A Multifaceted Collaborative Filtering(CF) Model Recommendation Systems

CSE 471 - Statistical Methods in AI

April 29, 2019

0.1 Contents

- Introduction
- Recommender Systems
- Dataset
- Methodology
- Neighborhood Model
- SVD++ Model
- Integrated Model
- Results
- Future Scope
- Challenges and Limitations
- Work Distribution

0.2 Introduction

Modern consumers are inundated with choices. Electronic retailers and content providers offer a huge selection of products, with unprecedented opportunities to meet a variety of special needs and tastes. Matching consumers with most appropriate products is not trivial, yet it is a key in enhancing user satisfaction and loyalty. This emphasizes the prominence of recommender systems, which provide personalized recommendations for products that suit a users taste . Internet leaders like Amazon, Google, Netflix, TiVo and Yahoo are increasingly adopting such recommenders.

Recommender systems are often based on Collaborative Filtering (CF) , which relies only on past user behaviore.g., their previous transactions or product ratingsand does not require the creation of explicit profiles. Notably, CF techniques require no do- main knowledge and avoid the need for extensive data collection. In addition, relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes. As a consequence, CF attracted much of attention in the past decade, resulting in significant progress and being adopted by some successful commercial systems, including Amazon, TiVo and Netflix.

In order to establish recommendations, CF systems need to compare fundamentally different objects: items against users. There are two primary approaches to facilitate such a comparison, which constitute the two main disciplines of CF: the neighborhood approach and latent factor models.

Neighborhood methods are centered on computing the relation- ships between items or, alternatively, between users. An item- oriented approach evaluates the preference of a user to an item based on ratings of similar items by the same user. In a sense, these methods transform users to the item space by viewing them as baskets of rated items. This way, we no longer need to compare users to items, but rather directly relate items to items.

Latent factor models, such as Singular Value Decomposition (SVD), comprise an alternative approach by transforming both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. For example, when the products are movies, fac- tors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined di- mensions such as depth of character development or quirkiness; or completely uninterpretable dimensions.

Neighborhood models are most effective at detecting very lo- calized relationships. They rely on a few significant neighborhood- relations, often ignoring the vast majority of ratings by a user. Con- sequently, these methods are unable to capture the totality of weak signals encompassed in all of a users ratings. Latent factor models are generally effective at estimating overall structure that relates si- multaneously to most or all items. However, these models are poor at detecting strong associations among a small set of closely related items, precisely where neighborhood models do best.Neighborhood models are most effective at detecting very lo- calized relationships. They rely on a few significant neighborhood- relations, often ignoring the vast majority of ratings by a user. Con- sequently, these methods are unable to capture the totality of weak signals encompassed in all of a users ratings. Latent factor models are generally effective at estimating overall structure that relates si- multaneously to

most or all items. However, these models are poor at detecting strong associations among a small set of closely related items, precisely where neighborhood models do best.

In this project we build a combined model that improves prediction accuracy by capitalizing on the advantages of both neighborhood and latent factor approaches.

0.2.1 Problem Statement

Predict movie ratings for all the users and thus recommend the relevant movies to the users making use Multifaceted Collaborative Filtering.

This is realized by building a combined model that improves prediction accuracy by capitalizing on the advantages of both neighborhood and latent factor approaches.

0.3 Recommender System

A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications.

Taxonomy of Recommender System :

Recommender systems usually make use of either or both collaborative filtering and content-based filtering (also known as the personality-based approach)[7], as well as other systems such as knowledge-based systems. Collaborative filtering approaches build a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in.[8] Content-based filtering approaches utilize a series of discrete, pre-tagged characteristics of an item in order to recommend additional items with similar properties.[9] Current recommender systems typically combine one or more approaches into a hybrid system (see Hybrid Recommender Systems).

