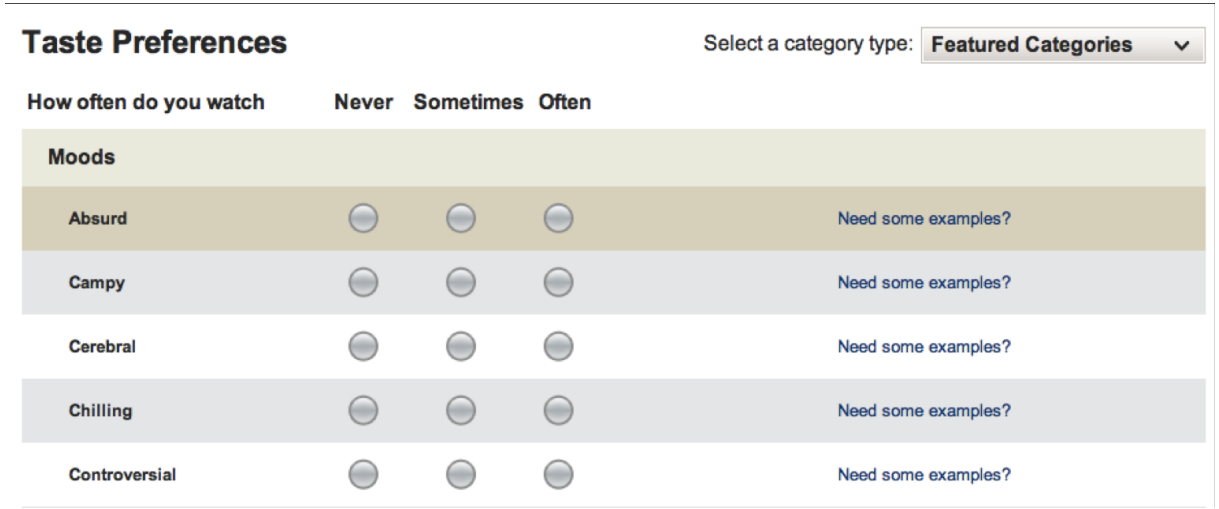
- Content-based filtering: items are mapped into a feature space, and recommendations depend on item characteristics
- Collaborative filtering: recommendations are based only on user-item ratings

Examples

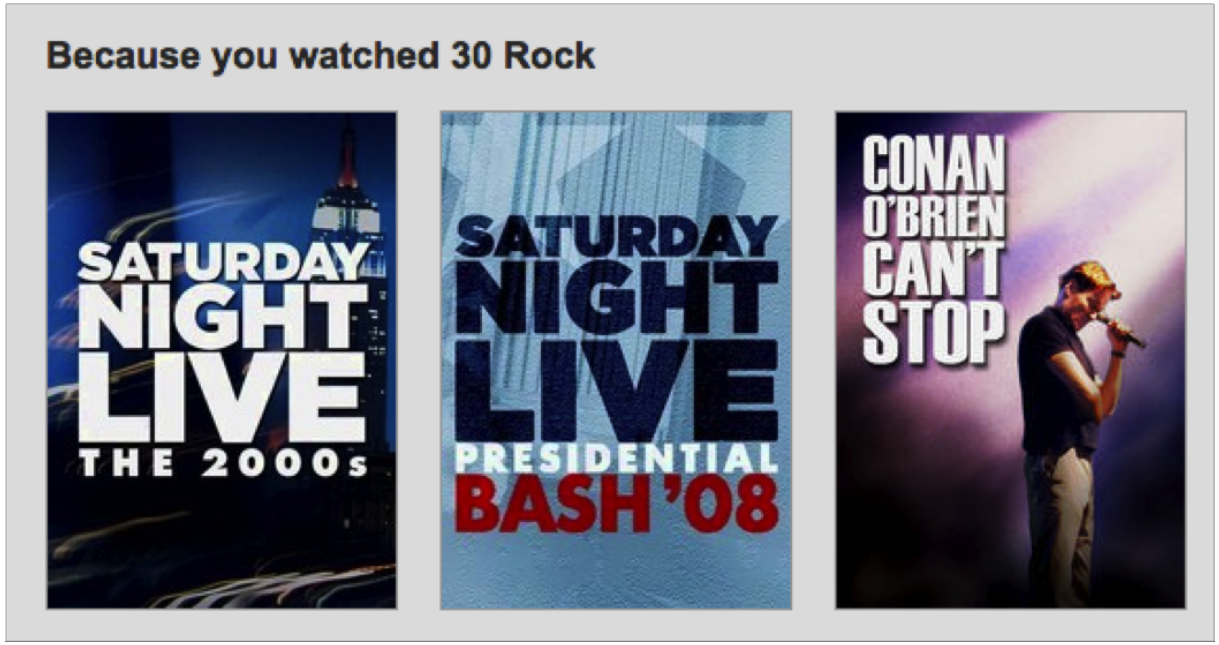
Example: Netflix



Example: Netflix (Content-based filtering)

