
A two-dimensional decomposition approach for matrix completion through gossip

Mukul Bhutani

Amazon.com

mbhutani@amazon.com

Bamdev Mishra

Amazon.com

bamdevm@amazon.com

Abstract

Factoring a matrix into two low rank matrices is at the heart of many problems. The problem of matrix completion especially uses it to decompose a sparse matrix into two non sparse, low rank matrices which can then be used to predict unknown entries of the original matrix. We present a scalable and decentralized approach in which instead of learning two factors for the original input matrix, we decompose the original matrix into a grid blocks, each of whose factors can be individually learned just by communicating (gossiping) with neighboring blocks. This eliminates any need for a central server. We show that our algorithm performs well on both synthetic and real datasets.

1 Introduction

The problem of approximating a matrix by decomposing it into two low rank factor matrices is useful in solving many machine learning problems [1] including that of recommendation systems [2]. The problem of building a recommendation system involves having a large, sparse matrix \mathbf{X} of dimensions $m \times n$ and finding its low rank decomposition $\mathbf{X} = \mathbf{U}\mathbf{W}^T$ where \mathbf{U} and \mathbf{W} are non sparse matrices of dimensions $m \times r$ and $n \times r$ respectively and $r \ll m, n$. The product of these matrices can then be used to find the missing entries of \mathbf{X} . Since recommendation systems generally deal with user data, the security aspect may also become a major concern as one doesn't want to compromise the private data of users.

The problem of matrix completion is treated as an optimization problem and can be solved using gradient search [3, 4]. Parallel versions of gradient search have been used but they still require a central server [5, 6]. This dependency on a central server during most part of learning is one which we intend to eliminate. [7] follows an approach in which the input matrix is decomposed (as groups of columns) into l parts which were all processed by different agents. Each agent i estimates its own version of the original matrix, \mathbf{X}_i , as $\mathbf{U}\mathbf{W}_i^T$. Also, the matrix \mathbf{U} has to be synchronized between all the agents after each round of iteration. And thus each agent has the same (and also complete) view of \mathbf{U} after each iteration. In the approach followed by [8], in the decomposition $\mathbf{X} = \mathbf{U}\mathbf{W}^T$, each row of \mathbf{X} and \mathbf{U} is stored in different nodes and a public matrix \mathbf{W} is exchanged between them. Random walks are performed to bring convergence in \mathbf{W} . However, here too a single agent takes care of the complete row.

In contrast to the works discussed above, we present a decomposition strategy in a way such that the matrix is decomposed not only row wise [9], but also in a column wise fashion. We decompose the matrix \mathbf{X} into a two dimensional rectangular grid such that each decomposed part (which we henceforth call a block) can be factorized into its own local, row rank factors \mathbf{U} s and \mathbf{W} s. These blocks can then be processed by different agents. The individual blocks just *gossip* with their neighbors, trying to reach consensus and thus no communication happens with a central server during the learning phase. Once the learning is done, a final culmination of \mathbf{U} s and \mathbf{W} s is performed.

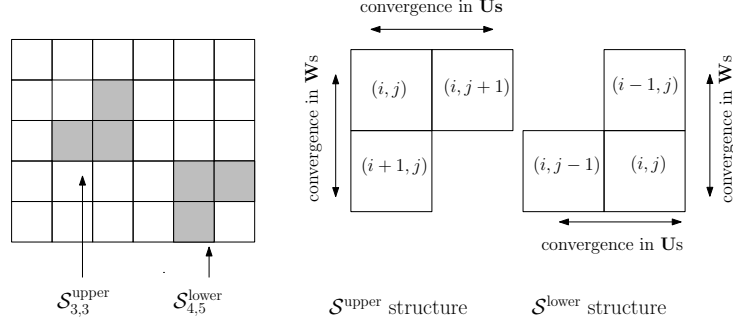


Figure 1: General decomposition of a matrix \mathbf{X} into a grid of 5×6 blocks. Two particular structures, $\mathcal{S}_{45}^{\text{upper}}$ and $\mathcal{S}_{33}^{\text{lower}}$, are highlighted. If \mathbf{X} had dimensions 500×600 , then each of the 5×6 block would have 100×100 entries.