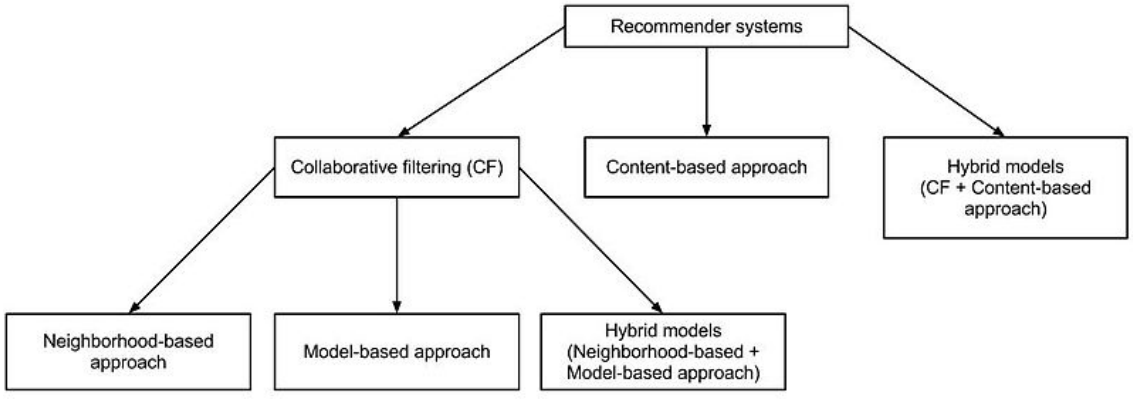

Figure 1: Taxonomy of Recommender System

0.3.1 Collaborative Filtering

One approach to the design of recommender systems that has wide use is collaborative filtering. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations using only information about rating profiles for different users or items. By locating peer users/items with a rating history similar to the current user or item, they generate recommendations using this neighborhood. The user- and item-based nearest neighbor algorithms can be combined to deal with the cold start problem and improve recommendation results using this data Collaborative filtering methods are classified as memory-based and model-based. A well-known example of memory-based approaches is the user-based algorithm, while that of model-based approaches is the Kernel-Mapping Recommender.

Key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighbor (k-NN) approach and the Pearson Correlation as first implemented by Allen.

When building a model from a user's behavior, a distinction is often made between explicit and implicit forms of data collection.

Examples of explicit data collection include the following:

Asking a user to rate an item on a sliding scale. Asking a user to search. Asking a user to rank a collection of items from favorite to least favorite. Presenting two items to a user and asking him/her to choose the better one of them. Asking a user to create a list of items that he/she likes (see Rocchio classification or other similar techniques). Examples of implicit data collection include the following:

Observing the items that a user views in an online store. Analyzing item/user viewing times.[38] Keeping a record of the items that a user purchases online. Obtaining a list of items that a user has listened to or watched on his/her computer. Analyzing the user's social network and discovering similar likes and dislikes.

0.3.2 Content based Filtering

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the users preferences. These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.

In this system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past, or is examining in the present. It does not rely on a user sign-in mechanism to generate this often temporary profile. In particular, various candidate items are compared with items previously rated by the user

and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research.

To create a user profile, the system mostly focuses on two types of information:

- A model of the user's preference.
- A history of the user's interaction with the recommender system.

0.3.3 Hybrid recommender systems

Most recommender systems now use a hybrid approach, combining collaborative filtering, content-based filtering, and other approaches . There is no reason why several different techniques of the same type could not be hybridized. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. Several studies that empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrated that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem, as well as the knowledge engineering bottleneck in knowledge-based approaches.

Netflix is a good example of the use of hybrid recommender systems. The website makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).

0.4 Dataset

For this project,we have used MovieLens dataset, which is one of the standard datasets used for implementing and testing recommender engines.

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota.

This data set consists of:

- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.

Features of the dataset -

- The ratings are at intervals of 0.5 on a 5-point scale,starting from 0.5 and going to 5.
- The data is randomly ordered.
- The time stamps are unix seconds since 1/1/1970 UTC
- Each user is represented by an id, and no other information is provided.

