Improving matrix factorization

Shrainik Jain (1323338)
shrainik@cs.washington.edu

Arunkumar Byravan (1222561)
barun@cs.washington.edu

Abstract

Content Recommendation systems (eg. Netflix, Amazon) try to predict user responses for content to make a decision on recommending content to the user. One way to model this is to consider it as a matrix approximation problem. It has been shown that this problem can be solved using matrix factorization techniques by assuming that the matrix is of low rank. Lee et al [1] relax this assumption and consider the matrix is locally low rank. They proceed to approximate this matrix as a smooth convex combination of low rank matrices each of which approximate local regions of the original matrix. We are interested in implementing this method and exploring ways to use context information to improve the predictions. We present our results for prediction accuracy using various types of context information and compare the results to the base algorithm.

1 Introduction

The Netflix Challenge, i.e., predicting a user's rating for a particular movie, has led to a significant amount of research in the field of Matrix factorization. This is because of the fact that the problem of predicting a user's rating of a movie, based on his ratings for other movies and ratings by other users for similar movies can be framed as solving a matrix approximation problem. If each user represents a row and each movie represents a column, then the problem is equivalent to finding the missing entries of this matrix. There are several approaches to solving this problem, most common among which are SVD [2] (minimization of Frobenius norm) and Compressed sensing (minimization of nuclear norm).

Most of the algorithms assume that the matrix is of a low rank. Lee et al [1] relax this by assuming that the matrix is only locally low rank i.e., specific sub-regions of the matrix are of low rank. This assumption is strictly weaker than the global low rank approximation and the global approximation problem can now be decomposed into finding good low rank approximations for the sub-matrices. This results in a fast and easily parallelizable framework that works well on large sparse datasets. In this work, we implement the algorithm proposed in [1] and test its performance on the MovieLens datasets. The algorithm in [1] does not use contextual information for learning. By adding additional information on movie similarity, user age, gender etc we add priors to the learning algorithm to make better predictions. We are interested in exploring the effect of this biasing on the performance of the method in [1].

2 Related Work

There is a huge body of related work on solving the matrix approximation problem for recommendation systems. Common strategies can be broken to two broad categories [3], *content filtering* and *collaborative filtering*. Content filtering requires trained analysts to rate each content on various metrics and then match these metrics with the user choices. The obvious problem with this approach is that such metrics are often incredibly hard to collect or not available at all.

Collaborative filtering takes into account the user's past behaviour in terms of his preferences and does not require explicit profiling of content. Two primary methods in collaborative filtering are *neighbourhood methods* and *latent factor methods*. Neighbourhood methods look for other users with similar preferences and suggest the user with content that these other similar users like. Latent Factor methods represent a user's preference as a combination of several hidden factors. The final preference is dependent on the amount each factor matters to the user and how a movie fares on each of those factors.

In light of the netflix challenge, the problem was to build a recommendation system based on collaborative filtering using latent factor methods. This breaks down to the matrix approximation problem. After Billsus et al. [4] initially proposed the use of SVD to solve collaborative filtering, there have been a number of similar approaches. Koren et al. [3] talk about using SVD while including the effect of user bias and temporal dynamics in their model. Jaggi et al. [5] propose a novel tuning free approach to solve the problem by bounding the nuclear norm. The winners of the Netfilx prize [6] [7] used an ensemble approach where the final prediction is a weighted combination of several methods like SVD, k-NN and Restricted Boltzmann Machines.

Given the scale of the data for the netflix challenge, the speed of prediction is also a very important factor and a number of works focus on finding distributed and parallizable solutions. Zhou et al. [8] use the Alternating-Least-Squares with Weighted-λ-Regularization technique to solve a form of incomplete SVD and focus specifically on parallelizability. Mackey et al. [9] provide a divide and conquer approach in which they break the problem into a number of smaller factorization problems and use SVD on each smaller problem. Lee et al. [1] use a similar approach but with a key difference that they use distance based kernels to find subsets of dataset to work on and the subsets can overlap. In our work, we plan to use the idea from Lee et al. [1] and incorporate contextual information to improve prediction accuracy.

Using contextual information in recommender systems has been proposed before. Adomavicius et al. [10] [11] research this extensively and give the notion of context pre-filtering, post-filtering and modeling, which are different paradigms of incorporating context into predictions. We will be using similar approach to use context provided in the MovieLens1M and MovieLens10M to see if biasing the model produces better results.

