Inequality-Constrained Matrix Completion: Adding the Obvious Helps!

Martin Takáč, Jakub Mareček, and Peter Richtárik [*]

August 12, 2014

Abstract

We propose imposing box constraints on the individual elements of the unknown matrix in the matrix completion problem and present a number of natural applications, ranging from collaborative filtering under interval uncertainty to computer vision. Moreover, we design an alternating direction parallel coordinate descent method (MACO) for a smooth unconstrained optimization reformulation of the problem. In large scale numerical experiments in collaborative filtering under uncertainty, our method obtains solution with considerably smaller errors compared to classical matrix completion with equalities. We show that, surprisingly, seemingly obvious and trivial inequality constraints, when added to the formulation, can have a large impact. This is demonstrated on a number of machine learning problems.

1 Motivation

Matrix completion is a well-known problem, with applications ranging from image processing to recommender systems. When dimensions of a matrix X and some of its elements $X_{i,j}, (i,j) \in \mathcal{I}$ are known, the goal is to find the unknown elements. Without imposing any further requirements on X, there are infinitely many solutions. In many applications, however, the matrix completion that minimizes the rank:

$$\begin{array}{rl} \min & \operatorname{rank}(Y), \\ \text{subject to} & Y_{i,j} = X_{i,j}, (i,j) \in \mathcal{I}, \end{array} \tag{1}$$

works the best. In this paper, we present a variant of the problem, where there are inequalities, instead of equalities. This variant has a number of important applications:

*Martin Takáč is at Lehigh University, Jakub Mareček is at IBM Research, and Peter Richtárik is at the University of Edinburgh. Their addresses aretakac.mt@gmail.com, jakub@marecek.cz, and peter.richtarik@ed.ac.uk, respectively.

Collaborative Filtering under Uncertainty. Collaborative filtering is a well-established application of matrix completion problems Srebro (2004), largely thanks to the success of the Netflix Prize (`netflixprize.com`). Let us have a matrix, where each row corresponds to one user and each column corresponds to a product or service. There are only a small number of entries known, considering that every user rates only a modest number of products or services. Further, notice that one user may provide two different ratings for one and the same product at two different times, depending on the current mood and other circumstances at the two times. One may hence want to consider an interval $[\underline{x}, \overline{x}]$ instead of a fixed value x, e.g., $[x-\epsilon, x+\epsilon]$ or rather $[\max\{L, x-\epsilon\}, \min\{x+\epsilon, U\}]$, when x is known to be a rating on the scale of $[L, U]$. One may hence want to solve:

$$\begin{aligned} &\min_Y \max_{X_{i,j} \in [\underline{X_{i,j}}, \overline{X_{i,j}}] \forall (i,j) \in \mathcal{I}} \operatorname{rank}(Y), \\ &\text{subject to} \qquad Y_{i,j} = X_{i,j}, (i,j) \in \mathcal{I}. \end{aligned} \tag{2}$$

Notice that this generalizes the robust linear programming of Soyster (1973) to rank minimization.

Low-Rank Approximations in Image Processing. Another use of matrix completion can be found in image processing. In inpainting problems, a subset of pixels from an image are given and the task is to fill in the missing pixels. Matrix completion with equalities (1), where $\mathcal{I}$ is the index set of all known pixels, has been used numerous times in this setting. If the original matrix comes from the real life, it probably will be full rank, albeit with quickly decreasing singular values in the spectrum. In this case, instead of solving the equality-constrained problem (1), one should like to find a low-rank approximation Y^* of X, such that the known entry of X is not far away from Y^*, i.e., $\forall (i,j) \in \mathcal{I}$ we have $Y_{i,j} \approx X_{i,j}$. Let us illustrate this with a small matrix

$$X = \begin{pmatrix} 68.16 & 78.12 & 24.04 \\ 78.12 & 90.09 & 30.03 \\ 24.04 & 30.03 & 20.01 \end{pmatrix},$$

which has rank 3 and its singular values $\Sigma = (167.9945, 10.2553, 0.0102)^T$. It is easy to verify that

$$Y_2^* = \begin{pmatrix} 68.1546 & 78.1250 & 24.0389 \\ 78.1250 & 90.0853 & 30.0310 \\ 24.0389 & 30.0310 & 20.0098 \end{pmatrix}$$

is the best rank 2 approximation of X in Frobenius norm. Observe that no single element of Y_2^* is identical to X, but that $Y_2^* \approx X$. It is an easy exercise to show that for any $X \in \mathbb{R}^{m \times n}$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}}$, and Y_r^* as its best rank-r approximation, we have $|X_{i,j} - (Y_r^*)_{i,j}| \leq \sum_{i=r+1}^{\min\{m,n\}} \sigma_i =: \mathcal{R}(r)$ for all (i,j). Therefore, we should not require equality constrains in (1), but rather inequalities $|Y_{i,j} - X_{i,j}| \leq \mathcal{R}(r), \forall (i,j) \in \mathcal{I}$. One should like to stress that this approach is not the same as minimizing $\sum_{(i,j)\in\mathcal{I}} (X_{i,j} - Y_{i,j})^2$ over all rank r matrices, because we do not penalize the elements of Y, which are already close to X. It is also different from the usual treatment of noise in the observations Candès & Plan (2010). One could rather formulate this as the minimization of $\sum_{(i,j)\in\mathcal{I}} \max\{0, |X_{i,j} - Y_{i,j}| - \mathcal{R}(r)\}^2$ over all rank r matrices.

