Collaborative filtering for sparse explicit binary data

Solvi Mar Magnusson,
Computer Science Department, Pace University, New York, NY, USA
sm65664n@pace.edu

1 Introduction

Physical stores are becoming a thing of the past. Consumers have never had as much freedom to shop, what they want, when they want, and wherever they are. The growth of online shopping, and content consumption has created a vast and vibrant market space [1]. With this increased growth of content consumption, consumers are increasingly moving away from the traditional weekly and seasonal programming schedules. According to research by Pricewaterhouse Coopers, 63% of US households, in 2013, used a video streaming service [2]. This tells us that modern consumers are overwhelmed with choices. These e-commerce retailers, and content providers offer consumers a large selection of products to buy. For those retailers and content providers, matching consumers with the most appropriate products is the key to enhancing user satisfaction and loyalty. This is where recommendation systems come into play[3]. Recommendation systems are a set of tools and techniques for providing suggestions for items that are of particular interest of the consumer.

There are multiple types of recommendation systems available, most notably there are content-based and collaborative filtering recommendation systems. Content-based systems learns to recommend items that are similar to other items a user has rated in the past, while collaborative filtering recommends items to the user items that other users, with similar tastes have liked in the past . The transaction, the connection between users and items, can be made out of explicit or implicit feedback. Explicit data is where users are prompted to rate items explicitly while implicit transactions can be extracted from user's actions [4]. In this paper I an going to focus on collaborative filtering for sparse [5] explicit binary data as current popular techniques, such as singular value decomposition [4], do not handle binary matrices as well as real-valued matrices [6].

2 Background and related work

Here I am going to go into more detail about neighborhood-based collaborative filtering and latent factor matrix factorization in recommendation systems.

2.1 Neighborhood models

Neighborhood collaborative filtering methods are based on the K-Nearest Neighbor rules where similarities between users and/or items are used to generate a set of items that are relevant for the active user. This method relies entirely on a good similarity metric that can handle sparse data. There have been proposed several similarity methods that can handle the sparse nature of the data, the Pearson Correlation Coefficient and Cosine-based similarity. When it comes to imputing a rating of an item for a user, all the ratings of the users that are in the neighborhood of the active user are aggregated into the predicted rating using a weighted majority prediction [7].

2.1.1 Advantages and disadvantages of Neighborhood models

Advantages of Neighborhood models

- Simple. At best they only need one parameter to be tuned, the number of neighbors.
- They are justifiable. Because you are predicting a rating based on other users or items, you can justify that this prediction is because you are similar to these users, or the items you have rated are similar to these items.
- Popular items get recommended more frequently.
- Knowledge of the domain is not necessary.
- The item recommendations will improve over time once more data is collected.

Disadvantages of Neighborhood models

- If an item has not been rated or few have rated said item, that won't be rated to users.

Source: [8]

2.2 Latent factor matrix factorization

Matrix factorization models have gained a lot of popularity over the years, thanks to its accuracy and scalability [4]. Matrix factorization models try to factorize the [User x Item] matrix into a joint latent factor space of dimentionality f, such that the [User x Item] matrix interactions are inner products of that space. Each item i is associated with a vector $q_i \in \mathbf{R}^f$, and each user is associated with a vector $p_u \in \mathbf{R}^f$. The resulting dot product $q_i^T p_u$ captures the interaction between user u and item i. The final rating is then created by adding baseline predictors that depend only on the user or item. We can therefore assume that a rating is generated through the following equation.

$$\hat{r}_i = \mu + b_i + b_u + q_i^T p_u$$

Then usually stochastic gradient decent is used to learn the model parameters by minimizing the regularized squared error [4]. As mentioned above this method is accurate and scaleable, however it lacks the ability to factorize binary matrices [6].

3 Method

Here I am going to propose a possible solution for a recommendation system for binary data using a neighborhood model and a matrix factorization model.

The solution is in two steps.

1. Impute as many missing values in the dataset using the neighborhood method.
2. Use matrix factorization on the imputed dataset.

3.1 Step 1

The imputation step takes the neighborhood model, and imputation from [7], AutAI.
The AutAI method can identify what missing values should be imputed, with the imputed set adaptively determined according to the active user's own rating history. To make a prediction on item t_s for user u_a, the imputed set is determined by two factors, the users related to u_a and items related to t_s, U_a and T_s respectively. The missing values that should be imputed are then taken from the key neighborhood of the active user, defined as $N_{a,i} = \{r_{a'i} | u_{a'} \in U_a, t_i \in T_s\}$ [7].

3.2 Step 2

Matrix factorization on the imputed dataset given from Step 1 using stochastic gradioent decent as described above.

4 Experiment and Result

4.1 The Dataset

The dataset used in this experiment comes from Grouplens research. They have been collecting rating data from the MovieLens website. The dataset contains about 100.000 ratings on 9.000 movies by 610 users [9].

