Project 3: Algorithms

STAT GU4243

Applied Data Science

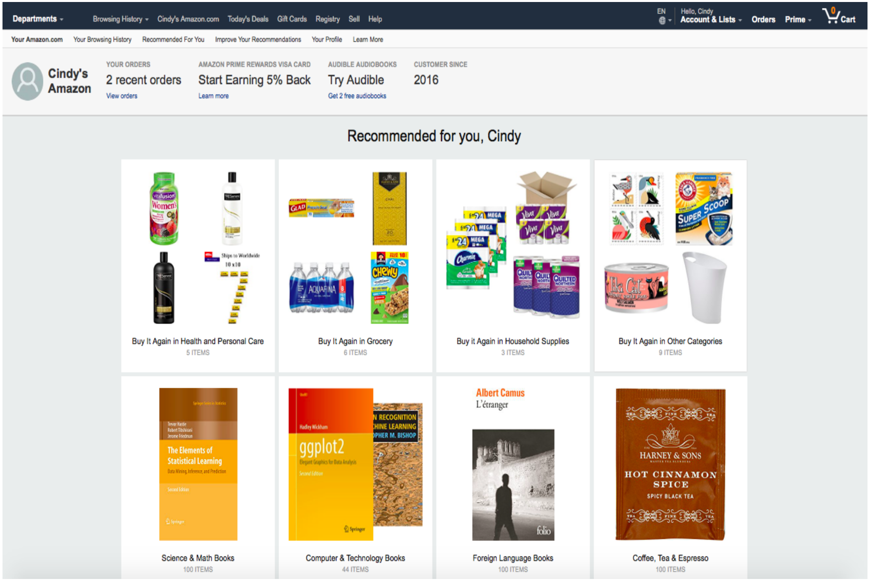
Cynthia Rush
Columbia University

March 19, 2018

Collaborative Filtering

Recommender Systems

- Often rely on recommendations from others via spoken word, reference letters, travel guides, news reports, etc.
- **Recommender systems** assist this natural process by sifting through data (books, webpages, articles, etc.) and automating the process of finding information most interesting and valuable to the user.
- Internet has made such systems necessary – vast amounts of information available.


Departments · Browsing History · Cindy's Amazon.com · Today's Deals · Gift Cards · Registry · Sell · Help · EN · Hello, Cindy Account & Lists · Orders · Prime · 0 Cart

Your Amazon.com · Your Browsing History · Recommended For You · Improve Your Recommendations · Your Profile · Learn More

Cindy's Amazon

YOUR ORDERS: 2 recent orders (View orders)

AMAZON PRIME REWARDS VISA CARD: Start Earning 5% Back (Learn more)

AUDIBLE AUDIOBOOKS: Try Audible (Get 2 free audiobooks)

