Project Report for CS798, 2016 Fall
Optimization for Machine Learning

Tim Tse
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3E6
trttse@uwaterloo.ca

Abstract

In this project, I studied the problem of matrix completion which can roughly defined as the task of filling in missing entries from a partially observed matrix. Specifically, I performed a replication study of [1] wherein I implemented three of the four standard optimization algorithms and I analyzed and compared them in terms of their recovery rank with respect to the number of entries revealed and the rank of the true matrix. I also attempted to outline a method of a solving a problem left open by [1] with regards to the optimal convergence rate of gradient descent with constant step size for matrix completion.

1 Introduction

Matrix completion can roughly be defined as the task of filling in the missing entries of a partially observed matrix. A wide range of datasets can naturally be represented in the form of a matrix such as for example, consider the movie recommendation problem in the form of the Netflix Prize. In this problem, Netflix has a group of users that have each rated a subset of movies and using these ratings, Netflix wishes to predict the ratings of all other movies from these users. This problem can naturally be represented in a matrix format wherein the rows of the matrix represent the users and the columns represent the movies and the (i, j) cell of the matrix is the rating that the i-th user gave to the j-th movie. The task is then to fill in the missing entries (i.e., unrated movies) of the matrix. In addition to collaborative filtering and recommendation systems, matrix completion has also found applications in system identification in control, and multi-class learning in machine learning.

Mathematically, we can define the matrix completion problem as follows: Given a matrix $M \in \mathbb{R}^{m\times n}$ with only a subset of its entries $M_{i,j}, (i, j) \in \Omega \subseteq \{1, 2..., m\} \times \{1, 2, ..., n\}$, we wish to recover the missing entries of M, while at the same time, impose a low-rank structure on the estimation matrix $\hat{M}$. There are two popular approaches to the matrix completion problem. The first approach is roughly known as the nuclear norm approach wherein we solve the optimization problem

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ \text{subject to} & X_{i,j} = M_{i,j},\ (i, j) \in \Omega, \end{array}$$

where $\|X\|_*$ denotes the nuclear norm. The benefit to this approach is that there are strong theoretical results that guarantee the recoverability of M. However, the major drawback is that it requires computing the singular value decomposition (SVD) per-iteration and it also requires the entire matrix to be stored and updated [2] making this technique unattractive to large matrices.

The second approach is roughly known as the matrix factorization model. In this method, we represent a rank r estimation of M as the product of two other matrices X and Y and we recover M by optimizing over the two factors. Specifically, we solve the regularized minimization problem

$$\underset{X\in\mathbb{R}^{m\times r},Y\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad \frac{1}{2}\sum_{(i,j)\in\Omega}[M_{i,j}-(XY^\top)_{i,j}]^2+\lambda(\|X\|_F^2+\|Y\|_F^2).$$

The advantage of this approach is that representing the estimation matrix $\hat{M}$ over the factors X and Y imposes a given rank on $\hat{M}$ (i.e., via the compact SVD). Another major strength is that this formulation is efficient both in per-iteration complexity as well as in storage space, making this technique viable in very large problem sets with n in the millions (as claimed in [1]). Finally, this formulation is attractive because standard optimization techniques such as (stochastic) gradient descent and alternating minimization can be applied to solve the minimization problem. A slight downside compared to the nuclear norm formulation is that the matrix factorization model has weaker theoretical guarantees, though this drawback can be mitigated with careful initialization. In fact, [1] showed that after proper initialization, the problem is effectively strongly convex and hence, standard optimization algorithms can be applied. In this project, I strictly focus my investigations on algorithms of the latter approach.

2 Related Works

The theoretical results of matrix completion was first established by [3] where they showed that under the incoherence assumption, a low-rank matrix can be exactly recovered if the number q of sampled entries obey $q \geq Cn^{1.2}r\log n$ for some positive numerical constant C. The authors also provided empirical verification of their theory through numerical experiments where they solved the nuclear norm minimization by applying semidefinite programming (SDP), though indeed, this only was a proof of concept as this approach became infeasible for matrices with $n > 100$. The scalability was improved by the work of [2] where they introduced a nuclear norm minimization algorithm using soft-thresholding on the singular values of the estimate matrix, making matrices with $n \approx 1000$ now viable. Next, we saw a series of work that focused on the factorization model of the problem such as the OptSpace algorithm [4] which roughly operates with an intialization with SVD and then a gradient descent over a Grassmann manifold and the AltMinComplete algorithm [5] which again works by first initializing with a SVD but unlike OptSpace, it then performs an alternating minimization over two factors. These methods work with $n \approx 30000$, but they also lack strong theoretical guarantees though this was mitigated for alternating minimization with the theoretical work of [6]. Finally, the work of [1] introduced novel initialization scheme and regularizer that makes matrix completion feasible with very large matrices (n in the millions) and also have strong theoretical guarantees (problem is effectively strongly-convex with the right initialization). On a side note, it has come to my attention that the very recent work of [7] has begun to provide theoretical analysis of the non-convex formulation of matrix completion via the *geometry* of the problem.

3 A Replication of Some Matrix Completion Algorithms

For part of this project, I implemented three of the four algorithms proposed by [1] to solve the matrix completion problem, namely, gradient descent (GD), two-block alternating minimization (TBAM) and stochastic gradient descent (SGD). I compared the recovery rates of each algorithm with respect to p, the probability of an entry being revealed and r, the rank of the matrix M.

