Project Report
Matrix Completion Dream Team

Sergey Makarychev and Aleksandr Rozhnov

https://github.com/MSergeyV/Matrix-Completion

Abstract

The matrix completion problem is the problem of finding or approximating a low-rank matrix based on a few samples of this matrix. We implemented several competitive algorithms and performed their exhaustive analysis: compared the speed of convergence, applied these algorithms for recovering image problem and recommender systems. For the last problem we measured their performance using HR and RMSE metrics.

Contents

1 Contribution of members

1.1 Sergey Makarychev

Implementing FastALS. Implementing Riemannian Optimization. Evaluation of the quality of algorithms (convergence, time). Contribution to the presentation and presenting the project at the projects defense session.

1.2 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.

2 Background

A wide range of datasets are naturally organized in matrix form. Some of the elements of these matrices can be unobserved, others can be lost. Thus, the problem of reconstruction missed values appears. For example, we can consider the term-document matrix: The frequencies of words used in a collection of documents can be represented as a matrix, where each entry corresponds to the number of times the associated term appears in the indicated document. There are many others cases where we need to fill matrices, thus there is high demand in robust and fast algorithms to solve this problem.

3 Problem Statement

Let $X \in \mathbb{R}^{m \times n}$ denote a matrix of the size $(m \times n)$ that is only known on the set Ω. Let's define the following projection operator P_Ω:

$$P_\Omega(X) : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}, X_{i,j} = \begin{cases} X_{i,j}, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise} \end{cases}$$

The matrix completion problem is the following optimization problem:

$$\begin{aligned} &\text{minimize } \operatorname{rank}(Z) \\ &\text{s.t } P_\Omega(X) = P_\Omega(Z) \end{aligned}$$

The last problem is NP-hard. So, we can consider convex relaxation of this problem by introduction nuclear norm:

$$\begin{aligned} &\text{minimize } \|Z\|_* \\ &\text{s.t } P_\Omega(X) = P_\Omega(Z), \end{aligned}$$

here $\|\cdot\|_*$ is a nuclear norm - the sum of singular values of a matrix. This convex relaxations is widely used and there are some strong theoretical probabilistic results in this topic. For instance, authors in ([ECaTT09]) showed that when Ω is sampled uniformly at random, then the nuclear relaxation of the problem can recover with high probability any matrix X of rank k.

Due to the presence of noise, consider a more robust version with the tolerance $\epsilon \geq 0$.

$$\|P_\Omega(X) - P_\Omega(Z)\|_F \leq \epsilon$$

4 Implemented approaches

4.1 Alternating Least Squares (ALS)

Alternating Least Squares method solves matrix completion problem minimizing objective function

$$\underset{U,V}{\text{minimize}} \, \|P_\Omega\left(X - UV^T\right)\|_F^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right),$$

where $U \in \text{Mat}^{k\times m}(\mathbb{R}), V \in \text{Mat}^{k\times n}(\mathbb{R})$ are unknown matrices which we try to obtain. To avoid overfitting we use regularization term with coefficient λ.

One can notice that this objective is non-convex, however, if we fix U and treat it as constant matrix, then the objective is a convex function with respect to V and vice versa. Consequently, if we fix one of the matrices and take derivatives with respect to columns of the second one and assign them to zero, then the roots of this equations will give global minimum of objective with fixed first matrix. After that we repeat these operations considering another matrix as constant. Thus, on each step we push objective to minimum and we obtain the following formulae for iterative update of U and V:

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

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

4.2 FastALS

Consider the problem in the form:

$$\underset{U,V}{\text{minimize}} \, \|P_\Omega\left(X - UV^T\right)\|_F^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right).$$

The algorithm exploits the decomposition:

$$P_\Omega\left(X - UV^T\right) = P_\Omega\left(X\right) + P_\Omega^\perp\left(UV^T\right) - UV^T,$$

where $P_\Omega^\perp$ projects onto the complement of the set Ω.

Suppose we have current estimates for U and V, and wish to compute the new $\tilde{V}$. We will replace the first occurrence of UV^T in the right-hand side of the previous equation with the current estimates, leading to a filled in $X^* = P_\Omega\left(X\right) + P_\Omega^\perp\left(UV^T\right)$, and then solve for $\tilde{V}$ in

$$\underset{\tilde{B}}{\text{minimize}} \, \|X^* - U\tilde{V}\|_F^2 + \lambda\|\tilde{V}\|_F^2.$$

So, as we can see this methods try to maintain the efficient *sparse + low-rank* representation for high-dimensional problems.

