Recommendation System Review Report

Jiefei Xia

Context

Recommendation algorithms are best known for their use on e-commerce Websites (J.B. Schafer, 2001), where they use input about a customer's interests to generate a list of recommended items. It has become a hot area since the appearance of the first papers on collaborative filtering in the mid-1990s. And based on google search trends (Figure 1), there's an increasing interest in this area in the last five years. Therefore, it is a great topic to discuss in our E-commerce course.

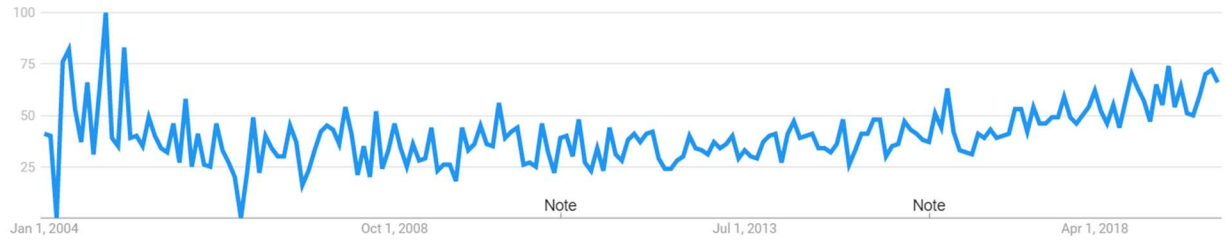

Figure 1 Google Trend of "recommendation system"

Problem

A typical recommendation system can be formulated as follows:

Let C be the set of all users, S be the set of all items, and u be a utility function that measures the usefulness of item s to user c, i.e., $u: C \times S \to R$, where R is a totally ordered. The utility of an item can be represented by rating, click, purchase frequency and so on. However, it can be represented as other formats as well, like click, purchase and so on. Then, for each user $c \in C$, we want to choose such item $s \in S$ that maximizes the user's utility. More formally:

$$\forall c \in C, s'_c = \underset{s \in S}{\operatorname{argmax}} u(c, s)$$

Each element of the user space C can be defined with a profile p_{c_i} that includes various user characteristics, such as age, gender, income, marital status, etc. Similarly, each element of the item space S is defined with a set of characteristics p_{s_i}.

*The central problem of recommender systems lies in that utility u is usually not defined on the whole $C \times S$ space, but only on some subset of it. This means u needs to be **extrapolated** to the whole space $C \times S$. In recommender systems, the utility is typically represented by ratings and is initially defined only on the items previously rated by the users.* (Adomavicius, 2005)

An example of a user-item rating matrix for a movie recommendation application is presented in Table 1. The recommendation engine should be able to estimate (predict) the empty utility (X) item/user combinations and issue appropriate recommendations based on these predictions. In the real case, m and n are usually extreme large (millions), so the matrix is very sparse (lots of 0).

User ID	*Item ID*	$\boldsymbol{s_1}$ $\boldsymbol{p_{s_1}}$	$\boldsymbol{s_2}$ $\boldsymbol{p_{s_2}}$	$\boldsymbol{s_3}$ $\boldsymbol{p_{s_3}}$	… …	$\boldsymbol{s_m}$ $\boldsymbol{p_{s_m}}$
$\boldsymbol{c_1}$	$\boldsymbol{p_{c_1}}$	0	X	u_{13}	…	0
$\boldsymbol{c_2}$	$\boldsymbol{p_{c_2}}$	u_{21}	0	0	…	u_{2m}
$\boldsymbol{c_3}$	$\boldsymbol{p_{c_3}}$	0	u_{32}	X	…	0
…	…	…	…	…	…	…
$\boldsymbol{c_n}$	$\boldsymbol{p_{c_n}}$	u_{n1}	0	u_{n3}	…	u_{nm}

Table 1 Data of Recommendation System

Algorithm

Basket Analysis

The most famous algorithm in the recommendation system is called collaborative filtering. But before going deep into the recommendation system, let's look at something before the invention of the collaborative filtering – the basket analysis.