Inpainting with Side Information. Let us present another image processing application. If our original matrix is a gray scale image, then one knows more about the missing pixels than just that they are missing! In particular, one knows that those missing pixels take values from the interval $[0, 1]$. This can hence be seen as “side information” which, as we will show in numerical section, improves recovery of a low-rank approximation considerably. One can extend this approach further, e.g. if the pixel is missing within a light region of the image, one can assume that the intensity should be at least 0.8.

Forecasting with Side Information. A related application comes from the forecasting of seasonal data, e.g. sales. Let us assume that in process $\{X_t\}$, one knows $k + 1 = \tau$ such that $F_X(x_{t_1+\tau}, \ldots, x_{t_k+\tau}) = F_X(x_{t_1}, \ldots, x_{t_k})$ for the cumulative distribution function $F_X(x_{t_1+\tau}, \ldots, x_{t_k+\tau})$ of the joint distribution of $\{X_t\}$ at times $t_1 + \tau, \ldots, t_k + \tau$. One can then formulate the forecasting into the future as a matrix completion problem, where there the historical datum at time t is at row $\lfloor t/\tau \rfloor$, column $t \mod k$ specified by an equality or a pair of inequalities, and where inequalities represent side information. For an example of such side information in sales forecasts, notice that one often has bookings for many months in advance and knows that the sales for the respective months will not be less than the bookings taken.

A number of other applications, e.g., in the recovery of structured matrices Chen & Chi (2013) and in sparse principal component analysis with priors on the principal components, can be envisioned.

2 The Problem

In this section we introduce our notation and formalize the problem. Let X be an $m \times n$ matrix to be reconstructed. Assume that elements $(i,j) \in \mathcal{E}$ of X we wish to fix, for elements $(i,j) \in \mathcal{L}$ we have lower bounds and for elements $(i,j) \in \mathcal{U}$ we have upper bounds. We propose the following natural formulation for the equality and inequality constrained matrix completion problem:

$$\begin{array}{lll} \min\limits_{X\in\mathbb{R}^{m\times n}} & \operatorname{rank}(X) & \\ \text{subject to} & X_{ij} = X^{\mathcal{E}}_{ij}, & (i,j)\in\mathcal{E}, \\ & X_{ij} \geq X^{\mathcal{L}}_{ij}, & (i,j)\in\mathcal{L}, \\ & X_{ij} \leq X^{\mathcal{U}}_{ij}, & (i,j)\in\mathcal{U}. \end{array} \tag{3}$$

This problem is NP-hard, even with $\mathcal{U} = \mathcal{L} = \emptyset$ Natarajan (1995). This special case of (3) has been widely studied, e.g., in Recht et al. (2011); Goldfarb et al. (2009); Ma et al. (2011).

A popular heuristic enforces low rank in a synthetic way by writing X as a product of two matrices, $X = LR$, where $L \in \mathbb{R}^{m\times r}$ and $R \in \mathbb{R}^{r\times n}$. Hence, X is of rank at most r. This has been proposed and analyzed by Lee et al. (2010); Recht et al. (2010); Srebro et al. (2004); Tanner & Wei (2013). Let $L_{i:}$ and $R_{:j}$ be the i-th row and j-h column of L and R, respectively. Instead of (3), we consider the problem

$$\min\{f(L,R) \;:\; L\in\mathbb{R}^{m\times r},\; R\in\mathbb{R}^{r\times n}\}, \tag{4}$$

where

$$\begin{aligned} f(L,R) := {} & \tfrac{\mu}{2}\|L\|_F^2 + \tfrac{\mu}{2}\|R\|_F^2 \\ & + f_{\mathcal{E}}(L,R) + f_{\mathcal{L}}(L,R) + f_{\mathcal{U}}(L,R), \end{aligned}$$

and

$$\begin{aligned} f_{\mathcal{E}}(L,R) &:= \tfrac{1}{2}\textstyle\sum_{(ij)\in\mathcal{E}}(L_{i:}R_{:j} - X^{\mathcal{E}}_{ij})^2, \\ f_{\mathcal{L}}(L,R) &:= \tfrac{1}{2}\textstyle\sum_{(ij)\in\mathcal{L}}(X^{\mathcal{L}}_{ij} - L_{i:}R_{:j})_+^2, \\ f_{\mathcal{U}}(L,R) &:= \tfrac{1}{2}\textstyle\sum_{(ij)\in\mathcal{U}}(L_{i:}R_{:j} - X^{\mathcal{U}}_{ij})_+^2, \end{aligned}$$

and $\xi_+ = \max\{0,\xi\}$.

Parameter μ helps to prevent scaling issues[1]. Hence, we could optionally set μ to zero and then from time to time rescale matrices L and R, so that their product stays constant Tanner & Wei (2013). The term $f_{\mathcal{E}}$ (resp. $f_{\mathcal{U}}$, $f_{\mathcal{L}}$) encourages the equality (resp. inequality) constraints to hold.