CUSTOMER SINCE: 2016

Recommended for you, Cindy

Buy It Again in Health and Personal Care — 5 ITEMS

Buy It Again in Grocery — 6 ITEMS

Buy it Again in Household Supplies — 3 ITEMS

Buy It Again in Other Categories — 9 ITEMS

Science & Math Books — 100 ITEMS

Computer & Technology Books — 44 ITEMS

Foreign Language Books — 100 ITEMS

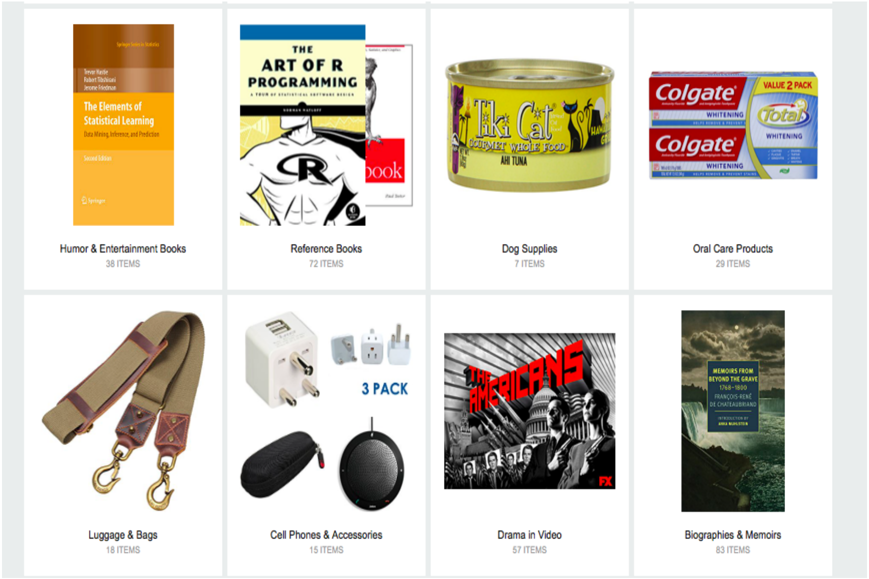
Coffee, Tea & Espresso — 100 ITEMS

Recommender Systems

Often classified into two categories:

1. **Content-based**: Utilize contents of items and find similarities among them. Users are given a profile about items that interest them and then similar items are recommended.
    - Example: Want to recommend articles to a specific reader.
    - Such a system might analyze the contents of all books the reader has ever read, summarizing them by key words, and then representing the interests of the reader by frequent keywords.
    - Recommendations made for books matching the reader's 'profile' keywords.

2. **Collaborative filtering**: Relies only on item ratings from all users. Assumption that users who have rated same items with similar ratings are likely to have similar preferences.
    - Example: Finds other users with similar article tastes, and recommends what they have read.

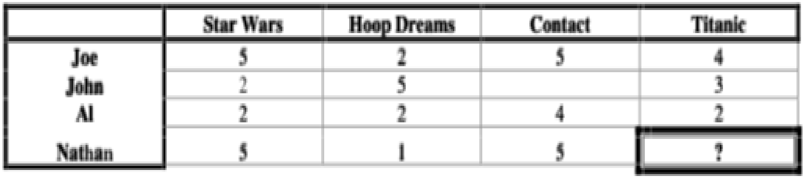

Humor & Entertainment Books
38 ITEMS
Reference Books
72 ITEMS
Dog Supplies
7 ITEMS
Oral Care Products
29 ITEMS
Luggage & Bags
18 ITEMS
Cell Phones & Accessories
15 ITEMS
Drama in Video
57 ITEMS
Biographies & Memoirs
83 ITEMS

Collaborative Filtering Set-up

Collaborative filtering refers to the process of making automatic predictions (filtering) about the preferences of a new user from a database containing the preferences from many users (collaborating).

1. m users $\{u_1, \ldots, u_m\}$ and n items $\{i_1, \ldots, i_n\}$.
2. Each user u_i has rated a list of items.
3. Ratings have two types:
    - **Explicit**. E.g. ratings on a 1-5 scale.
    - **Implicit**. E.g. preferences determined based on purchase history.

User-item Matrix



	Star Wars	Hoop Dreams	Contact	Titanic
Joe	5	2	5	4
John	2	5		3
Al	2	2	4	2
Nathan	5	1	5	?

Figure 1: Collaborative filtering can be represented as the problem of predicting missing values in a user-item matrix. This is an example of a user-item rating matrix where each filled cell represents a user's rating for an item. The prediction engine is attempting to provide Nathan a prediction for the movie 'Titanic'.

Netflix Prize

- Open competition for best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films.
- Grand prize of one million dollars was given to team of Austrain scientists and scientists from ATT Bell Labs. Better than Netflix's own algorithm by 10.06%.
- Training data of 100,480,507 ratings (1-5 scale) that 480,189 users gave to 17,770 movies.
- (Our training data: 809,526 ratings that 5,055 users rating 1,619 movies.)

