How might my ratings change my recommendations?

Goals of Recommender Systems

- Show content that we’re interested in
- Suggest new content that would interest us
- Suggest new content that is generally popular
- Adjust recommendations based on our feedback

$1M winning algorithm not actually used by Netflix

Researchers were able to de-anonymize data by comparing with IMDB ratings, resulting in a lawsuit.
RecSys Challenge 2018

Welcome ACM RecSys Community! For this year’s challenge, use the Spotify Million Playlist Dataset to help users create and extend their own playlists.

Read on for all the details. Good luck!

The RecSys Challenge 2018 is organized by Spotify, The University of Massachusetts, Amherst, and Johannes Kepler University, Linz.

Have a question, suggestion or concern? Let us know by emailing us at recsyschallenge@spotify.com

Recommender Systems

- What makes two (Amazon) users similar?
  - Purchased the same set of items
  - Liked and disliked the same set of items
- What makes two items similar?
  - The same set of users purchased/liked them
  - Their titles, description, prices, other metadata

Collaborative Filtering

Create a user-item matrix

<table>
<thead>
<tr>
<th></th>
<th>Sohie</th>
<th>Brian</th>
<th>Christine</th>
<th>Orit</th>
<th>Catherine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
**Similarity: Jaccard**

- Measure similarity between a pair of user vectors (or a pair of item vectors)

\[ U_A = [1, 0, 1, 0] \]
\[ U_B = [0, 0, 1, 0] \]

\[ \text{Jaccard}(U_A, U_B) = \frac{|U_A \cap U_B|}{|U_A \cup U_B|} \]

Problem: does not work for non-binary vectors

When is result 0? When is it 1?

**Similarity: Cosine**

- Measure similarity between a pair of user vectors (or a pair of item vectors)

\[ U_A = [0, 5, 2, 0] \]
\[ U_B = [1, 0, 4, 2] \]

\[ \text{CosineSim}(U_A, U_B) = \frac{U_A \cdot U_B}{||U_A|| \ ||U_B||} \]

When is result 0? When is it 1?

**User-Based Collaborative Filtering**

**Task**: predict rating on new user-item entry in matrix: \( U_{A'} \), \( I_P \)

- Among users that have rated \( I_P \), select a set \( S_K \) of the K most similar users to \( U_A \)
- Predicted rating for \( U_{A'}, I_P \) is average rating of \( I_P \) from users in \( S_K \):

\[ R(U_{A'}, I_P) = \frac{\sum_{U_B \in S_K} R(U_B, I_P)}{K} \]

Some users have a tendency to be more or less generous

Use deviation from a user’s average rating, rather than a user’s absolute rating

\[ R(U_{A'}, I_P) = \frac{\sum_{U_B \in S_K} R(U_B, I_P) - \text{mean}(U_B)}{K} \]
Item-Based Collaborative Filtering

**Task:** predict rating on new user-item entry in matrix: $U_A, I_P$

- Among items that have been rated by $U_A$, select a set $S_K$ of the $K$ most similar items to $I_P$.
- Predicted rating for $U_A, I_P$ is average rating of $U_A$ from items in $S_K$:

$$R(U_A, I_P) = \frac{\sum_{I_i \in S_K} R(U_A, I_i)}{K}$$

**Weighted Average**

Compute final score in some class:

<table>
<thead>
<tr>
<th>Score</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class participation</td>
<td>60%</td>
</tr>
<tr>
<td>Homework</td>
<td>95%</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>50%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>87%</td>
</tr>
</tbody>
</table>

Weighted Mean = \[\frac{\sum \text{Weight} \times \text{Score}}{\sum \text{Weight}}\] = \[\frac{50 \times 0.60 + 200 \times 0.95 + 100 \times 0.50 + 150 \times 0.87}{50 + 200 + 100 + 150}\]

**Problems with Collaborative Filtering?**

- If user-item matrix is too sparse, may not be useful
- "Cold-start problem": how to handle new users and items?
- Won’t encourage diverse results (echo chamber effect)
Content-Based Recommendations: Approach 1

- Define similarity between users (or similarity between items) in terms of content features, not rating patterns
  - Examples of item features: restaurant cuisine type, director or actors in movie, product details
  - Examples of user features: demographic information
- Apply same methods as for collaborative filtering

Content-Based Recommendations: Approach 2

- Featurize users and items under the same set of features
  - Features: words
    - user feature values = word counts in reviews
    - item feature values = word counts in descriptions
  - Features: demographics
    - user feature values = demographic info
    - item feature values = target demographics
- Compute similarity between a given user and item

Featurizing Text

- Bag of words: tokenizing, counting, tf-idf weighting

|                     | bland | but | fast | food | good | no | parking | service | but | bland | food | good | no | parking | service | but | bland | food | good | no | parking | service | but | bland | food | good | no | parking | service | but | bland | food | good | no | parking |
|---------------------|-------|-----|------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|-----|-------|------|------|----|---------|---------|
| Fast service but bland food | 1     | 0   | 1    | 1    | 0    | 0  | 0       | 1       | 0   | 0     | 0    | 0    | 0  | 0       | 1       | 0   | 0     | 0    | 0    | 1  | 1       | 1       | 0   | 1     | 0    | 0    | 1  | 1       | 1       |
| Good fast food       | 0     | 0   | 1    | 1    | 0    | 0  | 0       | 0       | 0   | 1     | 0    | 0    | 0  | 0       | 0       | 1   | 1     | 0    | 0    | 1  | 1       | 1       |
| No service, no parking, no good | 0     | 0   | 0    | 0    | 1    | 1  | 3       | 1       | 1   | 1     | 1    | 1    | 1  | 1       | 1       | 1   | 1     | 1    | 1    | 1  | 1       | 1       |

- N-Grams

Evaluation

- **Task Type A:** Given test set of (user, item) pairs, predict ratings
  - Raw accuracy, e.g., percentage of ratings predicted exactly
  - Root mean squared error (RMSE)
    \[
    \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}
    \]
- **Task Type B:** Given test set of users, predict set of items to recommend
  - Precision, Recall, F1 Score
Vectorization (Array Programming)

- Many scientific and numerical computing libraries, such as NumPy in Python, provide **vectorized** operations, i.e., operations that can be applied to an entire array (matrix).

\[
\begin{align*}
np.random.randint(...) & \quad np.median(a) & \quad a[a>10] \\
np.sum(a) & \quad np.mean(a) & \quad np.dot(a,b) \\
a**2 & \quad np.ones(...) & \quad...
\end{align*}
\]

- Whenever possible, it is usually a good idea to use **vectorization** rather than looping through an array and applying an operation to each element.