3 The Algorithm

Coordinate descent algorithms (CDA) are effective in solving large-scale problems, due to their low per-iteration computational cost. Although each iteration of CDA is cheap, many more iterations are required for convergence, compared to second-order algorithms or similar. The stochastic CDA has received much

[1]Let $X = LR$, then also $X = (cL)(\frac{1}{c}R)$ as well, but we see that for $c \to 0$ or $c \to \infty$ we have $\|L\|_F^2 + \|R\|_F^2 \ll \|cL\|_F^2 + \|\frac{1}{c}R\|_F^2$.

attention, recently, because it has numerous benefits, compared to the deterministic version Nesterov (2012); Tseng (2001). Notably, it has been shown that stochastic CDA can be efficiently parallelized and one can obtain almost linear speed-up Bradley et al. (2011); Richtárik & Takáč (2012); Recht et al. (2011), e.g., in regimes when the number of parallel updates τ is much smaller that the dimension of the optimization problem.

We now present our alternating parallel coordinate descent method for MAtrix COmpletion ("MACO") in Algorithm 1.

Algorithm 1 Matrix Completion via Alternating Parallel Coordinate Descent

```
input $\mathcal{E}, \mathcal{L}, \mathcal{U}, X^{\mathcal{E}}, X^{\mathcal{L}}, X^{\mathcal{U}}$, rank $r$
 1: choose $L \in \mathbb{R}^{m \times r}$ and $R \in \mathbb{R}^{r \times n}$
 2: for $k = 0, 1, 2, \dots$ do
 3:   choose random subset $\hat{\mathcal{S}} \in \{1, \dots, m\}$
 4:   for $i \in \hat{\mathcal{S}}$ in parallel do
 5:     choose $\hat{r} \in \{1, \dots, r\}$ uniformly at random
 6:     compute $\delta_{i\hat{r}}$ using formula (5)
 7:     update $L_{i\hat{r}} \leftarrow L_{i\hat{r}} + \delta_{i\hat{r}}$
 8:   end for
 9:   choose random subset $\hat{\mathcal{S}} \in \{1, \dots, n\}$
10:   for $j \in \hat{\mathcal{S}}$ in parallel do
11:     choose $\hat{r} \in \{1, \dots, r\}$ uniformly at random
12:     compute $\delta_{\hat{r}j}$ using (7)
13:     update $R_{\hat{r}j} \leftarrow R_{\hat{r}j} + \delta_{\hat{r}j}$
14:   end for
15: end for
```

In Steps 3–8 of our algorithm, we fix R, choose random $\hat{r}$ and a random set $\hat{S}$ of rows of L, and update, in parallel for $i \in \hat{S}$: $L_{i\hat{r}} \leftarrow L_{i\hat{r}} + \delta_{i\hat{r}}$. In Steps 9–14, we fix L, choose random $\hat{r}$ and a random set $\hat{S}$ of columns of R, and update, in parallel for $j \in \hat{S}$: $R_{\hat{r}j} \leftarrow R_{\hat{r}j} + \delta_{\hat{r}j}$.

Let us now comment on the computation of the updates, $\delta_{i\hat{r}}$ and $\delta_{\hat{r}j}$. First, note that while f is not convex jointly in (L, R), it is convex in L for fixed R and in L for fixed R.

If we now fix $i \in \{1, 2, \dots, m\}$ and $\hat{r} \in \{1, 2, \dots, r\}$, and view f as a function of $L_{i\hat{r}}$ only, it has a Lipschitz gradient with constant

$$W^{\mathcal{L}}_{i\hat{r}} = \mu + \sum_{v\,:\,(iv)\in\mathcal{E}} R^2_{\hat{r}v} + \sum_{v\,:\,(iv)\in\mathcal{L}\cup\mathcal{U}} R^2_{\hat{r}v}.$$

That is, for all L, R and $\delta \in \mathbb{R}$, we have

$$f(L + \delta E_{i\hat{r}}, R) \leq f(L, R) + \langle \nabla_L f(L, R), E_{i\hat{r}} \rangle \delta + \tfrac{W^{\mathcal{L}}_{i\hat{r}}}{2} \delta^2,$$

where E is the $n \times r$ matrix with 1 in the $(i\hat{r})$ entry and zeros elsewhere. Likewise, if we now fix $j \in \{1, 2, \dots, n\}$ and $\hat{r} \in \{1, 2, \dots, r\}$, and view f as a function of

$R_{\hat{r}j}$ only, it has a Lipschitz gradient with constant

$$V^{\mathcal{U}}_{\hat{r}j} = \mu + \sum_{v\,:\,(vj)\in\mathcal{E}} L^2_{v\hat{r}} + \sum_{v\,:\,(vj)\in\mathcal{U}\cup\mathcal{L}} L^2_{v\hat{r}}.$$

That is, for all L, R and $\delta \in \mathbb{R}$, we have

$$f(L, R + \delta E_{\hat{r}j}) \leq f(L, R) + \langle \nabla_R f(L, R), E_{\hat{r}j}\rangle \delta + \tfrac{V^{\mathcal{U}}_{\hat{r}j}}{2}\delta^2,$$

where E is the $r \times m$ matrix with 1 in the $(\hat{r}j)$ entry and zeros elsewhere.

