Recommender System

- What is it?
- How to build it?
- Challenges
- R package: `recommenderlab`

What is a recommender system

- Wiki definition:

  A **recommender system** or a **recommendation system** (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item.

- A system that automatically recommend items to users, which likely to be of interest to the users, by utilizing historical information.

- Daily recommender systems: Netflix, Amazon, Online Ads, Google news, Twitter, Facebook, Pandora, Pinterest, ...

Data Representation

- **User** $i = 1 : m$
- **Item** $j = 1 : n$
- **Utility matrix** R_{ij} (aka the User-Item Ratings Matrix): user i's preference for item j, which could be binary (click or not click), numerical (movie ratings), or NA.
- Goal: predict the missing entries in R, which are likely to take high values. So not exactly a matrix completion problem, although MSE or its variants are often used as evaluation criteria.

A Naive Recommender System

- Aggregate the ratings for each item
- Find the top K highly liked items, and then recommend them to every user
- Drawback: not personalized.
- How to tailor the recommendations for a user?
- Key ideas: users tend to like similar items; items tend to be liked by similar users.

How to Build a Recommender System

- Content-based method
- Collaborative filtering (CF) method
  - Item-based CF
  - User-based CF
- Hybrid method
- Dimension reduction via SVD
- A key ingredient: similarity measure

Content-Based Recommendation

- Item profile: represent each item by a d-dim feature vector. For example, how to characterize a movie by a feature vector?
- User profile: represent each user by a d-dim feature vector by aggregating the feature vectors of items this user like.

So we embed the m users and n items in a Euclidean space $\mathbb{R}^d$. Then we can recommend items that are close to user i to user i.

Measuring Similarity

- Jaccard similarity: useful for binary ratings

$$\frac{|A \cap B|}{|A| \cdot |B|}, \quad \text{where } A, B \text{ are two sets.}$$

- Cosine similarity: useful for numerical ratings

$$\frac{u^t v}{\|u\| \cdot \|v\|}, \quad \text{where } u, v \text{ are two vectors}$$

- Other similarity measures: Pearson correlation

Item-Based CF

- Instead of using feature vectors to measure the similarity between items, we use similarity between their ratings.
- That is, we use the j-th column of the user-item Utility matrix R as the feature vector for user i, and compute similarity between items using one of the similarity measures.
- Then recommend item j to users who like items similar to j.

User-Based CF

- Similar to what we have done in Item-Based CF, we use the i-th row of the user-item Utility matrix to compute similarity between users.
- For user i, find his/her K-nearest users, aggregate their item ratings and then recommend highly liked items to user i.

Singular Value Decomposition

Approximate $R_{m\times n} \approx U_{m\times d} V^t_{d\times n}$ by minimizing

$$\sum_{R_{ij}\neq \text{NA}} (R_{ij} - u_i^t v_j)^2 + \lambda_1 \text{Pen}(U) + \lambda_2 \text{Pen}(V),$$

where u_i is the i-th row of matrix U and v_j is the j-th row of matrix V. Then we can predict any missing entries in R by the corresponding inner product of u_i and v_j.

I've also seen systems that use u_i's and v_i's as the new reduced dimension representation of users/items, and then apply content-based or CF methods.

Some Practical Issues

- Correct bias by normalization
- Cluster users and items
- Combine multiple recommender systems
- Consider content (location, time, device) and interface (mobile, laptop)
- Serendipity/Diversity versus accuracy
- Incorporate user feedback

Challenges

- Scalability: large amount of items and users
- Imbalance: data sparsity
- Utility matrix: how to construct it based on the problem at hand?
- Cold-start: how to recommend a new item or make recommendations to a new user, since they do not have historical information?

R package: recommenderlab

- Recommender systems 101 [link]
- Recommendation system in R [link]
- Building a movie recommendation engine with R [link]