Project Report
Low-rank Approximations for Large Incomplete Matrices

Liliya Ageeva
Sergey Makarychev
Aleksandr Rozhnov
Anton Zhevnerchuk

https://github.com/zhevnerchuk/matrix-filling

Abstract

There are some challenging industrial problems in which only incomplete data is available and the goal is to "complete" the data. This problem can be stated as an optimization problem in at least two different ways, and a variety of methods can be used for solving the problem in each of the formulations. We implemented three competitive algorithms for matrix-filling problem and compared their performances to each other.

Contents

1 Contribution of members

1.1 Lilya Ageeva

Conduction of numerical experiments, implementation of some necessary functions and aid in looking for bugs, designing the presentation (structure and contents), working with the theory of Alternating Least Squares.

1.2 Sergey Makarychev

Studying of the theory of Riemannian optimization for solving an optimization problem on manifold of matrix of rank k. Implementation of this algorithm in Python. Evaluation of the quality of algorithms (convergence, time). Contribution to the presentation.

1.3 Alekasndr Rozhnov

Studying the Alternating Least Squares Method for matrix completion and implementation of this algorithm in Python. Preprocessing of images for evaluation of implemented algorithms with them.

1.4 Anton Zhevnerchuk

Implementation of Soft-Input algorithm and studying its theoretical basis. Debugging of the Riemannian Optimization. Contribution to the presentation and presenting the project at the projects defense session.

2 Background

The growth of the Internet influenced the development of recommender systems, data compression and reconstruction. Some of the problems appearing in that fields deal with data, which can be naturally represented as a matrix with some observed values and others values unknown. The goal in such problems is to guess the data at not observed entries. Usually it is assumed that our data has low-rank structure: this assumption follows naturally from the data structure or from the algorithms used for data compression.

3 Problem formulation

We have a matrix $X \in \mathbb{R}^{m \times n}$ with some unknown entries. Let us denote by Ω the set of positions (i, j) for which $X_{i,j}$ is known. We want to construct a low-rank approximation Z of X. In our project we considered two approaches to solving that problem:

- Fix rank of an approximation and minimize error at known entries;
- Fix acceptable error at known entries and minimize rank of an approximation.

For the following we need to introduce some notation. For a $m \times n$ matrix A $P_\Omega(A)$ is a projection of A on the set of known entries:

$$P_\Omega(A)_{i,j} = \begin{cases} A_{i,j}, & (i,j) \in \Omega \\ 0, & (i,j) \notin \Omega \end{cases}$$

When we fix rank, our problem can be formulated as follows:

$$\text{minimize} \quad f(Z) := \frac{1}{2}\|P_\Omega(X - Z)\|_F^2 \qquad \text{subject to} \quad \text{rank}(Z) = k. \tag{1}$$

Here k is a parameter of an algorithm and should be set before running. Two of three implemented algorithms (Alternating Least Squares and Riemannian Optimization) corresponds to that approach.

When we fix acceptable error at observed entries $\delta > 0$ and minimize rank of an approximation, we have the following problem:

$$\text{minimize} \quad \text{rank}(Z) \qquad \text{subject to} \quad \sum_{(i,j)\in\Omega} (X_{i,j} - Z_{i,j})^2 \leq \delta. \tag{2}$$

Since this optimization problem 2 becomes too hard in general case, the following modification is considered:

$$\text{minimize} \quad \|Z\|_* \qquad \text{subject to} \quad \sum_{(i,j)\in\Omega} (X_{i,j} - Z_{i,j})^2 \leq \delta. \tag{3}$$

Here $\|\cdot\|_*$ is a nuclear norm, which is just a sum of singular values of a matrix. Such a relaxation is a classic approach for matrix-filling problem and is used in many papers (Fazel, 2002; Candes and Recht, 2008; Candes and Tao, 2009; Recht et al., 2007). It occurs that problem 3 is convex, which makes it much easier to solve. Nuclear norm is clearly connected with the rank, so in general a solution for 3 seems to be also a rather good solution for 2. It can be shown that for every $\delta > 0$ there is a $\lambda > 0$ such that 3 is equal to

$$\text{minimize} \quad \frac{1}{2}\sum_{(i,j)\in\Omega} (X_{i,j} - Z_{i,j})^2 + \lambda\|Z\|_*. \tag{4}$$

So, our goal is to solve problem 4. One of the implemented algorithms (Soft-Input) corresponds to that formulation of a matrix completion problem.

