A Stochastic Alternating Minimization Approach for Large Scale Low Rank Matrix Factorization

Ben Cao, Qinzhen Xu, Wen Yan, Luxi Yang
School of Information Science and Engineering
Southeast University
Nanjing, China 210096
Email: bigben@seu.edu.cn

***Abstract*—Low Rank Matrix Factorization (LRMF) is a classical problem that arises in a wide range of practical contexts, especially in collaborative filtering, dimension reduction, etc. In this paper, a stochastic alternating minimization approach applied to LRMF problem is proposed. The main idea of the approach is to randomly sample partial rows of the matrix to perform parameter update during training using alternating minimization, which not only reduces the computational requirements but also declines the over-fitting risk. The simulation results on synthetic datasets show that both alternating minimization-based algorithm and the proposed stochastic variant are applicable for LRMF task, while the proposed algorithm is more competitive for large-scale datasets due to its low-complexity and scalability.**

I. Introduction

Low Rank Matrix Factorization (LRMF) task, which frequently occurs in data analysis, generally refers to learning a sparse representation of a low rank matrix $\mathbf{A}$. The target low rank matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is factorized into two much smaller matrices $\mathbf{U} \in \mathbb{R}^{m\times s}$ and $\mathbf{V} \in \mathbb{R}^{n\times s}$, i.e., a bi-linear form $\mathbf{A} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$, and $s < \min\{m, n\}$.

However, the above model as described is non-convex (it is, more specifically, bi-convex. A function $f : \mathbf{X} \times \mathbf{Y} \mapsto \mathbb{R}$ is called bi-convex, if $f(x, y)$ is convex in y for fixed $x \in \mathbf{X}$, and in x for fixed $y \in \mathbf{Y}$ [1]). Previous work has shown that alternating minimization is a widely applicable and empirically successful approach for bi-convex optimization problem (e.g., low rank matrix factorization task) in practice [1,10]. Alternating minimization proceeds by alternatingly fixing one of the latent factors and optimizing the other. Once one of the factors is fixed, solving for the other is a tractable convex problem. Alternating minimization is an iterative process in which two phases are involved in each iteration: phase I, fix $\mathbf{U}$, update $\mathbf{V}$; and phase II, fix $\mathbf{V}$, update $\mathbf{U}$. The basic steps are repeated until termination criteria reached, where the termination criteria are: 1) maximum iteration is reached; 2) the gap (absolute or relative) between adjacent iterations is less than the threshold (tolerance).

Due to its simplicity, low memory requirement and flexibility, alternating minimization has been a popular approach for low rank matrix factorization. However, some difficulties in training deserve special attention when the size of matrix $\mathbf{A}$ increases. 1) There are many matrix manipulations in the training; therefore, computation grows rapidly as the size expands. 2) Optimization-based algorithms regard every entry of matrix as parameter to be optimized, the amount of to-be-optimized parameter will be huge for large matrices. It is a common sense that over-fitting risk rises when the number of parameters grows. For this reason, some tricks should be introduced to deal with over-fitting. Regularization is a common and useful technique to relieve over-fitting [2], but extra computation is required.

A stochastic alternating minimization approach is proposed in this paper to focus on the above two issues. The main idea is to randomly sample partial rows of the matrix to perform parameter update during each iteration step, which not only reduces the computational requirements but also declines the over-fitting risk.

The idea is mainly motivated by denoising autoencoder [3] which is based on the hypothesis that partially destructed inputs should yield almost the same representation (robustness to partial destruction of the input). This is done by corrupting the initial input X to get a partially destroyed version X by means of stochastic mapping. It is refreshing that the structure can be recoverable from partial observation. In the proposed algorithm, we exploit such characteristic of the low rank matrix to reduce the computation and relieve the over-fitting risk. In fact, almost every model applied to LRMF exploits the low rank structure more or less. Moreover, previous related works on matrix completion [2], [4]–[7], alternating minimization [8]–[11], and train tricks in deep learning [12], [13] also inspire us a lot. Details about the proposed algorithm are discussed in the next section.

II. Method

A. Model Reformation

The low rank target matrix $\mathbf{A}$ is usually denoted as a bi-linear form as follow:

$$\mathbf{A} = \mathbf{U}\mathbf{V}^{\mathrm{T}} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{U} \in \mathbb{R}^{m\times s}, \mathbf{V} \in \mathbb{R}^{n\times s}$, and we have $s \ll \min\{m, n\}$.