Below, I briefly outline the three algorithms from [1]. For full details of the algorithms, please refer to the paper. I implemented three of the four algorithms of [1], leaving out row block successive upper bound minimization (BSUM) for the reason that row BSUM required solving a tricky subproblem which I opted not to tackle for the reasons of spending more time on tackling theory.

3.1 Gradient Descent

This is a standard gradient descent algorithm that performs update on X_k and Y_k for every k-th iteration as follows:

$$X_k \longleftarrow X_k(\eta_k) \triangleq X_{k-1} - \eta_k \nabla_X \tilde{F}(X_{k-1}, Y_{k-1}),$$
$$Y_k \longleftarrow Y_k(\eta_k) \triangleq Y_{k-1} - \eta_k \nabla_Y \tilde{F}(X_{k-1}, Y_{k-1}).$$

The algorithm allows three options for choosing the step size η_k: constant step size, restricted Armijo rule and restricted line search. In this project, I only implemented the constant step size version of the algorithm. For constant step size, $\eta_k = \eta \leq \bar{\eta}_1, \forall\, k$, where $\bar{\eta}_1$ is a constant defined in the paper.

3.2 Two-block Alternating Minimization

A standard algorithm that performs alternating minimization on X_k and Y_k for every k-th iteration as follows:

$$X_k \longleftarrow \arg\min_X \tilde{F}(X, Y_{k-1}),$$
$$Y_k \longleftarrow \arg\min_Y \tilde{F}(X_{k-1}, Y).$$

In the case that there is no regularizer, there is a closed form update for X_k and Y_k. The paper argues that when there is a regularizer, we can first start with a closed form update for X_k and Y_k and then we can follow the algorithm up with a gradient descent step.

3.3 Stochastic Gradient Descent

The objective function $\tilde{F}$ can be decomposed into a summation of a sequence of functions f_i, namely, $\tilde{F} = f_1 + \cdots + f_{|\Omega|+m+n+2}$. SGD makes use of this fact by sequentially performing a gradient descent update on every f_i (as opposed to all of $\tilde{F}$ in gradient descent) on X_k and Y_k for every k-th iteration as follows:

$$\begin{aligned}
&\texttt{For} \quad i = 1 \quad \texttt{to} \quad |\Omega| + m + n + 2 \\
&\qquad X_{k,i} \longleftarrow X_{k,i-1} \triangleq X_{k-1} - \eta_k \nabla_X f_i(X_{k,i-1}, Y_{k,i-1}), \\
&\qquad Y_{k,i} \longleftarrow Y_{k,i-1} \triangleq Y_{k-1} - \eta_k \nabla_Y f_i(X_{k,i-1}, Y_{k,i-1}). \\
&\text{End}
\end{aligned}$$

η_k satisfies $\sum_k \eta_k = \infty$, $\sum_k \eta_k^2 < \eta_{\text{sum}}$ and $0 < \eta \leq \bar{\eta}$, where η_{sum} and $\bar{\eta}$ are constants specified in the paper.

3.4 Experiments and Results

The first experiment that I performed is a simple execution of the algorithms on a contrived dataset and then observing the recovery results. In this experiment, I generated a 1000×1000 matrix M of rank 10 by sampling each entry from $\mathcal{N}(0, 1)$ and I generated a subset of revealed entries Ω by uniformly revealing each entry with probability $p = 0.04$. The error metric that I used is root-mean-square error (RMSE) which is defined as RMSE $= \frac{\|M - XY^\top\|_F}{\sqrt{mn}}$, where m and n is the number of rows and columns of M, respectively (which in this case $m = n = 1000$). Figure 1 shows the RMSE with respect to the number of iterations of the three algorithms. The plots show that the convergence rate is much faster for TBAM than it is for GD and SGD: for the given configuration, TBAM took ≈ 50 iterations to converge as opposed to ≈ 2000 for GD and SGD. $p = 0.04$ was chosen for the reason that in this configuration, a uniform reveal rate of 4 % was the threshold with which exact recovery was beginning to fail. The rank of the matrix is also correlated with recovery results with the relation being the higher the rank with respect to n, the lower the recovery success rates. Preliminary experiments show that this is consistent with theoretical results and in the second experiment, I investigate in more detail the correlation between p, r and the recovery success rate. I also included a sample (first 74 rows and 10 columns) numerical output of the GD algorithm in Table 1 (last page). Each cell is demarcated by a solid line and within each cell, there are three values demarcated by a dotted line. The first value is $\Omega_{i,j}$ (value of matrix with missing entries), the second value is $M_{i,j}$ (value of the fully revealed matrix) and the last value is $\hat{M}_{i,j}$ (value of the estimate matrix). We see that even with $p = 0.04$, recovery is almost exact.