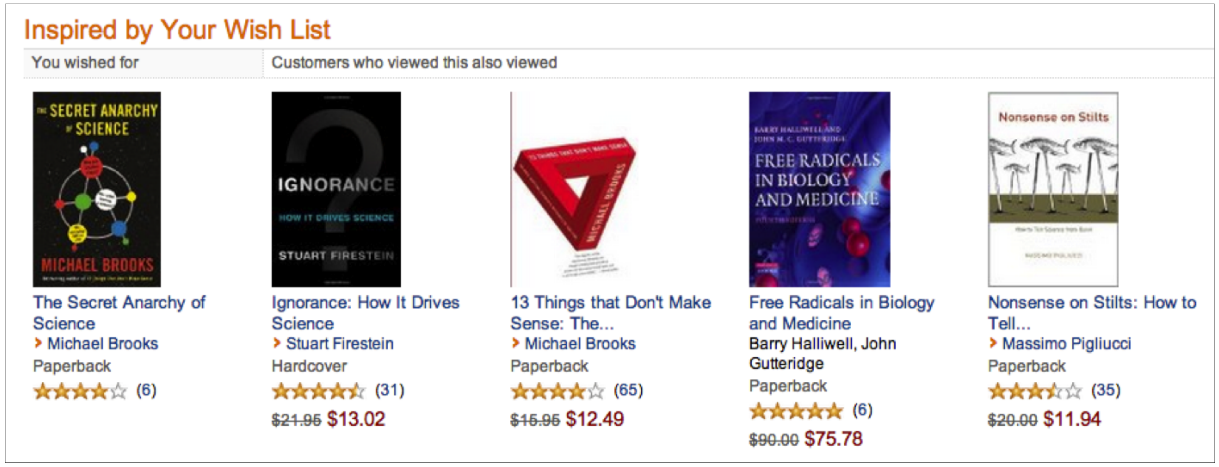


Example: Netflix (Item-based collaborative filtering)



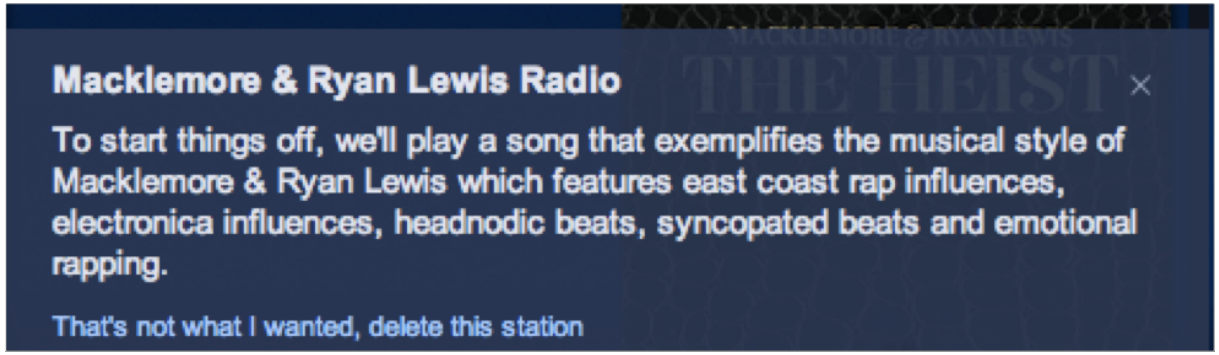
Example: Netflix (User-based collaborative filtering)

Example: Amazon (Item-based collaborative filtering)