4 Implemented approaches

4.1 Soft-Input

This algorithm is designed for solving matrix-completion problem in the formulation 4. If all matrix X is known, a solution for 4 can be found analytically. If $X = U\Lambda V^T$ is SVD for X, a solution is given by

$$Z = U\Lambda_\lambda V^T, \text{ where } \Lambda_\lambda = \text{diag}\Big(\max\{\sigma_1 - \lambda, 0\}, \ldots, \max\{\sigma_{\min\{m,n\}} - \lambda, 0\}\Big).$$

Soft-Input is an iterative method for solving 4 in general setting with very simple main idea: at each iteration we replace unknown entries of X with the values of a current approximation Z_ℓ and solve problem 4 for a matrix with all entries known. In [MHT10] it is shown that this iterative algorithm converges to a solution for 4.

At each iteration we need to find SVD of $m \times n$ matrix $P_\Omega(X) - P_\Omega(Z_\ell) + Z_\ell$. Note that Z_ℓ is a matrix with low-rank, so MATVEC operations can be done in $O((m+n)k)$ flops, where k is rank of Z_ℓ. Matrix $P_\Omega(X) - P_\Omega(Z_\ell)$ is sparse. Therefore, if X has good sparse structure, MATVEC operations with matrix $P_\Omega(X) - P_\Omega(Z_\ell)$ are comparatively cheap as well, so truncated SVD of $P_\Omega(X) - P_\Omega(Z_\ell) + Z_\ell$ can be found much faster than in general case.

4.2 Riemannian Optimization

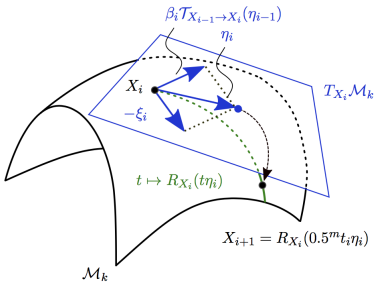
Let us note that constraints in the problem 1 form a smooth manifold $\mathcal{M}_k$ of matrices of rank k: $\mathcal{M}_k := \{Z \in \mathbb{R}^{m\times n} : \operatorname{rank}(Z) = k\}$. Since the objective function f is also smooth, problem 1 is a smooth optimization problem, and so Riemannian optimization is the generalization of standard unconstrained optimization, where the search space is $\mathbb{R}^n$, to optimization of a smooth objective function on a Riemannian manifold.

This algorithm for matrix completion minimizes the least-square distance on the sampling set over the Riemannian manifold of fixed-rank matrices. This method use a generalization of classical non-linear conjugate gradients (CG) on Euclidean space to perform optimization on manifolds.

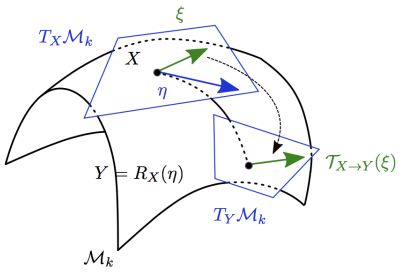
As a tangent vector only gives a direction but not the line search itself on the manifold, a smooth mapping called the retraction, is needed to map tangent vectors to the manifold.

To improve convergence, this method requires taking a linear combination of the Riemannian gradient with the previous search direction, that does not lie in current tangent space. So, it needs to be transported from the previous tangent space to the current one. This is called vector transport.

Visualization of non-linear CG on a Riemannian manifold



Vector transport on a Riemannian manifold



4.3 Alternating Least Squares (ALS)

Alternating Least Squares method solves matrix completion problem minimizing objective function 1. To avoid overfitting we use regularization term with coefficient λ. We use the same notations to formulate the problem:

Let us denote $U \in \mathrm{Mat}^{k\times m}(\mathbb{R})$ and $V \in \mathrm{Mat}^{k\times n}(\mathbb{R})$ as factor matrices, so our goal is then to estimate the complete matrix $X \approx U^T V$. We can formulate this problem as an optimization problem in which we aim to minimize an objective function and find optimal

U and V. In particular, we aim to minimize the least squares error:

$$\left(\sum_{(i,j)\in\Omega}(X_{i,j}-U_i^TV_j)+\lambda\left(\sum_i\|U_i\|^2+\sum_j\|V_j\|^2\right)\right)\to\min_{U,V},$$

where single index denotes the number of column of the corresponding matrix.