Collaborative Filtering Challenges

- Often the data is sparse – user-item matrix large, with few rated values.
- Difficult for new users or items, i.e. **cold start** problem.
- Algorithm must scale well with increasing number of items and users: e.g. tens of millions of users and millions of items.
- Predictions need to be made rapidly.
- Difficuty with **synonyms** – when the same or similar items have different names or entries.
- Want algorithms to increase diversity in the sense that they help users discover new items.

Types of Collaborative Filtering Algorithms

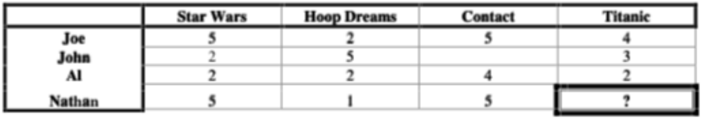
Memory-based algorithms use the entire user-item database to generate predictions that are based on the **active user's** neighbors (those other users with similar interests).

Model-based algorithms develop and learn models to fit the data and then used the learned models to make predictions.

Basic Memory-Based CF

- Early gen algorithms calculate similarity weight between users or items, and make predictions with these weights.
- Basic idea:
  - Calculate weight w_{ij} betweens users u_i and u_j representing the distance or correlation between two-users.
  - Produce a prediction for the active user by taking a weighted average of all ratings by other users on the item of interest.
- Figuring out a good similarity weight is crucial for memory-based CF algorithms.
- Sometimes referred to as **neighborhood methods**, since a subset of users are chosen based on similarity with the active user and predictions use weighted average of neighbors votes.

Example: Correlation-based Similarity



	Star Wars	Hoop Dreams	Contact	Titanic
Joe	5	2	5	4
John	2	5		3
Al	2	2	4	2
Nathan	5	1	5	?

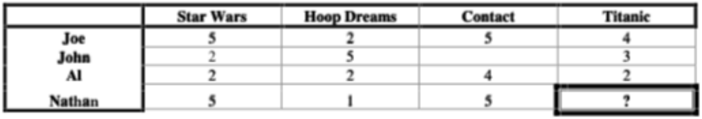
Let $r_{\text{u,m}}$ be the rating by user 'u' for movie 'm' and $\bar{r}_{\text{u}}$ be user u's average rating.

$$\hat{r}_{\text{Nathan,Titanic}} = \bar{r}_{\text{Nathan}} + \frac{\sum_{\text{u}\in\{\text{Joe, John, Al}\}}(r_{\text{u,Titanic}} - \bar{r}_{\text{u}}) \times w_{\text{u,Nathan}}}{\sum_{\text{u}\in\{\text{Joe, John, Al}\}} w_{\text{u,Nathan}}},$$

where $w_{\text{u,v}}$ is the Pearson correlation coefficient

$$w_{\text{u,v}} = \frac{\sum_{\text{m}\in\text{movies}}(r_{\text{u,m}} - \bar{r}_{\text{u}}) \times (r_{\text{v,m}} - \bar{r}_{\text{v}})}{\sqrt{\sum_{\text{m}\in\text{movies}}(r_{\text{u,m}} - \bar{r}_{\text{u}})^2 \sum_{\text{m}\in\text{movies}}(r_{\text{v,m}} - \bar{r}_{\text{v}})^2}}.$$

Example: Correlation-based Similarity



	Star Wars	Hoop Dreams	Contact	Titanic
Joe	5	2	5	4
John	2	5		3
Al	2	2	4	2
Nathan	5	1	5	?

$$
\begin{aligned}
&\hat{r}_{\text{Nathan,Titanic}} \\
&= 3.66 + \frac{(4-4)w_{\text{Joe,Nathan}} + (3-3.5)w_{\text{John,Nathan}} + (2-2.67)w_{\text{Al,Nathan}}}{w_{\text{Joe,Nathan}} + w_{\text{John,Nathan}} + w_{\text{Al,Nathan}}} \\
&= 3.66 + \frac{(4-4)(1) + (3-3.5)(-1) + (2-2.67)(0.5)}{1-1+0.5} \\
&= 4.
\end{aligned}
$$

where, for example, we calculate $w_{\text{John,Nathan}}$ as

$$
w_{\text{John,Nathan}} = \frac{(5-3)\times(2-3.5) + (1-3)\times(5-3.5)}{\sqrt{((5-3)^2 + (1-3)^2)\times((2-3.5)^2 + (5-3.5)^2)}} = -1.
$$

