A Riemannian gossip approach to decentralized matrix completion

Bamdev Mishra
Amazon.com,
Bangalore, India
bamdevm@amazon.com

Hiroyuki Kasai
Graduate School of Information Systems,
The university of Electro-Communications
Chofu-shi, Tokyo, 182-8585, Japan
kasai@is.uec.ac.jp

Atul Saroop
Amazon.com,
Bangalore, India
asaroop@amazon.com

Abstract

In this paper, we propose novel gossip algorithms for the low-rank decentralized matrix completion problem. The proposed approach is on the Riemannian Grassmann manifold that allows *local* matrix completion by different agents while achieving asymptotic consensus on the *global* low-rank factors. The resulting approach is scalable and parallelizable. Our numerical experiments show the good performance of the proposed algorithms on various benchmarks, e.g., the Netflix dataset.

1 Introduction

The problem of low-rank matrix completion amounts to completing a matrix from a small number of entries by assuming a low-rank model for the matrix. The problem has many applications in control systems and system identification [1], collaborative filtering [2], and information theory [3], to name a just few. Consequently, it has been a topic of great interest and there exist many large-scale implementations for both *batch* [4, 5, 6, 7, 8, 9, 10] and *online* scenarios that focus on parallel and stochastic implementations [11, 12, 13, 14].

In this paper, we are interested in a *decentralized* setting, where we divide the matrix completion problem into smaller subproblems that are solved by many agents locally while simultaneously enabling them to arrive at a *consensus* that solves the full problem. In order to minimize the communication overhead between the agents, we constrain each agent to communicate with *only* one other agent as in the *gossip* framework [15]. One motivation is that this addresses privacy concerns of sharing sensitive data [16]. Another motivation is that the gossip framework is robust to scenarios where certain agents may be inactive at certain time slots, e.g., consider each agent to be a computing machine. We propose a preconditioned variant that is particularly well suited for ill-conditioned instances. Additionally, we also propose a parallel variant that allows to exploit parallel computational architectures. All the variants come with asymptotic convergence guarantees. To the best of our knowledge, this is the first work that exploits the gossip architecture for solving the decentralized matrix completion problem. Numerical comparisons show that the proposed algorithm show good performance on various benchmarks including on the Netflix dataset. The Matlab codes of the proposed algorithms are available at `https://bamdevmishra.com/codes/gossipmc/`.

2 Decentralized matrix completion problem formulation

The matrix completion problem is formulated as

$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{X}) - \mathcal{P}_\Omega(\mathbf{X}^\star)\|_F^2 \quad \text{subject to } \operatorname{rank}(\mathbf{X}) = r, \tag{1}$$

where $\mathbf{X}^\star \in \mathbb{R}^{m \times n}$ is a matrix whose entries are known for indices if they belong to the subset $(i, j) \in \Omega$ and Ω is a subset of the complete set of indices $\{(i, j) : i \in \{1, ..., m\} \text{ and } j \in \{1, ..., n\}\}$.

9th NIPS Workshop on Optimization for Machine Learning (OPT 2016), Barcelona, Spain.

The operator $\mathcal{P}_\Omega(\mathbf{X}_{ij}) = \mathbf{X}_{ij}$ if $(i,j) \in \Omega$ and $\mathcal{P}_\Omega(\mathbf{X}_{ij}) = 0$ otherwise is called the orthogonal sampling operator and is a mathematically convenient way to represent the set of known entries. The rank constraint parameter r is usually set to a low value, i.e., $\ll (m,n)$ to seek low-rank completion.

