ISyE 6740 – Spring 2021
Project Report

Team Member Names: Christopher Dae-Hyun Kim @ ckim612 @ gtID#: 903751328

Association Rules as Sparsity Alleviation in Recommendation Systems

Problem Statement
Netflix held an open competition in 2007 inviting teams to beat their in-house recommendation system CineMatch by 10% with the root mean squared error (RMSE) as the evaluation metric [1]. Sophistication of models varied greatly; some teams had futilely approached the problem by extracting features, such as genres, cast, etc., outside of the published dataset, some used a simple average based approach that proved to be only marginally better than CineMatch, and the top performing teams submitted an ensemble of models [1].

The focus of this project will be to expand upon the singular value decomposition (SVD) algorithmic approach by three students Michael Harris, Jeffrey Wang, and David Kamm [1] using weighted, grouped averages derived from pairwise association rules [2] as pseudo imputations to the sparse utility matrix.

Data Source
Netflix had recently republished the dataset that was used during the competition on Kaggle. The data consists of seven files with a total of 2.13 GB in size. According to the documentation, there are over 100,000,000 ratings from over 480,000 users over 17,000+ movie titles. Ratings are from October 1998 to December 2005. The team will also use the popular MovieLens 100k dataset from GroupLens. Released on April 1998, the data consists of 100,000 ratings from 943 users on 1682 movie titles.

Data sources can be retrieved from the following links:
1. Kaggle: https://www.kaggle.com/netflix-inc/netflix-prize-data
2. MovieLens: https://grouplens.org/datasets/movielens/100k/

Due to lack of computational resources, an undetermined subset of the data from Netflix will be used whereas the entirety of the MovieLens data will be used.

Methodology
Data is originally in the following format:

User_id	Movie_id	Rating	Timestamp
1	1	5	874965758
1	2	1	876893171
1	3	2	878542960
2	1	3	876893119

Figure 1. Example of original data without preprocessing.

Data will then be processed to a utility matrix as seen in the following figure:

	I_1	I_2	I_3	I_4
U_1	5	1	2	
U_2	3	3	3	3
U_3	5		3	
U_4				5

Figure 2. Utility matrix M where U_n indicates users and I_m indicates movie titles.

SVD is the factorization of a real matrix $M \in \mathbb{R}^{n\,x\,m}$ where $n \leq m$:

$$M = U\,\Sigma\,V^T := U\Sigma\Sigma^T V^T,$$
$$M = [u_1, u_2, \ldots u_n] \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix} [v_1, v_2, \ldots v_m]^T.$$

The objective function [3] is defined as minimizing the sum of squared errors (SSE):

$$\min_{U,V,\Sigma} \sum_{i,j \in M} (M_{ij} - [U\Sigma\Sigma^T V^T]_{ij})^2$$

Since SVD is undefined for a sparse utility matrix M [3] as presented in the data of users and their respective ratings for movies seen in Figure 1, we must rewrite the SVD and its new objective function with the introduction of a regularization term by using a stochastic gradient descent optimization process [3,4]:

$$R = Q^T \cdot P \text{ where } M \approx R,\ U\Sigma \approx P, \text{ and } \Sigma V^T \approx Q.$$

$$\min_{p,q} \sum_{u,i \in k} (r_{ui} - q_i^T p_u)^2 + [\lambda_1 \sum_u \|p_u\|^2 + \lambda_2 \sum_i \|q_i\|^2\ ]\ s.t.$$
$$\varepsilon_{ui} = r_{ui} - q_i^T p_u$$
$$q_i \leftarrow q_i + \gamma(\varepsilon_{ui}\, p_u - \lambda q_i)$$
$$p_i \leftarrow p_i + \gamma(\varepsilon_{ui}\, q_i - \lambda p_u)$$

Furthermore, we can introduce bias terms in the above objective function [3]:

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

$$\min_{p,q} \sum_{u,i \in k} (r_{ui} - \hat{r}_{ui})^2 + [\lambda_1 \sum_u \|p_u\|^2 + \lambda_2 \sum_i \|q_i\|^2 + \lambda_3 \sum_i \|b_i\|^2 + \lambda_4 \sum_u \|b_u\|^2\ ],$$

where μ is the global mean of all movies, b_i is the bias for a specific movie, and b_u is the bias for a specific user.

