COMPARATIVE STUDY OF MODEL BASED COLLABORATIVE FILTERING RECOMMENDER SYSTEMS

A PREPRINT

Allen(Shuo) Lin
sl604@duke.edu

May 1, 2020

ABSTRACT

With the rapid growing volume of online information, recommender systems are leveraged more heavily than ever to bring customers satisfaction and convenience. Given its widespread adoption in the fields of e-commerce, entertainment and essentially education, the importance of recommender systems cannot be overstated. This report aims to provide a general algorithm-wise comparison across different types of model based collaborative filtering recommenders. In this report, four models of recommenders are being examined: K-Nearest Neighbors, Singular Value Decomposition, and Probabilistic Matrix Factorization. The comparisons are evaluated in aspects of execution time, accuracy, and scalability. Based on the comparison, this report also provides the rationales of how the underlying algorithmic structure of each model contributes to the result.

1 Introduction

Recommender Systems are everywhere in our lives. They personalize our shopping experience by suggesting us of potential things to buy on Amazon, interesting movies to watch on Netflix and even what music to listen on Spotify. They leverage our historical patterns and make prediction on items that we could potentially like. Items are ranked according to their relevancy and the most relevant ones are shown to the us. The most fundamental recommender systems are called collaborative filtering. collaborative filtering recommender systems are based on assumption that people like items similar to other items they like in the past, and items that are liked by others who have similar tastes. There are two types of collaborative filtering: memory based approach and model based approach. Memory based approach finds similar users based on cosine similarity or Pearson Correlation and then take the weighted average of ratings. Since no learning or optimization is being done in the memory based approach, such approach is easy to implement and the results obtained is highly self-explainable; however, unfortunately it does not handle sparsity or missing data in the data making it not scalable and thus not practical for most of the real-world problems. Model based approach, on the other hand, handles sparsity and missing data much better. Model based approach uses latent variable through machine learning to predict user ratings of unrated items. Despite the fact that the results could be intractable due to the use of latent variables, model based approach generally performs better than memory based approach since real-world recommendation data typically have many missing values.
In this report, two model based algorithms and one memory based algorithm are implemented and cross validated on the Movie-Lens dataset consisting of 1 million ratings. The evaluation of each model takes consideration of execution time, accuracy, and scalabilty. Amongst the three implemented models, the probabilistic Matrix Factorization model achieves the best overall performance.

2 Data

This report used the MovieLens 1M dataset which consists of 1 million user movie ratings from 6,000 users on 4,000 movies. The data is released on 2/2003 and has been used as a benchmark dataset from many researches. Below is the top 5 samples from MovieLens:

	userId	movieId	rating	timestamp
0	1	1193	5.0	978300760
1	1	661	3.0	978302109
2	1	914	3.0	978301968
3	1	3408	4.0	978300275
4	1	2355	5.0	978824291
5	1	1197	3.0	978302268

Figure 1: samples from MovieLens

3 Models

3.1 K-Nearest Neighbors

K-Nearest Neighbors is a clustering based algorithm which does not make any assumptions on the underlying data distribution. When KNN makes a prediction about a movie, it calculates the distance between that movie and every other movies in the data then ranks and returns the top k nearest neighbor movies whose ratings are then averaged to obtain the target rating. This KNN model uses twenty nearest movies to compute the target rating.

3.2 Singular Value Decomposition

In model based approach, we essentially would like to turn the recommendation problem into an optimization problem by viewing it as how accurate do our model predict the rating for items given a user. One metric that is commonly used for recommender systems is the Root Mean Square Error. By viewing it as an optimization problem, we are essentially minimizing RMSE on the known user ratings. SVD has a great property of minimizing reconstruction Sum of Square Error and RMSE happen to be monotonically related with SSE. Since SVD minimizes SSE, we know that it also minimizes RMSE. SVD factorizes matrix A as:

$$A = USV^T$$

$$\underset{m \times n}{A} = \underset{m \times m}{U} \; \underset{m \times n}{S} \; \underset{n \times n}{V^T}$$

$$\begin{pmatrix} x_{11} & x_{12} & & x_{1n} \\ & & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & & u_{m1} \\ & \ddots & \\ u_{1m} & & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_1 & \ddots & 0 \\ & \sigma_r & \ddots \\ 0 & & 0 \end{pmatrix} \begin{pmatrix} v_{11} & & v_{1n} \\ & \ddots & \\ v_{n1} & & v_{nn} \end{pmatrix}$$

Figure 2: SVD

where U and V are orthogonal matrices with orthonormal eigenvectors and S is the diagonal singular value matrix. The U matrix represents the relationship between users and latent factors. S is a diagonal matrix describing the strength of each latent factor, and V transpose indicates the similarity between items and latent factors.

3.3 Probabilistic Matrix Factorization

PMF incorporates the same approach as SVD and it is in fact a probabilistic extension of the SVD model; however, the main difference is that PMF treats the latent vectors as parameters in a Bayesian model.