The minimizer of the right hand side of the bound on $f(L+\delta E_{i\hat{r}}, R)$ is given by

$$\delta_{i\hat{r}} := -\tfrac{1}{W^{\mathcal{L}}_{i\hat{r}}}\langle \nabla_L f(L, R), E_{i\hat{r}}\rangle, \tag{5}$$

where $\langle \nabla_L f(L, R), E_{i\hat{r}}\rangle$ equals

$$\begin{aligned} &\mu L_{i\hat{r}} + \textstyle\sum_{v\,:\,(iv)\in\mathcal{E}} (L_{i:}R_{:v} - X^{\mathcal{E}}_{iv})R_{\hat{r}v} \\ &+ \textstyle\sum_{v\,:\,(iv)\in\mathcal{U}\ \&\ L_{i:}R_{:v}<X^{\mathcal{U}}_{iv}} (L_{i:}R_{:v} - X^{\mathcal{U}}_{iv})R_{\hat{r}v} \\ &+ \textstyle\sum_{v\,:\,(iv)\in\mathcal{L}\ \&\ L_{i:}R_{:v}>X^{\mathcal{L}}_{iv}} (L_{i:}R_{:v} - X^{\mathcal{L}}_{iv})R_{\hat{r}v}. \end{aligned}$$

Note that

$$f(L + \delta_{i\hat{r}}E_{i\hat{r}}, R) \leq f(L, R) - \tfrac{(\langle \nabla_L f(L,R), E_{i\hat{r}}\rangle)^2}{2W_{i\hat{r}}}. \tag{6}$$

The minimizer of the right hand side of the bound on $f(L, R+\delta E_{\hat{r}j})$ is given by

$$\delta_{\hat{r}j} := -\tfrac{1}{V^{\mathcal{U}}_{\hat{r}j}}\langle \nabla_R f(L, R), E_{\hat{r}j}\rangle, \tag{7}$$

where $\langle \nabla_R f(L, R), E_{\hat{r}j}\rangle$ equals

$$\begin{aligned} &\mu R_{\hat{r}j} + \textstyle\sum_{v\,:\,(vj)\in\mathcal{E}} (L_{v:}R_{:j} - X^{\mathcal{E}}_{vj})L_{v\hat{r}} \\ &+ \textstyle\sum_{v\,:\,(vj)\in\mathcal{L}\ \&\ L_{v:}R_{:j}<X^{\mathcal{L}}_{vj}} (L_{v:}R_{:j} - X^{\mathcal{L}}_{vj})L_{v\hat{r}} \\ &+ \textstyle\sum_{v\,:\,(vj)\in\mathcal{U}\ \&\ L_{v:}R_{:j}>X^{\mathcal{U}}_{vj}} (L_{v:}R_{:j} - X^{\mathcal{U}}_{vj})L_{v\hat{r}}. \end{aligned}$$

Note that

$$f(L, R + \delta_{\hat{r}j}E_{\hat{r}j}) \leq f(L, R) - \tfrac{(\langle \nabla_R f(L,R), E_{\hat{r}j}\rangle)^2}{2V_{\hat{r}j}}. \tag{8}$$

The random set $\hat{\mathcal{S}}$ can be chosen uniformly at random, or can be chosen nonuniform, as is common in importance sampling. In our experiments we have chosen the uniform variant. If we have a multicore machine available with τ cores, then a reasonable subset $\hat{\mathcal{S}}$ should have cardinality τ, or some integral multiple of τ, so that every core has a reasonable (not too small so that it is underutilized, but not too large so that processing takes a long time) load at every iteration.

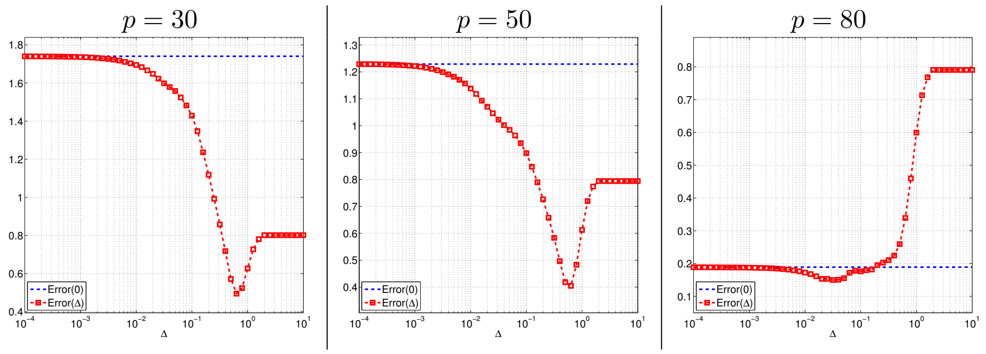


Figure 1: Dependence of Error(Δ) as function of Δ for various $p \in \{30, 50, 80\}$.