This project will attempt to make the utility matrix less sparse by introducing/imputing weighted, grouped means using pairwise association rules as "real" data.

At a high level, pairwise association rule learning is commonly used for transactional data and can be interpreted as the determination of significant event pairs and their associated probabilities. Significant events or rules are determined by the $support, confidence$, and $lift$, where support and confidence thresholds are user defined.

$$rule = \{A \Rightarrow B\}$$
$$supp\{A \Rightarrow B\} = P(A \cap B)$$
$$conf\{A \Rightarrow B\} = P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$
$$lift\{A \Rightarrow B\} = \frac{P(A \cap B)}{P(A)P(B)}$$

We will then extend these significant rules derived from the user-item utility matrix M by calculating their item-item group averages and impute these as data into the appropriate positions. The following is the pseudo code algorithm that encompasses the project methodology:

Algorithm 1 SVD-A

$D \equiv$ Data of users, movies, and ratings
procedure ASSOCIATION RULE LEARNING(D, $supp$, $conf$, $lift$)
    Create transactional data X from D
    $X \equiv \{x_0, x_1, ...x_{m-1}\}$
    $A \equiv \{a_0, a_1, ...a_{n-1}\}$
    $T \equiv$ Table of movie pair counts
    $C \equiv$ Table of movie counts
    $T[a,b], C[a] \leftarrow 0 \; \forall a, b \in A$
    for every $x \in X$ **do**
        for every $\{a \in x, b \in x\}$ **do**
            $T[a,b] \leftarrow T[a,b] + 1$
            $T[b,a] \leftarrow T[b,a] + 1$
            $C[a] \leftarrow C[a] + 1$
            $C[b] \leftarrow C[b] + 1$
        end for
    end for
    for every $\{a \in A, b \in A\}$ **do**
        if $\frac{T[a,b]}{m-1} \geq supp$, $\frac{T[a,b]}{C[a]} \geq conf$, $\frac{P(a \cap b)}{P(a)P(b)} > lift$ **then**
            return a, b
        end if
    end for
end procedure
procedure SVD-A(D, (a,b))
    Create utility matrix M from D
    Perform ASSOCIATION RULE LEARNING procedure
    for every $i, j \in M$ **do**
        if $M[i,j] = (a,b) \wedge M[i,j] \neq$ NaN **then**
            $M[i,j] = \mu_{\mu_{j,i}}$
        end if
    end for
    Perform SVD on M
end procedure

Figure 3. Pseudo code for project methodology.

We will demonstrate this method explicitly by using Figure 2 as an example. The first step would be to map the user ratings of movies as transactions:

$$U_1 = \{I_1, I_2, I_3\},$$
$$U_2 = \{I_1, I_2, I_3, I_4\},$$
$$U_3 = \{I_1, I_3\},$$
$$U_4 = \{I_4\}.$$

Supposing the rule $\{I_1 \Rightarrow I_2\}$ is determined, support of this rule would be $supp\{I_1 \Rightarrow I_2\} = \frac{2}{4}$, confidence of the rule would be $conf\{I_1 \Rightarrow I_2\} = \frac{2}{3}$, and lift would be $lift\{I_1 \Rightarrow I_2\} = \frac{8}{6}$. Supposing the rule $\{I_3 \Rightarrow I_2\}$ is determined, the support, confidence, and lift of this specific rule would be equal to the rule $\{I_1 \Rightarrow I_2\}$. Because $lift\{I_1 \Rightarrow I_2\} > 1$ and $lift\{I_3 \Rightarrow I_2\} > 1$, the antecedents and consequents of the rules are dependent and positively correlated since the joint probability of A and B is greater than the product of the marginal probabilities.

Significant rules are determined by sequentially filtering by the support, confidence, and lift thresholds. In this example, the supports and confidences for both rules are higher than an arbitrary, user defined threshold of 10% and 50% respectively. We can now proceed to calculate the value for the pseudo imputation.