A way to handle the rank constraint in (1) is by a fixed-rank matrix parameterization. In particular, we use $\mathbf{X} = \mathbf{U}\mathbf{W}^T$, where $\mathbf{U} \in \mathrm{St}(r,m)$ and $\mathbf{W} \in \mathbb{R}^{n\times r}$, where $\mathrm{St}(r,m)$ is the set of $m \times r$ matrices with *orthonormal* columns. Additionally, fixing $\mathbf{U}$, $\min_{\mathbf{W}\in\mathbb{R}^{n\times r}} \|\mathcal{P}_\Omega(\mathbf{U}\mathbf{W}^T) - \mathcal{P}_\Omega(\mathbf{X}^\star)\|_F^2$ can be solved in closed form. Equivalently, (1) reads

$$\min_{\mathbf{U}\in\mathrm{St}(r,m)} \quad \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{U}\mathbf{W}_{\mathbf{U}}{}^T) - \mathcal{P}_\Omega(\mathbf{X}^\star)\|_F^2 \tag{2}$$

where $\mathbf{W}_\mathbf{U}$ is the solution to the inner optimization problem $\min_{\mathbf{W}\in\mathbb{R}^{n\times r}} \|\mathcal{P}_\Omega(\mathbf{U}\mathbf{W}^T) - \mathcal{P}_\Omega(\mathbf{X}^\star)\|_F^2$.

We distribute the task of solving the problem (2) among N agents, which perform certain computations independently. To this end, we partition the incomplete matrix $\mathbf{X}^\star = [\mathbf{X}_1^\star, \mathbf{X}_2^\star, \ldots, \mathbf{X}_N^\star]$ along the columns such that the size of $\mathbf{X}_i^\star$ is $m \times n_i$ with $\sum n_i = n$ for $i = \{1,2,\ldots,N\}$. Each agent i has knowledge of the incomplete matrix $\mathbf{X}_i^\star$ and its *local* set of indices Ω_i of known entries. We also partition the weight matrix $\mathbf{W}$ as $\mathbf{W}^T = [\mathbf{W}_1^T, \mathbf{W}_2^T, \ldots, \mathbf{W}_N^T]$ such that the matrix $\mathbf{W}_i$ has size $n_i \times r$. A straightforward reformulation of (2) is

$$\min_{\mathbf{U}\in\mathrm{St}(r,m)} \frac{1}{2}\sum_i \underbrace{\|\mathcal{P}_{\Omega_i}(\mathbf{U}\mathbf{W}_{i\mathbf{U}}^T) - \mathcal{P}_{\Omega_i}(\mathbf{X}_i^\star)\|_F^2}_{\text{problem handled by agent } i}, \tag{3}$$

where $\mathbf{W}_{i\mathbf{U}}$ is the least-squares solution to $\min_{\mathbf{W}_i\in\mathbb{R}^{n_i\times r}} \|\mathcal{P}_{\Omega_i}(\mathbf{U}\mathbf{W}_i^T) - \mathcal{P}_{\Omega_i}(\mathbf{X}_i^\star)\|_F^2$, which can be computed by agent i independently of other agents.

Although the computational workload gets distributed in (3), all agents still require the knowledge of the common $\mathbf{U}$ (to compute the matrices $\mathbf{W}_{i\mathbf{U}}$). To circumvent this issue, each agent i stores a local copy $\mathbf{U}_i$, which it then updates based on information from its *neighbors*. For minimizing the communication overhead between agents, we additionally put the constraint that at any time slot only two agents communicate, i.e, each agent has exactly only one neighbor. To this end, the agents are numbered according to their proximity, e.g., for $i \leqslant N-1$, agents i and $i+1$ are neighbors. Equivalently, agents 1 and 2 are neighbors and can communicate. Similarly, agents 2 and 3 communicate, and so on. This communication between the agents allows to reach a *consensus* on $\mathbf{U}_i$. Specifically, it suffices that the *column spaces* of all $\mathbf{U}_i$ converge. (The precise motivation and formulation are in Section 3.) Our proposed decentralized matrix completion problem formulation is