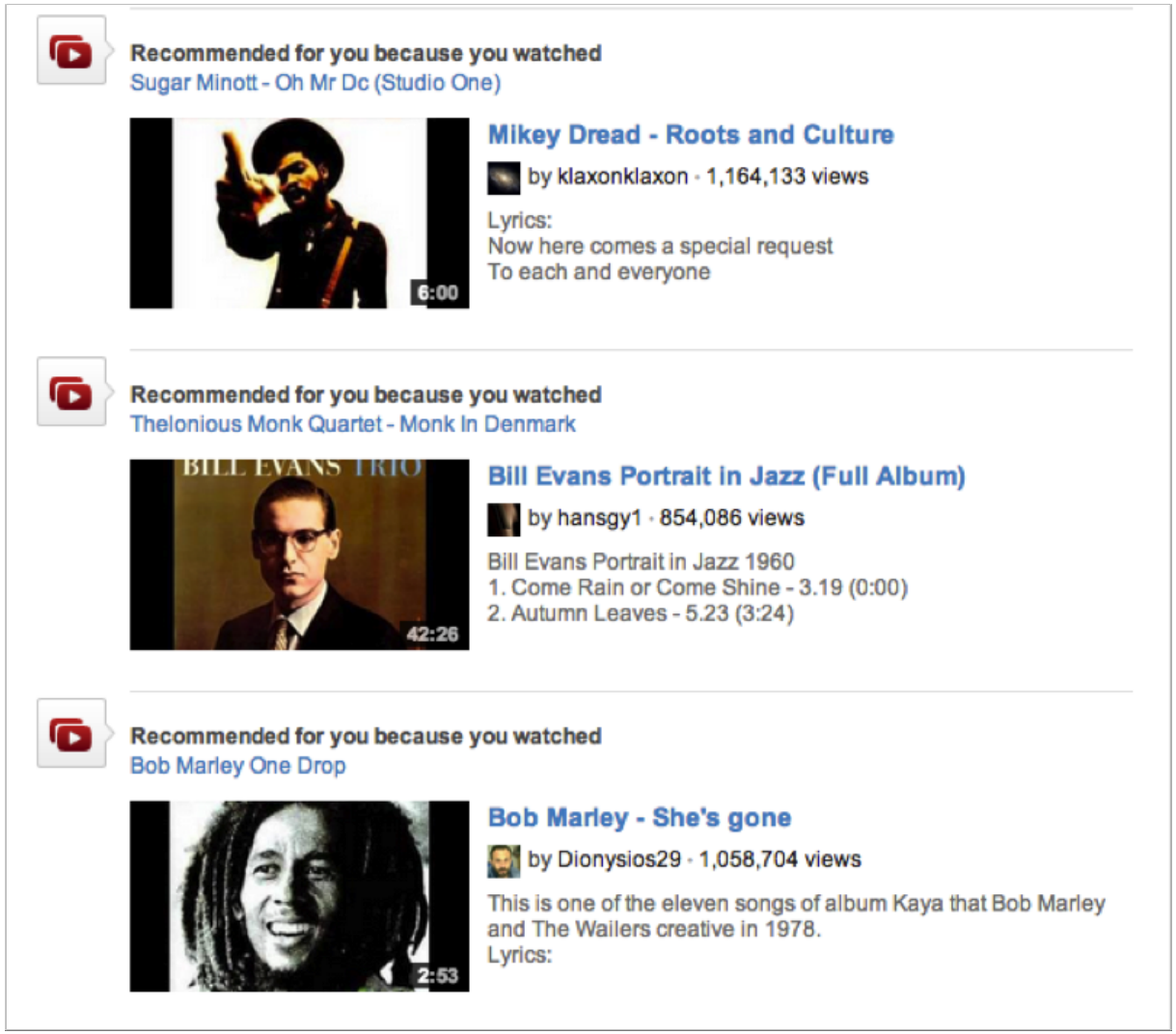
Example: Pandora

Content-based



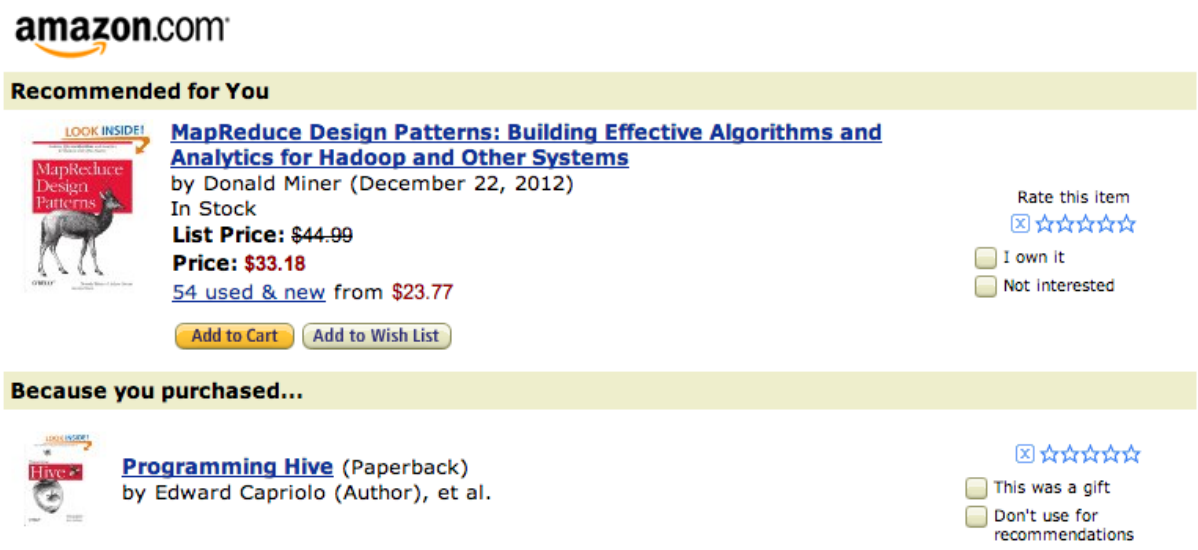
Example: Youtube

Item-based collaborative filtering



Example: Amazon

Item-based collaborative filtering



Example: NY Times

MOST E-MAILED | RECOMMENDED FOR YOU

1. **How Big Data Is Playing Recruiter for Specialized Workers**
2. SLIPSTREAM
   When Your Data Wanders to Places You've Never Been
3. MOTHERLODE
   The Play Date Gun Debate
4. **For Indonesian Atheists, a Community of Support Amid Constant Fear**
5. **Justice Breyer Has Shoulder Surgery**
6. BILL KELLER
   Erasing History

Example: NY Times

Content-based

8. How do you determine my Most Read Topics? Back to top ▲

Each NYTimes.com article is assigned topic tags that reflect the content of the article. As you read articles, we use these tags to determine your most-read topics.

To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

Content-based filtering

- Content-based filtering begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.
- Item vectors measure the degree to which the item is described by each feature, and user vectors measure a user's preferences for each feature.
- Ratings are generated by taking dot products of user and item vectors.

Example: Content-based filtering

Movie feature matrix

Movie	Big box office	Aimed at kids	Famous actors
Finding Nemo	5	5	2
Mission Impossible	3	-5	5
Jiro Dreams of Sushi	-4	-5	-5

User matrix

User	Big box office	Aimed at kids	Famous actors
Alice	-3	2	-2
Bob	5	-4	5

Example: Content-based filtering

Movie feature matrix x user vector

Movie	Big box office	Aimed at kids	Famous actors	Total score
Finding Nemo	5 x -3	5 x 2	2 x -2	-9
Mission Impossible	3 x -3	-5 x 2	5 x -2	-29