4.1.1 Preprocessing

The dataset contains userId, movieId, ratings and a timestamp. The column timestamp was dropped as that adds no additional information for the recommendation system.
Originally, the ratings are on the scale $1 - 5$ with 0.5 increments. In order to make the dataset binary, all ratings over 3.5 were considered as a liked movie, or 1, and all ratings 3.5 and lower were considered as disliked movies, or 0. This split results in about half of the ratings are considered positive and half negative.

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

100836 rows × 4 columns

Figure 1: MovieLens dataset.

4.2 The experiment

After the preprocessing, the AutAI algorithm is applied to the dataset with a neighborhood size of 10. After this step the ratings went from 100.386 to 181.945 total ratings, an increase of 81.109 imputed ratings. The imputed dataset is then turned into an [user x item] matrix where all the zeroes were turned to -1 and missing values were set to 0. The matrix was then factorized with 30 latent factors and 200 iterations of stochastic gradient decent. The matrix factorization step was also applied to the original dataset without the imputation step.

```
Iteration: 10 ; error = 9.4481
Iteration: 20 ; error = 5.6170
Iteration: 30 ; error = 5.2375
Iteration: 40 ; error = 4.9856
Iteration: 50 ; error = 4.6739
Iteration: 60 ; error = 4.6972
Iteration: 70 ; error = 4.5750
Iteration: 80 ; error = 4.5786
Iteration: 90 ; error = 4.6407
Iteration: 100 ; error = 4.4873
Iteration: 110 ; error = 4.5330
Iteration: 120 ; error = 4.5831
Iteration: 130 ; error = 4.4267
Iteration: 140 ; error = 4.4706
Iteration: 150 ; error = 4.4631
Iteration: 160 ; error = 4.5373
Iteration: 170 ; error = 4.4495
Iteration: 180 ; error = 4.3537
Iteration: 190 ; error = 4.4934
Iteration: 200 ; error = 4.4309
```

Figure 2:
Matrix factorization on the original dataset

```
Iteration: 10 ; error = 7.1546
Iteration: 20 ; error = 4.1522
Iteration: 30 ; error = 3.5164
Iteration: 40 ; error = 3.3361
Iteration: 50 ; error = 3.1461
Iteration: 60 ; error = 3.1169
Iteration: 70 ; error = 2.9848
Iteration: 80 ; error = 2.9287
Iteration: 90 ; error = 2.8988
Iteration: 100 ; error = 2.8431
Iteration: 110 ; error = 2.9506
Iteration: 120 ; error = 2.8597
Iteration: 130 ; error = 2.8017
Iteration: 140 ; error = 2.9128
Iteration: 150 ; error = 2.8271
Iteration: 160 ; error = 2.9169
Iteration: 170 ; error = 2.8104
Iteration: 180 ; error = 2.8132
Iteration: 190 ; error = 2.7959
Iteration: 200 ; error = 2.7632
```

Figure 3:
Matrix factorization on the imputed dataset

We can see that the matrix factorization on the imputed dataset converges faster to a smaller error rate than matrix factorization on the original dataset, however the imputed dataset starts with a smaller error rate after the first 10 iterations. The reason for that is we almost doubled the size of the dataset after the imputation stage and therefore it is easier to find a better factorization.

5 Future work

I definately need to look better into this method of collaborative filtering on binary sparse data. Because of time limits I was not able to fine tune the parameters needed for the two steps. A neighborhood size of 10 was arbitrarilly chosen for a faster computational time, therefore more neighborhood sizes are needed for testing. The same goes for the matrix factorization, 200 iterations were chosen for the same reason as the neighborhood size. More research is needed for choosing a similarity metric for the binary data, as the AutAI uses the Pearson Correlation Coefficient, that metric was used without any consideration to other metrics.

References

[1] Kuan-Pin Chiang and Ruby Roy Dholakia. Factors driving consumer intention to shop online: an empirical investigation. *Journal of Consumer psychology*, 13(1):177–183, 2003.

[2] Sidneyeve Matrix. The netflix effect: Teens, binge watching, and on-demand digital media trends. *Jeunesse: Young People, Texts, Cultures*, 6(1):119–138, 2014.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[5] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[6] Prashant Yadava, Pauli Miettinen, and Gerhard Weikum. Boolean matrix factorization with missing values. *Universitat des Saarlandes Max-Planck-Institut fiir In formatik AG5*, 2012.

[7] Yongli Ren, Gang Li, Jun Zhang, and Wanlei Zhou. Lazy collaborative filtering for data sets with missing values. *IEEE transactions on cybernetics*, 43(6):1822–1834, 2013.

[8] Anvitha Hegde and Savitha K Shetty. Collaborative filtering recommender system. *International Journal of Emerging Trends in Science and Technology*, 2(07), 2015.

[9] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.