In the second experiment, I attempted to reproduce the numerical results of [3] wherein they ran their experiments for different combinations of (n, r, p) in an effort to verify their theoretical result on the conditions necessary for exact recovery, namely, $q \geq Cn^{1.2} r \log n$ where in this case, q is the number of sampled entries and C is a positive numerical constant. Specifically, for the configurations

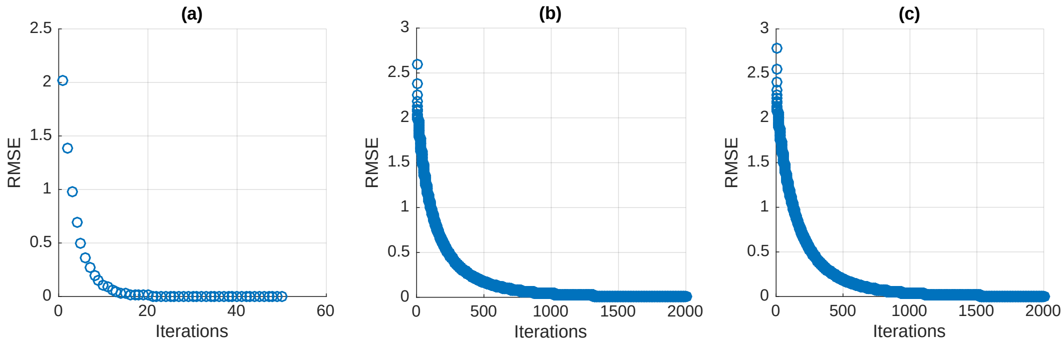

Figure 1: RMSE against Iterations: (a) Two-block Alternating Minimization, (b) Gradient Descent, (c) Stochastic Gradient Descent.

of my experiment, I used $n = 1000$ square matrix and the combination $r = \{1, ..., 50\} \times q = \{1, ..., 50\}$. For each combination, an instance of the algorithm was executed and using a jet color scheme to indicate value, the final RSME was plotted (Figure 2) against $\frac{r}{n}$ on the y-axis and $\frac{q}{n^2}$ on the x-axis. This was performed for each of the three algorithms and 50, 2000 and 2000 iterations were used for TBAM, GD and SGD, respectively. The results for GD and SGD are quite faithful to the results in [3]: there is high recoverability with low r and/or high q ("cold region") and the recoverability diminishes as r increases and/or q decreases with complete irrecoverability at the extreme ("hot region"). TBAM is slightly more interesting: the algorithm produced highly varied values and for the regions that clearly diverged, I capped the maximum off at 10 (to prevent the color spectrum from being "stretched" too much) and hence, we see high divergence in areas where $\frac{r}{n} \approx \frac{q}{n^2}$ for reasons that I am not too sure.

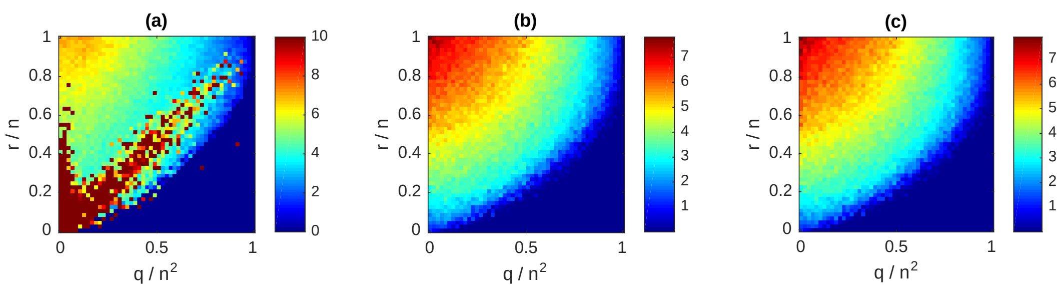

Figure 2: RMSE for each combination of $\frac{r}{n} \times \frac{q}{n^2}$: (a) Two-block Alternating Minimization, (b) Gradient Descent, (c) Stochastic Gradient Descent.

4 Theoretical Attempts

In this section I wish to make an informal attempt/discussion/outline of a theoretical problem left open by [1]. Specifically, in their proof of the time complexity bound of gradient descent with constant step size, the authors arrive at the time complexity of $\tilde{O}(\text{poly}(n) \log \frac{1}{\epsilon})$ for achieving any ϵ-optimal solution, where the $\tilde{O}$ notation hides factors polynomial in r, κ, α. The authors claim that they believe the optimal time complexity can be improved to $\tilde{O}(|\Omega| \log \frac{1}{\epsilon})$ which is what they observed in practice. In this section, I attempt to outline my approach of proving this. First, however, I would like to discuss some background of their proofs.

One lemma that appears to be quite applicable to my goal is **Lemma 4.2.1** [1] which states that under certain assumptions (i.e., Ω is uniformly generated at random with size $|\Omega|$ and with probability at least $1 - \frac{1}{n^4}$), their initialization procedure will produce an initial point within what is known as an "incoherent neighborhood" and the algorithm(s) will converge to a stationary point inside said incoherent neighborhood. They define an incoherent neighborhood as $K_1 \cap K_2 \cap K(\delta)$, where K_1,

K_2 and $K(\delta)$ are definitions of sets that constrain the norms of X and Y (and hence enforce incoherence). Under this lemma, the problem is effectively locally strongly convex but actually not really as a more precise description might be the so-called "cost-to-go estimate" in optimization literature. Nonetheless, the authors applied first order methods, proving two conditions: (1) cost-to-go estimate (same as Theorem 1.20 Lecture 07) and (2) sufficient decrease (same as Theorem 1.7 Lecture 05) to arrive at the conclusion that the algorithm converges at the linear rate of $\tilde{O}(\text{poly}(n)\log\frac{1}{\epsilon})$. The goal then is to somehow replace the poly(n) factor in the bound with $|\Omega|$. My understanding is that gradient descent converges at a sublinear rate but under certain assumptions (such as strong convexity) then the rate can be improved to a linear rate and hence, I will make the assumption that the problem is both strongly convex and L-smooth (though one might imply the other in Theorem 1.34 Lecture 08?) Under this assumption we have