Efficient implementation. Formulas (5) and (7) suggest that the computation of the final step requires a lot of computation. This can, however, be avoided if we define matrices $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$ such that $A_{iv} = W_{iv}^{\mathcal{L}}$ and $B_{vj} = V_{vj}^{\mathcal{U}}$. After each update of the solution, we can also update those matrices. Similarly, one can store sparse residuals matrices $\Delta_{\mathcal{E}}$, $\Delta_{\mathcal{L}}$, $\Delta_{\mathcal{U}}$, where

$$(\Delta_{\mathcal{E}})_{i,j} = \begin{cases} L_{i:}R_{:j} - X_{ij}^{\mathcal{E}}, & \text{if } (ij) \in \mathcal{E} \\ 0, & \text{otherwise,} \end{cases}$$

and $\Delta_{\mathcal{U}}$, $\Delta_{\mathcal{L}}$ are defined in similar way. Subsequently, the computation of $\delta_{i\hat{r}}$ or $\delta_{\hat{r}j}$ is reduced to just a few multiplications and additions.

Lock-free implementations. Since each iteration is cheap and does not depend on the size of the problem, one could possibly solve problems of any dimension. Because each thread deals with a different row of L (column of R), there is no risk of race conditions at run-time. Hence no atomic operations are required. At some point, no single computer will have sufficient memory capacity, and hence one will have to distribute the computation across a cluster. Fortunately, techniques developed in Yun et al. (2013) are also applicable to Algorithm 1, and hence this algorithm can be extended to the distributed setting.

Related work. Let us note that Cai et al. (2010) analyzed matrix completion with an arbitrary convex constraint and proposed to solve the problem using Singular Value Thresholding (SVT) algorithm. This, however, requires the computation of a singular value decomposition (SVD) in each iteration. A number of other approaches, e.g., augmented Lagrangian methods Tomioka et al. (2010), could also be extended, but would require a singular value decomposition or a number of iterations of the power method Jaggi & Sulovský (2010); Shalev-shwartz et al. (2011). Even considering the recent progress in randomized methods for approximating singular value decompositions Halko et al. (2011), the approximation becomes very time-consuming very quickly as the dimensions of matrices grow.

Our algorithm can be seen as a coordinate-wise version of the alternating least squares (ALS) algorithm. If $r = 1$ and $\mathcal{U} \equiv \mathcal{L} \equiv \emptyset$ and one always chooses all elements of $\hat{\mathcal{S}}$, then this algorithm is equivalent with classical ALS.

3.1 Convergence Analysis

Due to the non-convex nature of (4), one has to be satisfied with convergence to a stationary point.

Theorem 1. *Let $\mu > 0$ and $(L^{(k)}, R^{(k)})$ be the (random) matrices produced by Algorithm 1 after k iterations. Then Algorithm 1 is monotonic, i.e., for all $k \geq 0$,*

$$0 \leq f(L^{(k+1)}, R^{(k+1)}) \leq f(L^{(k)}, R^{(k)}), \tag{9}$$

Moreover, almost surely,

$$\nabla_L f(L^{(k)}, R^{(k)}) \to 0, \qquad \nabla_R f(L^{(k)}, R^{(k)}) \to 0.$$

Sketch of the proof. Monotonicity can be deduced from (6) and (8). Then assumption that $\mu > 0$ together with monotonicity (9) implies that the levelset $\{(L, R) : f(L, R) \leq f(L^{(0)}, R^{(0)})\}$ is bounded. Hence, the Lipschitz constants W and V are bounded above. The rest follows again from (6) and (8). □

4 Numerical Experiments

In this Section, we present the results of various experiments, including a comparison with classical matrix completion with $\mathcal{U} \equiv \mathcal{L} \equiv \emptyset$. We focus on how much can one benefit from imposing obvious inequalities.

4.1 Dependence of classical MC and the one with inequality with Δ margin

Motivated by the fact that the best r-rank approximation of the original matrix can have each element different from the observed elements, we decided to propose an experiment, where we generate a random matrix $X \in \mathbb{R}^{20\times 20}$ with rank 8. Afterwards, we sample p% of entries of that matrix, which we store in index set $\mathcal{I}$, and solve (4) with just the inequality constrains, i.e., $\mathcal{E} \equiv \emptyset, \mathcal{U} \equiv \mathcal{L} \equiv \mathcal{I}$, $X^{\mathcal{U}} = X - \Delta\mathbf{1}$ and $X^{\mathcal{L}} = X + \Delta\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{m\times n}$ is a matrix with all elements equals to 1. Let us denote by $Y^*(\Delta)$ the solution of that optimization problem after 10^5 serial iterations ($|\hat{\mathcal{S}}| = 1$) and with $\mu = 10^{-5}$. Figure 1 shows the dependence of error defined as follows

$$\text{Error}(\Delta) = \frac{\|Y^*(\Delta) - X(7)\|_F}{\|X(7)\|_F},$$

where $X(r)$ is the best rank r approximation of X obtain using SVD decomposition of the whole matrix. Figure 1 clearly suggest that, e.g., if 50% of elements are observed then by allowing each entry $\in \mathcal{I}$ of reconstructed matrix to lie in

Δ neighborhood of observed values, we can decrease the relative error of reconstruction from approximately 1.22 to 0.4 for $\Delta \approx \mathcal{R}(r)$. In this case, the value of $\|X(7)\|_F$ was 21.3245 and $\mathcal{R}(r) = 0.1075$.