Notice that this objective is non-convex, however, if we fix the set of variables U and treat them as constants, then the objective is a convex function of V and vice versa. If we fix U and take derivatives with respect to columns of V and assign them to zero, then the roots of this equation will give values of columns of V which give the local minimum. After that we repeat these operations considering V as constants. Thus we obtain the following formulae for iterative update of U and V:

$$U_j=\left(\sum_{(j,h)\in\Omega_{(j,*)}}V_hV_h^T+\lambda\mathcal{I}_k\right)^{-1}\cdot\sum_{(j,h)\in\Omega_{(j,*)}}X_{jh}V_h,\quad j=1,\dots m$$

$$V_j=\left(\sum_{(h,j)\in\Omega_{(*,j)}}U_hU_h^T+\lambda\mathcal{I}_k\right)^{-1}\cdot\sum_{(h,j)\in\Omega_{(*,j)}}X_{jh}U_h,\quad j=1,\dots n$$

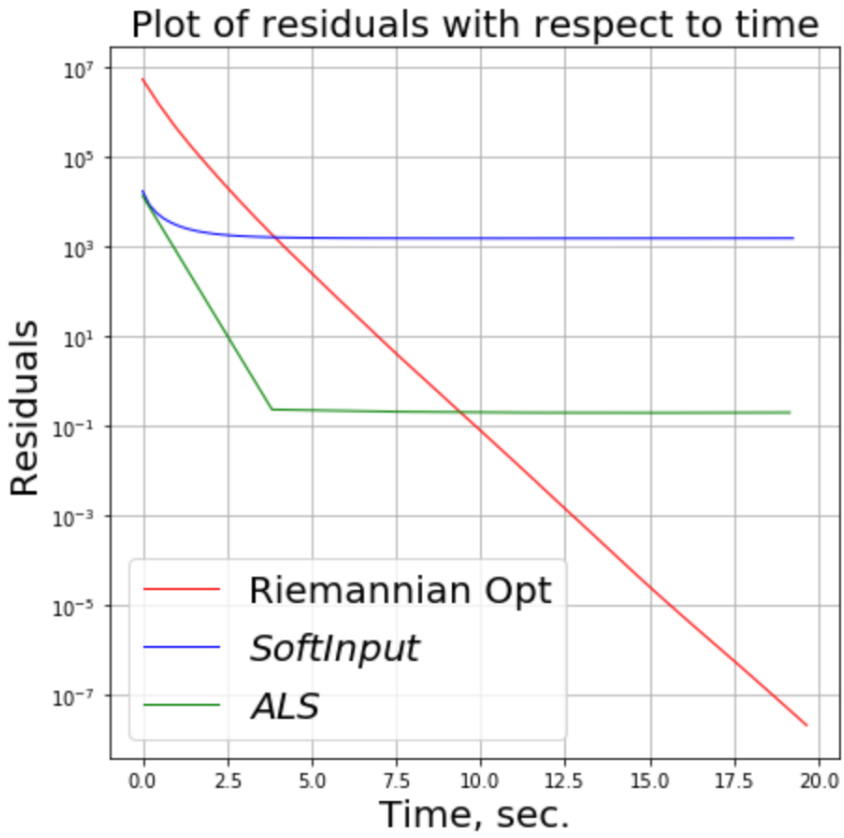
5 Comparative Analysis

	ALS	**Riemannian**	**SOFT-INPUT**
Parallelizable	✓	✓	✗
Adaptive rank choice	✗	✗	✓
Flops per iteration	$O(\|\Omega\|K^2+K^3)$	$O((M+N)K^2+\|\Omega\|K))$	$O((M+N+\|\Omega\|)K^3)$

This table shows general pros and contras of implemented algorithms. Note that finding truncated SVD (which is a subroutine of Soft-Input) can be parallelized, but in ALS and Riemannian Optimization much more complicated subroutines can be done by independent threads simultaneously, so their parallelization seems to be much more effective. Moreover, both ALS and Riemannian Optimization can be slightly modified to get adaptive rank choice. Generally, it can be made as follows: when we converge to a solution with rank k we start looking for a solution with rank $k+1$, and use a retraction of a found rank-k solution as a first guess.