$$\begin{aligned}\min_{\mathbf{U}_1,\ldots,\mathbf{U}_N\in\mathrm{St}(r,m)} \quad & \sum_i \underbrace{f_i := \frac{1}{2}\|\mathcal{P}_{\Omega_i}(\mathbf{U}_i\mathbf{W}_{i\mathbf{U}_i}^T) - \mathcal{P}_{\Omega_i}(\mathbf{X}_i^\star)\|_F^2}_{\text{completion task handled by agent } i} \\ & + \frac{\rho}{2}\underbrace{(d_1^2(\mathbf{U}_1,\mathbf{U}_2) + d_2^2(\mathbf{U}_2,\mathbf{U}_3) + \ldots + d_{N-1}^2(\mathbf{U}_{N-1},\mathbf{U}_N))}_{\text{consensus}},\end{aligned} \tag{4}$$

where d_i is a certain distance measure between $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$ for $i \leqslant N-1$ and $\rho \geqslant 0$ is a parameter that trades off matrix completion with consensus. It should be noted that the consensus term has $N-1$ particular pairwise distances. The particular structure in the consensus term helps in achieving consensus in a straightforward manner [17, Section 4.4]. For a large ρ, the consensus term in (4) dominates, minimizing which allows the agents to arrive at consensus. For $\rho = 0$, the optimization problem (4) solves N independent completion problems and there is no consensus. For a sufficiently large ρ, the problem (4) achieves the goal of approximate matrix completion along with consensus.

3 The Riemannian gossip algorithm

It should be noted that the optimization problem (2) only depends on the *column space* of $\mathbf{U}$ rather than $\mathbf{U}$ itself [9, 11]. Subsequently, the optimization problem (2), and similarly (3), is conceptually on the *Grassmann* manifold $\mathrm{Gr}(r,m)$, which is the set of r-dimensional subspaces in $\mathbb{R}^m$ [19]. However, numerical algorithms, by necessity, are implemented with matrices $\mathbf{U}$ on $\mathrm{St}(r,m)$. It should be stated that the Grassmann manifold is a *Riemannian* manifold and optimization on the Grassmann manifold is a well studied topic in literature [19]. Notions such as the Riemannian gradient (first order derivative of a cost function), geodesic (shortest curve), and logarithm mapping ("difference"

Table 1: Proposed online gossip algorithm for (4)

1. At each time slot t, pick an agent $i \leqslant N-1$ randomly with uniform probability. Compute the Riemannian gradients $\mathrm{grad}_{x_i} f_i, \mathrm{grad}_{x_{i+1}} f_{i+1}, \mathrm{grad}_{x_i} d_i$, and $\mathrm{grad}_{x_{i+1}} d_i$ as

$$\mathrm{grad}_{x_i} f_i = \mathrm{Grad}_{x_i} f_i - \mathbf{U}_i^T(\mathbf{U}_i^T \mathrm{Grad}_{x_i} f_i), \mathrm{Grad}_{x_i} f_i = (\mathcal{P}_{\Omega_i}(\mathbf{U}_i \mathbf{W}_{i\mathbf{U}_i}^T) - \mathcal{P}_{\Omega_i}(\mathbf{X}_i^\star))\mathbf{W}_{i\mathbf{U}_i},$$
$$\mathrm{grad}_{x_i} d_i = -\mathrm{Log}_{x_i}(x_{i+1}), \text{ and } \mathrm{grad}_{x_{i+1}} d_i = -\mathrm{Log}_{x_{i+1}}(x_i),$$

   where $x_i = \mathbf{U}_i$, $x_{i+1} = \mathbf{U}_{i+1}$, and $\mathrm{Log}_{x_i}(x_{i+1})$ is the *logarithm* mapping, which is well defined on $\mathrm{Gr}(r, m)$. It should be noted that the Riemannian gradient of the Riemannian distance is the negative logarithm mapping [18].
2. Given a stepsize γ_t, update $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$ as

