Comparing the Effect of Matrix Factorization Techniques in Reducing the Time Complexity for Traversing the Big Data of Recommendation Systems

Animesh Pandey and Siddharth Shrotriya

***Abstract*—In this fast trending technological era, data is growing very fast all around the globe. Today by analyzing large data sets, one can spot business trends, detect environmental changes, predict forthcoming social agendas and combat crime. This so-called "Big Data" analytics is challenging and dependent on time complexity. Amidst all this, we also know that space is becoming lesser day by day. In a time when computer imaging and web browsing demand the use of many different data, it is very important for us to be able to store all the types in reasonably sized matrices. A database is like a matrix but due to the growing data it will have a never ending addition of rows. To tackle this situation, we have tried to compare some matrix compression techniques that will help in efficient and fast working of analytic web recommendation systems and will decrease the effort of information retrieval from large data sets.**

***Index Terms*—Big Data, data mining, image processing, information retrieval, matrix factorization, recommendation systems, time complexity.**

I. Introduction

The tremendous growth in the amount of available information and the number of visitors to Web sites in recent years poses some key challenges for recommendation systems. Data is growing very fast. Today, by analyzing large data sets, one can spot business trends, detect environmental changes, predict forthcoming social agendas and combat crime. This so-called "Big Data" analytics is challenging [1]. In recommendation systems, we focus on the questions like "Which movie should I see?", "Which food should I try?" The system uses collaborative based recommendation techniques for producing recommendations. It is based on similarity of user likes or ratings.

Recommendation systems apply data analysis techniques to the problem of helping the users to find items they would like to purchase. This data is present in tremendous amount, so traversing through all the data would take up a lot of time and space. In most cases the number of columns in a database is constant but the tuples may keep on increasing. Hence, it is important to either lessen the amount of data to be parsed or to compress it. Compression is better option because lessening might affect the quality of final result. In order to emulate a big database we have used an image which be subjected to some compression algorithms. The Matrix Factorization is a very useful tool for image compression [2]. We can take an image which originally has a rank of R and store it in a reasonably good representation matrix that has only rank R', where R'< R. Sometimes, we can even cut-off more of the original image without losing clarity. In a time when computer imaging and web browsing demand the use of many images, it is very important for us to be able to store all the image types in reasonably sized matrices. In this paper, we analyze the effect of matrix factorization on large databases or images that can be useful for recommendation systems [3]. The large databases are M x N matrices so not all factorization techniques could be applied to them. But we have used a method which can be helpful even when the matrix is not a square matrix. We will be focusing on how to reduce the time taken or the time complexity of Matrix traversal. Image compression can also be useful in image searching in search engines as image matching takes a lot of time, compression can reduce the time taken for this process.

Manuscript received September 10, 2012; revised October 8, 2012.

Animesh Pandey is with the Department of Information Technology, and Siddharth Shrotriya is with the department of Electronic and Communication Engineering, Jaypee Institute of Information Technology, A- 10,Sector-62, Noida,UP, India (e-mail: animeshpandey@acm.org and siddharth@ieee.org)

II. Matrix Factorization Models

We have used five factorization models for our research which are:

1) Singular Value Decomposition
2) LU Decomposition
3) QR Decomposition
4) Schur Decomposition
5) Bayesian Probabilistic Matrix Factorization

In order to emulate a large database we have used a 1024 x 600 jpeg image which can resemble a sparse rating matrix of a recommendation system. One could say that there are 600 items and 1024 ratings/users. But only Singular Value Decomposition and QR Decomposition are applicable to a rectangular sparse matrix so for other techniques we made a square matrix by zero padding.

Fig. 1. 1024 × 600 Rectangular Matrix.

Only Singular Value Decomposition and QR Decomposition is applied to the Fig. 1 and rest other applied to Fig. 2.

A. Singular Value Decomposition

Applicable to: m-by-n matrix A, $A=UDV^H$ where D is a nonnegative diagonal matrix, and U and V are unitary matrices, and V^H denotes the conjugate transpose of V (or simply the transpose, if V contains real numbers only). Comment: The diagonal elements of D are called the singular values of A.

B. LU Decomposition

Applicable to a square matrix (Fig.2) A which is decomposed into $A=LU$, where L is lower triangular and U is upper triangular. This decomposition summarizes the process of Gaussian elimination in matrix form. Matrix P represents any row interchanges carried out in the process of Gaussian elimination. If Gaussian elimination produces the row echelon form without requiring any row interchanges, then P=I, so an LU decomposition exists.

C. QR Decomposition

Applicable to m-by-n matrix (Fig. 1) A, $A=QR$ where Q is an orthogonal matrix of size m-by-m, and R is an upper triangular matrix of size m-by-n. Unlike SVD, the QR decomposition provides an alternative way of solving the system of equations or without inverting the matrix A.

