Three approaches to recommender systems

Martin Powers

University of Minnesota - Morris
Morris, Minnesota 56267

power182@morris.umn.edu

December 4, 2010

Outline

Why we care about recommender systems

Without recommender systems we recieve suggestions from:

- word of mouth
- reading reviews
- researching
- trial and error

With recommender systems we:

- seamlessly interact with the browsing tools we already use
- more frequently find items we will enjoy
- spend less time looking for items
- spend less money trying out items

What is an recommender system?

Recommender System: A system whose purpose is to take in information and output suggestions to a user.

- Billboard top 10
- Oprah's bookclub

Collaborative Filtering: Using a large number of different user's preferences to find recommendations for a specific user.

- Netflix
- Amazon

Netflix

Movie rental and streaming service that suggests movies to users based on what movies they have rated.

Uses a recommender system called Cinematech to predict ratings.

Both Netflix and customers benefit from accurate recommendations.

Netflix Prize

In 2006 Netflix released a enormous dataset containing over

100 million ratings given by

480,000 users on

17,770 movies

$1,000,000 prize to the team that improves Cinematech's accuracy by 10%

Won in 2009 by "Bell-Kors Pragmatic Chaos" with a 10.05% improvement.

Input

Users provide input through implicit and explicit means

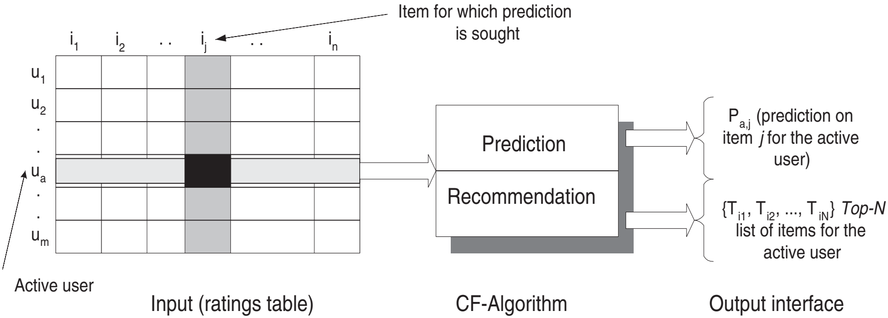
Implicit:

- Pageviews
- What website they arrive at a page from
- Frequency that an item is used

Explicit:

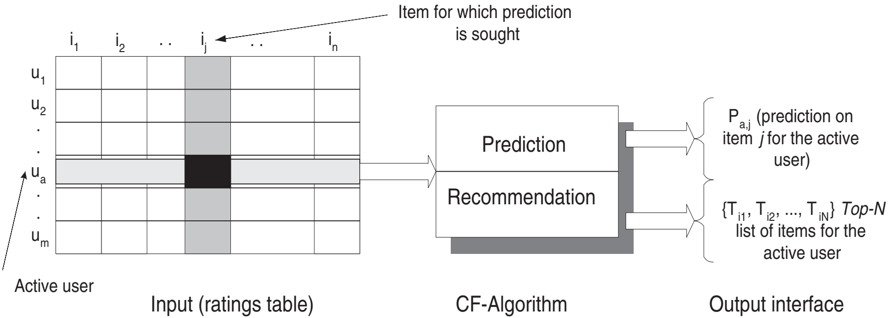
- Product review
- Rating
- Purchasing an item

The collaborative filtering process



m users $U = \{u_1, u_2, \cdots, u_m\}$
n items $I = \{i_1, i_2, \cdots, i_n\}$

The collaborative filtering process



Prediction: A value that the user is expected to give an unrated item.
Recommendation: A list of items that the user is expected to like.

Finding neighbors

To find a user's neighbors we compare the active user with all other users and find the ones with most similar taste.

We use the method userSim(u, n) to determine how close the user u and its neighbor n are.

Different algorithms can be used in userSim(u, n), we use the pearson correlation algorithm.

The result will be within the range from -1, showing perfect disagreement, and 1, being perfect agreement.