4.3 Riemannian Optimization

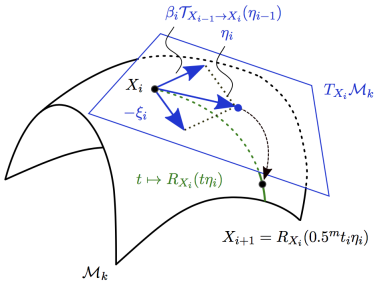
Let us note that constraints in the problem form a smooth manifold $\mathcal{M}_k$ of matrices of rank k: $\mathcal{M}_k := \{Z \in \mathbb{R}^{m \times n} : \text{rank}(Z) = k\}$.

This method use a generalization of classical non-linear conjugate gradients (CG) on Euclidean space to perform optimization on manifolds.

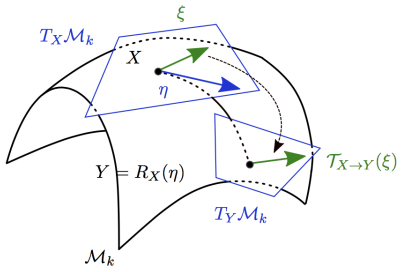
As a tangent vector only gives a direction but not the line search itself on the manifold, a smooth mapping called the retraction, is needed to map tangent vectors to the manifold.

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

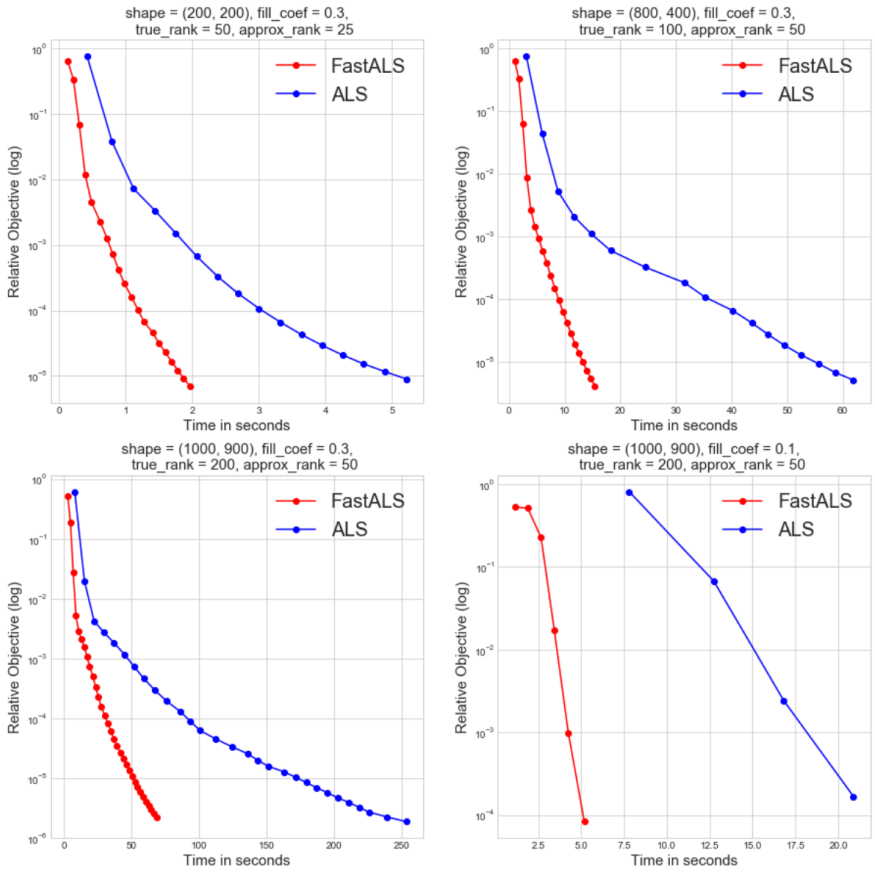


5 Experiments

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

5.1 1-st experiment

In the first experiment we tried to figure out dependence of convergence of the size of the problem and filling coefficient. For this task we randomly generate 4 different matrices. The first pair of two random matrices have different sizes, but number of nonzero elements is equal for both of them. The second pair of two random matrices have equal sizes, but different number of nonzero elements. So, our results are on the following plot:



Comments: As we can see FastALS "faster" stabilize changing in the objective. In averaging, as we can see, ALS requires two times bigger time to achieve the same accuracy in changing of the objective function as FastALS.