4.2 Recovery

It is well known that a recovery of rank r matrix $X \in \mathbb{R}^{m\times n}$ from just $p < r(m+n-r)$ observed entries is an ill-posed problem Candès & Tao (2010); Tanner & Wei (2013), because there can exist infinitely many rank r matrices with the entries observed.

In the next experiment, we constructed matrices $X \in \mathbb{R}^{m\times n}$ with different ranks $r \in \{1, 2, \ldots, \min\{m, n\}\}$ and tried 10 different random samplings of p elements. For each random sampling, we ran Algorithm 1 for the maximum of 10^6 serial iterations ($|\hat{\mathcal{S}}| = 1$). Figure 2 shows how many times (out of 10) we managed obtain reconstruction with relative error less than 5% of the original matrix. The red line is a theoretical line, above which there is no guarantee of recovery (when $p < r(m+n-r)$).

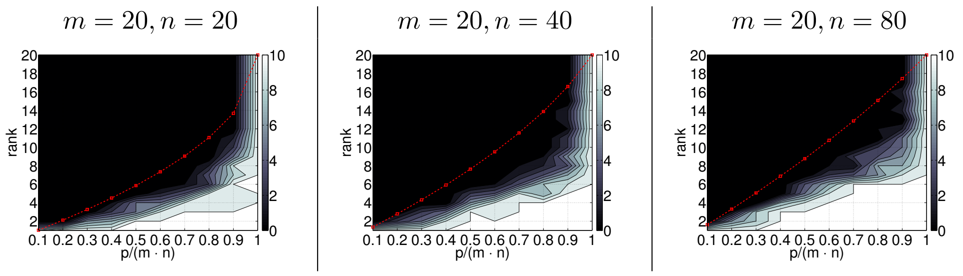


Figure 2: Recovery of rank k matrix $X \in \mathbb{R}^{m\times n}$ for different number of observed elements p.

4.3 Inpaiting

Inpaiting is a process of reconstruction of parts of images or videos. Again, we can think about (e.g. grayscale) image as a matrix X with values in $[0, 1]$. We again observe just a subset of elements indexed by $\mathcal{I}$ and we want to find a low rank matrix Y such that $\mathcal{P}_{\mathcal{I}}(Y) \approx \mathcal{P}_{\mathcal{I}}(X)$. However, this is actually not all we know! We also know that $\forall(i, j) : Y_{i,j}$ should lay in $[0, 1]$. Actually, because we are searching for rank r approximation of X we know that for sure $-\mathcal{R}(r) \geq Y_{i,j} \geq 1 + \mathcal{R}(r)$, but for simplicity we assume that r is big enough and therefore $\mathcal{R}(r)$ is small.

To show how this simple and obvious side-information can help us to find a better rank r matrix, we undertook the following experiment. We took a 512×512 grayscale image (Lenna) and chose 50% of the pixels randomly, indexed as $\mathcal{I}$. Then, we ran Algorithm 1 for 10^7 serial iterations ($|\hat{\mathcal{S}}| = 1$). We obtained solutions $X_E(\text{rank})$ and $X_{IN}(\text{rank})$, where $X_E(rank)$ was obtained when we used only equality constrains ($\mathcal{E} = \mathcal{I}, \mathcal{U} \equiv \mathcal{L} \equiv \emptyset$) and $X_{IN}(rank)$ was obtained when we used also inequality constrains ($\mathcal{E} = \mathcal{I}$, $\mathcal{U} \equiv \mathcal{L} \equiv -\mathcal{I}$, $X^{\mathcal{U}} = \mathbf{0} \in \mathbb{R}^{512\times512}$, $X^{\mathcal{L}} = \mathbf{1} \in \mathbb{R}^{512\times512}$, where $-\mathcal{I}$ is a set of all elements of X except those in $\mathcal{I}$). Figure 3 shows for different rank $\in \{30, 50, 100\}$ the best rank approximation obtained by SVD ($X(\text{rank})$) and solutions $X_E(\text{rank})$ and $X_{IN}(\text{rank})$. The benefit of obvious inequality constrains is nicely visible, e.g., at rank $= 100$, where the relative error of reconstruction is more than twice smaller. Further, the image is more smooth, upon visual inspection.

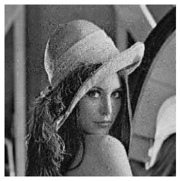
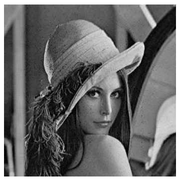
rank	$X(\text{rank})$	$X_E(\text{rank})$	$X_{IN}(\text{rank})$
30	 $\|X(\text{rank})\|_F = 223.9999$	 $\|X(\text{rank}) - X_E(\text{rank})\|_F = 13.1394$	 $\|X(\text{rank}) - X_{IN}(\text{rank})\|_F = 12.6303$
50	 $\|X(\text{rank})\|_F = 224.6876$	 $\|X(\text{rank}) - X_E(\text{rank})\|_F = 18.2070$	 $\|X(\text{rank}) - X_{IN}(\text{rank})\|_F = 13.1859$
100	 $\|X(\text{rank})\|_F = 225.2117$	 $\|X(\text{rank}) - X_E(\text{rank})\|_F = 39.1631$	 $\|X(\text{rank}) - X_{IN}(\text{rank})\|_F = 15.2551$

Figure 3: Adding obvious constrains can help to get better solution.

