EE239 PROJECT 1

Collaborative Filtering

Amogh Param, Brandon Wu
brandonwu@cs.ucla.edu

1 INTRODUCTION

In this report we study a matrix representation of a Recommendation System that utilizes Collaborative Filtering. There is much industry interest in such systems; for example the Netflix suggestion system for its users may use a scheme similar to that presented here. The problem we will study in detail is the following: based on feedback data from n users indicating a rating of m items, we construct a $n \times m$ matrix R, where the (i, j) element corresponds to the rating given by user i on item j.

We would like to extrapolate this matrix to infer which unrated items that a user will likely enjoy. Therefore, we adopt the following construction: $R_{n\times m} = U_{n\times k}V_{k\times m} + \epsilon_{n\times m}$ where the rows of U are vectors that characterize a particular user, the columns of V characterize a particular item, and ε represents the error, parameterized by a factor k.

The factorization algorithm used throughout this report is the *Alternating Least Squares Method* which seeks to minimize the following cost function:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(r_{ij} - (UV)_{ij})^2 \qquad \text{(Equation 1)}$$

We will build the recommendation system based on a dataset consisting of 100K movie ratings collected by GroupLens[1].

2 PART 1: A SIMPLE FACTORIZATION USING ALS

Our Recommendation System uses the NMF MATLAB toolbox[2] to perform the matrix factorization that minimizes the cost function in Equation 1. In this section, we used the NMF tools to factorize R. The metric we used to determine the closeness of the factorization was the squared error (Equation 1). We only considered entries that of the prediction matrix that corresponded to actual prediction the user actually made, so Equation 1 was modified by an additional weight matrix $\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}(r_{ij} - (UV)_{ij})^2$ where $w_{ij} = 1$ if user i rated movie j and 0 otherwise. We also varied the value of parameter k to determine the optimal size of matrices U and V. Small values of k encode less information which makes the reconstruction of R less accurate but large values of k could result in over-fitting the dataset. The results for squared error versus parameter k are shown in Table 1 below:

Table 1: Squared Error vs k

k	10	50	100
sq. error	5.81E+04	4.39E+05	3.71E+05

Although the squared error decreased with increasing k, the difference is not significant.

[1] http://grouplens.org/datasets/movielens/
[2] https://sites.google.com/site/nmftool/

3 PART 2: A MODIFIED CONSTRUCTION USING NON-TRIVIAL WEIGHTS

In the second part, we apply the same cost function but this time, we reverse the roles of the R and W (weight) matrices in the factorization step. We again applied the ALS factorization algorithm for different values of parameter k and measured the squared error. The results are summarized below in Table 2.

Table 2: Squared error vs k

k	10	50	100
sq. error	5.71E+04	2.50E+04	1.18E+04

4 PART 3: 10-FOLD CROSS VALIDATION

In this section, we show results from the same system built using 10-Fold Cross validation. The original dataset is transformed via random permutation, then divided into a testing and training set. The training set is used to construct the R matrix which is factorized into U and V. This process is repeated for ten iterations (10 folds). The values of the prediction matrix $U \cdot V$ are validated against the ratings in the testing set. We define a new metric based on this prediction error shown below in Equation 2.

$$\sum_{i,j \in S} \left| r^p{}_{ij} - r_{ij} \right| \qquad \text{(Equation 2)}$$

Where S is the testing set and R^p is the prediction matrix defined by U and V. The results of the 10-Fold Cross Validation is shown below in Table 3.

Table 3: Ten-Fold Cross Validation Prediction Error

fold	error
1	1.5592
2	1.5476
3	1.5413
4	1.5685
5	1.537
6	1.5629
7	1.5686
8	1.538
9	1.5632
10	1.5688

The minimum prediction error occurred in the 5th fold and the maximum error occurred in the 10th fold. The average prediction error across all ten folds is $avg_e = 1.5555$.

This process is repeated for the weighted ALS factorization matrices from Part 2 (factorize using the {0, 1} matrix). These results are shown below (note that the errors were normalized by the weight matrix to be comparable with the previous results).

Table 4: Prediction Error for ALS on 1/0 Matrix

fold	error
1	2.4846
2	2.4791
3	2.4655
4	2.5008
5	2.456
6	2.4858
7	2.473
8	2.475
9	2.4787
10	2.4673

In this case, the 4th fold provided the largest Prediction error while the 5th fold had the lowest.

5 PART 4: PRECISION AND RECALL

Next, we apply the 10-Fold Cross Validation techniques developed in Part 3 to a classification problem where we conclude that ratings above a certain threshold are movies the user "Liked". We use this to quantify the Precision and Recall metrics which are defined as follows:

$$precision = \frac{tp}{tp+fp} \qquad \text{(Equation 3)}$$

$$recall = \frac{tp}{tp+fn} \qquad \text{(Equation 4)}$$

A true positive (tp) is defined as an element in the R matrix where $r_{ij} > t \; AND \; r^p{}_{ij} > t$ for some threshold t. A false positive (fp) is defined as an element of R where $r_{ij} \leq t \; AND \; r^p{}_{ij} > t$ and so on.