$\tilde{\mathbf{A}} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$ is the estimation (approximation) of observation matrix $\mathbf{A}$. Let $\mathbf{E}$ denote the reconstruction error:

$$\mathbf{E} = \mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}} \tag{2}$$

As common practice, we minimize the root mean squared error (RMSE) over all entries in matrix, thereby the objective (loss) function to be optimized can be written as follow:

$$\mathrm{J} = \frac{1}{2} trace\{\mathbf{E}^{\mathrm{T}}\mathbf{E}\} = \frac{1}{2} trace\{\mathbf{E}\mathbf{E}^{\mathrm{T}}\} \tag{3}$$

A variety of optimization techniques can be used to minimize J [14], [15]. In this paper, the analysis is based on Gradient Descent (GD). Note that alternative optimization methods (e.g., Conjugate Gradient, CG) could also be considered. The analysis in this paper is not specific to a particular kind of optimization method.

The gradient of J with respect to factor $\mathbf{U}$ and $\mathbf{V}$ are:

$$\begin{aligned} \frac{\partial \mathrm{J}}{\partial \mathbf{V}} &= \left(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)^{\mathrm{T}}\mathbf{U} = \mathbf{E}^{\mathrm{T}}\mathbf{U} \\ \frac{\partial \mathrm{J}}{\partial \mathbf{U}} &= \left(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right)\mathbf{V} = \mathbf{E}\mathbf{V} \end{aligned} \tag{4}$$

Equation (4) can be easily derived from the formula of matrix calculus. We can also make simple validation here: given that $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{U} \in \mathbb{R}^{m\times s}, \mathbf{V} \in \mathbb{R}^{n\times s}$, we have $\mathbf{E}^{\mathrm{T}}\mathbf{U} \in \mathbb{R}^{n\times s}$, $\mathbf{E}\mathbf{V} \in \mathbb{R}^{m\times s}$, which is the same size as that of matrix $\mathbf{U}$ and $\mathbf{V}$, hence they can be used to update $\mathbf{U}$ and $\mathbf{V}$ in the update stage of GD.

After yielding the gradient, we obtain the basic framework of GD-based alternating minimization algorithm (α refers to learning rate):

Algorithm 1 Alt-Min for LRMF.

```
1: Input: $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{U} \in \mathbb{R}^{m\times s}, \mathbf{V} \in \mathbb{R}^{n\times s}$.
2: Initialize $\mathbf{U}$ and $\mathbf{V}$.
3: Do
   (1). Phase I, fix $\mathbf{U}$, update $\mathbf{V}$ via Gradient Descend method, $\mathbf{V} = \mathbf{V} - \alpha\big(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\big)^{\mathrm{T}}\mathbf{U}$;
   (2). Phase II, fix $\mathbf{V}$, update $\mathbf{U}$, $\mathbf{U} = \mathbf{U} - \alpha\big(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\big)\mathbf{V}$;
   Until termination criteria are reached.
4: Output: $\mathbf{U}$ and $\mathbf{V}$.
```

Before discussing the stochastic variant, we introduce sample operator f, whose rows are a subset of identity matrix's rows, which means that only one 1 in each row while others 0. For example:

$$f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

There are three rows in f , each of them is a one-hot encoded vector sampled from rows of $\mathbf{I}_4$. It is clear that f operator can extract row(s) or column(s) from a matrix. When f_k applied to the left of matrix $\mathbf{A}$, it takes the k-th row of $\mathbf{A}$; and when the operator applied to the right, columns of $\mathbf{A}$ are extracted.

In phase I, apply sample operator f to $\mathbf{U}$, then we have:

$$\mathbf{E}_1 = f\mathbf{E} = f\left(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right) = f\mathbf{A} - (f\mathbf{U})\mathbf{V}^{\mathrm{T}} \tag{5}$$

The objective function (the Frobenius norm of $\mathbf{E}_1$) and the gradient w.r.t $\mathbf{V}$ can be written as:

$$\mathrm{J}_1 = \frac{1}{2} trace\{\mathbf{E}_1^{\mathrm{T}}\mathbf{E}_1\} = \frac{1}{2} trace\{\mathbf{E}_1\mathbf{E}_1^{\mathrm{T}}\} \tag{6}$$

$$\frac{\partial \mathrm{J}_1}{\partial \mathbf{V}} = (f\mathbf{E})^{\mathrm{T}} f\mathbf{U} = \big(f\mathbf{A} - (f\mathbf{U})\mathbf{V}^{\mathrm{T}}\big)^{\mathrm{T}} f\mathbf{U} \tag{7}$$