3 Theory

In many content recommendation problems, there are observed ratings of different content by many users and given a new user, we have to choose specific content to recommend based on our model of the user's preference. We can think of this preference as being encoded in the user's ratings of other content and the ratings of similar users. We can construct a matrix M of size $n_1 \times n_2$ (n_1 is the total *number of users* and n_2 is the total *number of different content*) with the observed ratings (set A). In practice, this matrix is usually sparse as we have ratings from only a few users on each content. The problem of predicting the user's preference for the remaining content can now be formulated as the problem of completing this matrix M i.e. we want to find suitable values for the unknown entries of the matrix M given the known set of ratings A.

The predominant approaches to solving this problem construct an approximation $\hat{M}$ of the matrix M by assuming that M (and therefore $\hat{M}$) is of low rank. This means that we can approximate the matrix M as:

$$M \approx \hat{M} = UV^T \; ; \; U \in \mathbb{R}^{n_1 \times r}, V \in \mathbb{R}^{n_2 \times r}; r << min(n_1, n_2) \tag{1}$$

It has been shown empirically that this assumption works well in practice [2], [3]. Two popular approaches to constructing this low rank approximation are:

- $\mathcal{H}_1$ *Incomplete SVD*:where we explicitly constrain the rank of the approximation to be r and
- $\mathcal{H}_2$ *Nuclear Norm minimization*:where minimizing the nuclear norm enforces low rankedness.

As an extension to these methods, Lee et al. [1] propose a relaxation of the low rank assumption. Instead, they posit that the matrix M is characterized by multiple low rank matrices each of which approximate M in a local region. Given a specific point (a, b) on the matrix, we have a mapping $\mathcal{T}(a, b)$ that approximates M in the **local** region around the point (a, b) with a low rank matrix.

Key to this is an assumption of a metric structure over the rows and columns of M (users and content respectively) with a specific distance metric $d((a,b),(c,d))$ measuring the similarity between users a, c and items b, d. The metric d defines the **local** region around any point and $\mathcal{T}(a,b)$ approximates M in this local region. The authors make a key assumption that this mapping $\mathcal{T}$ is slowly varying (Holder continuous) which allows them to use a limited number of matrices $\mathcal{T}$ to approximate M fully (otherwise, we need as many as $n_1 n_2$ samples to estimate $\mathcal{T}$ accurately). Additionally, the authors define a smoothing kernel $K_h^{(a,b)}$ based on d to smooth the contributions of other users and items while learning (a high value for a specific (user,movie) pair leads the algorithm to try to estimate it better). With this general idea in mind, we can look at local versions of the two algorithms mentioned above as they try to estimate $\mathcal{T}(a,b)$ as a low rank matrix $\hat{\mathcal{T}}(a,b)$ for a point (a,b):

$\mathcal{H}'_1$ *Incomplete SVD*:

$$\hat{\mathcal{T}}(a,b) = \arg\min_{\hat{\mathcal{T}}} \sum_{(a',b')\in A} K_h^{(a,b)}(a',b') \left([UV^T]_{(a',b')} - M_{(a',b')}\right)^2 \tag{2}$$

$$s.t. \quad rank(\hat{\mathcal{T}}(a,b)) = rank(UV^T) = r \tag{3}$$

$\mathcal{H}'_2$ *Nuclear Norm minimization*:

$$\hat{\mathcal{T}}(a,b) = \arg\min_{\hat{\mathcal{T}}} \left\|\hat{\mathcal{T}}(a,b)\right\|_* \tag{4}$$

$$s.t. \quad \sum_{(a',b')\in A} K_h^{(a,b)}(a',b') \left([UV^T]_{(a',b')} - M_{(a',b')}\right)^2 < \epsilon \tag{5}$$

This gives us a low rank approximate of M in a local region around (a,b). Repeating this process for a few different $(a,b) \in Q$ values gives us low rank approximates in different areas of M. Given a new test entry (c,d) we can predict the mapping around the point as:

$$\hat{\hat{\mathcal{T}}}(c,d) = \sum_{(a,b)\in Q} \frac{K_h^{(a,b)}(c,d)\hat{\mathcal{T}}(a,b)}{\sum_{(a',b')\in Q} K_h^{(a',b')}(c,d)} \tag{6}$$

This leads to a simple algorithm for Local LOw Rank Matrix Approximation (LLORMA) where we choose a random set of points Q and learn the low rank mappings $\hat{\mathcal{T}}$ around them using the local version of incomplete SVD ($\mathcal{H}'_1$) with additional regularization on the matrices U and V. As the points Q are chosen randomly, learning at each point can be done in parallel.