D. Schur Decomposition

Applicable to square matrix A, $A=VSV^T$ where A, V, S and are matrices that contain real numbers only. In this case, V is an orthogonal matrix, V^T is the transpose of V, and S is a block upper triangular matrix called the real Schur form. The blocks on the diagonal of S are of size 1×1 (in which case they represent real eigenvalues) or 2×2 (in which case they are derived from complex conjugate eigenvalue pairs).

E. Bayesian Probabilistic Matrix Factorization [4]

Low-rank matrix approximation methods provide one of the simplest and most effective approaches to collaborative filtering. Such models are usually fitted to data by finding a MAP estimate of the model parameters, a procedure that can be performed efficiently even on very large datasets. However, unless the regularization parameters are tuned carefully, this approach is prone to over fitting because it finds a single point estimate of the parameters. In this paper we present a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model in which model capacity is controlled automatically by integrating over all model parameters and hyper parameters.

F. Use of Zero Padding in Fig. 2

LU decomposition is applicable only to square matrices. So, in order to make the rectangular matrix of dimensions m x n, a square Zero matrix is added to the rectangular matrix that has the dimensions of Max(m, n) x Max(m, n). Before decomposition we know that which cells are zero so that, afterwards they can be removed and we can get a compressed matrix.

TABLE I: COMPARISON TABLE FOR THE MATRIX FACTORIZATION MODELS

Factorization Model	Matrix Dimension	Matrix Type	Rank (R) before Decomposition	Rank (R') After Reconstruction	Correlation with original image	Rank Reduction
Singular Value Decomposition	600 X 1024	Rect	602	21	0.9896	96.5%
LU Decomposition	1024 X 1024	Square	602	438	0.9854	27.24%
QR Decomposition	600 X 1024	Rect	602	150	0.9890	75%
Schur Decomposition	300 X 300	Square	300	200	0.8551	33.33%
Bayesian Probabilistic	400 X 600	Rect	400	30	0.9820	92.5%

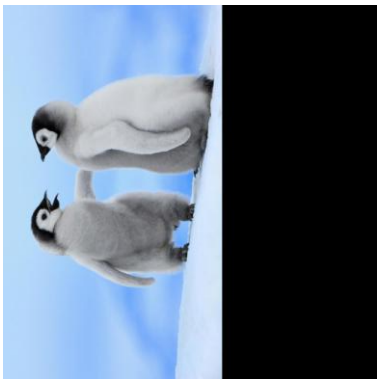


Fig. 2. 1024 × 1024 Square Matrix with Zero Padding.

III. EXPERIMENTAL SETTING

A. Dataset

In our research, we have used a jpeg image that simulates a large database of 600 items and 1024 users that is mostly the real time scenario in case of large datasets. The pixel color values that range from 0-255 can be assumed to be the ratings that a user has given to a product as this is a recommendation system dataset. Two more datasets that are cropped parts of Fig. 1.

There are two datasets - Fig. 1 and Fig. 2.

However, in research one cannot be sure whether the dataset used will be feasible for further operations.

B. Tools for Matrix Factorization

We have programmed all the matrix factorization models in Matlab 2010a software. They have specific functions where a matrix is fed to it and it returns its decomposed matrices. But as we were using an image dataset so we had to use some image processing techniques, functions of the Image Processing Toolbox and Statistics Toolbox of Matlab [5]. Some of the functions used were- [U,S,V] = svd(Matrix), [L,U] = lu(Matrix) etc.

IV. RESULTS AND DISCUSSIONS

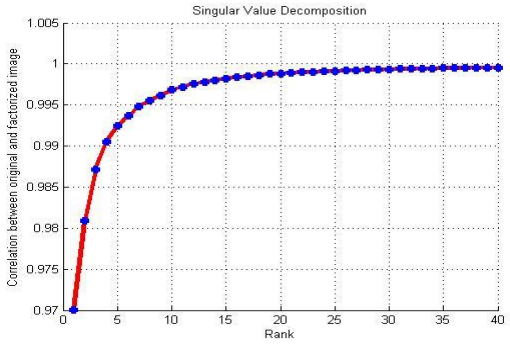
We had set the minimum correlation for the images to be 0.98.