Example: Correlation-based Similarity

```
UI <- matrix(c(5, 2, 5,4, 2, 5, NA, 3, 2, 2, 4, 2, 5, 1, 5, NA),
             nrow = 4, ncol = 4, byrow = TRUE)

rownames(UI) <- c("Joe", "John", "Al", "Nathan")
colnames(UI) <- c("Star Wars", "Hoop Dreams", "Contact", "Titanic")

w_JoeNat  <- cor(UI["Joe", ], UI["Nathan", ], use = "complete.obs")
w_JohnNat <- cor(UI["John", ], UI["Nathan", ], use = "complete.obs")
w_AlNat   <- cor(UI["Al", ], UI["Nathan", ], use = "complete.obs")

movies <- c("Star Wars", "Hoop Dreams", "Contact")

muJoe  <- mean(UI["Joe", movies], na.rm = TRUE)
muJohn <- mean(UI["John", movies], na.rm = TRUE)
muAl   <- mean(UI["Al", movies], na.rm = TRUE)
muNat  <- mean(UI["Nathan", movies], na.rm = TRUE)
```

Example: Correlation-based Similarity

```
numerator <- (UI["Joe", "Titanic"] - muJoe)*w_JoeNat
             + (UI["John", "Titanic"] - muJohn)*w_JohnNat
             + (UI["Al", "Titanic"] - muAl)*w_AlNat

est <- muNat + (numerator) / (w_JoeNat + w_JohnNat + w_AlNat)
```

The Project

Project Description

In this project, working in teams, you will evaluate and compare a pair of collaborative filtering algorithms.

Each team will:

1. implement, from scratch, one model-based and the other memory-based algorithm from the collaborative filtering literature,
2. evaluate the algorithms' performance on two datasets,
3. and investigate specific components of the memory-based model.

Submission Details

Submission

- The GitHub repo of your codes.
- A testing report (must be a reproducible R notebook or a similar format) implementing the algorithms.
- A writeup of a side-by-side comparison of the performance and computational efficiency of the algorithms.

Presentations

Focus on group-specfic assignments.

Briefly explain the details of the component your group is assigned to test, how the evaluation is carried out, and the main results.

Submission Details

Evaluation criteria

(9pts) Readabiity and reproducibility of codes (including correct implementation)
(9pts) Validity of evaluation (well-defined measures of performance; experiment set up)
(7pts) Presentation (final report, Github documentation, and in-class presentation)

Schedule

Week 1 [3/19 and 3/21]: Intro and project description. An overview of collaborative filtering and computational aspects.
Week 2 [3/26 and 3/28]: Paper dicussions and an in-depth look at the algorithms.
Week 3 [4/2 and 4/4]: Work week.

Suggested Team Workflow

Week 1 is the **Reading** Week

Read the papers and get familiar with the data. Implement the simple neighborhood model on both datasets.

As a team, brainstorm an evaluation plan and assign tasks. Each team is strongly recommended to demonstrate project progress by posting a project plan with task assignments on GitHub.

Suggested Team Workflow

Week 2 is the **Coding** Week

Make sure you understand the details of the algorithms and start coding. Look towards evaluating the algorithms.

It is ok to separate into two sub-teams, one working on each algorithm, as long as the two teams have the same criteria for evaluating the algorithms. The two sub-teams can also serve as others' validators.

Week 3 is the **Evaluation** Week

By using R Notebook to carry out coding and evaluation, your final report can just be adding explanation and comments to your Notebook.