4 Adding Contextual information

Incorporating context information into predictions has been researched separately [10] [11], but most matrix factorization techniques do not make use of context. Increasing the rank r of the decomposition for incomplete SVD should give an effect similar to the addition of context as the number of latent features increase with r. But a larger rank results in increasing computational complexity and this becomes difficult for large datasets. Using context information, we should be able achieve similar results at lower rank values.

In LLORMA, similarity between the users and movies is used to compute a local region around a given anchor point to which the low rank decomposition is applied. At test time, this similarity is used to guide the prediction given a new user and movie pair. Therefore, the measure of similarity is critical to the performance of the algorithm (assuming that using similarity is a good idea in general) and we posit that biasing this measure with additional contextual information may give us better prediction performance. We look at combining this information with the original similarity measure to gauge its effect on prediction performance.

We looked at two methods for adding contextual information into LLORMA:

Method 1 : Biasing the kernel (K_h) using contextual information in addition to cosine similarity before training and using this biased measure for training and testing

Method 2 : Biasing the kernel after training and using it only for the testing phase (the original similarity measure from [1] is used for training)

Fig.1 shows a pictorial depiction of the methods. The Movielens-1M dataset has information on user age, gender, location and movie genre while the Movielens-10M dataset has information on movie genres. We do not test against the Netflix dataset as we could not find good contextual information (except rating date) for it.

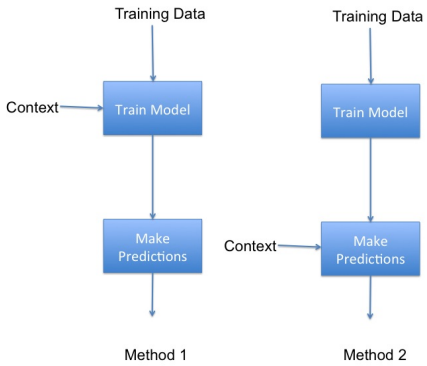


Figure 1: Methods of incorporating contextual information into LLORMA

5 Experiments

In the original paper [1], the authors tested the algorithm on the MovieLens and Netflix datasets. The distance metric d was computed as the angle between the singular vectors (rows & columns of U and V respectively) and K_h was chosen to be an epanechnikov kernel that decomposes over users and items.

5.1 Baseline

We implemented the LLORMA algorithm in MATLAB with the same settings as in the original paper and tested it on the MovieLens1M and MovieLens10M datasets. As expected, we were able to match the RMSE ranges shown in the original paper (the values are not exact due to randomization). Table 1 and 2 shows the RMSE values for LLORMA on the MovieLens datasets with 50 anchor points and varying rank. These results serve as the baseline for further comparison of the performance of LLORMA with the addition of contextual information.

5.2 Age

We first use age to bias the user similarity. For a given user, the similarity to other users based on age is computed in the following manner:

$$s_{age} = 6/(|Age_{user1} - Age_{user2}|) \;\; for, \;\; Age_{user1} \neq Age_{user2} \tag{7}$$

$$s_{age} = 1 \;\; for, \;\; Age_{user1} = Age_{user2} \tag{8}$$

and this is added to the kernel using:

$$K_h = K_h * s_{age} \tag{9}$$

The idea behind using the similarity measure as defined above is that MovieLen1M dataset has Age given as age groups, in groups like 18-25, 25-32, so on, and age value of 18 means that the user's age is the in group 18-25. The above measure considers same age group similarity as 1, adjacent age groups similarity as 0.85 and so on. We also tried similarity functions using exponential of age differences, which gives similar results. The resulting RMSE values with the addition of age information on the MovieLens 1M dataset for different rank r is shown in the second row of table 1. Training using the updated kernel (method 1) seems to perform worse than training normally and using the kernel at test time (method 2), but both of them do worse than the baseline meaning that adding age similarity actually worsens the performance of LLORMA.