Theorem. If f is L-smooth and σ-strongly convex, then gradient descent with fixed step size $\eta \leq \frac{2}{\sigma+L}$ or with backtracking line search search satisfies

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leq c^k \frac{L}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_F^2, \text{ where } 0 < c < 1. \tag{1}$$

Now, we can replace L with $|\Omega|$ in (1) if we can show that $L \leq |\Omega|$. **Claim 2.1** [1] states that suppose $\beta_0 \geq \beta_T$ and

$$L(\beta_0) \triangleq 4\beta_0^2 + 54p\frac{\beta_0^2}{\beta_1^4}.$$

Then $\nabla\tilde{F}(X,Y)$ is Lipschitz continuous over $\Gamma(\beta_0)$ with Lipschitz constant $L(\beta_0)$, i.e.

$$\|\nabla\tilde{F}(X,Y) - \nabla\tilde{F}(U,V)\|_F \leq L(\beta_0)\|(X,Y)-(U,V)\|_F, \forall(X,Y),(U,V) \in \Gamma(\beta_0),$$

where $\|(X,Y)-(U,V)\|_F = \sqrt{\|X-U\|_F^2 - \|Y-V\|_F^2}$ and for any positive number β, $\Gamma(\beta)$ is a bounded set defined as

$$\Gamma(\beta) \triangleq \{(X,Y)|X \in \mathbb{R}^{m\times r}, Y \in \mathbb{R}^{n\times r}, \|X\|_F \leq \beta, \|Y\|_F \leq \beta\}.$$

The objective is then to show that $L(\beta_0) \leq |\Omega|$ and if we can do that, then perhaps we can show that gradient descent with fixed η converges at rate $\tilde{O}(|\Omega|\log\frac{1}{\epsilon})$ within the bounded set $\Gamma(\beta_0)$. This is all I have for now, apologies for the half-baked idea and/or if this is completely wrong.

References

[1] Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via non-convex factorization. *CoRR*, abs/1411.8003, 2014. URL http://arxiv.org/abs/1411.8003.

[2] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982, March 2010. ISSN 1052-6234. doi: 10.1137/080738970. URL http://dx.doi.org/10.1137/080738970.

[3] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008. URL http://arxiv.org/abs/0805.4471.

[4] Raghunandan H. Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries. *CoRR*, abs/0901.3150, 2009. URL http://arxiv.org/abs/0901.3150.

[5] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 665–674, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488693. URL http://doi.acm.org/10.1145/2488608.2488693.

[6] Moritz Hardt. On the provable convergence of alternating minimization for matrix completion. *CoRR*, abs/1312.0925, 2013. URL http://arxiv.org/abs/1312.0925.

[7] Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. *CoRR*, abs/1605.07272, 2016. URL http://arxiv.org/abs/1605.07272.