Pearson correlation algorithm

$$\text{userSim}(u, n) = \frac{\sum_{i \in I_{u,n}} (r_{ui} - \overline{r}_u)(r_{ni} - \overline{r}_n)}{\sqrt{\sum_{i \in I_{u,n}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{i \in I_{u,n}} (r_{ni} - \overline{r}_n)^2}}$$

where:

- $I_{u,n}$ is all the items that both have rated
- r_{ui} and r_{ni} are the ratings users u and n have given item i
- $\overline{r}_u$ and $\overline{r}_n$ are the average rating for users u and n
- Ranges between -1 and 1

Predicting a rating

We use userSim() to determine a predicted rating, $P_{u,i}$, for an item, i, that the active user, u, hasn't rated yet.

The prediciton algorithm we use is a weighted sum algorithm.

Prediction algorithm

$$P_{u,i} = \bar{r}_u + \frac{\sum_{n \in N_u} \text{userSim}(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \in N_u} \text{userSim}(u,n)}$$

This is calculated by finding the sum of all ratings for i given by u's neighbors, each weighted by how similar u is to each neighbor.

To make sure that $P_{u,i}$ is in the same scale as all other ratings, we normalize the above by dividing it by the sum of u's similarity with their neighbors.

We add the average of u's ratings to the total while also subtracting the average of each neighbor's ratings to compensate for each user's rating bias.

Why item-based?

Two main challenges for user-based algorithms:

Scalability:

- Growing number of users
- Larger neighborhoods

Sparsity:

- New users have rated no items
- Low rating frequency of users

Item-based algorithms

Instead of comparing users based on their similarity we compare items

Items work better than users because

- items are static
- there are less items than users

To find if two items are similar we look at all users who have rated both
If the items have similar ratings from a user, we say that they are similar

We modify our user-based algorithm, pearson correlation, to work with items

Pearsons correlation algorithm

$$\text{itemSim}(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_u)^2}}$$

where

- items i and j are two items being compared
- U is a list containing all users who have rated both i and j
- $r_{u,i}$ and $r_{u,j}$ are a user's ratings for items i and j
- $\bar{r}_u$ is user u's average rating

Item-based prediction

To predict what a user, u, will rate an item, i, we look at what the user has rated items similar to i.
If u rates similar items highly, then it is likey they will rate i highly as well.

To find this rating, we can again use a weighted sum algorithm

Item-based prediction algorithm

$$P_{u,i} = \frac{\sum_{N \in \text{ similar rated items}} \text{itemSim}(i, N) \cdot r_{u,N}}{\sum_{N \in \text{ similar rated items}} \text{itemSim}(i, N)}$$

where

- u and i are the active user and the item we're predicting the rating for
- N is an item similar to i
- $r_{u,N}$ is the rating u gave to item N

What is a temporal model?

Temporal model: A prediction algorithm that takes into account the time that a user rated items and adjusts its prediction accordingly.

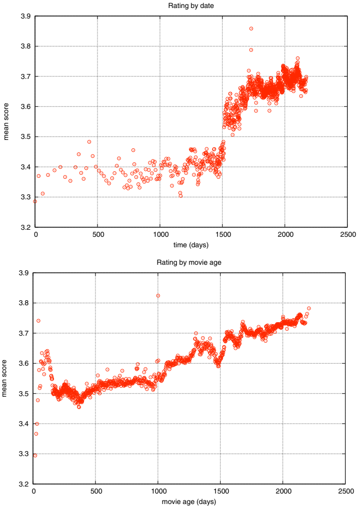
Reasons to use a temporal model:

- users change how they rate items over time
- item's ratings change over time

Temporal trends in the Netflix data

User ratings jump in early 2004 (1500 days)

Item ratings increase with age



How to use the temporal dynamic

Things to take in account:

- User bias changes over time
- Item bias changes over time
- User preference changes over time

Bias is the deviation from the average rating an item or user has.

We can localize each of these trends by sampling smaller spans of time. Instead of looking at the entire dataset, we only look at ten weeks of data at a time. We create bins that relate to a specific time span and create a prediction model uses these bins.

Parts of a temporal model

$$b_{ui}(t) = \mu + b_u(t) + b_i(t)$$

where:

- μ is the average rating of items by all users
- $b_u(t)$ is the bias for user u at time t
- $b_i(t)$ is the bias for item i at time t

In conclusion

User-based algorithms work well and are easily implemented, but can be improved upon by exploring different methods and finding trends in the data. The two techniques explored here make use of these trends.

- The item-based algorithm makes use of the fact that items are static.
- The temporal model takes into account that users and movies are affected by time.

When looking for ways to improve recommender systems it becomes important to recognize attributes of a dataset so that improvements can be made.

Questions?