Recommender Systems
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
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>Ada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie 1</td>
<td>5</td>
<td></td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Movie 2</td>
<td>2</td>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Movie 3</td>
<td></td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Movie 4</td>
<td></td>
<td>4</td>
<td></td>
<td>4</td>
<td></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)
  
  \[
  U_A = [0, 5, 2, 0] \\
  U_B = [1, 0, 4, 2]
  \]

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

**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 \(S_K\):

  \[
  \hat{R}(U_A, I_P) = \frac{\sum_{U_i \in S_K} R(U_i, I_P)}{K}
  \]

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

- 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

  \[
  R(U_A, I_P) = \frac{\sum_{U_i \in S_K} R(U_i, I_P) - \text{mean}(U_i)}{K}
  \]
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 \( S_K \):

\[
R(U_A, I_p) = \frac{\sum_{I_q \in S_K} R(U_A, I_q)}{K}
\]

**Weighted Average**

Mean is 73%

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 is 80%

**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

<table>
<thead>
<tr>
<th>Word</th>
<th>bland</th>
<th>but</th>
<th>fast</th>
<th>food</th>
<th>good</th>
<th>no</th>
<th>parking</th>
<th>service</th>
<th>but</th>
<th>bland</th>
<th>food</th>
<th>good</th>
<th>no</th>
<th>parking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast service</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bland food</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Good fast food</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>TP: Recommended items user actually buys</th>
<th>FP: Recommended items user does not buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TN: Items not recommended and user does not buy</td>
<td>FN: Items not recommended and user buys</td>
</tr>
</tbody>
</table>
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)
  
  ```python
  np.random.randint(...) np.median(a) a[a>10]
a**2 np.sum(a) np.mean(a) np.dot(a,b) np.ones(...)  
  ```

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

---

Overview

- **ML Algorithms**
  - Supervised Learning
  - Unsupervised Learning
    - Hierarchical Clustering
    - K-Means
    - Gaussian Mixture Models
    - Dimensionality Reduction

- **Non-Parametric**
  - Decision Trees
  - kNN
  - Support Vector Machines
  - Collaborative Filtering

- **Parametric**
  - Regression Models
    - Linear Regression
    - Logistic Regression
  - Linear Classifiers
    - Perceptron
    - Neural Networks
  - Non-Linear Classifiers
    - Support Vector Machines
    - Gaussian Mixture Models
    - Hidden Markov Models