Order ID	*Item*	*User ID*
o_1	Banana, Apple	c_1
o_2	Beer, Diaper, Cookie	c_3
o_3	Beer, Chips	c_2
o_4	Potato, Tomato, Egg	c_1
o_5	Orange juice, Beer, Diaper	c_7
…	…	…
o_k	…	c_n

Table 2 Data of Basket Analysis

In basket analysis, the data is in its original format recorded in order level, shown in table 2. The most popular algorithm in basket analysis is called association rule, which aims to find out the relevance in the

dataset. The main algorithm in association rules is called Apriori[1]. And there's an improvement version of it called FP Growth[2]. The main comparison between these two algorithms is shown in table 3.

	Apriori	***FP Growth***
Data Structure	Apriori property	Frequency Pattern Tree
Algorithm	Breadth-first search	Divide and conquer
Memory	Large	Small
Time Complexity	$O(Tk + (1 - m^T)/(1 - m))$	$O(m^2)$

Table 3 Two Association Rule Algorithms: Apriori vs FP Growth

Recommendation System

Recommendation systems are usually classified into the following categories, based on how recommendations are modeled:

- *Content-based recommendations: The user will be recommended items similar to the ones the user preferred in the past (based on item profile p_s);*
- *Collaborative recommendations: The user will be recommended items that people with similar tastes and preferences liked in the past;* (Adomavicius, 2005)

In this report, we will mainly focus on collaborative filtering as it is the most state-of-art one. Big tech companies including Amazon, Netflix and Spotify are all using the variant collaborative filtering algorithm.

Collaborative Filtering

The two primary areas of collaborative filtering are the neighborhood methods and latent factor models. Neighborhood methods are centered on computing the relationships between users. Latent factor models are an alternative approach that tries to explain the utilities by characterizing both items and users on some factors inferred from the rating patterns. In a sense, such factors comprise a computerized alternative to the profile.

Figure 2 left illustrates the user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies and then determines which other movies they liked. In this case, all three liked Saving Private Ryan, so that is the first recommendation. Two of them liked Dune, so that is next, and so on. (Koren, 2009)

[1] Pseudo code: https://www.geeksforgeeks.org/apriori-algorithm/
[2] Pseudo code: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm

Figure 2 right shows the latent factor approach. Consider two hypothetical dimensions characterized as female- versus male-oriented and serious versus escapist. The figure shows where several well-known movies and a few fictitious users might fall on these two dimensions. For this model, a user's predicted rating for a movie, relative to the movie's average rating, would equal the dot product of the movie's and user's locations on the graph. For example, we would expect Gus to love Dumb and Dumber, to hate The Color Purple, and to rate Braveheart about average. Note that some movies—for example, Ocean's 11—and users—for example, Dave—would be characterized as neutral on these two dimensions. (Koren, 2009)