The dataset is split into 2 partitions, train and test sets.(80-20) split.

0.5 Methodology

0.5.1 Baseline Estimate

Typical CF data exhibit large user and item effects i.e., system- atic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. It is customary to adjust the data by accounting for these effects, which we encapsulate within the baseline estimates. Denote by the overall average rating. A baseline estimate for an unknown rating rui is denoted by bui and accounts for the user and item effects:

$$b_{ui} = \mu + b_u + b_i \tag{1}$$

The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline estimate for the rating of the movie Titanic by user Joe. Now, say that the average rating over all movies,mu, is 3.7 stars. Furthermore, Titanic is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the baseline estimate for Titanics rating by Joe would be 3.9 stars by calculating 3.7 0.3 + 0.5. In order to estimate b_u and b_i, one can solve the least squares problem:

$$\sum_{(u,i)\in(K)}(r_{ui} - \mu - b_u - b_i)^2 + \lambda_1(\sum_u {b_u}^2 + \sum_i {b_i}^2) \tag{2}$$

Here, the first term $\sum_{(u,i)\in(K)}(r_{ui} - \mu - b_u - b_i)^2$ strives to find the b_u's and b_i's that fit the given ratings. The regularizing term $\lambda_1(\sum_u {b_u}^2 + \sum_i {b_i}^2)$ avoids overfitting by penalizing the magnitudes of the parameters.

0.5.2 Neighborhood Models

The most common approach to CF is based on neighborhood models. Its original form, which was shared by virtually all earlier CF systems, is user-oriented. Such user-oriented methods estimate unknown ratings based on recorded ratings of like minded users.

Central to most item-oriented approaches is a similarity measure between items. Frequently, it is based on the Pearson correlation coefficient, $_ij$, which measures the tendency of users to rate items i and j similarly. Since many ratings are unknown, it is expected that some items share only a handful of common raters. Computation of the correlation coefficient is based only on the common user support. Accordingly, similarities based on a greater user support are more reliable.

Our goal is to predict r_{ui} the unobserved rating by user u for item i. Using the similarity measure, we identify the k items rated by u, which are most similar to i. This set of k neighbors is denoted by $S_{k(}i;u)$. The predicted value of r_{ui} is taken as a weighted average of the ratings of neighboring items, while adjusting for user and item effects through the baseline estimates:

$$r_{ui} = \frac{b_{ui}+\sum_j \epsilon S^k(i;u)s_{ij}(r_{uj}-b_{uj})}{\sum j\epsilon S^k(i;u)s_{ij}} \tag{2}$$

0.5.3 Improved Neighborhood Model

In this section we introduce a new neighborhood model, which allows an efficient global optimization scheme. The model offers improved accuracy and is able to integrate implicit user feedback. We will gradually construct the various components of the model, through an ongoing refinement of our formulations.

Previous models were centered around user-specific interpolation weights $-\theta_{ij}^{u}$

0.5.4 SVD and SVD++ model

We have focused on models that are induced by Singular Value Decomposition (SVD) on the user-item ratings matrix. Recently, SVD models have gained popularity, thanks to their attractive accuracy and scalability. A typical model associates each user u with a user- factors vector $\mathrm{p}_u Rf, and each item i with an item - factors vector qiRf. The prediction is done by taking an inner product bui + pTuqi. The more involved part is parameter estimation.$

0.5.5 Integrated model

The new neighborhood model is based on a formal model, whose parameters are learnt by solving a least squares problem. An advantage of this approach is allowing easy integration with other methods that are based on similarly structured global cost functions. Latent factor models and neighborhood models nicely complement each other. Accordingly, in this section we will integrate the neighborhood model with our most accurate factor model SVD++. A combined model will sum the predictions of neighborhood and Latent factor model, thereby allowing neighborhood and factor models to enrich each other, as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$
$$+ |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}$$