For each of the 10 folds, we sweep the threshold between 1 and 5 and obtain a value for precision and recall. We average these values across the folds and plot Precision vs Recall parameterized by threshold. This result is shown below in Figure 1.

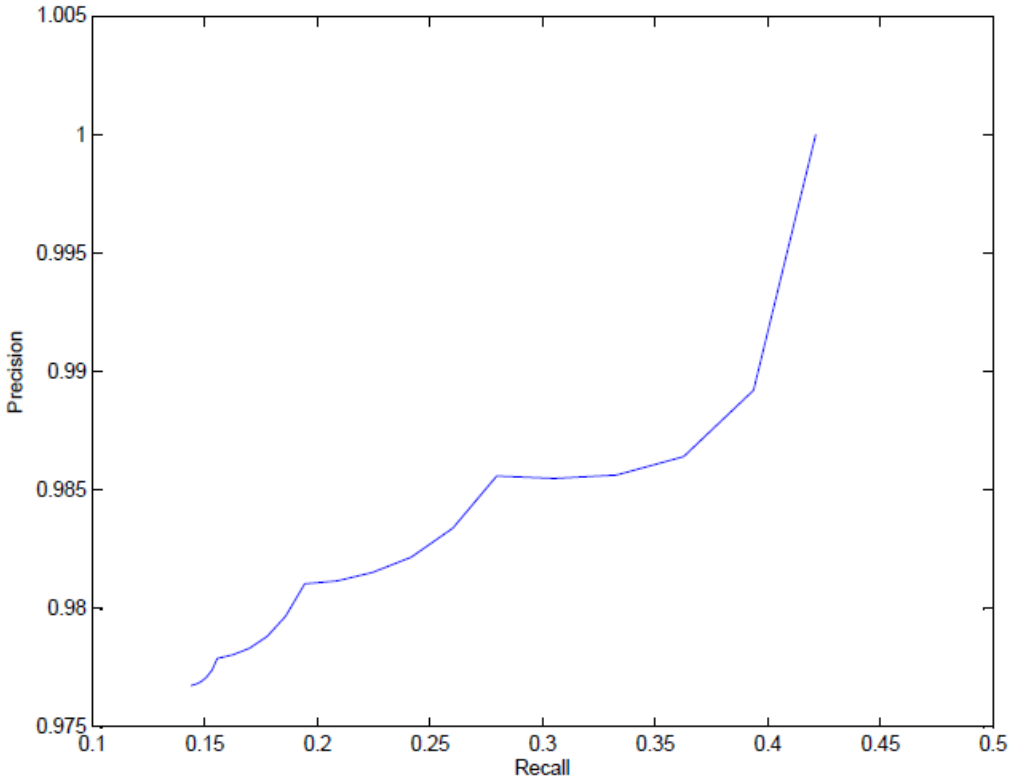

Figure 1: Precision vs Recall for Standard ALS Factorization

We do not repeat the process for the modified factorization using the non-trivial weight matrix from Part 2. Performing classification on this matrix would not produce useful results since we train on a matrix consisting only of 1's and 0's, to which classification of "likes" versus "dislikes" would be meaningless. Therefore, we proceed to the next part of the report, where we analyze the improvement on the ALS algorithm based on introducing regularization factor.

6 PART 5: IMPROVED ALS WITH REGULARIZATION

In this section, we introduce an improvement on the ALS cost function, called a Regularization term λ which reduces the effect of over-fitting the data. The new cost function is modeled by the following Equation:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}(r_{ij} - (UV)_{ij})^2 + \lambda \cdot (\sum_{i=1}^{n}\sum_{j=1}^{k} {u_{ij}}^2 + \sum_{i=1}^{k}\sum_{j=1}^{m} {v_{ij}}^2) \qquad \text{(Equation 5)}$$

The NMF code was modified to implement a new update rule based on this cost function and the results were re-evaluated for Parts 1 & 2 for $\lambda = 0.01, 0.1, 1$.

Table 5: Squared Error vs k and λ (Part 1)

λ\k	10	50	100
0.01	57420	57363	58712
0.1	25286	24878	26543
1	11636	11738	13549

The squared error is reduced compared to the case with no regularization factor. It also is reduced as λ increases.

Again, the process is repeated on the R matrix and W matrix in Part 2.

