Netflix Challenge - 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 testing its performance on the MovieLens and Netflix datasets. Further, we would like to see if adding additional contextual information improves the performance of the 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 poerformance on the MovieLens and Netflix datasets (same as in [1]). The algorithm in [1] does not use contextual information for learning. By adding additional information on movie similarity (and possibly rating date), we can bias the learning algorithm. 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 2 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 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 the 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 the latent factor methods. This breaks down to the matrix approximation problem. After Billsus D. 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. [8] propose a novel easy to parallelize and tuning free aproach to solve the problem by bounding the nuclear norm. The winners of the Netfilx prize [9][10] 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 in 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 Y. et al. [5] use the Alternating-Least-Squares with Weighted-λ-Regularization technique to solve a form of incomplete SVD and focus specifically on the parallelizability. Mackey et al. [6] 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 given by Lee et al. [1] to achieve parallelization and incorporate contextual information to improve prediction accuracy.

3 Theory

In many content recommendation problems, we have 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*:

$$(U, V) = \arg\min_{U,V} \sum_{(a,b)\in A} \left([UV^T]_{(a,b)} - M_{(a,b)}\right)^2 \tag{2}$$

$$s.t. \quad rank(\hat{M}) = rank(UV^T) = r \tag{3}$$

where we explicitly constrain the rank of the approximation to be r and

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

$$(U, V) = \arg\min_{U,V} \left\| U\hat{V}^T \right\|_* \tag{4}$$

$$s.t. \quad \sum_{(a,b)\in A} \left([UV^T]_{(a,b)} - M_{(a,b)}\right)^2 < \epsilon \tag{5}$$

where minimizing the nuclear norm helps 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*:

$$\hat{\mathcal{T}}(a,b) = \arg\min_{\hat{\mathcal{T}}} \sum_{(a',b')\in A} K_h^{(a,b)}(a',b')\left([UV^T]_{(a',b')} - M_{(a',b')}\right)^2 \tag{6}$$

$$s.t. \quad rank(\hat{\mathcal{T}}(a,b)) = rank(UV^T) = r \tag{7}$$

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

$$\hat{\mathcal{T}}(a,b) = \arg\min_{\hat{\mathcal{T}}} \left\|\hat{\mathcal{T}}(a,b)\right\|_* \tag{8}$$

$$s.t. \quad \sum_{(a',b')\in A} K_h^{(a,b)}(a',b')\left([UV^T]_{(a',b')} - M_{(a',b')}\right)^2 < \epsilon \tag{9}$$

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:

$$\hat{\hat{\mathcal{T}}}(c,d) = \sum_{(a,b)\in Q} \frac{K_h^{(a,b)}(c,d)\hat{\mathcal{T}}(a,b)}{\sum_{(a',b')\in Q} K_h^{(a',b')}(c,d)} \tag{10}$$

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.

3.1 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. We used the same settings as in the original paper in our implementation. We have implemented most of the system in MATLAB and are testing it using a subset of the Netflix dataset and the MovieLens dataset. We are still in the process of debugging our implementation and we do not have any reasonable results at this point. One of the main problems that we have encountered is the huge size of the Netflix dataset which has caused us significant trouble with testing. We are confident that we will fix the issues with our system and have results in the next few days.

4 Proposed Future Work

The distance metric in [1] does not make use of any contextual information to measure user and item similarity (this is hinted in the paper). We are interested in adding additional contextual information to help in computing similarity between items. In the simplest case, we will try to use the movie genre or some expert added (by us) movie similarity imformation to modify the distance metric (and Kernel). If this works well, we will try to use additional user information like age, gender as well (MovieLens dataser). We are interested in seeing how this affects the performance of the algorithm, in terms of change in RMSE with respect to the original implementation. This adition of information to the algorithm is like a mixture of colloborative and content filtering and it would be exciting to see how this fares in terms on prediction accuracy. If time permits, we would also like to make an efficient parallel implementation of this algorithm.

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] Joonseok Lee et al., Local Low-Rank Matrix Approximation. *Proceedings of the 30th Internation Conference on Machine Learning, Atlanta, Georfia, USA, 2013.*

[2] A. Paterek, Improving Regularized Singular Value Decomposition for Collaborative Filtering. *Proc. KDD Cup and Workshop, ACM Press, 2007, pp. 37-42*

[3] Yehuda Koren et al., Matrix Factorization Techniques for Recommender Systems. *Cover Feature, IEEE Computing Society, 2009*

[4] Billsus, D. and Pazzani, M. J, Learning collaborative information filters. *Proc. of the Internation Conference on Machine Learning, 1998*

[5] Yunhong Zhou et al. Large-Scale Parallel Collaborative Filtering for the Netflix Prize.

[6] Mackey et al. Divide-and-Conquer Matrix Factorization.

[7] Rainer Germulla, Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. *http://www.mpi-inf.mpg.de/ rgemulla/publications/gemulla11dsgd-slides.pdf, Aug, 2011*

[8] Jaggi, M. and Sulovsky Marek, A Simple Algorithm for Nuclear Norm Regularized Problem. Proc. of the 27th International Conference on Machine Learning, Haifa, Israel, 2010.

[9] Bell, R., Koren, Y., and Volinsky, C. Modeling relationships at Multiple scales to improve accuracy of large recommender systems. *Proc. of the ACM SIGKDD Conference, 2007*

[10] Sill, J., Takacs, G., Mackey, L., and Lin, D. Feature-Weighted Linear Stacking.

[11] Salakhutdinov, R., Mnih, A., and Hinton, G., E. Restricted boltzmann machines for collaborative filtering. *ACM International Conference Proceeding Series, pages 791798. ACM, 2007.*