$$(\mathbf{U}_i)_+ = \mathrm{Exp}_{x_i}(-\gamma_t(\alpha_i \mathrm{grad}_{x_i} f_i + \rho \mathrm{grad}_{x_i} d_i))$$
$$(\mathbf{U}_{i+1})_+ = \mathrm{Exp}_{x_{i+1}}(-\gamma_t(\alpha_{i+1} \mathrm{grad}_{x_{i+1}} f_{i+1} + \rho \mathrm{grad}_{x_{i+1}} d_i)),$$

   where $x_i = \mathbf{U}_i$ and $x_{i+1} = \mathbf{U}_{i+1}$. $\alpha_i = 1$ if $i = \{1, N\}$, else $\alpha_i = 0.5$, which balances the terms. $\mathrm{Exp}_{x_i}(\xi_{x_i})$ is the *exponential* mapping, which is well defined on $\mathrm{Gr}(r, m)$.

between elements) have closed-form expressions [19]. The distance measure d_i in (4) is chosen as the Riemannian distance between the column spaces of $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$. It is easy to see that the critical points of the consensus term in (4), by construction, achieve consensus [17].

We exploit the stochastic gradient descent (SGD) setting framework proposed in [17] for (4). We make the following assumptions.

A1 Agents i and $i+1$ are neighbors for all $i \leqslant N-1$.

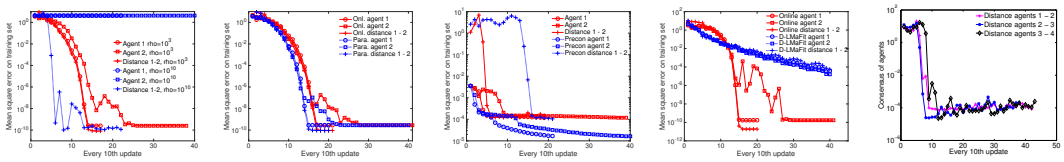
A2 At each time slot, say t, we pick an agent $i \leqslant N-1$ randomly with uniform probability. From assumption **A1**, this means that we also pick agent $i+1$ (the neighbor of agent i). Subsequently, agents i and $i+1$ update $\mathbf{U}_i$ and $\mathbf{U}_{i+1}$, respectively, by taking a *gradient descent step* with stepsize γ_t on $\mathrm{Gr}(r, m)$. That is, our sample cost is $f_i + f_{i+1} + \rho d_i^2$. The stepsize sequence satisfies the standard conditions, i.e., $\sum \gamma_t^2 < \infty$ and $\sum \gamma_t = +\infty$.

Although the particular structure of the consensus term in (4) assumes agents topology, there is still flexibility in terms of sampling for SGD. **A1** here specifically helps define a "sample". Finally, the proposed algorithm is presented in Table 1. Overall, the computational cost of each agent update is $O(|\Omega_i| r^2 + n_i r^2 + mr)$. The Grassmann manifold related ingredients cost $O(mr^2 + r^3)$.

Convergence analysis. Asymptotic convergence analysis of the algorithm in Table 1 follows directly from the analysis in [17, Theorem 1]. The key idea is that for a compact Riemannian Grassmann manifold all continuous functions of the parameter are bounded. Subsequently, under a decreasing stepsize condition and noisy gradient estimates (an unbiased estimator of the batch gradient), the algorithm in Table 1 converges to a critical point of (4) *almost surely*.