The term $f\mathbf{U}$ extracts row(s) from $\mathbf{U}$ where f serves as sample operator. In the case when f is an identity matrix, (7) reduce to (4). But if we set f as a small proportion of $\mathbf{I}$'s rows, the computation per iteration will decrease sharply, which brings two immediate benefits: 1) speed up the training; 2) easy to scale up the algorithm.

By the same token, in phase II, we have:

$$\begin{aligned} \mathbf{E}_2 &= \mathbf{E}g^{\mathrm{T}} = \left(\mathbf{A} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right) g^{\mathrm{T}} = \mathbf{A}g^{\mathrm{T}} - \mathbf{U}(g\mathbf{V})^{\mathrm{T}} \\ \mathrm{J}_2 &= \frac{1}{2} trace\{\mathbf{E}_2^{\mathrm{T}}\mathbf{E}_2\} = \frac{1}{2} trace\{\mathbf{E}_2\mathbf{E}_2^{\mathrm{T}}\} \\ \frac{\partial \mathrm{J}_2}{\partial \mathbf{U}} &= \mathbf{E}g^{\mathrm{T}}g\mathbf{V} = \Big(\mathbf{A}g^{\mathrm{T}} - \mathbf{U}(g\mathbf{V})^{\mathrm{T}}\Big) g\mathbf{V} \end{aligned} \tag{8}$$

where g is sample operator applied to the right of $\mathbf{E}$.

The stochastic variant to the primal GD-based alternating minimization algorithm can be described as follows:

Algorithm 2 stochastic Alt-Min for LRMF.

```
1: Input: $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{U} \in \mathbb{R}^{m\times s}, \mathbf{V} \in \mathbb{R}^{n\times s}$.
2: Initialize $\mathbf{U}$ and $\mathbf{V}$.
3: Do
   (1). Phase I, decide random sample operator $f$, update $\mathbf{V}$, $\mathbf{V} = \mathbf{V} - \alpha\big(f\mathbf{A} - (f\mathbf{U})\mathbf{V}^{\mathrm{T}}\big)^{\mathrm{T}} f\mathbf{U}$;
   (2). Phase II, decide random sample operator $g$, update $\mathbf{U}$, $\mathbf{U} = \mathbf{U} - \alpha\Big(\mathbf{A}g^{\mathrm{T}} - \mathbf{U}(g\mathbf{V})^{\mathrm{T}}\Big) g\mathbf{V}$;
   Until termination criteria are reached.
4: Output: $\mathbf{U}$ and $\mathbf{V}$.
```

In standard alternating minimization algorithm, one assumes that sample operator f or g is identity matrix, which means all rows of $\mathbf{U}$ or $\mathbf{V}$ contribute to the parameter update while in the proposed stochastic variant, only partial rows are sampled. In each iteration, we randomly select subset row(s) of $\mathbf{U}$ or $\mathbf{V}$, that is what 'stochastic' in the algorithm means.

Here are two heuristics for sample policy:

1) For each row, suppose the probability to be chosen is p. Generate a random number r, which is uniformly distributed in $[0, 1]$. If $r < p$, the row is chosen; otherwise not. Let random variable X denote the number of rows chosen from the whole rows, we have $X \sim Bernoulli(N, p)$, and we can expect that Np rows will be extracted per iterative step.
2) Generate a random permutation of the sequence $1 : N$, then we take the first r entries as the row index to be chosen. This is based on the assumption that the chance for every row to be chosen is uniform.

B. Performance Analysis

Algorithm Alt-Min, a naive alternating minimization-based optimization applied to LRMF task, is simple and easy-to-implement. But it heavily relies on the matrix manipulation, which is hard to scale up for large-scale datasets. When the size of target matrix increases, the training will be computational prohibitive, or even worse – impossible to implement. The improved stochastic counterpart proposed in this paper could effectively reduce the computation of the training, making it more applicable for large-scale datasets, that is to say, the proposed algorithm is more flexible and scalable.