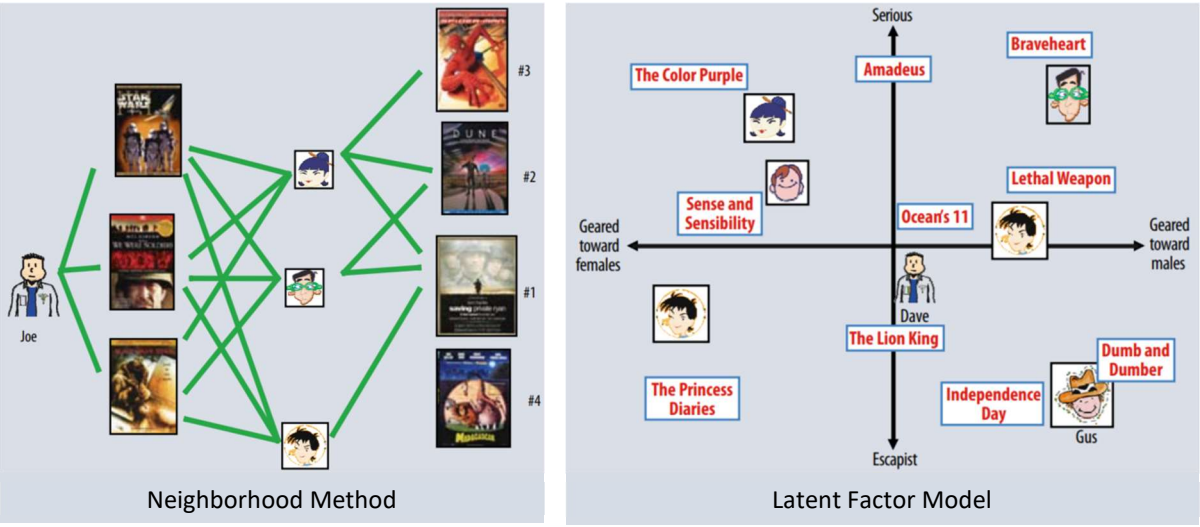

Figure 2 Two Collaborative Filtering Methods: Neighborhood Method and Latent Factor Model (Koren, 2009)

As we mentioned in Table 1, the real data is usually very large (a million-million matrix), while most neighborhood methods are model-free and only utilize small off-line computation, which is not feasible in the real case. Therefore, we will discuss more the latent factor model.

Latent Factor Model

Matrix Factorization

Some of the most successful realizations of latent factor models are based on matrix factorization. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. *This method has become popular in recent years by combining good scalability with predictive accuracy. In addition, they offer much flexibility for modeling various real-life situations.* (Koren, 2009)

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f. For a given item s, the elements of $q_s \in f$ measure the extent to which the item possesses those

factors, positive or negative. For a given user c, the elements of $p_c \in f$ measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $q_c^T \cdot p_u$, captures the interaction between user c and item s—the user's overall interest in the item's characteristics. This approximates user c's utility of item s, which is denoted by u_{cs}, leading to the estimate $\widehat{u_{cs}} = q_c^T \cdot p_u$.

The reconstruction process is very similar to singular value decomposition (SVD)[3]. But it might encounter a problem due to the sparseness caused by the high portion of missing value, give some missing value imputation for SVD might bring in unnecessary noise.

Negative Matrix Factorization (NMF)[4] is another decomposition technique where we try to split the matrix R into the product of two matrices U and V. Our objective here is to get the two matrices U and V, so that their matrix multiplication gives us the matrix R. Here R is typically very sparse.

Restricted Boltzmann Machine

As we mentioned in Figure 1, there's increasing popularity on the recommendation system in the last five years, part of the reason is the raising of machine learning, especially deep learning. So, we will also look at the most advanced combination of collaborative filtering and deep learning, the Restricted Boltzmann Machine (RBM)[5]. RBM is an energy-based model, and it uses Boltzmann distribution to picture the status of the neuron. "Restricted" means every visible node is only connected to every hidden node compare with an ordinary Boltzmann Machine, so the RBM into a bipartite graph.

RBM is a Markov Random Field Graph Model, trying to lean the hidden representation of the visible input graph. In the case of recommendation system, if all N users rated the same set of M items, we could treat each user as a single training case for an RBM which had M "SoftMax" visible units symmetrically connected to a set of binary hidden units. *Each hidden unit could then learn to model a significant dependency between the utilities of different items*. (Salakhutdinov, 2007)

Salakhutdinov, et al. provide three modifications to the base RBM model to fit the recommendation problem (Salakhutdinov, 2007) and the champion of the Netflix Prize open competition[6] rely heavily on this model. First, they use a special neural to represent the missing utilities as the input is a sparse

[3] Mathematical details: https://en.wikipedia.org/wiki/Singular_value_decomposition
[4] Mathematical details: https://en.wikipedia.org/wiki/Non-negative_matrix_factorization
[5] Network structure details and ordinary training algorithm(Simulated Annealing): https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine
[6] https://en.wikipedia.org/wiki/Netflix_Prize

vector, so the RBM can learn to use missing ratings to influence its hidden features, even though it does not try to reconstruct these missing ratings and it does not perform any computations that scale with the number of missing ratings. Second, based on the Netflix Prize Dataset, they take special extra information into account. They use a vector r to indicate which movies the user rated (even if these ratings are unknown). Third, as the current parameterization of $W \in R^{M\times K\times F}$ results in many free parameters, they factorize the parameter matrix W into a product of two lower-rank matrices A and B. In particular: $W_{ij}^{k} = \sum_{c=1}^{C} A_{ic}^{k} B_{cj}$.

