Collaborative Filtering using the Augmented Lagrangian Method for matrix completion

Sven Hammann, Stefan Dietiker, Jannick Griner
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—We evaluate the performance of a collaborative filtering algorithm called MC-ALM that uses the Augmented Lagrangian Method (ALM) to minimize the nuclear norm of the reconstructed matrix. We present a slightly simpler version of this algorithm and show that it performs better than the baseline SVD and RPCA algorithms in terms of mean squared error, but has a longer runtime than SVD. We present our own model for generating artificial data that can be used to test collaborative filtering algorithms.**

I. Introduction

Almost all online shops nowadays employ some form of a recommender system to predict consumer preferences based on collected data on consumer behavior and product ratings. The enormous amount of data require high-performance algorithms capable of dealing with high-dimensional data. The problem of predicting the unknown ratings is called collaborative filtering, and the search for better algorithms in this field is a prominent problem, as can be seen in the Netflix prize challenge (netflixprize.com), where a million dollars were offered for the best such algorithm.

In our model, we consider an $m \times n$ matrix consisting of the ratings of m users for n items, where many values are missing. The goal of collaborative filtering is to predict the missing values as accurately as possible. When we want to measure the performance of a collaborative filtering algorithm, we remove some entries from a given, full data matrix D, then have the algorithm reconstruct the missing entries as accurately as possible. We finally compare the reconstructed matrix A with the original matrix D.

The underlying assumption of most collaborative filtering algorithms, including MC-ALM that is used in this paper, is that the original matrix D has low rank, or at least approximately low rank, i.e., many dimensions have small singular values. The reason for this lies in the assumed correlations between ratings: If two or more users have similar ratings on some items, it is likely that there is some underlying smaller dimension responsible for these ratings. These dimensions can be interpreted as topics of interest.

A natural procedure is now to find a solution A that respects the known entries of D, i.e., the reconstruction is equal to the original on the entries that are known, and that has minimal rank. A good approximation for minimization of the rank is minimizing the nuclear norm, i.e., the sum of singular values. This can be solved by convex optimization. The algorithm we use was presented in [1] and uses the Augmented Lagrangian Method (ALM). We call it MC-ALM short for Augmented Lagrangian Method for matrix completion.

An additional consideration is that of corrupted entries in the original data matrix. These values are not missing, but they may not reflect a true rating of a user. An important real-world example are shilling attacks, where manufacturers give good ratings to their own products and bad ratings to the products of competitors.

We compare the algorithm using ALM from [1] to an algorithm using RPCA (as in [2]) and to another baseline algorithm using one Singular Value Decomposition. We present our own model to generate artificial test data that simulates users rating hotels that belong to different groups, and attackers that may want to benefit one group and hurt the others. We test our algorithms on our own artificial data as well as on a different data set.

II. Models and Methods

A. Related Work

The algorithm we use is based on Algorithm 6 in [1]. We use a simplified version of it that does not include the update step for μ and uses the economy-size singular value decomposition directly provided by MATLAB rather than an external library.

We compare these results to those of another algorithm based on [2], Section 1.6, that explains how Robust PCA can be used for matrix completion.

B. Objective

We formulate our problem as in Section 3.2 of [1]. Given an original matrix D and a number of observed entries Ω, we wish to reconstruct the complete matrix D. Under certain conditions on the rank of D and the number of samples, D can be recovered exactly [3]. In the case of collaborative filtering, we may be given a matrix

D that is only approximately low-rank, i.e., it has many dimensions with small singular values. In this case, a good approximation is still possible.

Let A be our matrix with the objective to reconstruct the original D. We cannot use convex optimization to minimize $rank(A)$ directly, and we might not want to do this if A has many small, but nonzero singular values. We instead minimize the nuclear norm of A, denoted by $||A||_*$, which is the sum of singular values of A. This provides a good approximation for minimizing $rank(A)$. This is proven with respect to Robust PCA in [2] and adapted for our setting in [1].