-1.4719	-1.4719	-1.4733	0	0.45841	0.45823	0	-2.7147	-2.7135	0	-2.2906	-2.2897	0	-3.2402	-3.2385	0	2.0311	2.0307	0	2.9777	2.9779	0	-0.67823	-0.67887	0	3.8429	3.8416	0	2.1032	2.1034
0	-0.044193	-0.046711	0	5.9271	5.9268	0	3.4471	3.4458	0	-0.79696	-0.79744	0	-3.5895	-3.5889	0	-4.3447	-4.3399	0	6.3236	6.3244	0	-2.7737	-2.774	0	-2.2165	-2.2157	0	6.0618	6.0614
0	3.4505	3.4542	0	-0.52515	-0.52622	0	-4.6873	-4.6903	0	-1.7837	-1.7899	0	7.1998	7.198	0	-5.61	-5.6104	0	0.060094	0.060701	0	-2.212	-2.2059	0	-1.9511	-1.9556	0	-6.5924	-6.5876
0	-0.55153	-0.55228	0	5.0109	5.0111	0	-2.5708	-2.5712	0	-3.7658	-3.7668	0	0.47072	0.47378	-4.4208	-4.4208	-4.4201	0	2.9053	2.9054	0	-2.3934	-2.3936	0	-1.71	-1.7106	0	0.54546	0.54598
0	6.6275	6.6245	0	-2.7435	-2.7401	0	1.3613	1.3643	0	1.421	1.4201	0	5.4995	5.4951	0	-2.8229	-2.8207	0	2.0387	2.0373	0	4.3532	4.3577	0	-0.96895	-0.96651	0	-1.5851	-1.5858
0	-4.9087	-4.9079	3.7374	3.7374	3.7378	0	-5.7037	-5.7038	0	-0.42854	-0.43088	0	-7.3155	-7.313	0	5.1961	5.1925	0	2.0154	2.0161	0	-6.4014	-6.4001	0	5.0826	5.0837	0	-5.5199	-5.517
0	2.2234	2.2212	0	-3.7947	-3.7952	0	5.9391	5.9391	0	4.681	4.6784	0	-5.1299	-5.1285	0	-0.42299	-0.42312	0	-0.95148	-0.95564	0	-1.8343	-1.8301	0	-1.6389	-1.6377	0	0.70512	0.70082
0	-1.7486	-1.7447	0	-2.5094	-2.5085	0	3.479	3.4832	0	7.6048	7.605	0	-6.8422	-6.841	0	4.7832	4.7746	0	-2.3364	-2.3354	0	-4.7087	-4.7082	0	2.3888	2.387	0	-5.5356	-5.5348
0	-0.32413	-0.32498	0	0.4717	0.47011	0	3.4839	3.4828	0	2.7023	2.7057	0	-0.80526	-0.80552	0	2.0773	2.0761	0	0.11929	0.11897	0	0.22843	0.22686	0	4.8637	4.8637	0	-0.22528	-0.22771
0	1.8313	1.8292	0	-1.6504	-1.6503	0	3.0674	3.0681	0	0.90511	0.9056	0	0.38324	0.38252	0	-0.37423	-0.37149	0	2.3268	2.3267	0	1.7615	1.7619	0	0.36538	0.36549	0	5.2074	5.2064
0	0.49278	0.49136	0	3.4673	3.4641	0	-2.5371	-2.5395	0	-5.7697	-5.7686	0	0.69381	0.69294	0	-1.6839	-1.677	0	5.4676	5.4639	0	5.5149	5.5149	0	-1.3658	-1.3621	0	2.3754	2.3751
0	-1.0473	-1.0529	0	8.1123	8.1155	0	-4.8737	-4.8678	0	-1.5136	-1.5176	0	-9.7737	-9.7678	0	-5.4597	-5.4611	0	8.5619	8.5588	0	-9.1878	-9.1857	0	-0.20718	-0.20922	0	-6.3121	-6.3056
0	0.37231	0.37191	0	1.3456	1.3449	0	5.5212	5.521	0	1.8531	1.854	0	1.0952	1.0938	0	-2.4897	-2.4892	0	-1.6149	-1.6148	0	-0.63818	-0.63755	0	-1.4387	-1.439	0	2.8729	2.8718
0	5.3031	5.2994	0	6.0182	6.0159	0	1.1137	1.1099	0	-2.5187	-2.5182	0	2.963	2.9632	0	-6.7089	-6.7071	0	5.4254	5.4219	0	-2.69	-2.688	0	3.0022	3.0035	0.66842	0.66842	0.66894
0	-5.053	-5.0512	0	3.6891	3.6891	0	1.6782	1.6782	0	-2.4258	-2.4269	1.416	1.416	1.4154	0	0.1395	0.13675	0	-0.72609	-0.72387	0	2.5143	2.5142	0	0.061032	0.059646	0	0.31669	0.31903
0	-1.0381	-1.0442	0	3.8348	3.8338	0	-0.28133	-0.28153	0	-0.23719	-0.23498	0	-6.5911	-6.5899	0	2.5296	2.5309	0	6.6936	6.6927	0	-6.3305	-6.3315	8.0793	8.0793	8.0794	0	4.8981	4.8968
0	8.3015	8.3047	0	-3.8495	-3.8468	0	1.6251	1.6255	0	4.8726	4.8694	0	6.0482	6.0483	0	-6.6296	-6.6315	0	-0.94748	-0.94668	0	-1.821	-1.8206	0	-6.2455	-6.2471	0	-3.9162	-3.9151
0	1.1341	1.1394	0	-2.7861	-2.7871	0	-3.6026	-3.6029	0	1.8513	1.8493	0	4.2506	4.2516	0	-0.92619	-0.92756	0	-4.1133	-4.1122	0	-2.5851	-2.5846	0	-4.1842	-4.1862	0	-4.9115	-4.9092
0	-3.0696	-3.0711	0	0.47355	0.47404	0	2.1647	2.1651	0	1.5867	1.5879	0	5.707	5.7058	0	-1.8728	-1.8714	0	-6.1155	-6.115	0	-0.5954	-0.59774	0	-1.718	-1.7192	0	-1.6363	-1.6367