To illustrate the effect of the obvious constrains further, we took a 50×50 image and sample randomly 50% of pixels. (The image is the top-left corner of the Lenna image.)

Figure 4 shows the original image X and the best rank 10 approximation $X(10)$. The solutions $X_{\mathcal{E}}$, $X_{\mathcal{E}+\mathcal{U}}$, $X_{\mathcal{E}+\mathcal{L}}$ and $X_{\mathcal{E}+\mathcal{U}+\mathcal{L}}$ were obtained by running Algorithm 1 for 3×10^5 serial iterations ($|\hat{\mathcal{S}}| = 1$), where $\mathcal{E}$ contains the observed pixels and $\mathcal{U}$ and $\mathcal{L}$ contains all other pixels. We have used $X_{\mathcal{U}} = \mathbf{0}$ and $X_{\mathcal{L}} = \mathbf{1}$. The result again suggest that adding simple and obvious constrains leads to better low rank reconstruction and helps to keep reconstructed elements of matrix in expected bounds.

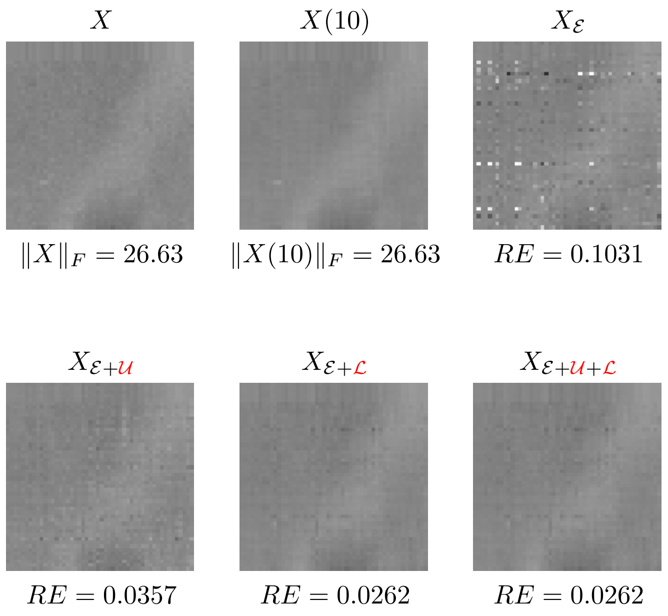


Figure 4: Original 50×50 image, the best rank 10 approximation and reconstruction using Algorithm 1 with different settings. The RE is a relative error defined as $RE(X.) = \|X. - X(10)\|_F / \|X(10)\|$.

4.4 The Netflix Problem

Within collaborative filtering, we focus on the problem presented in the Netflix Prize, which bears the name of Netflix, a company which provides streaming media (e.g. movies and TV series) on-demand on-line. Customers of Netflix can rate movies, which they have seen already, and Netflix uses such recommendations to suggest which movies to watch next. If you have ever rated movies on Netflix, though, you may have noticed that whether you give a movie three stars or four depends on your current mood, viewing conditions, etc. Formally, there is a matrix X, where each row corresponds to one user and each column corresponds to a movie. We know values at $X_{i,j}$ for all $(i,j) \in \mathcal{I}$, but consider interval uncertainty sets around the actual ratings $X_{i,j}$ and solve:

$$\begin{array}{rl} \text{minimize} & \operatorname{rank}(Y), \\ \text{subject to} & Y_{i,j} \le \min\{5, X_{i,j}+1\}, (i,j) \in \mathcal{I}, \\ & Y_{i,j} \ge \max\{1, X_{i,j}-1\}, (i,j) \in \mathcal{I}. \end{array} \tag{10}$$

Given that Netflix uses the scale of 1 to 5 stars in the ratings, we use width 2 interval uncertainty set, but this can be changed freely.

In our computational experiments, we have used

- `smallnetflix_mm.train` for training and `smallnetflix_mm.validate` for testing. The training dataset contains $c_{\text{tr}} = 3,298,163$ rating $\in$

$\{1, 2, 3, 4, 5\}$ of $m = 95,526$ users for $n = 3,561$ movies. There we look for a 95526×3561 matrix of rank 2 or 3.

- a dataset, which contains $100,198,805$ ratings of $480,189$ users for $17,770$ movies. There, we look for a 480189×17770 matrix of rank 20, but we do not have a validation matrix to match.

All data were obtained from a repository hosted by Carnegie Mellon University[2].