The Data

The Data

Anonymous Microsoft Web Data

An example of implicit voting data, with each part of the website characterized as being visited (vote of one) or not (no vote) by a user.

EachMovie

An explicit voting example using data, with votes ranging in value from 0 to 5.

The full data is available at the above links, however we will be using a subsample of the data for easier computation that can be downloaded from my Dropbox

Anonymous Microsoft Web Data

- Log of anonymous users of www.microsoft.com.
- Want to predict areas of web site a user will visit based on data on other areas the user visited.
- Data created by sampling and processing www.microsoft.com logs. Data records use of www.microsoft.com by 38000 anonymous, randomly-selected users. For each user, data lists all the areas of the web site (Vroots) that user visited in one week timeframe.

Anonymous Microsoft Web Data

```
> head(train, 15)
   V1    V2    V3
1   C 10010 10010
2   V  1010     1
3   V  1000     1
4   V  1011     1
5   V  1012     1
6   V  1013     1
7   V  1014     1
8   C 10019 10019
9   V  1017     1
10  V  1004     1
11  V  1018     1
12  V  1029     1
13  V  1008     1
14  V  1030     1
15  V  1031     1
```

```
Dataset format:
Each line of the data file starts with a letter
The line types of interest are:
-- Case and Vote Lines:
   For each user, there is a case line followed
   For example:
     C,"10164",10164
     V,1123,1
     V,1009,1
     V,1052,1
   Where:
    'C' marks this as a case line,
    '10164' is the case ID number of a user,
    'V' marks the vote lines for this case,
    '1123', 1009', 1052' are the attributes ID's
                                  user visited.
    '1' may be ignored.
```

Anonymous Microsoft Web Data

Training Data

- 4151 users and 269 unique Vroots.
- Each user visited between 6 and 35 Vroots (average = 8.16, median = 7) with a total of 33,875 Vroots visited (3% of all user/Vroot combinations).

Test Data

- 665 users (a subset of the training users) and 171 unique Vroots (a subset of the training Vroots).
- Idea: You predict all missing user/Vroot pairs in the training data and test with data available in the test dataset.
- Each test user visited between 1 and 17 Vroots (average = 3.08, median = 2) with a total of 2.053 Vroots visited.

EachMovie Data

- 61,265 users entered a total of 2,811,983 numeric ratings on 1,623 movies, i.e. about 2.4% entries are rated by zero-to-five stars.
- User indices range from 1 to 74,424.
- Movie indices range from 1 to 1,648.

```
> round(table(train$Score)/nrow(train), 2)

   1    2    3    4    5    6
0.16 0.06 0.13 0.25 0.25 0.15
```

Anonymous Microsoft Web Data

```
> head(train, 15)
   Movie User Score
1      1    1     4
2      2    1     4
3     17    1     5
4     18    1     2
5     25    1     5
6     31    1     2
7     32    1     4
8     34    1     5
9     36    1     3
10    39    1     5
11    41    1     4
12    43    1     4
13    44    1     1
14    47    1     3
15    50    1     4
```

```
> tail(train, 15)
       Movie  User Score
809512  1247 74418     5
809513  1258 74418     5
809514  1276 74418     6
809515  1288 74418     6
809516  1333 74418     5
809517  1357 74418     4
809518  1358 74418     5
809519  1363 74418     1
809520  1367 74418     1
809521  1391 74418     2
809522  1393 74418     3
809523  1405 74418     4
809524  1407 74418     3
809525  1416 74418     3
809526  1476 74418     1
```

Anonymous Microsoft Web Data

Training Data

- 809,526 ratings from 5,055 users on 1,619 movies. Around 10% of all possible user/movie pairs.
- Users rate between 97 and 1,164 movies (average = 160.14, median = 137).

Test Data

- 199,876 ratings from 5,055 users (same users from training data) rating 1,597 movies (a subset of the training movies).
- Idea: You predict all missing user/movies ratings in the training data and test with data available in the test dataset.
- Test users rate between 24 and 291 movies (average = 39.54, median = 34).