2 Decomposition pattern

The input matrix \mathbf{X} is decomposed into a $p \times q$ dimensional rectangular grid of blocks (Figure 1). Each block can be referred by using the indices i and j corresponding to its row and column in the decomposition respectively. Each \mathbf{X}_{ij} can then be factored as $\mathbf{U}_{ij} \mathbf{W}_{ij}^{\top}$ as usual. We try to learn \mathbf{U} s and \mathbf{W} s corresponding to each of these blocks and then in the end, combine them appropriately to form universal \mathbf{U} and \mathbf{W} which can then be used to find the missing entries of the original matrix \mathbf{X} . These individual blocks gossip with their neighbors (blocks sharing an edge) and try to reach a consensus. Each row tries to reach to a consensus in terms of \mathbf{U} and each column tries to reach a consensus for \mathbf{W} . All these \mathbf{U} s and \mathbf{W} s are appended together to form the universal \mathbf{U} and \mathbf{W} which represents factors of original matrix \mathbf{X} . This pattern of communication results in a natural structure of a group of blocks, which can be thought of as gossiping. Let's call these groups of blocks or structures as $\mathcal{S}^{\text{upper}}$ and $\mathcal{S}^{\text{lower}}$ (shown in Figure 1). We call one of the blocks of the structure as pivot block indexed as (i, j) and the other two blocks are indexed relatively. Each block can belong to one or more structures depending on its position in the grid. Each of these structures is an independent computational unit and thus any two non overlapping structure can potentially be processed independently by different agents.

3 Problem formulation

We model the matrix completion problem as an optimization problem and the objective function of our two dimensional decomposed formulation can be derived by doing the analysis of our $\mathcal{S}^{\text{upper}}$ and $\mathcal{S}^{\text{lower}}$ structures. For $\mathcal{S}^{\text{upper}}$, for blocks (i, j) and $(i + 1, j)$, we try to bring convergence between their \mathbf{W} s and for the blocks (i, j) and $(i, j + 1)$ we try to bring the convergence to their \mathbf{U} s. We define the cost of a structure as comprising of two components: f and d . The f cost component of a block measures how close it is to the original matrix and the d cost component measures the consensus between two adjacent \mathbf{U} s (denoted as d^U) or \mathbf{W} s (denoted as d^W). For a block indexed at (i, j) it can be written as:

$$f_{ij} = \|\mathbf{X}_{ij} - \mathbf{U}_{ij} \mathbf{W}_{ij}^{\top}\|_F^2, d_{ij}^U = \|\mathbf{U}_{ij} - \mathbf{U}_{i+1,j}\|_F^2, \text{ and } d_{ij}^W = \|\mathbf{W}_{ij} - \mathbf{W}_{i,j+1}\|_F^2, \quad (1)$$

where $\|\mathbf{Z}\|_F^2$ denotes the square of the Frobenius norm of \mathbf{Z} . Consequently, the total cost (g) for a structure turns out to be:

$$g^{\text{structure}} = f_{\text{(for all the three blocks)}} + \rho d^U + \rho d^W,$$

where ρ is the *weight* factor. Hence, the total cost for an $\mathcal{S}^{\text{upper}}$ structure turns out to be:

$$g_{ij}^{\text{upper}} = f_{ij} + f_{i+1,j} + f_{i,j+1} + \rho \|\mathbf{U}_{ij} - \mathbf{U}_{i+1,j}\|_F^2 + \rho \|\mathbf{W}_{ij} - \mathbf{W}_{i,j+1}\|_F^2. \quad (2)$$

For $\mathcal{S}^{\text{lower}}$, we can derive the costs in similar fashion. For a decomposition of matrix \mathbf{X} into $p \times q$, our end goal is to minimize the sum of costs for all $\mathcal{S}^{\text{upper}}$ and $\mathcal{S}^{\text{lower}}$ possible, i.e.,

$$\min_{\mathbf{U}_{ij}, \mathbf{W}_{ij}} \sum_{i=1, j=1}^{p, q} g_{ij}^{\text{upper}} + g_{ij}^{\text{lower}} + \lambda \|\mathbf{U}_{ij}\|_F^2 + \lambda \|\mathbf{W}_{ij}\|_F^2, \quad (3)$$

1	2	2	2	1
1	2	2	2	1
1	2	2	2	1
1	2	2	2	1
1	2	2	2	1
1	2	2	2	1

(a) The relative number of times a block is selected while calculating the gradient of d^U .

1	1	1	1	1
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
1	1	1	1	1

(b) The relative number of times a block is selected while calculating the gradient of d^W .