Following are the variations of Correlations with the Matrix ranks:

1) Singular Value Decomposition

1 < Rank < 40

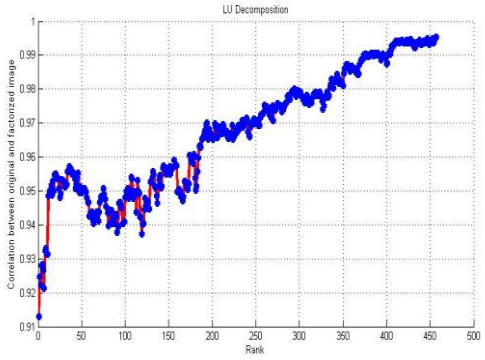
0.97 < correlation < 0.996



2) LU Decomposition

1 < Range < 457

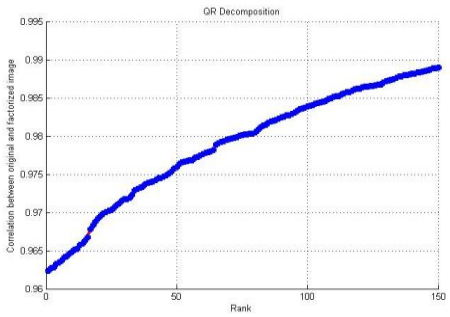
0.915 <Correlation < 0.995



3) QR Decomposition

1 < Range < 150

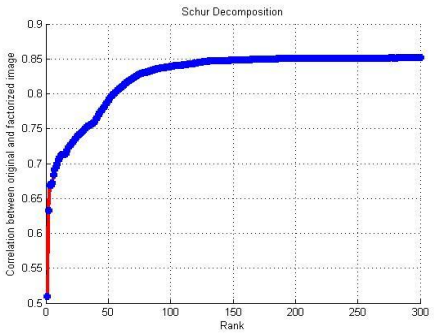
0.962 < Correlation < 0.99



4) Schur Decomposition

1 < Rank < 40

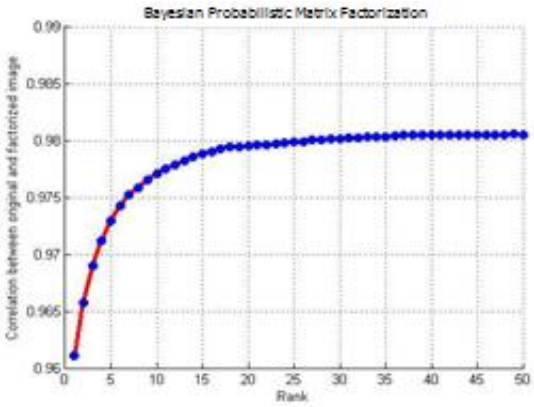
0.97 < correlation < 0.996



5) Bayesian Probabilistic Matrix Factorization

1 < Rank < 40

0.97 < correlation < 0.996



One can clearly see that the variation of correlation with rank is smoother in cases of a rectangular matrix. The square matrix decomposition has a lot of noise as a result of which one cannot be sure whether the data will remain the same after factorization.

In Singular Value Decomposition the graph is linear initially after words tends towards 1.00 correlation which means that at a very less rank one can be sure that the data is still the same even after factorization. Similarly in QR decomposition the variation is also a bit linear but not as good as that of SVD [6].

But by these observations one can be sure that SVD is best decomposition model for a large database and QR decomposition could be taken as the second option and LU decomposition should be avoided.

According to Table1, the rank reduction of SVD is nearly 96% and that of QR is about 75%. This indicates that SVD can help traversing the database in lesser time as compared to other two models. This will increase efficiency of the recommendation systems [7].

Those values are acceptable. We would say that in case of a very large database SVD could prove fruitful and in smaller database QR can be used. For even smaller databases Schur can be used as it gave a better result only when the database was reduced. The result of Bayesian Probabilistic model is nearly similar to that of SVD [8].

V. Conclusions

Recommendation systems generally deals with large data and hence face a problem called “Big Data”. One way to cope with this problem is compression. Image compression is a very good way to emulate the effect of matrix factorization on large datasets. Rating matrix of a recommendation system is a rectangular matrix and Singular Value Decomposition, QR decomposition and Bayesian Probabilistic MF prove to be the best for such a matrix. Schur decomposition is good for smaller lesser sparse square matrices. LU is for Square matrices but due to introduction of noise, it is not reliable and moreover the rank reduction is minimal for this one. Application of such models in web based recommendation systems can make them accurate as well as more efficient.

REFERENCES

[1] I. H. Witten and E. Frank, *Data mining: Practical Machine Learning Tools and Techniques*, 3rd Edition. San Francisco: Morgan Kaufmann, 2011.

[2] G. Strang, *Introduction to Linear Algebra*, 3rd Edition, Wellesly-Cambridge, 2005

[3] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," presented at ACM WWW10 Conference, May, 2001.

[4] R. Salakhutdinov and A. Minh "Bayesian Probabilistic Matirx Factorization using Markov Chain Monte Carlo," ACM, 2008

[5] D. F. Griffiths, *An Introduction to Matlab*, 2005

[6] Y. Koren, "Matrix Factorization techniques for Recommender System," *IEEE*,2009

[7] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "On the Gravity recommendation system," in *Proc. of KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 22-30, San Jose, CA, USA, 2007

[8] T. Hofmann, "Latent semantic models for collaborative Filtering,"*ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89-115, 2004.