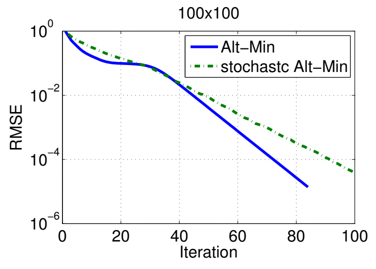
For LRMF task $\mathbf{A} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{U} \in \mathbb{R}^{m\times s}$, $\mathbf{V} \in \mathbb{R}^{n\times s}$, GD-based alternating minimization approach and our stochastic variant have been discussed. For both algorithms, gradient computation is the dominant step which is very computationally expensive. In algorithm Alt-Min, matrix $\mathbf{U}$ and $\mathbf{V}$ are directly multiplied, which requires $m \times n \times s$ multiplication(s) and $m \times n \times (s-1)$ addition(s) at each iteration. For large m and n, it is a considerable number of $O(mn)$ (suppose s is a fixed constant) that we cannot afford. In algorithm stochastic Alt-Min, only r (r is typically quite smaller than both m and n) rows of the total rows of matrix $\mathbf{U}$ and $\mathbf{V}$ are selected randomly at each iteration, which can reduce the multiplication to $r \times n \times s$ for phase I and $m \times r \times s$ for phase II.

In algorithm Alt-Min, one takes all entries of $\mathbf{U}$ and $\mathbf{V}$ into consideration at each iteration, which is vulnerable to meet over-fitting issue when the size of $\mathbf{A}$ expands (number of the to-be-estimated parameter explodes). To guarantee the convergence, a smaller learning rate for GD is preferred. Nevertheless, in stochastic Alt-Min algorithm, only a small proportion of the total rows of $\mathbf{U}$ or $\mathbf{V}$ are randomly sampled, which helps to reduce the risk of over-fitting, thereby we can set a relatively higher learning rate.

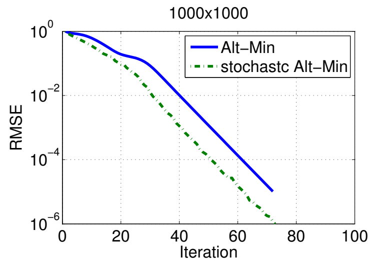
On the other hand, algorithm Alt-Min takes all entries into account, which is like a batch process to the observation matrix while algorithm stochastic Alt-Min only takes several of the whole. Theoretically speaking, algorithm Alt-Min takes less iterations to converge (Just consider the gradient descend case, full-batch GD is faster than mini-batch GD and stochastic gradient descent (SGD) in that batch process is more computation-efficient). Here is an empirical explanation: algorithm stochastic Alt-Min (random) requires more iterations to achieve an ergodic access to all rows of matrix $\mathbf{A}$. But algorithm stochastic Alt-Min prefers higher learning rate, so we can expect that there is no significant difference in the training time of both algorithms to achieve the same convergence. Even in a worse case that stochastic Alt-Min is slower than Alt-Min, the former algorithm is still competitive for its low-complexity and scalability, which becomes very important when faced with large-scale datasets.

III. Result

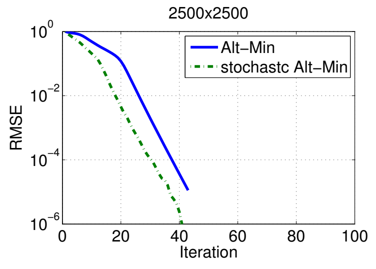
In this section, we present some experiment results on synthetic datasets.



(a) 100x100



(b) 1000x1000



(c) 2500x2500

Fig. 1. Convergence speed for the two algorithms on synthetic datasets.

The dataset is generated by a matrix product of two Gaussian random matrices of sizes $m \times s$ and $s \times n$. To validate the scalability of both algorithms, we test three datasets of size $m \times n$ from 100×100, 1000×1000, and 2500×2500. In all cases, we set the rank $s = 20$ (The rank of generated matrix is guaranteed to be less than s, so the target matrix could be treated as low rank matrix).

Under each specification (say, algorithm, dataset), the simulation run for 10 times, then RMSE (the average recovery error J as denoted in 3) metric is collected to make a performance comparison. The RMSE is normalized by the initial RMSE value. The relative drop rate in log-scale is shown in Fig.1. For both algorithms, maximum iteration is 100, the goal relative RMSE is set to be 1e-5. Note that the learning rate varies for different datasets to guarantee the convergence.

