CS410 Tech Review: Collaborative Filtering Algorithms

squires4 | Samantha Squires

Introduction: Recommender Systems and Collaborative Filtering

Recommender systems, such as those used by Netflix, YouTube, Spotify, and Amazon, are designed to predict which items a user will like (for example, which products they are likely to purchase on Amazon). There are two main approaches to this problem: *content-based filtering*, which uses similarities between items to recommend items similar to what a user has previously liked[1], and *collaborative filtering*, which also takes into account similarities between users to recommend items like those that similar users have liked[2]. For example, a content-based filtering algorithm on a movie streaming service might recommend a movie to User A because User A previously watched other movies in the same genre, while a collaborative filtering algorithm might recommend a movie to User A because other users with a similar watch history to User A also watched this movie.

This tech review will focus on *collaborative filtering*, and specifically on algorithms used for two sub-categories of collaborative filtering: *memory-based* methods and *model-based* methods.

Collaborative Filtering Algorithms

Memory-Based Algorithms for Collaborative Filtering

Memory-based algorithms directly use historical data of interactions between users and items (e.g. rating history, purchase history, watch history) to find similar users and predict how much a user will like an item based on how much their "nearest neighbor" users have liked it[3]. One example of a memory-based algorithm is a user-based nearest-neighbor approach:

User-based Collaborative Filtering using Nearest Neighbors

Consider the use case of a system that wants to recommend the "top N" items that a user has not yet seen, but would be interested in (for example, Netflix recommendations of nnew shows to watch that a user would rate most highly). One method to accomplish this is a user-based nearest-neighbor approach:

The basic idea behind this approach is this: given a user A and an item I which the user has not yet seen, find a set of users who are similar to A (say, users who have watched many of the same shows on Netflix as A) and have rated I. Predict user A's rating of I by combining the ratings of these similar users (for example, taking the average). Repeat this process for each item that user A has not yet seen, and recommend the top N based on predicted rating[4].

A common metric for measuring similarity between users is *Pearson correlation*, which takes into account differences in rating patterns between users by incorporating each user's average rating, with the following formula[5]:

[1] https://developers.google.com/machine-learning/recommendation/content-based/basics
[2] https://developers.google.com/machine-learning/recommendation/collaborative/basics
[3] http://www10.org/cdrom/papers/519/node7.html
[4] https://www.math.uci.edu/icamp/courses/math77b/lecture_12w/pdfs/Chapter%2002%20-%20Collaborative%20recommendation.pdf
[5] https://grouplens.org/blog/similarity-functions-for-user-user-collaborative-filtering/

$$sim(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{ui} - \mu_u)(r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{ui} - \mu_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{vi} - \mu_v)^2}}$$

In the formula above, r_{ui} and r_{vi} indicate user U's and user V's ratings of item I, and μ_u and μ_v represent user U's and user V's mean ratings[6].

While easy to understand, memory-based algorithms for collaborative filtering such as k-NN have a disadvantage: they don't tend to scale well to large amounts of data[7]. For modern applications that must make filtering decisions for many users and items, model-based algorithms may be more appropriate.

Model-Based Algorithms for Collaborative Filtering

Model-based collaborative filtering algorithms use historical data of interactions between users and items to create a model, which is then used to make recommendations. In model-based algorithms, the entire dataset is only required in the model building stage, and not required to make each prediction of whether a user will like an item or not[8]. One common model-based approach is *matrix factorization*:

Matrix Factorization

Imagine a similar scenario to the one discussed above, when we covered nearest-neighbors filtering: a site like Netflix wishes to recommend shows for a user to watch next, recommending the shows they will like the most. The following is an example of using matrix factorization to approach this problem:

Let's say we have an m by n matrix A, where m is the number of users and n is the number of items (such as Netflix shows). In matrix A, the row for each user contains the rating they have given to each show they previously rated. This matrix will be large, because there are many users and many shows, and it will also be sparse–a typical user will only have rated a small fraction of the shows available on Netflix.

In matrix factorization, we find two smaller matrices, U and V, such that:

$$A = UV^T$$

Here, U is the *user embedding matrix*, with dimension m by k, and V is the *item embedding matrix*, with dimension n by k. We use k to represent the number of latent features to discover about our items[9].

To find matrices U and V such that UV^T is approximately equal to the original ratings matrix A, we define an objective function to minimize (such as the sum of squared errors over all user/item ratings) and minimize it using a method like stochastic gradient descent or Weighted Alternating Least Squares (WALS) [10].

After obtaining estimates for U and V using the available historical ratings data, we can compute UV^T to obtain estimated ratings that each user would give to items they have not yet seen. This allows us to recommend to a user the items with the highest predicted ratings for that user.

Challenges in Collaborative Filtering: The Cold-Start Problem

One challenge faced by both memory-based and model-based collaborative filtering systems is known as the *cold-start problem*, which occurs when a new user or item is added to the system and there is no historical data available (for example, a new user joins Netflix and hasn't watched or rated anything yet, so there is no data yet about their likes/dislikes).

One common solution to the cold-start problem is active learning, where a new user is asked to rate a selection of items to gather some data on their preferences. These initial items that the user rates may be chosen using

[6] https://grouplens.org/blog/similarity-functions-for-user-user-collaborative-filtering/
[7] https://www.aaai.org/Papers/AAAI/2006/AAAI06-218.pdf
[8] http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html
[9] https://developers.google.com/machine-learning/recommendation/collaborative/matrix
[10] https://developers.google.com/machine-learning/recommendation/collaborative/matrix

a variety of strategies, including selecting "controversial" items with a wide variety of ratings, or preferentially selecting more "well-known" items that the new user is more likely to be familiar with. New items with no user interaction history can be dealth with using a system that takes into account known attributes of items (e.g. film genre, actors, etc.) to make recommendations, which is an example of a hybrid recommender system, combining collaborative filtering with content-based filtering[11].

Conclusion:

This article has focused on collaborative filtering approaches to recommender system, illustrating examples of both memory-based and model-based algorithms. Many additional collaborative filtering methods exist beyond those discussed in detail above: for example, some recent research has focused on a deep-learning approach to collaborative filtering using neural nets[12].The cold-start problem is a common one for both types of collaborative filtering system, and to overcome it, many modern recommender systems combine collaborative and content-based approaches into a hybrid system to achieve better performance. Building better reccomender systems is essential to many modern products, and methods for collaborative filtering will surely continue to be implemented and improved upon for years to come.

[11] https://www.researchgate.net/publication/332511384_Cold_Start_Solutions_For_Recommendation_Systems
[12] https://dl.acm.org/doi/10.1145/3038912.3052569