1	3	3	3	2
3	6	6	6	3
3	6	6	6	3
3	6	6	6	3
3	6	6	6	3
2	3	3	3	1

(c) The number of times a particular block gets repeated while calculating gradient for f .

Figure 2: Relative frequencies of selection of different blocks for a grid of size 6×5 .

where the cost for g_{ij}^{upper} (and similarly g_{ij}^{lower}) can be seen from (2) if a S^{struct} is valid or is 0 otherwise and λ is the regularization parameter added according to [10].

4 Algorithm

Our basic online sequential algorithm in Algorithm 1 is very simple and uses the stochastic gradient descent (SGD) algorithm for (3). We have an input matrix \mathbf{X} of dimensions $m \times n$, which we divide into a grid of $p \times q$ blocks. Here p and q are the hyper parameters which would govern how one wishes to distribute the data. This in turn can depend upon the number of agents which one wishes to employ. Each of these blocks can be factored into corresponding \mathbf{U}_{ij} and \mathbf{W}_{ij} having rank r . These \mathbf{U} s and \mathbf{W} s are initialized randomly. We then randomly sample a structure out of the various possible ones (details of which were described in Section 2), `updateThroughSGD` calculates the gradient and updates the corresponding three \mathbf{U} s and \mathbf{W} s (corresponding to the three blocks comprising that structure). The process of sampling and updating parameters is repeated until convergence is reached. To update the λ [10] we use $\gamma_t = a/(1 + (bt))$, where t is the number of iterations and a, b are scalars. After Algorithm 1 has converged, all the \mathbf{U} s and \mathbf{W} s are finally combined to form \mathbf{U} and \mathbf{W} of size $m \times r$ and $n \times r$, respectively.

Normalizing representations of blocks. Owing to our methodology of decomposition, the number of structures of which a block may be a part of is different for various blocks. Since, finally in (3) we want all the blocks to have equal representation, we multiply a coefficient to each block to normalize the times a block may be selected for an update. The relative frequency of a block getting selected in given in Figure 2 and thus the coefficients we use are the inverse of it.

Algorithm 1: Basic update algorithm via SGD

input : Decomposed blocks for \mathbf{X} and rank r .
output : \mathbf{U} s, \mathbf{W} s.

```

1 Initialize all  $\mathbf{U}$ s and  $\mathbf{W}$ s.
2 while convergence is not reached do
3    $S^{\text{struct}} =$  randomly pick a valid structure.
4    $[\mathbf{U}s, \mathbf{W}s] =$  updateThroughSGD( $\mathbf{X}s, S^{\text{struct}}$ ).
5   Check for convergence.
6 end

```

5 Numerical experiments

To demonstrate the efficacy of our algorithm, we perform two sets of experiments. In the first set, we run our algorithm on synthetically generated data sets and calculate the cost. For the second set of experiments, we demonstrate our algorithm working on some popular public datasets. We use the root mean squared error (RMSE) to gauge the performance of our algorithm.

Experiments on synthetic data sets. We randomly generate a synthetic matrix subject to a rank constraint. Of this we mask majority of the elements thus generating our train matrix. Similarly, a test matrix is also generated by choosing a fraction of elements in the original matrix which got masked and weren't selected for generating the train matrix and is used for evaluation.

Table 1: Parameters used for various experiments.

Parameter	Exp#1	Exp#2	Exp#3	Exp#4	Exp#5	Exp#6
ρ (weight factor)	1e3	1e3	1e3	1e3	1e3	1e3
λ (regularization parameter)	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9
$p \times q$ (dimensions of decomposed grid)	4×4	4×5	5×5	6×6	5×5	5×5
$m \times n$ (input matrix dimensions)	500×500	500×500	500×500	500×500	5000×5000	10000×10000
a (scalar in stepsize tuning)	5.0e-04	5.0e-04	5.0e-04	5.0e-04	5.0e-04	5.0e-04
b (scalar in stepsize tuning)	5.0e-07	5.0e-07	5.0e-07	5.0e-07	5.0e-06	5.0e-07

Table 2: Empirical proof of convergence of the algorithm.