		Rank-3	Rank-5	Rank-7	Rank-10
Baseline		0.8834	0.8584	0.8520	0.8499
Age	Method-1	0.8871	0.8585	0.8540	0.8510
	Method-2	0.8840	0.8586	0.8531	0.8508
Gender	Method-1	0.8840	0.8594	0.8538	0.8502
	Method-2	0.8841	0.8593	0.8534	0.8499
Movie Genre	Method-1	0.8797	0.8597	0.8526	0.8501
	Method-2	0.8793	0.8583	0.8533	0.8499
Combined	Method-1	0.8847	0.8623	0.8552	0.8500
	Method-2	0.8835	0.8594	0.8537	0.8500

Table 1: RMSE values for LLORMA with context added on the MovieLens-1M dataset. Results averaged over 20 tests.

5.3 Gender

Next we try using gender to bias the user similarity. For a given user, the similarity to other users based on gender is computed in the following manner:

$$s_{gender} = c \;\; for, \;\; Gender_{user1} \neq Gender_{user2} \tag{10}$$

$$s_{gender} = 1 \;\; for, \;\; Gender_{user1} = Gender_{user2} \tag{11}$$

$$where \;\; c \epsilon [0, 1) \tag{12}$$

and this is added to the kernel using:

$$K_h = K_h * s_{gender} \tag{13}$$

We ran the experiments with c in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 0.9]. The prediction acurracy (in terms of RMSE) worsens with decreasing c. This is expected as reducing c increases the bias of the model. The results for $c = 0.9$ are shown in third row of table 1.

5.4 Movie Genre

Movie Genre information is available for both the MovieLens1M and MovieLens10M datasets. Each movie entry has the names of the different classes it falls in (comedy, horror etc). To compute similarity based on movie genre, we created a binary vector (m^{genre}) for each movie where a 1 meant that the movie belonged to that genre and vice versa. The length of this vector is equal to the total number of genres in the dataset. We then combined it with the corresponding row in the V matrice as in section 3 using a simple horizontal merge to row vectors $m^{combined}$. The rationale behind this is that a rank r factorization yields a matrix V with r latent factors, we do not want to loose the information of these latent features. We then use the cosine similarity metric between these combined vectors to compute the similarity between different movies:

$$s_{genre} = 1 - \frac{2}{\pi} \arccos \left(\frac{m_j^{combined} \cdot m_k^{combined}}{\left\| m_j^{combined} \right\| \left\| m_k^{combined} \right\|} \right) \tag{14}$$

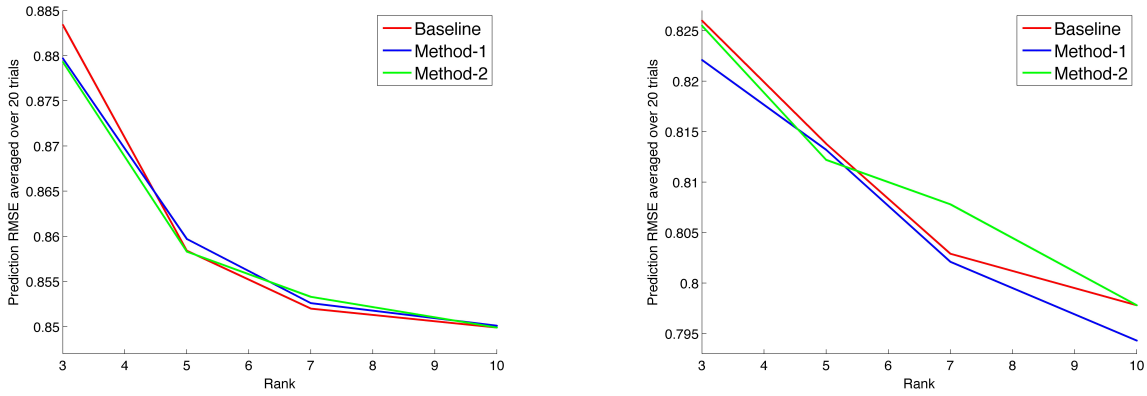
and add the result to the kernel as:

$$K_h = K_h * s_{genre} \tag{15}$$

Results for using genre similarity for the MovieLens1M and MovieLens10M datasets are given in tables 1 and 2 respectively. Fig.2a and Fig.2b depict these results pictorially. The prediction performance for both the methods is better than that of the baseline which tells us that the additional similarity information from movie genre is indeed useful for prediction. We also tried experimenting with similarity measure dependent on only the cosine similarity of m^{genre} binary vectors and not the $combined$ vector for the calculation of s_{genre}, but that biases the model too much and the results are worse than baseline algorithm.