Jiro Dreams of Sushi	-4 x -3	-5 x 2	-5 x -2	12

Pandora radio exercise

- What type of recommendation system is **Pandora**?
- How might the Pandora recommendation engine work?

Content-based filtering: Challenges

- Need to map each item into a feature space (manual process)
- Recommendations are limited in scope (items can only be recommended if it has one of the known features)
- Hard to create cross-content recommendations (would require comparing elements from different feature spaces)

Collaborative filtering

Collaborative filtering

- Instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.
- Our dataset is a ratings matrix of items and users.
- Assumption: users get value from recommendations based on other users with similar tastes

Ratings matrix (i.e the data)

18,000 movies

480,000 users

x	1	1	x	...	x
x	x	x	5	...	x
x	x	3	x	...	x
x	4	3	x	...	2
...	x	x	x	...	x
x	5	x	1	...	x
x	x	3	3	...	x
x	1	x	x	...	2

Collaborative filtering

- **Memory-based**: uses a sample of the users that are most similar to a given user to predict ratings on unrated items.
- **Model-based**: extracts complex patterns from the dataset, and uses that as a "model" to make recommendations without having to use the dataset every time

Memory-based collaborative filtering

- For a given user (u_i), find users that are most similar (e.g. **vector similarity** or **Pearson correlation coefficient**)
- For a given item i_j, the predicted rating of u_i is the average of the known ratings of i_j within the group of similar users.

Memory-based CF: User- vs Item-based

- We just talked about user-based CF, but the same process can be done for item-based CF.
- Item-based CF is more commonly used that user-based CF, why do you think that is?