0	0.2921	0.29452	0	-2.1134	-2.1121	0	11.6473	11.6456	5.0482	5.0482	5.0482	0	-7.606	-7.6109	-1.5883	-1.5883	-1.5894	0	-3.8445	-3.8392	0	1.0142	1.0147	0	-5.003	-5.0032	0	5.4998	5.4947
0	3.2379	3.2407	0	-3.5556	-3.5524	0	-0.89506	-0.89404	0	5.6534	5.6556	0	-0.52343	-0.52574	0	3.603	3.6014	0	-0.60164	-0.60004	0	1.6752	1.6719	0	-3.1588	-3.1571	0	-4.5844	-4.5876
0	-2.7907	-2.791	0	3.4537	3.4528	0	4.7978	4.7976	0	-2.4126	-2.4111	0	2.4237	2.4209	0	-1.822	-1.8193	0	-0.66619	-0.66691	0	3.1781	3.1775	0	-1.7954	-1.7951	0	5.834	5.8334
0	-4.7544	-4.7513	0	-0.2679	-0.26624	0	-4.4122	-4.4111	-2.903	-2.903	-2.9033	0	0.8559	0.85548	0	3.8224	3.8203	0	-4.2765	-4.2751	0	3.6992	3.6968	0	-2.1057	-2.1056	0	-2.118	-2.1173
0	0.86924	0.87392	0	-2.3203	-2.3194	0	2.9211	2.9193	0	-1.937	-1.9396	0	6.958	6.9577	-1.2527	-1.2527	-1.2525	0	-3.4998	-3.4984	0	9.8154	9.8128	0	-6.4554	-6.4528	0	4.8827	4.8826
0	2.5765	2.5767	0	-1.4474	-1.4482	0	2.2436	2.2437	4.76	4.76	4.7599	-0.17085	-0.17085	-0.17054	0	-0.14032	-0.13982	0	-1.8918	-1.8919	0	-5.9074	-5.9058	0	-1.0557	-1.056	0	1.6775	1.6776
0	-0.79547	-0.79507	0	-3.1471	-3.1452	0	-0.92332	-0.92248	0	1.7579	1.7581	0	-3.6812	-3.6811	0	4.5523	4.5512	0	-1.6586	-1.658	0	3.6198	3.6183	0	-1.4058	-1.4054	0	-2.1474	-2.148
0	2.5484	2.5467	3.8949	3.8949	3.8949	0	-1.5616	-1.5615	0	-0.087096	-0.088131	-0.23242	-0.23242	-0.23225	0	-6.3396	-6.3372	0	2.7889	2.7907	0	-4.5908	-4.59	0	-4.0711	-4.0717	0	-3.0761	-3.0729
0	2.4078	2.4055	0	2.6984	2.6995	0	1.1227	1.1226	0	2.7985	2.799	0	0.78208	0.78194	0	-1.8989	-1.8995	0	1.2869	1.2882	0	-4.5958	-4.5959	0	2.4064	2.4079	0	-1.4379	-1.4369
0	-0.52695	-0.52916	0	3.5179	3.5172	0	3.4244	3.4222	0	-1.1515	-1.1526	0	-3.3606	-3.3598	0	-2.1989	-2.1996	0	1.2306	1.2297	0	-1.6891	-1.6878	0	1.3993	1.4014	0	1.4502	1.4488
0	1.9527	1.9536	0	2.5876	2.583	0	5.025	5.0236	0	-1.1498	-1.1462	-2.719	-2.719	-2.7192	0	-3.751	-3.756	0	0.96432	0.96604	0	1.4351	1.4384	0	-0.96336	-0.96803	0	0.80636	0.8031
0	5.7218	5.7212	0	-5.5519	-5.5496	0	5.1114	5.1132	0	6.132	6.1329	0	-1.8795	-1.873	0	-3.1439	-3.1466	0	-1.289	-1.2907	0	-1.6003	-1.6036	0	-4.3863	-4.3867	0	0.45919	0.45511
-4.1382	-4.1382	-4.1385	0	-1.5884	-1.5894	0	2.3997	2.4011	0	4.3255	4.3279	-1.5997	-1.5997	-1.5996	6.0238	6.0238	6.023	0	-1.0624	-1.0632	0	-3.36	-3.3617	0	5.7205	5.7206	0	1.5474	1.5442
0	4.2276	4.2286	0	-1.1634	-1.1631	0	7.5807	7.5756	0	3.8217	3.8198	0	3.1463	3.1477	0	-1.8837	-1.8842	0	-3.0136	-3.0137	0	2.4272	2.4292	0	-1.8109	-1.8095	0	3.6897	3.6876
0	-2.2608	-2.2605	0	0.99209	0.99309	0	-3.0618	-3.0617	0	-0.53231	-0.53402	0	-5.408	-5.4068	0	0.65383	0.65313	0	0.10546	0.10631	0	-2.7351	-2.736	0	-1.6213	-1.6207	0	-3.5599	-3.5582
0	0.95677	0.95365	0	1.9823	1.9831	0	7.8158	7.8166	0	0.42446	0.42392	0	-0.071092	-0.072656	0	-5.2034	-5.1989	0	3.3673	3.3673	0	4.6927	4.6934	0	-4.1011	-4.1014	0	6.0919	6.0915
0	0.56209	0.56222	0	-1.5349	-1.5336	0	1.2399	1.2409	0	0.18216	0.18108	0	-2.5886	-2.59	0	0.48066	0.48015	0	-2.55	-2.55	0	1.087	1.0878	0	-3.2747	-3.2745	0	1.4127	1.4147
0	1.7696	1.7705	0	-1.0848	-1.0853	0	-1.7839	-1.7823	0	1.3936	1.3927	0	-0.12794	-0.1281	0	-0.99403	-0.99416	0	-1.3507	-1.351	0	-3.9607	-3.9589	0	-1.5049	-1.5041	0	-2.9352	-2.9318
0	2.9017	2.9009	0	-1.0246	-1.0247	0	3.915	3.9144	0	0.80965	0.80929	0	4.6845	4.6829	0	-3.3466	-3.3447	0	-0.60522	-0.6062	0	1.0536	1.0545	0	-1.0831	-1.0822	0	3.0064	3.0059