		Rank-3	Rank-5	Rank-7	Rank-10
Baseline		0.8260	0.8138	0.8029	0.7978
Movie Genre	Method-1	0.8221	0.8132	0.8021	0.7943
	Method-2	0.8255	0.8122	0.8078	0.7978

Table 2: RMSE values for LLORMA with added movie genre similarity on the MovieLens-10M dataset. Results averaged over 20 tests



(a) MovieLens1M

(b) MovieLens10M

Figure 2: RMSE values for addition of movie genre information to LLORMA on MovieLens1M and MovieLens10M datasets averaged over 20 tests.

5.5 Combined

Finally we tried combining the all these context measures to analyze the corresponding results. This was a simple update to the kernel K_h as follows:

$$K_h = K_h * s_{age} * s_{gender} * s_{genre} \tag{16}$$

The results are shown in last row of table 1.

5.6 Interpretation of results

For locally low rank matrix factorization, use of user context like age, gender doesn't help in predictions, infact the base algorithm always beats it. This is expected as the ratings by users are not heavily dependendent on age or gender, but they would depend a lot on the genre of the movie. Our experiments suggest that for the genre context, both methods of using context work better than the base algorithm until a certain rank. Context information as used, doesnt help much high ranks, a possible reason for this is that at higher ranks, the number of latent factors considered by the base algorithm automatically take into account the genres as hidden features of a movie. But, for very large datasets (Netflix, MovieLens 10M), when it might be costlier to use higher rank computations, we can use context to make computations faster at a small cost to RMSE. Finally, the use of all the context filters at the same time gives the worst results, this is because the model then becomes heavily biased towards specific users and movies.

6 Limitations and Future Work

We show that it is possbile to improve predictions using context, but the improvement is not very significant. We believe that we used very simple methods to add contextual information and better kernels and more complicated measures might give better results. Also, not all datasets provide context information (eg netflix), so using context is either not possible or requires merging data sets from multiple sources. Finally, for smaller datasets, LLORMA works better without context information as we can increase r to much higher values without incurring the computational costs.

One major area for future work is in testing the performance on different datasets with different contexts and better similarity measures. We are also interested in an efficient parallel implementation of this algorithm. Finally, we are looking at developing an online version of this algorithm.

7 Conclusion

We have shown that the addition of contextual information improves prediction accuracy ***(albeit slightly)*** for the local low rank matrix factorization technique. Among the different contextual information we used, movie genre seems to be the most informative. These results support the claim that contextual information can be useful for content recommendation systems and combining this with matrix factorization techniques can give good performance upto a certain rank. Since higher rank factorizations can be computationally expensive, we show that we can get similar performance at a lower rank if we use context information.

Acknowledgments

Most of the work in this report is derived from the Local Low-Rank Matrix approximation paper [1] by Lee et al.

References

[1] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 82–90.

[2] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] D. Billsus and M. J. Pazzani, "Learning collaborative information filters." in *ICML*, vol. 98, 1998, pp. 46–54.

[5] M. Jaggi, M. Sulovsk *et al.*, "A simple algorithm for nuclear norm regularized problems," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 471–478.

[6] R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 95–104.

[7] J. Sill, G. Takács, L. Mackey, and D. Lin, "Feature-weighted linear stacking," *arXiv preprint arXiv:0911.0460*, 2009.

[8] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management*. Springer, 2008, pp. 337–348.

[9] L. Mackey, A. Talwalkar, and M. I. Jordan, "Divide-and-conquer matrix factorization," *arXiv preprint arXiv:1107.0789*, 2011.

[10] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Transactions on Information Systems (TOIS)*, vol. 23, no. 1, pp. 103–145, 2005.

[11] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*. Springer, 2011, pp. 217–253.