Preconditioned variant. Given the Riemannian gradient $\xi_{x_i} = \mathrm{grad}_{x_i} f_i + \rho \mathrm{grad}_{x_i} d_i$ computed by agent i at $x_i = \mathbf{U}_i$, we propose the *preconditioner* $\xi_{x_i} \mapsto \xi_{x_i}(\underbrace{\mathbf{W}_{i\mathbf{U}_i}^T \mathbf{W}_{i\mathbf{U}_i}}_{\text{from completion}} + \underbrace{\rho \mathbf{I}}_{\text{from consensus}})^{-1}$, where $\mathbf{I}$ is the $r \times r$ identity matrix. Other than the scaling of the Riemannian gradient, the rest of the formulas in Table 1 remain the same for the preconditioned variant. Applying the preconditioner costs $O(n_i r^2 + r^3)$. The term $\mathbf{W}_{i\mathbf{U}_i}^T \mathbf{W}_{i\mathbf{U}_i}$ is computationally cheap to compute and it captures a *block diagonal approximation* of the Hessian of the simplified (but related) cost function $\|\mathbf{U}_i \mathbf{W}_{i\mathbf{U}_i}^T - \mathbf{X}_i^\star\|_F^2$. The works [6, 7, 9] use the term $\mathbf{W}_{i\mathbf{U}_i}^T \mathbf{W}_{i\mathbf{U}_i}$ as an effective *preconditioner*. The term $\rho \mathbf{I}$ is motivated from an approximation of the second order derivative of the consensus term in (4).

Parallel variant. Parallelization of the algorithm in Table 1 follows from a particular sampling of agents. We explain the idea for $N = 5$. Updates of the agents are divided into two *rounds*. In round 1, we pick agents 1 and 3, i.e., all the *odd* numbered agents. It should be noted the formulation (4), by construction, allows to update the agent pairs $(1, 2)$ and $(3, 4)$ in parallel. Similarly, in round 2, we pick agents 2 and 4, i.e., all the *even* numbered agents and update the agent pairs $(2, 3)$ and $(4, 5)$ in parallel. The key idea is that sampling is on the rounds and not on the agents. For example, we pick a round with uniform probability.



(a) Effect of ρ. (b) Online versus parallel. (c) Preconditioning. (d) Online versus D-LMaFit. (e) Netflix: consensus.

Figure 1: Performance of proposed algorithms in different scenarios.

4 Numerical comparisons

The proposed algorithm in Table 1 (Online Gossip) and its variants are compared on different problem instances. The implementations are based on the Manopt toolbox [20]. We also show comparisons with RTRMC-1 [9] (batch gradient descent) and D-LMaFit [16] (decentralized). $N = 6$ for our proposed algorithms. Online algorithms are run for a maximum of 1000 iterations. The parallel variant is run for 400 iterations. The stepsize sequence is defined as $\gamma_t = \gamma_0/t$, where t is the time slot and γ_0 is set using cross validation. D-LMaFit and RTRMC-1 are run for 400 iterations. For simplicity, all figures only show the plots for only few agents. All simulations are performed in Matlab. For each synthetic example, an $m \times n$ random matrix of rank r is generated as in [4]. A fraction of the entries, based on sampling ratio $\text{OS} := |\Omega|/(mr + nr - r^2)$, are randomly removed with uniform probability to construct the training set Ω and data $\mathbf{X}^\star$. Gaussian noise with mean zero and standard deviation 10^{-6} is added to the data. The matrices $\mathbf{X}_i^\star$ are created by dividing $\mathbf{X}^\star$ along columns equally among the agents. The training and test sets are also partitioned similarly.