The empirical results as shown in Fig.1 suggest that the stochastic version works as well as Alt-Min, or even better (a faster convergence), with the size of datasets expands. By carefully setting learning rate, both algorithms converge after limited iterations. The experiments have also reported that the proposed stochastic variant is less time-consuming, hence it is more flexible for large-scale datasets.

Besides, we further investigate the relationship between

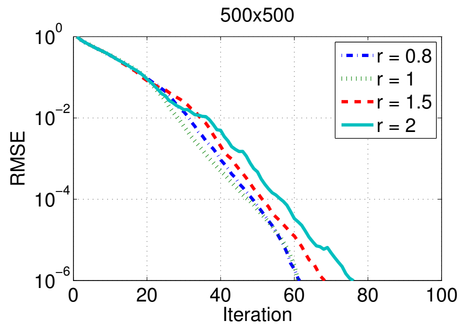


Fig. 2. RMSE versus sample ratio on synthetic dataset 500x500.

sample ratio r (defined as the multiplier of the number of sampled rows and the actual rank s) and log-scaled RMSE. Varying r as 0.8, 1.0, 1.5 and 2.0, the simulation results (Fig.2) show only slight differences for different r, implying that we can choose a relatively lower sample ratio, which would speed up the training but not impair the performance too much, making it possible for the proposed algorithm to scale up.

IV. Conclusion

In this paper we propose a stochastic alternating minimization approach for large-scale low rank matrix factorization task. The empirical results on synthetic datasets support the conclusion that both alternating minimization approach and its stochastic variant are applicable for LRMF task. But our stochastic version is more competitive for large-scale datasets due to its low-complexity and scalability. For even larger datasets, we can further extend our algorithm to a distributed framework, like Hadoop, Spark, etc. This could enable us to model larger-scale datasets and would be interesting scope for future work.

At last, we attempt to deliver a naive interpretation about why our algorithm works.

1) For matrix factorization task where the target matrix is denoted as a bi-linear form, the factorization is not unique. For example, suppose there are two matrices $\mathbf{U}$ and $\mathbf{V}$, such that $\mathbf{A} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$. By introducing a unary matrix $\mathbf{D}$ (i.e. $\mathbf{D}\mathbf{D}^{\mathrm{T}} = \mathbf{I}$), we rewrite $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{D}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} = (\mathbf{U}\mathbf{D})(\mathbf{V}\mathbf{D})^{\mathrm{T}}$, then $\mathbf{U}_1 = \mathbf{U}\mathbf{D}$, $\mathbf{V}_1 = \mathbf{V}\mathbf{D}$ is also the factorization. This feature is important for alternating minimization-based algorithm. Alternating minimization based algorithm, which is prone to get stuck in local optima [6], is not guaranteed to reach a global optima. Fortunately, the non-uniqueness of the factorization implies that there may be many global optima. As a result, the local optima are also the global optima in many cases. That is why the alternating minimization-based algorithms are successful in practice.
2) From the perspective of dimension reduction, matrix factorization task can be interpreted as extracting linear features from a high-dimensional data space. Column vectors in $\mathbf{U}$ are the axes of the subspace, and vectors in $\mathbf{V}$ are the components (or projections) in each axis. Since there are only r (rank $r \ll \min\{m,n\}$) linearly independent vectors which spans the column (row) space of low rank matrix $\mathbf{A}$, the 'redundancy' in the matrix could be fully exploited by sampling partial (not total) vectors for parameter update. It is the 'redundancy' that accounts for the robustness to the corrupted input of the low rank matrix.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 61372101 and National High Technology Project of China (2015AA01A703). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Mathematical Methods of Operations Research*, vol. 66, no. 3, pp. 373–407, 2007.
[2] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International Conference on Algorithmic Applications in Management*. Springer, 2008, pp. 337–348.
[3] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.
[4] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.
[5] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.
[6] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 665–674.
[7] X.-Y. Liu, S. Aeron, V. Aggarwal, and X. Wang, "Low-tubal-rank tensor completion using alternating minimization," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2016, pp. 984 809–984 809.
[8] J. A. Tropp, "An alternating minimization algorithm for non-negative matrix approximation," 2003.
[9] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, vol. 61, 2009.
[10] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, vol. 20, 2011, pp. 1–8.
[11] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.
[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[13] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.
[14] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.
[15] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.