In a sense, rule (16) provides a 3-tier model for recommenda- tions. The first tier, + bu + bi , describes general properties of the item and the user, without accounting for any involved interactions. For example, this tier could argue that The Sixth Sense movie is known to be good, and that the rating scale of our user, Joe, tends to be just on average. the next tier, provides the interaction between the user profile and the item profile.

$$, q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \textstyle\sum_{j \in \mathrm{N}(u)} y_j \right),$$

In our example, it may find that The Sixth Sense and Joe are rated high on the Psychological Thrillers scale. The final neigh- borhood tier contributes fine grained adjustments that are hard to profile, such as the fact that Joe rated low the related movie Signs. Model

parameters are determined by minimizing the associated regularized squared error function through gradient descent. Recall $e_{ui} = r_{ui} - r_{ui}^{-}$ We loop over all known ratings in K. For a given training case rui, we modify the parameters by moving in the opposite direction of the gradient, yielding:

- $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$
- $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j) - \lambda_7 \cdot q_i)$
- $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$
- $\forall j \in \mathrm{N}(u)$:
  $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |\mathrm{N}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_7 \cdot y_j)$
- $\forall j \in \mathrm{R}^k(i;u)$:
  $w_{ij} \leftarrow w_{ij} + \gamma_3 \cdot \Big(|\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_8 \cdot w_{ij} \Big)$
- $\forall j \in \mathrm{N}^k(i;u)$:
  $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot \Big(|\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_8 \cdot c_{ij} \Big)$

When evaluating the method on the ml-100k data, we used the following values for the meta parameters: $\gamma_1 = \gamma_2 = 0.007, \gamma_3 = 0.001, _6 = 0.005, _7 = _8 = 0.015$. It is beneficial to decrease step sizes (the γ's) by a factor of 0.9 after each iteration. The neighborhood size, k, was set to 300. The iterative process runs for around for 30 epochs till convergence.

By coupling neighborhood and latent factor models together, and recovering signal from implicit feedback, accuracy of results is improved beyond other methods. Recall that unlike SVD++, both the neighborhood model and Asymmetric-SVD allow a direct explanation of their recommendations, and do not require re-training the model for handling new users. Hence, when explainability is preferred over accuracy, one can follow very similar steps to integrate Asymmetric-SVD with the neighborhood model, thereby improving accuracy of the individual models while still maintaining the ability to reason about recommendations to end users.

0.6 Results

In 30 epochs, root mean square error converged to the following values :

- **Neighborhood model : 1.7757**
- **SVD++ model: 0.941**
- **Integrated model : 0.9283**

0.7 Future scope

Similarly, the movies total gross could be used to identify a users taste in terms of whether he/she prefers large release blockbusters, or smaller indie films. However, the above ideas may lead to overfitting, given that a users taste can be highly varied, and we only have a guarantee that 20 movies (less than 0.2%) have been reviewed by the user.

In addition, we could try to develop hybrid methods that try to combine the advantages of both content-based methods and collaborative filtering into one recommendation system.

0.8 Challenges and Limitation

- Insufficient hardware support to run large dataset (Netflix dataset), even in CSR format.
- Lack of better sources for implicit feedback.
- Data is extremely sparse.
- Not scalable.
- Cold start problem with respect to users and movies. (Can kill the recommender systems).

0.9 Work Distribution

At the beginning stage, all members explored the topic and spent time in learning about recommender systems and their types in relation to our project. Then, keeping in mind the deliverables of this project, all tasks were was divided into a number of modules. And after that,in order to better utilize the limited time and resources, workload of the project was divided in two workgroups:

- Sandeep and Aditya worked on Analysis of data and the Integrated model.
- Aishwarya and Akshansh worked on the SVD++ model, neighborhood model.

Since all modules have interdependencies, so integration of functionalities added after every update was done from time to time to avoid incompatibility of the whole code.