0	-1.7939	-1.7963	0	6.2526	6.252	0	-0.46603	-0.46583	0	-2.6471	-2.6444	0	-1.416	-1.4144	0	-1.0695	-1.0685	0	4.7344	4.7349	0	-1.2872	-1.2869	0	3.8512	3.8517	0	-0.62355	-0.62188
0	5.6881	5.6884	0	-2.0641	-2.0616	0	2.2759	2.2741	0	4.3993	4.3974	0	1.9251	1.9244	0	-3.2319	-3.2299	0	1.8558	1.8557	0	2.3816	2.3827	0	-5.0932	-5.0937	0	-4.4164	-4.4161
0	2.4289	2.4303	0	2.7498	2.7507	0	0.42749	0.42566	0	-3.1585	-3.1646	0	9.6034	9.6048	0	-7.3786	-7.3801	0	2.289	2.2919	0	1.6589	1.6606	0	-0.57447	-0.57453	0	-1.3183	-1.3103
0	2.1017	2.0913	0	3.5522	3.5521	0	6.1914	6.1964	0	2.6625	2.6595	0	2.663	2.6693	0	-8.4074	-8.3985	0	3.4984	3.4941	0	-0.14663	-0.14162	0	-6.6913	-6.6815	0	1.6947	1.6856
0	2.5252	2.5244	0	-0.73784	-0.73645	0	-0.77958	-0.77979	0	-1.6472	-1.6485	0	1.1849	1.1841	0	-2.0145	-2.0105	0	3.0493	3.0489	0	1.4535	1.4535	0	-3.7928	-3.7921	0	3.5749	3.5747
0	-1.3788	-1.3757	0	-1.3888	-1.3913	0	-6.2785	-6.2797	0	-2.9767	-2.9786	0	5.6062	5.6082	0	0.8111	0.81167	0	-2.731	-2.732	0	0.33055	0.33159	0	-1.0727	-1.0737	0	-2.5042	-2.5023
0	0.16946	0.16549	0	-0.059772	-0.061346	0	2.9521	2.9528	0	0.51015	0.51156	0	-1.639	-1.6382	0	-3.1155	-3.1097	0	2.7266	2.7251	0	0.052105	0.052138	0	-0.93689	-0.93609	0	2.7367	2.7333
0	5.1146	5.1137	0	0.26955	0.26813	0	-0.023116	-0.025826	0	-2.2367	-2.2371	0	5.4728	5.4719	0	-5.3812	-5.3805	0	1.516	1.5148	0	0.27006	0.27166	0	1.161	1.1618	0	0.16361	0.16306
0	-0.15517	-0.1582	0	-1.0834	-1.0815	0	3.7315	3.7303	0	-1.1868	-1.1886	0	0.66702	0.66497	0	-2.4424	-2.4401	0	-2.5312	-2.5313	0	3.9006	3.9002	0	-5.4773	-5.4783	0	4.6577	4.6555
0	-5.0074	-5.0118	0	-0.82481	-0.82229	0	2.1184	2.1229	0	3.4782	3.4772	0	-8.0575	-8.0609	0	5.6068	5.6077	0	1.5762	1.577	0	-0.89001	-0.8915	0	1.1551	1.1554	0	0.6943	0.69643
0	3.0523	3.0537	-8.1533	-8.1533	-8.1536	0	7.4175	7.419	0	4.0994	4.1035	0	-2.3246	-2.3242	0	5.0788	5.0788	0	-2.2444	-2.2413	0	8.2905	8.2877	0	-1.4158	-1.4187	0	9.9109	9.9117
0	-5.6658	-5.6668	0	0.50636	0.50703	0	3.461	3.4626	0	-2.425	-2.4238	0	-2.939	-2.9416	0	5.6441	5.6434	0	0.82035	0.82044	0	8.5382	8.5376	0	3.8844	3.8853	0	2.9458	2.9465
0	0.14481	0.14433	0	1.3736	1.374	0	6.3776	6.3761	0	3.9709	3.9706	0	3.9137	3.9138	0	-4.2443	-4.2429	0	0.47573	0.47506	0	2.2692	2.2701	0	-2.7362	-2.7389	0	-2.8001	-2.8035
0	-3.2766	-3.2773	0	0.44222	0.44324	0	1.4534	1.4547	0	1.3385	1.3385	0	0.38629	0.38643	0	2.0608	2.0663	0	3.1437	3.1441	0	1.7335	1.7312	0	-2.2996	-2.3005	0	5.4226	5.4217
0	2.9527	2.9496	0	-0.14599	-0.14677	0	3.7327	3.7334	-2.1355	-2.1355	-2.1352	0	-0.72316	-0.72318	0	-5.7859	-5.7815	0	2.1932	2.1914	0	2.1033	2.1033	0	-5.1949	-5.1947	0	6.8803	6.8782
0	-0.23982	-0.24097	0	-0.047767	-0.04508	0	-6.5459	-6.5456	0	-3.0545	-3.0594	0	-0.9179	-0.917	0	-0.67352	-0.67197	0	3.6079	3.6081	0	-0.40879	-0.40951	-3.2763	-3.2763	-3.2772	0	-1.4559	-1.4525
0	0.015204	0.01334	0	7.8236	7.8214	0	1.3728	1.3699	0	-0.41183	-0.41069	0	-3.1486	-3.1466	0	-3.6573	-3.66	0	4.4518	4.4513	0	-6.9262	-6.9264	0	5.4378	5.4375	0	-1.4762	-1.4771
0	6.5328	6.5345	-2.1676	-2.1676	-2.1661	0	-1.7502	-1.7529	0	2.6391	2.6366	0	9.2672	9.2678	-3.9954	-3.9954	-3.9961	0	-3.126	-3.1253	-2.013	-2.013	-2.0134	0	-2.9598	-2.9599	0	-3.3194	-3.3184