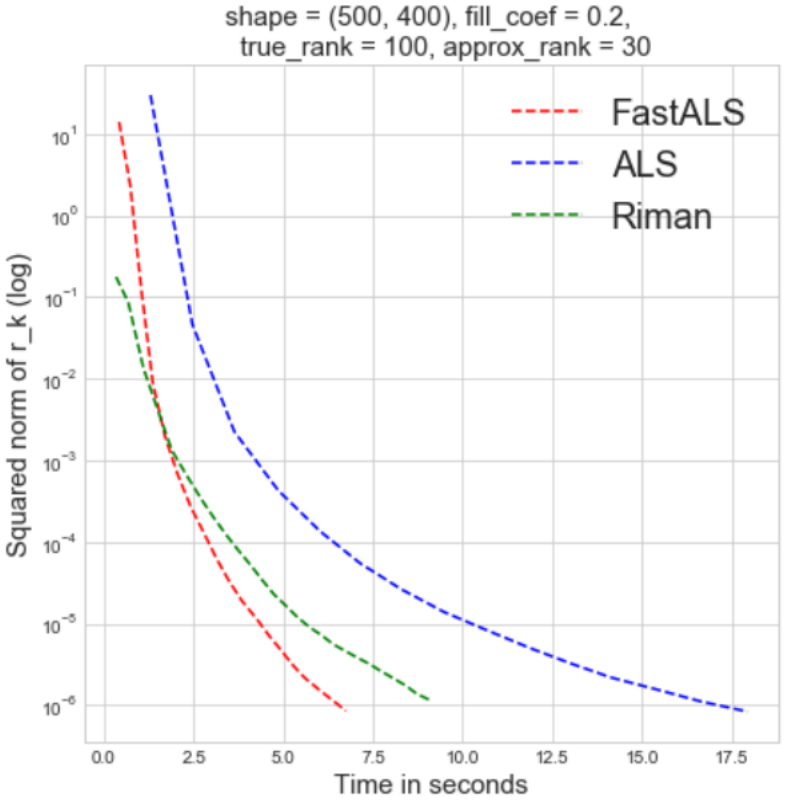
If we increasy the size of the matrix, staying the number of nonzero elements constant, we indeed will see, that both algorithms will require more computational time.

If we consider the problem of the same sizes and consider different amount of nonzero elements, we can see, that if we decrease the number of nonzero elements, than training time will also decrease. This happens because of efficient sparse calculations. As FastALS use sparse + non-sparse structure in the very efficient way.

Now, let's figure out, why computational time for ALS is much bigger that one for FastALS algorithm. It is clear from the description of the algorithm that ALS solves different regression problems for every row/column, because of their different amount of missingness. This can be costly. FastALS solves a single regression problem once and simultaneously for all rows/columns, because it operates on a filled-in matrix which is complete.

5.2 2-nd experiment

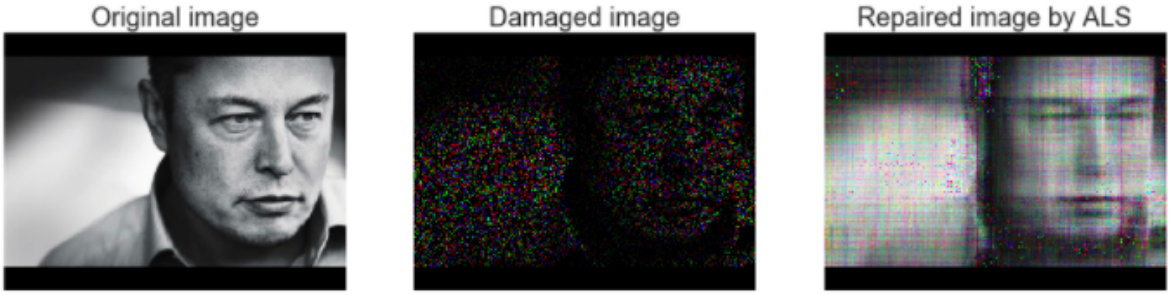
In this experiment we analyzed the rate of convergence for different implemented algorithms on random problem. We have obtained the following results:



Comments: As we can see the rate of convergence rate of residuals $\|r_k\|^2 \to 0$ for ALS is the slowest. Now consider other two algorithms. Riemannian Optimization faster than all reaches accuracy in 10^{-2}, but later the rate of convergence decreases. And after some iterations FastALS becomes faster than Riemannian Optimization. So, let's try to analyze why it is so. On each iteration Riemannian Optimization method do a lot of expensive operations, like producing SVD of really big sparse matrices, projection on Manifold and so on. This powerful methods give the Riemannian Optimization method the opportunity to reach a good accuracy, using only several steps. But furthermore, on each iteration it does line search, and as a every gradient method for it may be difficult to find an appropriate step for next iteration, when we are near local optimum. We started to do very small steps, that's why the rate of convergence decreased. So, we may try to consider some modifications of this method, to improve his rate of convergence near local optimum.