We want so solve the following problem:

$$\underset{A}{\text{minimize}} \ ||A||_* \text{ subject to}$$
$$A_{ij} = D_{ij} \ \forall (i,j) \in \Omega$$

Let D_Ω be the matrix such that $(D_\Omega)_{ij} = D_{ij} \ \forall (i,j) \in \Omega$, and $(D_\Omega)_{ij} = 0 \ \forall (i,j) \notin \Omega$. That is, the missing values are set to zero. We can then rewrite the constraint by introducing an auxiliary variable matrix E:

$$\underset{A}{\text{minimize}} \ ||A||_* \text{ subject to}$$
$$A + E = D_\Omega \quad E_{ij} = 0 \ \forall (i,j) \in \Omega$$

Since E may be nonzero only on the coordinates of missing values, its only purpose is to compensate for the values set in A on these coordinates. As D_Ω is zero there, if $A_{ij} = x$, we must have $E_{ij} = -x$ to fulfill the constraint.

C. Algorithm

We use a simplified version of Algorithm 6 in [1], ALM for matrix completion. We call this algorithm MC-ALM. We introduce the soft-thresholding operator used in [1]:

$$\mathcal{S}_\epsilon[x] := sign(x) \cdot max(abs(x) - \epsilon, 0)$$

It lets x go by ϵ closer to zero. The operator can be used on a matrix by applying it element-wise.

We use $E_{\bar{\Omega}}$ to denote that we only update E_{ij} for $(i,j) \notin \Omega$, i.e., only the values that are missing from D.

We refer to Algorithm 6 of [1] for the detailed workings on the algorithm, in particular for the augmented Lagrange function that is minimized by applying soft-thresholding to the singular values of A. The main difference of our version to the one in [1] is that we do not update μ; in our testings, we found this to be unnecessary for good results, and chose the simpler version to work with.

Input: Observation samples D_{ij}, $(i,j) \in \Omega$, of a matrix $D \in \mathbb{R}^{m \times n}$
$Y = 0$; $E = 0$; $\mu > 0$; $iter = 0$

while $\frac{||D_\Omega - A - E||_F}{||D_\Omega||_F} > \epsilon_1$ *AND* $iter < maxIter$ **do**
  $(U, S, V) = svd(D_\Omega - E + \mu^{-1}Y)$;
  $A = U\mathcal{S}_{\mu^{-1}}[S]V^T$;
  $E_{\bar{\Omega}} = (-A + \mu^{-1}Y)_{\bar{\Omega}}$;
  $Y = Y + \mu(D_\Omega - A - E)$;
  $iter = iter + 1$;
end
Output: A

D. Implementation details

We use MATLAB for the implementation of our algorithm, and do not use any extra software packages. For svd, we use the economy-size singular value decomposition provided by MATLAB.

We set $\mu = ||D_\Omega||_2^{-1}$, which is used in [1] as starting value μ_0 that later gets updated.

Our convergence criterion ϵ_1 is set to 10^{-7} and the maximum number of iterations $maxIter$ is set to 150. We found these values to give a good trade-off between runtime and accuracy, but encourage to experiment with these values: In some applications, a slightly higher error might be acceptable, so increasing ϵ_1 or decreasing $maxIter$ to achieve better runtime may be advisable.

E. Data generation model

We created a model to generate artificial data simulating users that rate hotels on a scale from 0 to 100. The model comes up with data in several steps. First, users and hotels are randomly divided into groups. In the next step, a coupling matrix, representing the similarity between the hotel groups is introduced. Based on this group similarity matrix and some randomness, a matrix representing the rating of hotel groups by user groups is created. Now we have a representation of the assumed underlying model of the rating process. User-hotel ratings can now be generated based on their respective groups. Again some extra randomness is added to represent the users individuality. Additionally the found rating is biased towards an underlying global score of the hotel, representing its "true" rating.
Attacks are added after the proper rating matrix has been created. A certain number of randomly selected hotels "attack" the system. The attackers are split into groups. Attack rows are added with the following ratings $R(h)$ for hotels:

$$R(h) = \begin{cases} \mathcal{N}(98,2) & \text{if } h \in \text{Attacker's group} \\ \mathcal{N}(2,2) & \text{if } h \notin \text{Attacker's group} \end{cases}$$

The attacker rows are added to the end of the data matrix.