For user U_3, $\mu_{[I_2|I_1=5]} = 1$ and $\mu_{[I_2|I_3=3]} = 3$ and the average of the grouped averages would equal two. We would then insert this value into the utility matrix as seen below:

	I_1	I_2	I_3	I_4
U_1	5	1	2	
U_2	3	3	3	3
U_3	5	2	3	
U_4				5

Figure 4. Utility matrix M with 2 inserted into $M_{3,2}$.

With this new matrix, we will then run the SVD algorithm as mentioned previously, which we will now refer to as the SVD-Alleviation (SVD-A) algorithm.

Evaluation

The SVD-A algorithm was successfully applied to the entirety of the MovieLens dataset albeit with negligible results. Five models were trained using a manual cross validation scheme where five disjoint datasets were partitioned in order to determine model efficiency as a function of the support threshold. Confidence and lift thresholds were kept constant at 80% and 1.00 respectively, and number of factors used in the matrix factorization process was kept constant at 5.

Model	RMSE
Unbiased SVD	0.9385
Unbiased SVD-A	0.9321
Biased SVD	0.9319
Biased SVD-A	0.9264
Mean based SVD	1.0563

Figure 5. Table of models using MovieLens data.

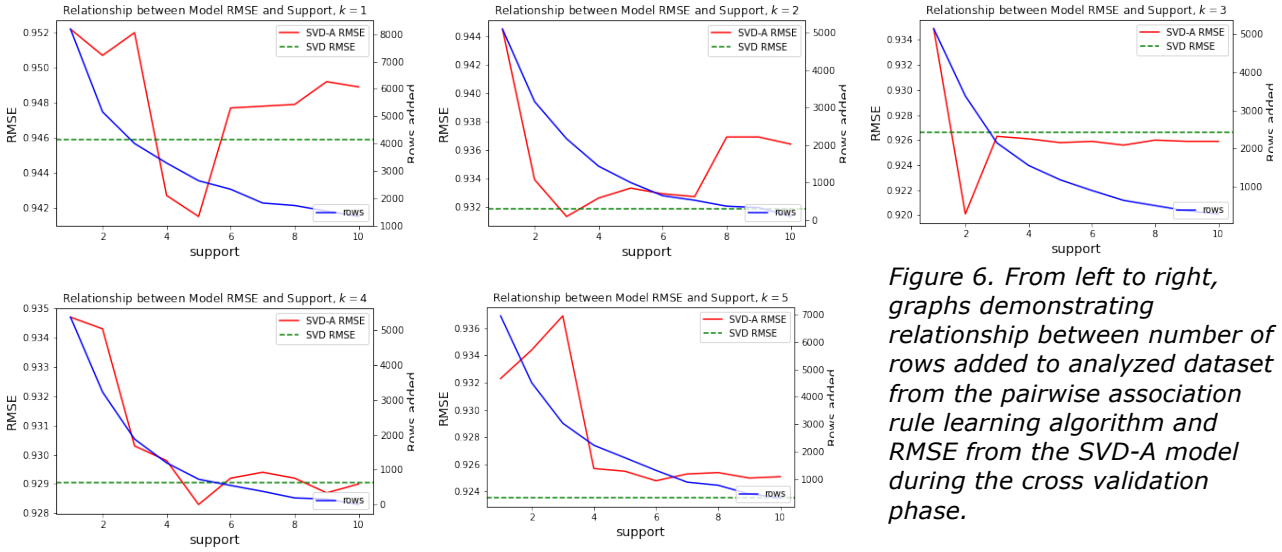

Figure 6. From left to right, graphs demonstrating relationship between number of rows added to analyzed dataset from the pairwise association rule learning algorithm and RMSE from the SVD-A model during the cross validation phase.

As we see in the graphs, we can interpret the strength of the SVD-A model as a function of high quality rows added to the original data or number of high quality values imputed into the utility matrix. In three out of five iterations of the cross validation phase, the result of the SVD-A model expectantly converges to the result of the SVD model as the utility matrix no longer gets imputed due to all rules being filtered out. Results suggest that SVD-A model strength as a function of the number of values imputed is a convex optimization problem.

After cross validation, the final biased SVD-A model used a support threshold of 2.5%, $\lambda = 0.04$, added 1874 rows, and RMSE = 0.9290. RMSE = 0.9353 was reported for the final biased SVD model.