Table 6: lambda and k versus Squared Error

λ\k	10	50	100
0.01	65384	66265	66462
0.1	31888	31772	32271
1	17027	17329	18012

Surprisingly, although the squared error was reduced in Part 2 as a result of swapping the roles of the R matrix with W, these results indicate the opposite. The squared error with regularization terms increase when you use the weight matrix.

7 PART 6: MOVIE RECOMMENDATIONS

In the final part of the project, we utilize our previous results to determine a set of recommended titles for each user. The number of recommended titles is set by parameter L which is set by 5. We also apply a similar classification scheme as in Part 4, where we say that ratings above a threshold are considered items that the user liked, otherwise they are considered disliked. We can construct a matrix that applies this threshold as follows:

$$t_{ij} = \begin{cases} 1 & if\ r_{ij} > T \\ 0 & otherwise \end{cases}$$

For some threshold T.

The recommendation step works as follows: first we apply the ALS algorithm (with regularization factor λ), training on the {1, 0} matrix representation of R. Then, for each user, we sort the row vector corresponding to the predicted matrix R_{pred} by the prediction value. The first items in the sorted list correspond to the items the ALS algorithm finds to be the most likely for the user to like.

We declare a recommendation a true positive, or a "hit" if the actual rating the user provided is also above the threshold value. Thus, we can classify the prediction and actual R matrices using the formulation presented above using the classification matrix for both the predictions and the actual ratings:

$$true\ positives = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} \cdot t_{ij}^{p}$$

$$actual\ likes = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij}$$

$$false\ alarms = \sum_{i=1}^{n}\sum_{j=1}^{m}(\sim t_{ij}) \cdot t_{ij}^{p}$$

$$suggested\ likes = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij}^{p}$$

Therefore we can define the "hit rate" to be:

$$Hit\ Rate = \frac{true\ positives}{actual\ likes} \qquad \text{(Equation 6)}$$

$$False\ Alarm\ Rate = \frac{false\ alarms}{suggested\ likes} \qquad \text{(Equation 7)}$$

The results of the Hit Rate versus False Alarm Rate is shown below in Figure 3:

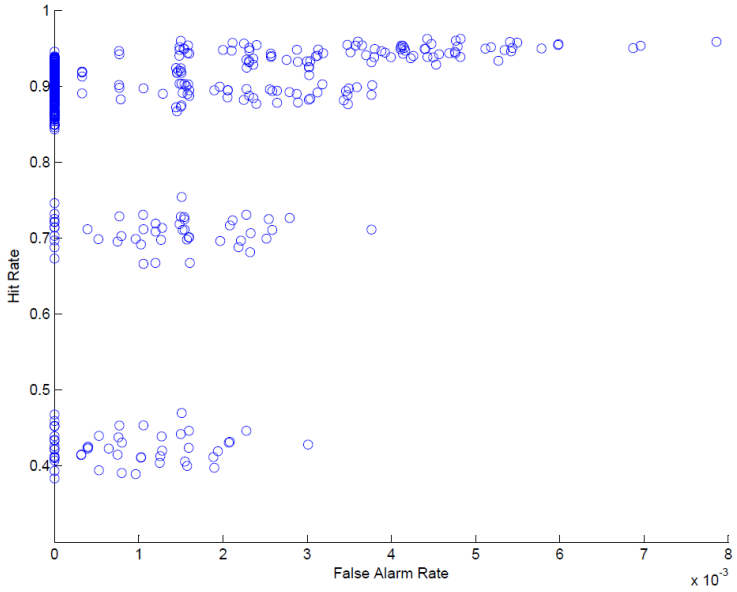

Figure 2: Hit Rate versus False Alarm Rate

It is not surprising that the function shows a stepwise relation when we parametrize based on threshold value. This is because the prediction matrix contains values with a fractional component but the actual ratings are only integral values. Therefore, thresholds at integral values will result in higher hit rates than those in between. Hit rate increases slightly as we increase L from 1 to 5 but the difference is not significant since both result in a sparse matrix with little data. The False alarm rate also increases as we decrease the threshold since this increases the number of items that we classify as "liked".

8 CONCLUSION

In this paper, we presented various modifications to a recommendation system that generates inferences for missing data points in a sparse matrix where column vectors represent user preferences. The Alternating Least Squares technique is an unsupervised learning technique which uses a least squares minimization to construct components U and V. We introduce modifications throughout the report and experiment with different techniques to obtain various metrics to characterize the validity of our techniques. The simplest is the least squared error, which quantifies the closeness of the factorized representation to the original R matrix. Next, we use 10-Fold Cross Validation to show that our design should be able to correctly qualify "new" data. We introduce a simple classification based on the inferred rating to quantify Precision and Recall. We show that adding a regularization term stabilizes the squared error of the factorization. Finally, we implement a practical system which actively introduces recommendations to a user with a reasonably high "hit rate".