Method	Error	Runtime
MC-ALM	4.2044	11.49594s
RPCA	4.5399	28.78280s
SVD	4.3549	1.069168s

Table I
MEASUREMENTS FOR THE FULL FIXED DATA SET. ERROR PRESENTS THE MEAN SQUARE ERROR BETWEEN THE ORIGINAL MATRIX AND THE ONE RECOVERED FROM 50% OF THE DATA. RUNTIME WAS MEASURED ON A STANDARD LAPTOP.

III. RESULTS

We tested our MC-ALM method against two other methods for matrix completion. The first one is based on the RPCA approach introduced in [2]. The minimization problem introduced in RPCA is solved using the alternating direction method of multipliers (ADMM - [4]), a variation of ALM. This approach was shown to work well for RPCA in [1]. RPCA is modified as proposed in section 1.6 of [2] to handle matrix completion.
The second baseline method we used is based on simple single value decomposition. The missing values are first replaced by the average rating of the corresponding user and the resulting matrix is then decomposed. The matrix is then recovered using the six largest singular values. This number was found to be optimal for our test data.

To compare the runtime and the error of the different algorithms in I, we used a fixed, real data set, containing 569'026 ratings from 7834 users for 100 items. This data set was also used to assess the efficiency of the different algorithms, by running them on a percentage of all matrix rows (figure 1).

Based on the user-hotel model introduced in II-E we constructed a series of extra data sets. A first matrix, consisting of 999 users and 42 hotels was created, based on 30 user groups and 5 hotel groups. By removing different numbers of values the influence of sparsity was evaluated. Results are shown in figure 2. A similar matrix (999 users, 81 hotels in 8 groups with 50% sparsity) was used to evaluate the influence of the number of user groups which in turn is connected to the rank of the matrix. Results are presented in figure 3.

IV. DISCUSSION

A. Runtime

The runtime of all three methods grows linearly in the number of rows in the matrix, as seen in Figure 1. This seems somewhat surprising, as for all methods, single value decomposition is being used the runtime of which, in general, is a function in $\mathcal{O}(m^2)$, where m is the number of rows. However, the transition to a linear behavior is achieved through the use of an optimized version of svd provided by MATLAB R2013b using economy-size decomposition to only retrieve the singular values needed. The different slopes

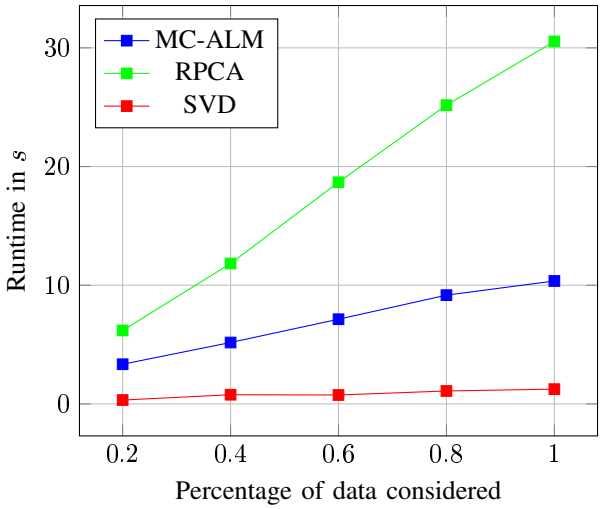

Figure 1. Comparison of the runtime of different methods based on the fixed data set.