5.3 3-rd experiment

In the third experiment we applied ALS algorithm for repairing Elon Musk photo. We ran it for two different images: the first one contained 10% of data, the second one contained 20% of data. In the first case it is difficult to recognize the person in the repaired picture, however in the second one it is easy to find Elon Musk himself.



In the send case we have 20% of data, thus the reconstruction is more close to the original picture.

5.4 4-th experiment

In the fourth experiment we show that FastALS and Riemannian Optimization are not the best choice for recommendation systems. To prove it we compute RMSE metric and two metrics which are commonly used in recommender systems HR-10(heat rate 10) and HR-20(heat rate 20). Here we give formula for HR-10:

$$\frac{1}{\#users}\sum_{\text{users}}\mathcal{I}\{recommendation \in top_{10}(user\ preferences)\},$$

HR-20 is defined in the same way.
To run this experiment we did the following steps:

1) preprocessed data, i.e. assigned 1 to all high rated films and 0 to others. And after that removed all rows where were no units.

2) we splited our data on train and test in proportion 80/20 and chose one element with the highest rate, saved his index to the hold out set and removed information about him from initial matrix.

3) After that we apply implemented matrix completion algorithms to recover matrix. To evaluate quality of prediction we compute metric on the hold-out set.

4) We tuned hyper parameters using self implemented GridSearchCV.

5) Evaluated the quality of algorithm with best parameters using self implemented 5-fold cross validation.

The results can be seen in the following table:

	ALS			RIEMANN OPTIMIZATION		
	HR-10	**HR-20**	**RMSE**	**HR-10**	**HR-20**	**RMSE**
Mean	0.175	0.27	0.97	0.120	0.17	0.98
STD	0.03	0.04	0.05	0.034	0.05	0.06

One can see that we have obtained quite good score using RMSE metric. However, the HR score, obtained by implemented algorithms, is quite poor. This happens because RMSE does not take into account the property of element to be in top, i.e. it can predict data very close to the real rate, but the order of predicted ratings will be dramatically incorrect. Thus RMSE is not a good choice for evaluating the quality of prediction in recommender systems . There are a lot of "recommender" metrics like HR, MRR, NDCG which are more suitable for this task. So, one should separate problems of matrix completion and recommender systems when filling in missed values in matrices. There are a lot of state-of-the-art algorithms for matrix completion problem, that can obtained a good quality, measured by RMSE metrics. But in general such algorithms perform poor quality in recommendation problems. So, ALS and Riemannian Optimization is a good choice for matrix completion problem, but one should try to look for better algorithms if he solves recommendation problem.

6 Conclusion

In conclusion of our work, we recall the results that we obtained. First, the ratio of missed values influences speed of algorithms because matrices become more sparse. From the second experiment we understood that ALS is the slowest algorithm among considered ones, however it was shown in the 3-rd experiment that ALS can repair missed values quite well. The 4-th experiment shows that FastALS and Riemannian Optimization are not adequate for Recommendation problems, as RMSE is not adequate for evaluation of algorithms applied to Recommendation systems.

It is worth to mention that this task is the first step to implementation Matrix Completion Library, which will afford generation of problems, prediction of missed values, evaluation of different metrics, exhaustive grid search among the parameters and cross validation.

7 Acknowledgements

We immensely appreciate help of Evgeny Frolov at all steps of the working on this project, especially for sharing useful articles and application cases of our algorithms in the context of Recommender Systems. We immensely appreciate help of TA Ivan Nazarov in early feedback for our project and useful advice about realization and preparing experiments.

References

[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`.

[ECaTT09] E.J Cande's and T. Tao, *The power of convex relaxation: Near-optimal matrix completion,* IEEE Trans. Inform. Theory **56** (2009), 2053-2080.

[TH14] R. Mazumder T. Hastie J. D. Lee, *Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares* (2014), available at `https://arxiv.org/pdf/1410.2596.pdf`.