0	10.7365	10.7356	0	-0.88722	-0.88848	0	1.6427	1.6403	0	2.9075	2.9082	0	3.7227	3.7238	0	-9.8546	-9.8517	0	0.83203	0.82907	0	-1.5995	-1.5981	0	-5.4845	-5.4837	0	-4.5098	-4.5119
0	1.7454	1.745	0	3.74	3.7403	0	1.341	1.3404	0	0.79637	0.7952	0	-2.5784	-2.576	0	-5.6205	-5.6219	0	1.9014	1.9012	0	-3.7693	-3.7677	0	-2.3832	-2.3845	-3.664	-3.664	-3.664
0	-0.42521	-0.42885	0	0.7146	0.71386	0	-0.28284	-0.28027	0	-1.851	-1.8488	0	-6.4325	-6.4313	0	-0.53046	-0.53025	1.3693	1.3693	1.3687	0	-2.2589	-2.2589	0	1.9269	1.9272	2.1441	2.1441	2.1436
0	1.1023	1.1006	0	-4.9625	-4.9609	0	5.2868	5.2895	0	7.6157	7.6174	0	1.3518	1.3516	0	2.5675	2.5702	0	-0.80571	-0.80587	0	0.54548	0.54376	0	-0.010246	-0.010062	0	1.8139	1.8117
0	-0.53882	-0.53854	0	0.45892	0.45923	0	-2.4669	-2.4671	0	-1.8318	-1.8323	0	0.092535	0.093538	0	-1.181	-1.181	0	-0.035657	-0.035593	0	-0.04287	-0.043929	0	-1.3468	-1.3474	0	-1.1784	-1.1787
0	0.77965	0.77989	0	0.37309	0.37306	0	-3.9338	-3.9344	0	-0.16711	-0.1668	0	1.0293	1.0307	0	0.067675	0.066998	0	0.19993	0.19976	0	-3.7695	-3.7709	0	1.3695	1.3695	0	-1.972	-1.9722
0	1.7833	1.7813	0	-0.98074	-0.97993	0	-1.6785	-1.677	0	4.1067	4.1051	0	-0.18983	-0.18779	0	-5.1708	-5.1715	0	-2.8403	-2.8401	0	-9.6931	-9.6945	0	-1.0817	-1.0822	0	-5.7938	-5.7946
0	-2.0386	-2.0397	0	5.7096	5.7071	0	-2.616	-2.6175	0	-6.4512	-6.4501	0	1.3823	1.3844	-2.3021	-2.3021	-2.3026	0	-0.51915	-0.52028	0	-1.2445	-1.2426	0	4.2843	4.2865	0	-0.52791	-0.52634
0	-1.14	-1.1412	0	2.9956	2.9963	0	-1.8082	-1.808	0	-3.1941	-3.1938	0	-1.3981	-1.3969	0	-1.2091	-1.2067	0	4.3671	4.3678	0	1.2832	1.2822	-0.43155	-0.43155	-0.4312	0	1.6284	1.6286
0	-6.0195	-5.9519	0	0.71879	0.734	0	2.3833	2.4435	0	0.098836	0.1186	0	-8.4964	-8.4522	0	6.1107	6.0338	0	-0.96153	-0.93568	0	0.33326	0.34451	0	2.5487	2.5149	0	2.996	3.0343
0	-2.9869	-2.9871	0	2.3792	2.3798	0	-2.8782	-2.8798	0	-4.9097	-4.9106	0	2.1477	2.1473	0	1.3383	1.3392	0	2.9721	2.9737	0	5.1416	5.1429	0	2.9125	2.913	0	-1.2368	-1.2351
0	0.50593	0.50875	0	-2.3973	-2.3968	0	-0.37923	-0.37977	0	3.8197	3.819	0	4.2846	4.2834	0	0.13929	0.1355	0	-5.6457	-5.6449	0	-2.7674	-2.7688	0	-0.82961	-0.83055	0	-5.6666	-5.6665
0	-0.3825	-0.386	0	1.5646	1.5633	0	1.641	1.641	0	-1.5934	-1.5931	0	1.1338	1.1317	0	-4.5729	-4.5683	0	-0.60449	-0.60473	0	-5.2743	-5.2751	0	-0.7386	-0.73959	0	6.8819	6.8807
0	2.4853	2.4872	0	-3.6626	-3.6637	0	-0.93677	-0.93855	0	1.1527	1.1505	0	6.7572	6.7581	0	-1.5384	-1.5386	0	-5.0797	-5.0791	0	-0.623	-0.61967	0	-1.1341	-1.1346	0	-1.3535	-1.3545
0	-2.9378	-2.9397	0	2.003	2.0069	0	3.6269	3.6269	0	-2.4324	-2.4308	0	3.1102	3.092	0	-2.5308	-2.5301	0	-0.054335	-0.043353	0	2.5398	2.5293	0	-1.172	-1.1673	0	4.3875	4.3871
0	-1.7517	-1.7535	0	3.1535	3.1579	0	2.7215	2.7236	0	-0.35898	-0.36113	0	1.2028	1.2042	0	-3.7014	-3.702	0	-0.80359	-0.80379	0	2.9658	2.9641	0	-5.5038	-5.5038	0	-1.4744	-1.4714
0	-1.4976	-1.4997	0	-1.905	-1.9039	0	-1.3081	-1.3053	-1.1807	-1.1807	-1.1804	-2.9409	-2.9409	-2.9411	0	1.6998	1.7002	0	-0.74444	-0.74462	0	-2.591	-2.592	0	0.61887	0.6176	0	4.7053	4.7055
-4.9883	-4.9883	-4.9878	0	5.7528	5.7538	0	-3.4272	-3.4292	0	-1.9747	-1.9739	0	4.6043	4.6053	-1.2753	-1.2753	-1.2765	0	-1.0202	-1.0193	0	-1.0795	-1.083	0	2.3378	2.3384	0	-8.0939	-8.0948

Table 1: Sample output (first 74 rows and 10 columns) of the GD algorithm: $\Omega_{i,j}$, $M_{i,j}$ and $\hat{M}_{i,j}$ separately respectively by the dotted line.