NumIterations	Exp#1	Exp#2	Exp#3	Exp#4	Exp#5	Exp#6
0	1.45e+05	1.45e+05	1.45e+05	1.44e+05	6.42e+05	6.66e+07
80000	6.92e-03	1.32e-01	1.45e+00	4.74e+02	1.26e+05	2.13e+04
160000	9.62e-06	7.65e-05	1.44e-04	9.94e-01	2.83e+02	4.06e+00
240000	convergence	1.07e-05	1.25e-05	1.04e-02	2.85e-01	9.96e-03
260000		convergence	1.21e-05	4.41e-03	7.39e-02	2.78e-03
280000			convergence	1.96e-03	2.09e-02	convergence
300000				9.28e-04	6.44e-03	
400000				convergence	convergence	

Table 1 describe various hyper parameters which were used for the experiments and Table 2 shows the costs (i.e., $\sum_{i=1}^{p,q} f_{ij} + \lambda \|\mathbf{U}_{ij}\|_F^2 + \lambda \|\mathbf{W}_{ij}\|_F^2$) as iterations increase.

Experiments on real data sets. We demonstrate the efficacy of our algorithm on widely used, high dimensional, and highly sparse public datasets which are frequently used for benchmarking. The input data is split in a 80 - 20 ratio. The training is done with the 80 percent part. The 20 percent part is kept for testing on which we calculate and report the RMSE. All the experiments are performed with tuned parameters. Table 3 lists the RMSE we received for various datasets with different decomposition pattern ($p \times q$).

Conclusion from the experiments. As we can see from the above experiments, the algorithm is able to learn different \mathbf{U} s and \mathbf{W} s corresponding to different blocks and reaches to convergence in all the cases. The order of reduction of the cost on synthetic datasets is 7 to 10 in all the cases. The experimentation on real datasets also provide enough evidence on learning proving that we can learn global factors even though we may be processing the data in many smaller and independent parts.

Table 3: Experiments using real datasets.

Rank	Number of blocks $p \times q$				
	2×2	3×3	4×4	5×5	10×10
MovieLens 1 million					
5	0.87	0.99	1.04	0.99	1.13
10	0.86	0.99	1.03	1.00	1.22
15	0.86	0.99	1.03	0.99	1.34
MovieLens 10 million					
5	0.97	0.95	0.98	0.97	1.07
10	0.97	0.95	1.00	0.99	1.25
15	0.98	0.96	1.03	1.02121	1.41
MovieLens 20 million					
5	0.95	0.92	0.93	0.99	1.01
10	0.96	0.93	0.93	1.02	1.11
15	0.96	0.94	0.93	1.05	1.24
Netflix					
5	1.03	0.98	1.13	1.06	1.02
10	1.00	0.98	1.14	1.02	1.02
15	1.00	1.11	1.16	1.02	1.03

6 Conclusion

We proposed a novel algorithm for matrix completion, where the data is decomposed along two dimensions, both row wise and column wise. It uses the gossip paradigm for communicating between the independent decomposed units. We get a decomposition in which individual decomposed units reach convergence by just talking to their neighbors, and hence, being independent of the master during the learning phase. Our initial experiments on the synthetic and real datasets shows the efficacy of our algorithm. Exploiting the fact that many of the $\mathcal{S}^{\text{struct}}$ do not contain any overlapping blocks, and hence can be processed in parallel, will be a topic of future research.

References

- [1] Azar Yossi, Fiat Amos, Karlin Anna R., McSherry Frank, and Saia Jared. Spectral analysis of data. In *33rd Symposium on Theory of Computing (STOC)*, pages 619–626, 2001.
- [2] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *34th annual ACS symposium on Theory of Computing (STOC)*, pages 82–90, 2002.
- [3] Genevieve Gorrell. Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2006.
- [4] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Nnmf: An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing* 60, 6:2882–2898, 2012.
- [5] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. In *Neural Information Processing Systems (NIPS)*, pages 281–288, 2006.
- [6] Quoc Le, Marc’Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *International Conference in Machine Learning*, pages 81–88, 2012.
- [7] Qing Ling, Yangyang Xu, Wotao Yin, and Zaiwen Wen. Decentralized low-rank matrix completion. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.
- [8] István Hegedűs, Márk Jelasity, Levente Kocsis, and András A. Benczúr. Fully distributed fully distributed robust singular value decomposition. In *14-th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2014.
- [9] Bamdev Mishra, Hiroyuki Kasai, Pratik Jawanpuria, and Atul Saroop. A Riemannian gossip approach to decentralized subspace learning on grassmann manifold. Technical report, arXiv preprint arXiv:1705.00467, 2016.
- [10] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. Technical report, arXiv preprint arXiv:1606.04838, 2016.