Due to the lack of computational resources, the Netflix data could not be fully analyzed due to the sheer size and dimensionality problem. In order to combat the data size, random subsets of the data were taken. However, the dimensionality problem was still persistent as the pairwise association rule learning algorithm generated rather interesting results:

	movie_id	movie_id	support
0	1871831	238525	0.023449
1	238525	1871831	0.023449
2	1546547	1361124	0.023449
3	1361124	1546547	0.023449
4	1285516	1008801	0.023449
...	...	...	...
127	1538355	2395672	0.023449
128	79160	205023	0.023449
129	205023	79160	0.023449
130	1923803	119196	0.023449
131	119196	1923803	0.023449

	movie_id	movie_id	confidence
0	1593992	457661	100.0
1	457661	1593992	100.0
2	1593992	68175	100.0
3	68175	1593992	100.0
4	1593992	15218	100.0
...	...	...	...
2538387	623281	816576	100.0
2538388	816576	2083014	100.0
2538389	2083014	816576	100.0
2538390	623281	2083014	100.0
2538391	2083014	623281	100.0

	movie_id	movie_id	lift
0	1593992	457661	8529.0
1	457661	1593992	8529.0
2	1593992	68175	8529.0
3	68175	1593992	8529.0
4	1593992	15218	4264.5
...	...	...	...
3566063	623281	816576	8529.0
3566064	816576	2083014	8529.0
3566065	2083014	816576	8529.0
3566066	623281	2083014	8529.0
3566067	2083014	623281	8529.0

Figure 7. From left to right: support, confidence, and lift values of pairs using random subset (m=100,000),

The SVD-A model was trained using a support threshold of 0.02%. This support threshold filtered out more than 99.93% of all rules, ending with only 132 significant rules. Furthermore,

we can see a repeating value for the supports, confidence, and lift due to the sparse utility matrix. This observation was repeated for all subsets.

For exploratory purposes, the SVD-A algorithm was still applied although the results from the association rule learning phase was rather unintelligible and computationally taxing. After cross validation (k =5), the RMSE for the regular biased SVD model was 1.0526 whereas the RMSE for the biased SVD-A model was 1.0412.

It is the conclusion of the project that although SVD-A algorithm marginally improves the SVD algorithm, the method is computationally expensive and not cost effective. Time complexity for association rule learning is $O(2^{d+1})$ [5] where d is the number of attributes of a dataset. The linear search algorithm used in the imputation phase adds another $O(N)$ complexity.

Furthermore, the SVD-A model is too user dependent as the user must determine the support, confidence, and lift thresholds; the user must sift through large ranges of thresholds in order to optimize the model. As we saw in the cross validation phase for the MovieLens data, there is a distinct relationship between the number of transactions or rows added and quality of the SVD-A model. Too many values imputed and the model performs significantly worse than the SVD model; too little values imputed have no substantial meaning. In other words, there is no purpose in performing the association rule learning process if no values are imputed. Moreover, significant rules are highly dependent on the sparsity of the data as seen in the Netflix data.

However, the results show that if a user were to impute values, grouped averages using significant item-item rules derived from association rule learning is far superior than imputing mere means as the mean based SVD reported a RMSE of 1.0563 compared to the SVD-A RMSE of 0.9264 (Figure 5).

For future works, it would be interesting to formulate the convex optimization problem within the SVD-A algorithm to eliminate the user dependent nature. It is highly recommended to apply the SVD-A algorithm to the entirety of the Netflix data.

References

1. Leskovec J, Rajaraman A, Ullman J. Mining of Massive Datasets. 3rd ed. Cambridge University; 2020.

2. Agrawal R, Imieliński T, Swami A. Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD Record. 1993;22(2):207-216.

3. Leskovec J, Rajaraman A, Ullman J. Recommender Systems: Latent Factor Models [Slides]. CS246: Mining Massive Datasets, Stanford University.

4. Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems. Computer. 2009; 42(8):30-37.

5. Tahyudin I, Haviluddin H, Nanbo H. Time Complexity of A Priori And Evolutionary Algorithm for Numerical Assocation Rule Mining Optimization. IJSTR. 2019;8(11):483-485.