The Algorithms

Project Tasks

Each has team has two tasks:

Task 1: **Implement** and **evaluate** the performance of two collaborative filtering algorithms – one model-based and the other memory-based – on both datasets.

- [**Model 1**] Model-based algorithm: clustering discussed in paper [1] section 2.3.
- [**Model 2**] Memory-based algorithm: user-based neighborhood model using Pearson's correlation for similarity weight. This is introduced in equations (1) and equations (2) in paper [2].

Project Tasks

- **[Model 1]** Model-based algorithm: clustering discussed in paper [1] section 2.3.
  - Want to estimate $r_{\mathrm{u,i}}$ (vote for user u on item i) with $\mathbb{E}[r_{\mathrm{u,i}}]$.
  - Use a Bayesian classifier where the probability of votes are conditionally independent given membership in an unobserved class with relatively few possible values.
  - Class variable represents groups or types of users with a common preferences or tastes.
  - Given the class, votes on various items are independent. i.e. Naive Bayes model.
  - Estimate parameters of model using training data.
  - Since we don't observe class variables, must use methods that can learn hidden variables. Use EM algorithm for fixed number of classes.
  - Choose number of classes via cross-validation.
- **[Model 2]** Memory-based algorithm: user-based neighborhood model using Pearson's correlation for similarity weight. This is introduced in equations (1) and equations (2) in paper [2].
  - Essentially just need to extend toy example from earlier in the slides to the entire dataset.

Project Tasks

Each has team has two tasks:

B. Evaluate potential performance enhancements on the baseline momeory-based algorithm (Model 2 above) by considering changes to various components of the algorithm.

- **[All Groups]** Consider different similarity weights: (1) Spearman's correlation, (2) vector similarity, (3) entropy, (4) mean-square difference, and (5) SimRank. Most are discussed in section 5.1 of paper [2]. SimRank is discussed in paper [4].
- **[Groups 1, 2, 3]** Consider significance and variance weighting. Section 5.2 and 5.3 of paper [2].
- **[Groups 4, 5]** Consider selecting neighborhoods. Section 6 of paper [2].
- **[Group 6, 7]** Consider rating normalization. Section 7 of paper [2].

Papers

1. Breese, J. S., Heckerman, D., & Kadie, C. (1998, July). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 43-52). Morgan Kaufmann Publishers Inc.

   This paper provides an introduction to collaborative filtering, and evaluates the data sets we will use. It also includes the details of the model-based algorithm you are asked to implement.

2. Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999, August). An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (pp. 230-237). ACM.

   This paper proposes the framework for the memory-based (neighborhood) models you are asked to analysis. It also contains details about the memory-based model components that each groups are assigned to investigate.

Papers

3. Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 4.

   A review paper of collaborative filtering.

4. Jeh, G., & Widom, J. (2002, July). SimRank: a measure of structural-context similarity. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 538-543). ACM.

   Proposes the SimRank similarity measure.

Model Evaluation

You need to compare the performance of the memory-based and model-based algorithms and also the memory-based model with various component combinations.

To do so you you will need to consider some sort of performance metric to estimate the accuracy of prediction.

Training set generates predictions, test set is used for evaluating predictive accuracy.

Performance Metrics

Effectiveness of collaborative filtering depends on manner in which recommendations will be presented to the user.

Mean Absolute Error

Used when algorithm gives the user a rating indicating potential interest in a topic.

$$MAE = \frac{\sum_{(\mathrm{u,i}) \in \mathrm{test}} |\hat{r}_{\mathrm{u,i}} - r_{\mathrm{u,i}}|}{|\mathrm{test}|},$$

where |test| is the number of ratings in the test set.

Use on movie data.

Ranked Scoring Evaluation

Used when algorithm gives the user an ordered list of recommendations. Goal is to estimate the expected utility of a particular ranked list to a user. Details in paper [1].

Use on website data.