Effect of ρ: we consider a problem instance of size $10\,000 \times 100\,000$ of rank 5 and OS 6. Two scenarios with $\rho = 10^3$ and $\rho = 10^{10}$ are considered. Figure 1(a) shows the performance of Online Gossip. Not surprisingly, for $\rho = 10^{10}$, we only see consensus (the distance between agents 1 and 2 tends to zero). For $\rho = 10^3$, we see both completion and consensus, which validates the theory. **Online versus parallel:** we consider the previous example with $\rho = 10^3$. Figure 1(b) shows that both online and parallel variants perform similarly. **Ill-conditioned instances:** we consider a problem instance of size $5\,000 \times 50\,000$ of rank 5 and impose an exponential decay of singular values with condition number 500 and OS 6. Figure 1(c) shows the performance of Online Gossip and its preconditioned variant for $\rho = 10^3$. Overall, the preconditioned variant shows a superior performance in Figure 1(c). **Comparisons with D-LMaFit:** we consider a problem instance of size $500 \times 12\,000$, rank 5, and OS 6. D-LMaFit is run with the default parameters. For Online Gossip, we set $\rho = 10^3$. As shown in Figure 1(d), Online Gossip quickly outperforms D-LMaFit. Overall, Online Gossip takes fewer number of updates to reach a high accuracy. **Netflix dataset:** the dataset (obtained from the code of [13]) consists of $100\,480\,507$ ratings by $480\,189$ users for $17\,770$ movies. We perform 10 random $80/20$ train/test partitions. The train ratings are centered around 0. We split both the train and test data among the agents along the number of users. We run Online Gossip with $\rho = 10^7$ (set with cross validation) and for $400(N-1)$ iterations and $N = \{2, 5, 10, 15, 20\}$ agents. We show the results for rank 10 [9]. Additionally, we add a regularization term $\|\mathbf{X} - \mathcal{P}_\Omega(\mathbf{X})\|_F^2$ with regularization parameter set to 0.01. Table 2 shows the root mean square error (RMSE) obtained on the entire test set averaged over 10 runs. For comparison, we show the test RMSE obtained by the batch algorithm RTRMC-1. Figure 1(e) shows the consensus of agents for the case $N = 10$.

Table 2: Performance of Online Gossip on the Netflix dataset at rank 10

	$N = 2$	$N = 5$	$N = 10$	$N = 15$	$N = 20$	RTRMC-1 (batch method)
Test RMSE	0.877	0.885	0.891	0.894	0.900	0.873

5 Conclusion

We have proposed a Riemannian gossip approach to the decentralized matrix completion problem. Specifically, the completion task is distributed among a number of agents, which are then required to achieve consensus. This is modeled as minimizing a weighted sum of *completion* and *consensus* terms on the Grassmann manifold. We propose a novel stochastic gradient descent algorithm, and its preconditioned and parallel variants, for the problem with simple updates. Numerical experiments show the competitive performance of the proposed algorithms on different benchmarks.

References

[1] I. Markovsky and K. Usevich. Structured low-rank approximation with missing data. *SIAM Journal on Matrix Analysis and Applications*, 34(2):814–830, 2013.

[2] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *International Conference on Machine learning (ICML)*, pages 713–719, 2005.

[3] J. Shi, Y.and Zhang and K. B. Letaief. Low-rank matrix completion for topological interference management by Riemannian pursuit. *IEEE Transactions on Wireless Communications*, PP(99), 2016.

[4] J. F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[5] K. C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6(3):615–640, 2010.

[6] T. T. Ngo and Y. Saad. Scaled gradients on Grassmann manifolds for matrix completion. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1421–1429, 2012.

[7] B. Mishra and R. Sepulchre. R3MC: A Riemannian three-factor algorithm for low-rank matrix completion. In *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC)*, pages 1137–1142, 2014.

[8] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

[9] N. Boumal and P.-A. Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra and its Applications*, 475:200–239, 2015.

[10] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[11] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *The 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 704–711, June 2010.

[12] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems*, 41(3):793–819, 2014.

[13] B. Recht and C Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[14] C. Teflioudi, F. Makari, and R. Gemulla. Distributed matrix completion. In *International Conference on Data Mining (ICDM)*, pages 655–664, 2012.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transaction on Information Theory*, 52(6):2508–2530, 2006.

[16] A.-Y. Lin and Q. Ling. Decentralized and privacy-preserving low-rank matrix completion. *Journal of the Operations Research Society of China*, 3(2):189–205, 2015.

[17] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[18] K. Hüper, U. Helmke, and S. Herzberg. On the computation of means on Grassmann manifolds. In *International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pages 2439–2441, 2010.

[19] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[20] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt: a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(Apr):1455–1459, 2014.