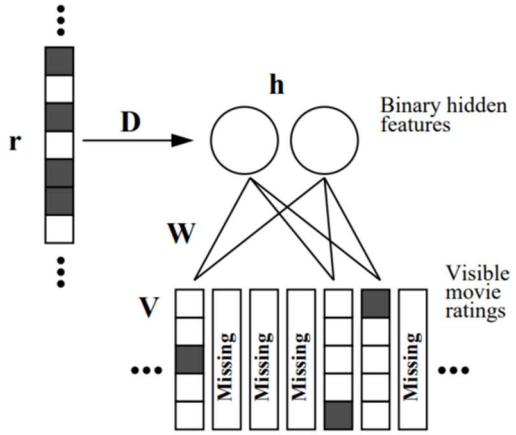

Figure 3 Conditioned Restricted Boltzmann Machine (Salakhutdinov, 2007)

Experiment

Dataset

We use Instacart grocery data[7] to conduct our experiment. The data contains 3.4m orders, 206k users and 50k products in total.

Result

The final experiment table is shown in Table 4. For the association rules algorithm (Apriori and FP Growth Algorithm), it's very hard to measure its performance because it needs some prior knowledge of the test dataset, i.e. it can only do prediction when we know part of the items in the order. Therefore, we create a new test dataset, which has split each order into two parts, the input and label. The input part contains two-third of the items in the order, while the label part contains the remaining one third. Then we match the frequency rules with the input part to see how many predicated items can hit the

[7] https://www.instacart.com/datasets/grocery-shopping-2017

label part. For the Collaborative filtering algorithm, we choose 80% of the total orders to build a $N \times M$ user-item training matrix and use the 100% of the total orders to build another test matrix, which is in $N \times M$ as well. And calculate the mean absolute error (MAE) of the predicted/reconstructed matrix and the test matrix.

Algorithm	*Train Time*	*Test Time*	*Performance*
Apriori	28 min	3 min	0.08 hit
FP Growth	5 min	3 min	0.08 hit
KNN (Cosine Similarity)	0 min	12 min	0.007 MAE
Singular Value Decomposition	Not feasible on sparse matrix		
Negative Matrix Factorization	8 min	≈0	0.006 MAE
Restricted Boltzmann Machine	21 min	≈0	0.003 MAE

Table 4 Experiment Result

Insight

From our experiment, we have the following findings,

- The objective definition is important in real machine learning problems.
  In the association rule, it's very hard to choose the support and confident cutoff value, because we cannot create a test data set to do a backtest. And without clear metrics, what we can do is only improving its computational efficiency, e.g., from Apriori to FP Growth, but help us to make a wiser decision is more important.
- Sometimes small math modification can fit new algorithms to new problems.
  In the past, RBM is mainly used for representation learning, but once we can put a SoftMax on the user-item utility, we can build a recommendation system based on RBM. And sometimes such new applications can bring unexpected results, e.g., a champion for Netflix Prize.
- Statistical representation is succinct.
  From Table 4, we can find that for the rule-based algorithm (Apriori and FP Growth), it generally takes a long time to learn the rules. However, if we use the statistical representation of the rules, like parameters in the RBM neural network, the training time might be similar, but the prediction process is faster.
- Offline computation is preferred in real cases.
  For algorithms without any offline computation (KNN), the prediction test is unacceptable in real life because it does use any offline computation to learn from data.

- Data Science spends 80% of time in preparing data.
  Most of the algorithms in the project already have an open-source package to use. However, which data to use, how to clean and transform the data into the right format can spend much more time than expected.

References

1. Adomavicius, G. &. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, 734-749.

2. J.B. Schafer, J. K. (2001). E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 115-153.

3. Koren, Y. R. (2009). Matrix factorization techniques for recommender systems. pp. 30-37.

4. Salakhutdinov, R. A. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th international conference on Machine learning.* ACM.