Model-based CF

- Model-based collaborative filtering abandons the neighbor approach and applies other techniques to the ratings matrix.
- The most popular model-based CF techniques use matrix decomposition techniques to find deeper structure in the ratings data.
- For example, we could decompose the ratings matrix via SVD to reduce the dimensionality and extract latent variables.

Model-based CF Example

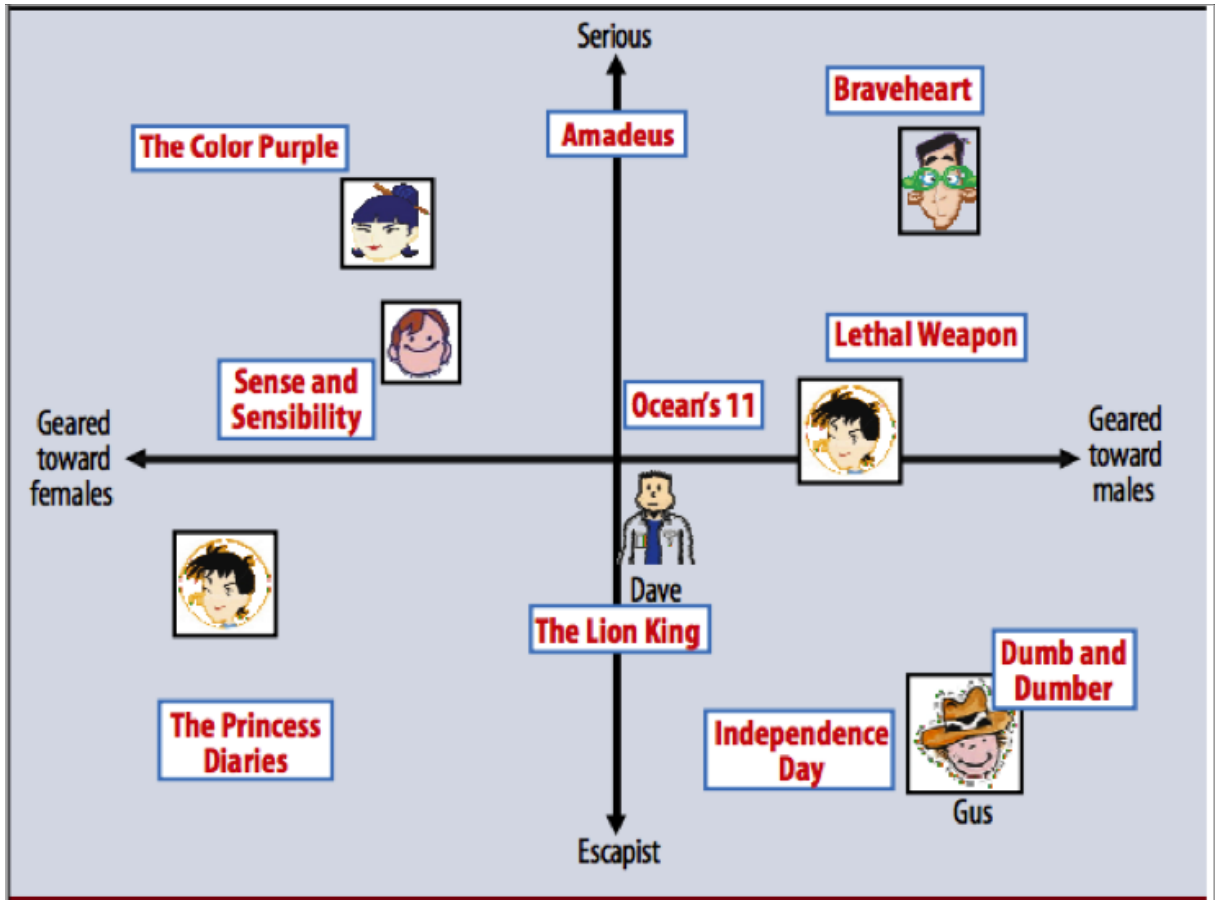


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Model-based CF: Recap

- Once we identify the latent variables in the ratings matrix, we can express both users and items in terms of these latent variables.
- As before, values in the item vectors represent the degree to which an item exhibits a given feature, and values in the user vectors represent user preferences for a given feature.
- Ratings are constructed by taking dot products of user and item vectors in the latent feature space.

Model-based CF: Questions

- What does this model remind you of?
- How is it different from content-based filtering?