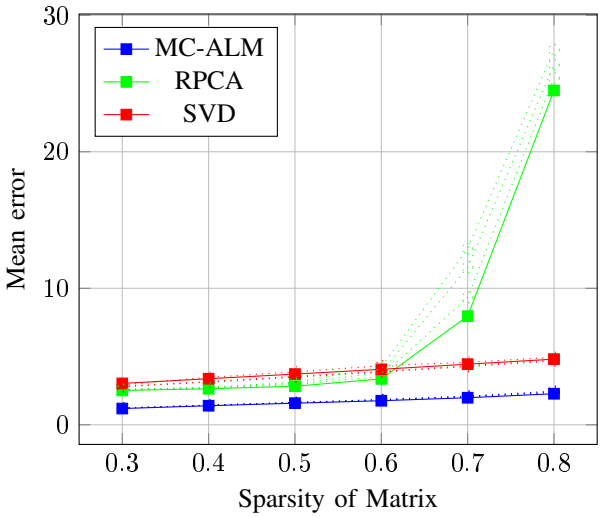

Figure 2. Mean square error for completing the matrix from a different amounts of missing values. Higher sparsity indicates more missing values. The dotted lines represent the results under the presence of attackers. The data was generated by our user-hotel model.

of growth are due to the iterations needed in MC-ALM and RPCA and the more complex update steps in RPCA.

B. Mean squared error

Our MC-ALM method outperforms SVD in terms of mean squared error on both the fixed data set, as seen in table I, and the data sets generated by our user-hotel model, as seen in figures 2 and 3. The implemented method for RPCA performs worst on the fixed data set. If the increased runtime can be tolerated, the lower error of MC-ALM makes it preferable to SVD.

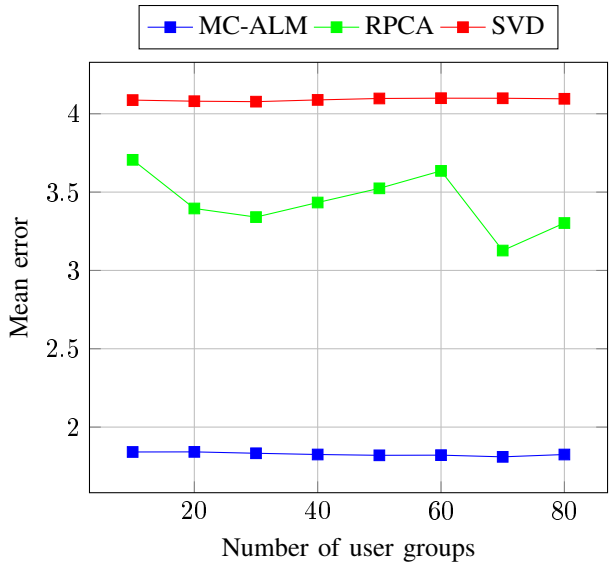

Figure 3. Mean square error for data sets from our user-hotel model with a different amount of user groups.

Figure 2 clearly demonstrates the deficiencies of RPCA for matrices with high sparsity. This coincides with the conditions given in [2]: If the input matrix is both low-rank and sparse, it becomes more difficult to decompose it into a low-rank and a sparse component.

We also extended our user-hotel model to include attackers. Unfortunately, no significant impact on the mean squared error could be measured. We conclude that either all methods already offer a good resistance to shilling attacks, or, more likely, that our model does not provide realistic attacks. In any case, further research in this direction is needed.

V. Summary

We presented and evaluated a simplified version of the MC-ALM algorithm from [1]. We showed that it performs better than the baseline algorithms RPCA and SVD in terms of mean squared error at the cost of increased runtime compared to SVD. We presented a model to generate data including shilling attacks. We used this data to measure performance on matrix with different sparsity and showed that the baseline RPCA algorithm breaks down when the matrix is too sparse, but MC-ALM does not.

We also measured how well the different algorithms perform on data from our model containing shilling attacks, but did not find a significant increase in mean square error on any of the algorithms. We conclude that our model needs improvement in order to provide a realistic simulation of shilling attacks. The question whether MC-ALM is a useful algorithm for collaborative filtering even under the presence of shilling attacks could not be answered conclusively. We believe that this is an interesting area for further research and hope that our model can provide a good starting point.

References

[1] Z. Lin, M. Chen, and Y. Ma, "The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices," *ArXiv e-prints*, Sep. 2010.

[2] Y. M. Emmanuel J. Candès, Xiaodong Li and J. Wright, "Robust principal component analysis?" *Journal of ACM*, vol. 58, no. 1, pp. 1–37, 2011.

[3] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1007/s10208-009-9045-5

[4] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," 2009.