Presented complexities of one iteration are practical. However, theoretical bounds are slightly different. For Riemannian Optimization there is also a term $O(\text{nb}_{Arm}((m+$

$n)k^2 + |\Omega|k))$, where nb_{Arm} is a number of done steps while Armijo search. In practice it almost always is equal to zero. For Soft-Input we obtain such a complexity only if MATVEC operations with $P_\Omega(X)$ can be done in $O(|\Omega|k)$ flops. Even then Soft-Input has the most expensive iteration among implemented algorithms, so if X has bad sparse structure, Soft-Input becomes uncompetitive in sense of working time.



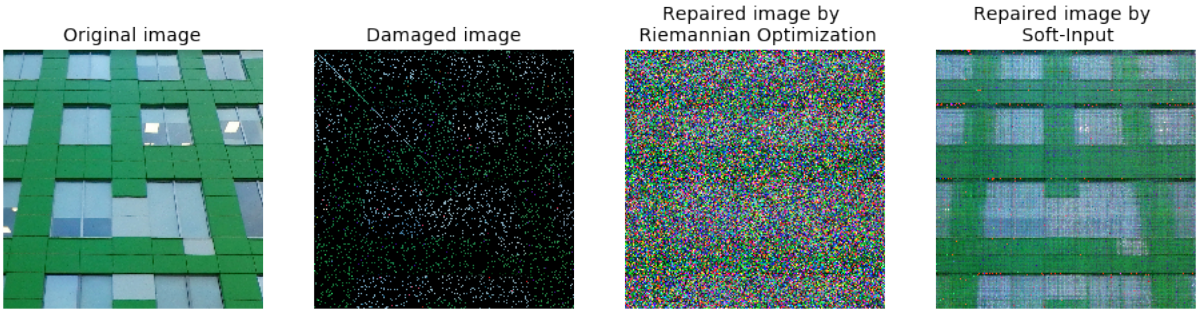
Despite the fact that all three algorithms deal with different objective functions, they can be still compared at the set of known entries. The plot above shows the behavior of the squared error on Ω. As we can see in the figure, both Riemannian Optimization and ALS converge fast. We would like to note that regularization used in the ALS does not allow to reduce the residuals to zero. Soft-Input works much longer than others, but at the same time, it should be noted that it chooses the rank adaptively, which the previous two algorithms do not.

6 Demonstration

More details of this section you can find in notebook "results.ipynb"

In the following experiment we apply SVD to 'Iphone.jpg' to create low-rank approximation of the image and remove apple from the back of Iphone. After that using ALS we recover 'damaged' image. Results are presented in the following sketches:

In the second experiment we download a photo of one of the buildings in Mendeleev Quarter. After that we set only about 5% of pixels to be known and then we try to repair the photo using some of implemented algorithms. We have achieved the following results:



We managed to find pretty completion using Soft-Input. Completion found with Riemannian Optimization doesn't seem to be very good, but note that the formulation of completion problem solving by Riemannian Optimization is not convex and can have more than one local optimas. So, we are likely to converge to a wrong local optima.

7 Acknowledgements

We immensely appreciate help of our mentor Maksim Rakhuba at all steps of the working on this project, especially for sharing useful articles and application cases of our algorithms.

References

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani, *Spectral Regularization Algorithms for Learning Large Incomplete Matrices*, Journal of Machine Learning (2010), available at `https://web.stanford.edu/%7Ehastie/Papers/mazumder10a.pdf`.

[Van12] B. Vandereycken, *Low-rank matrix completion by Riemannian optimization* (2012), available at `https://arxiv.org/pdf/1209.3834.pdf`.

[Zad15] R. Zadeh, *Lecture Notes on Distributed Algorithms and Optimization,* Stanford (2015), available at `https://stanford.edu/%7Erezab/classes/cme323/S15/notes/lec14.pdf`.