Model-based CF: Pros + Cons

- **Pro**: Scalable to many items and users because the model is much smaller
- **Pro**: Can help reduce overfitting
- **Con**: Model can quickly become outdated and is expensive to build
- **Con**: Dimentionality reduction can reduce useful information

Problems with Collaborative filtering

Cold-start problem

We can only recommend items that have already been rated

Solutions

- Enhance our recommendations using implicit feedback
- Use a hybrid content-based and collaborative filtering method

Cold-start problem

Implicit vs explicit data

- Explicit feedback is higher quality but sparse
- Implicit feedback is less accurate but more dense (and less invasive to collect)
- Implicit feedback includes browsing behavior, search history, and purchase behavior

Hybrid methods

Hybrid filtering methods provide another way to get around the cold start problem by combining filtering methods (eg, by using content-based info to "boost" a collaborative model).

Examples

- Create fake users based on content filters. Each new item is rated by *filter-bots* based on a set of features. see **Grouplens**
- An item is recommended to a user when (1) the item scored highly using a content-based method and (2) it has been rated highly by a similar user. see **Fab**

Class break

Review

- What are the two main types of recommendation systems?
- Name one pro and con of each.
- What is the "cold-start" problem? What are two solutions?

A Simple Matrix Factorization Model

Matrix Factorization

- Matrix factorization decomposes the ratings matrix and maps users and items into a low-dimensional vector space spanned by a basis of latent factors.
- Predicted ratings are given by inner products in this space, so for user u and item i we can write:

$$\hat{r}_{ui} = q_i^T p_u$$

Matrix Factorization

- Factoring the ratings matrix via SVD leads to difficulty, since the matrix is typically sparse and therefore our information about the data is incomplete.
- Interpolating missing values is an expensive process and can lead to inaccurate predictions, so we need another way to perform this factorization.

Matrix Factorization

- One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings matrix, we have to be careful to avoid overfitting.
- We can learn these feature vectors by minimizing the loss function, where k denotes the set of known ratings, and l is a hyperparameter.

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

Matrix Factorization

- It turns out that much of the variation in observed ratings is due to user or item biases (eg, some users are very critical, or some items are universally popular).
- We can capture these biases in our model by generalizing this equation, where μ is a global average rating, b_i is the item bias, b_u is the user bias, and $q_i^T r_u$ is the user-item interaction.

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

Matrix Factorization

- With this generalization, our minimization problem becomes

$$\min_{p^*,q^*,b^*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u{}^T q_i)^2 + \lambda$$

$$(\| p_u \|^2 + \| q_i \|^2 + b_u{}^2 + b_i{}^2)$$

- Further modifications can be made to this model (incorporating implicit feedback, capturing temporal effects, attaching confidence scores to predictions), and you can look up the details in the references.

Recommendation system in python

- **Objective**: Implement an item-item collaborative filtering recommender system using Python
- **Tools**: python

The Netflix Prize

The Netflix Prize

- The Netflix prize was a competition to see if anyone could make a 10% improvement to Netflix's recommendation system (accuracy measured by RMSE).
- The grand prize was $1mm dollars, with annual $50k progress prizes to the leader at the end of each year if the 10% threshold had not yet been met. Approx 50k teams participated from >180 countries.
- The ratings matrix contained >100mm numerical entries (1-5 stars) from ~500k users across ~17k movies. The data was split into train/quiz/test sets to prevent overfitting on the test data by answer submission (this was a clever idea!)

The Netflix Prize

- The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.
- Ultimately, the competition ended in a photo finish. The winning strategy came down to last-minute team mergers & creative blending schemes to shave 3rd & 4th decimals off RMSE (concerns that would not be important in practice).
- The competition did much to spur interest and research advances in recsys technology, and the prize money was donated to charity.

The Netflix Prize

Though they adopted some of the modeling techniques that emerged from the competition, Netflix never actually implemented the prizewinning solution.

Why do you think that’s true?

Group Discussion

Questions:

What are the pros and cons for content-based and collaborative filtering techniques?

Think of a recommendation problem that you have encountered. What recsys would you use and why?

Which system is more susceptible to attack? How might a malicious party implement an attack?

What are some practical considerations when using a recommendation system? from both a feasibility and user-experience perspective?

Other resources

- **Survey of recommender systems**
- **Another overview of recommender systems**
- **Great overview of CF techniques**
- **More details on matrix decomposition**
- **Description of item-item CF used by Amazon**
- **Shilling attacks - a weakness of CF techniques**
- **Google News Recommendation System**

More on Netflix

- **Winning team**
- **Blog post from Netflix discussing their recsys**
- **Slides from talk given by one of the Netflix prize winners**
- **Official Netflix prize website**
- **Cool visualization from the 2nd place team**