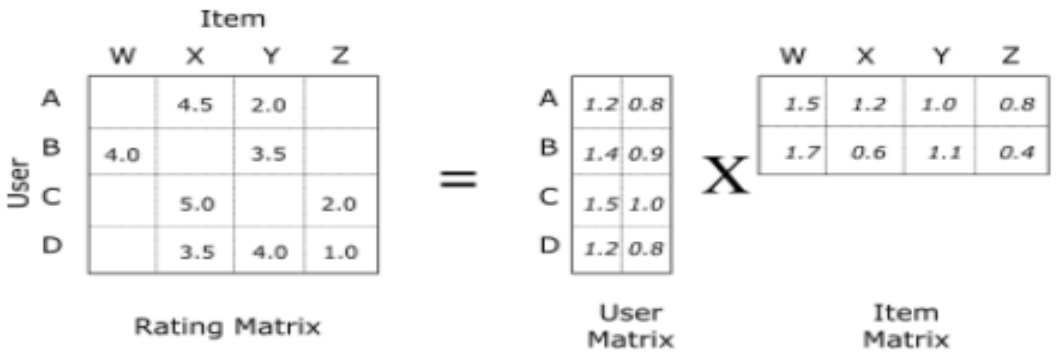

Figure 3: PMF

We define the conditional distribution over the observed ratings as:

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}[N(R_{ij}|U_i^T V_j, \sigma^2]$$

where R_{ij} represents a single entry in the user rating matrix R, U represent the decomposed User matrix, and V represent the movie latent vector matrix. We also place zero-mean spherical Gaussian priors on U and V.

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} N(U_i|0,\sigma_U^2 I)$$

$$p(V|\sigma_V^2) = \prod_{i=1}^{M} N(V_j|0,\sigma_V^2 I)$$

Now the log of the posterior distribution over the user and movie features is given by:

$$(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2) = -1/2\sigma^2 \sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 - 1/2\sigma_U^2 \sum_{i=1}^{N} U_i^T U_i - 1/2\sigma_V^2 \sum_{j=1}^{M} V_j^T V_j$$

$$-1/2(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}) \ln \sigma^2 + ND \ln \sigma_U^2 + MD \ln \sigma_V^2$$

Now maximizing the log-posterior over movie and user features is equivalent to minimizing the SSE objective function with regularization terms:

$$E = 1/2 \sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \lambda_U/2 \sum_{i=1}^{N} \|U_i\|^2 + \lambda_V/2 \sum_{i=j}^{M} \|V_j\|^2$$

where $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$. A local minimum of the above objective function can be found by performing gradient descent in U and V.

4 Results

Amongst the three models, KNN took the longest time to train and validate yet achieved the lowest accuracy. SVD took the shortest time to train but its accuracy is only a little higher than KNN. PMF, although took longer time than SVD to train, achieved best accuracy that is significantly lower than the other two models.

Model	RMSE	Training Time (sec)
KNN	.9229	867.94
SVD	.8736	268.2
PMF	.3302	682.62

Table 1: 5 fold cross validation results

5 Conclusion

The above cross validation results obtained are reasonable. KNN achieved lower accuracy than SVD and PMF since it is a memory based approach. There is no learning or optimization in the underlying algorithm. When bring trained on a dataset of 1 million samples, the KNN model took more than triple of the time as the SVD model. More importantly, since memory based approach does not handle sparsity well, the KNN model had the worst performance out of the three models tested. SVD had the shortest training time since the mechanic of the its underlying algorithm is well suited for such task. SVD is the standard benchmark method for decomposing a high dimensional and/or large matrix. The decomposition simply gives the minimal reconstruction SSE which is monotonically related with RMSE. It is not surprising that it achieved better accuracy than the KNN model. The PMF model is a probabilistic extension of the SVD model in which the latent variables are being treated as parameters in a Bayesian model. PMF performed much better than the KNN and SVD since it combines both Bayesian approach with shallow learning to achieve a local minimum. When using a sparse dataset with 1 million samples, both the the SVD and the PMF models are model based approach which has proven to be both faster to train and more accurate than the memory base approach KNN model.

6 Future Improvements

In this report, none of the recommender system used any state of art deep learning algorithm in the optimization process. Even though the PMF model, which achieved the best performance out of the three, used advanced optimization techniques, it only has a few neural network layers, which could be further improved by adding more layers with some alternation in its underlying structure.

References

Grover, Prince. "Various Implementations of Collaborative Filtering." Medium, Towards Data Science, 31 Mar. 2020, towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0.

Huang. "Introduction to Recommender System. Part 1 (Collaborative Filtering, Singular Value Decomposition)." Hacker Noon, 1 May 2020, hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75.

Mcleonard. "Mcleonard/Pmf-Pytorch." GitHub, github.com/mcleonard/pmf-pytorch/blob/master/Probability

Salakhutdinov, Ruslan, and Andriy Mnih. "Probabilistic Matrix Factorization." Probabilistic Matrix Factorization | Proceedings of the 20th International Conference on Neural Information Processing Systems, 1 Dec. 2007, dl.acm.org/doi/10.5555/2981562.2981720.

Sharma, Nikita. "Recommender Systems with Python-Part III: Collaborative Filtering (Singular Value Decomposition)." Medium, Heartbeat, 19 Feb. 2020, heartbeat.fritz.ai/recommender-systems-with-python-part-iii-collaborative-filtering-singular-value-decomposition-5b5dcb3f242b.