In order to illustrate the impact of the use of inequalities, we present the evolution of Root-Mean-Square Error (RMSE) on `smallnetflix` in Figure 5. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in\mathcal{I}_{\text{val}}}(X_{i,j} - Y^*_{i,j})^2}{c_{\text{val}}}},$$

where $\mathcal{I}_{\text{val}}$ contains $c_{\text{val}} = 545,177$ validates points and Y^* is the solution obtained by Algorithm 1 with $\mu = 0.001$ and constrains $X_{i,j} - \Delta \leq Y_{i,j} \leq X_{i,j} + \Delta$ for all $(i, j) \in \mathcal{I}_{\text{tr}}$. By epoch we mean c_{tr} element updates of matrix L and c_{tr} element updates of matrix R. Let us remark that RMSE is sensitive to the choice of Δ and the rank of the matrix we are looking for. If the underlying matrix has a higher rank than expected, $\Delta > 0$ can lead to smaller values of RMSE. We should also note that for some fixed Δ_1 and Δ_2, RMSE can be better with Δ_1 for a few epochs, but then get worse when compared with Δ_2. Hence, in practice, a cross validation should be used to determine suitable value of parameter Δ. One can use the following heuristic: Start solving the problem with a relatively large Δ. Decrease this parameter slowly, e.g. after each epoch. This corresponds to a process, where one seeks progressively less rough approximations of X, similar to decreasing penalty parameter λ in LASSO Tibshirani (1996).

In order to illustrate the run-time and efficiency of parallelization of Algorithm 1, Figure 6 shows the evolution of RMSE both per iteration and per runtime on the larger data set. As expected, the evolution per iteration is almost identical. The only difference stems from the fact that different random seeds were chosen. The evolution per runtime shows almost linear speed-up between 1 and 4 cores and marginally worse speed-up between 4 and 8 cores. Let us remark that in the recovery of a $m \times n$ matrix X, one iteration denotes m coordinate updates of matrix L and n coordinate updates of matrix R.

5 Extensions and Conclusion

We have presented the inequality-constrained matrix completion problem and an efficient algorithm, which converges to station points of the NP-Hard, non-convex optimisation problem, without ever trying to approximate the spectrum of the matrix. In our computational experiments, we have shown that even

[2] `http://www.select.cs.cmu.edu/code/graphlab/datasets/`

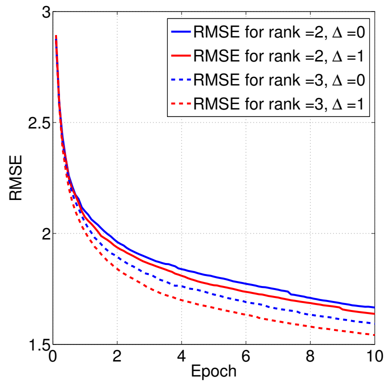


Figure 5: The evolution of RMSE on `smallnetflix` for rank of 2 and 3 and equalities ($\Delta = 0$) or inequalities ($\Delta = 1$) for interval uncertainty set of width 2Δ.

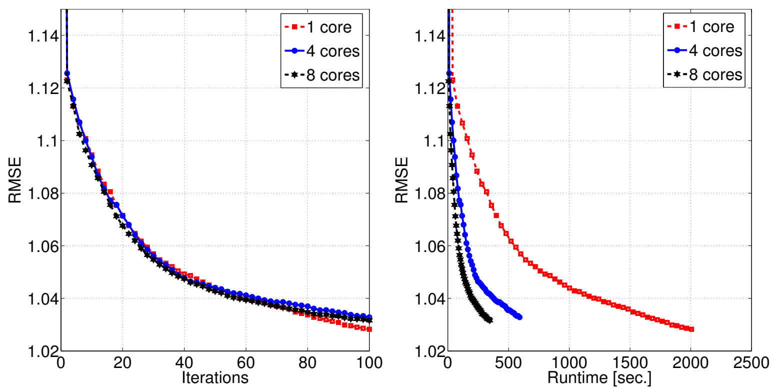


Figure 6: The evolution of RMSE on the larger dataset as function of the number of iterations and run-time using $1, 4$ and 8 cores. The rank parameter is 20 and $\mu = 10^{-3}$.

the most obvious inequality constraints are useful in a number of applications. Some of the applications, e.g. the collaborative filtering under uncertainty, may be of independent interest. This opens numerous avenues for further research:

Non-negative matrix factorization. The coordinate descent algorithm for the problem (4) is easy to extend, e.g., towards non-negative factorization. It is sufficient to modify lines 7 and 13 in Algorithm 1 as follows: $L_{i,\hat{r}} = \max\{0, L_{i,\hat{r}} + \delta_{i,\hat{r}}\}$, $R_{\hat{r},j} = \max\{0, R_{\hat{r},j} + \delta_{\hat{r},j}\}$. One could consider extensions beyond box constraints on the individual elements as well.

Getting rid of parameter μ. If we have some *a priori* bound on the largest eigenvalue of the matrix to reconstruct, let us denote it ζ, then we can modify lines 7 and 13 in Algorithm 1 as follows $L_{i,\hat{r}} = \max\{\min\{\zeta, L_{i,\hat{r}} + \delta_{i,\hat{r}}\}, -\zeta\}$, $R_{\hat{r},j} = \max\{\min\{0, R_{\hat{r},j} + \delta_{\hat{r},j}\}, -\zeta\}$.

We would be delighted to share our code with other researchers interested in the problem.

References

Bradley, Joseph K., Kyrola, Aapo, Bickson, Danny, and Guestrin, Carlos. Parallel coordinate descent for l1-regularized loss minimization. In *ICML*, June 2011.

Cai, Jian-Feng, Candès, Emmanuel J, and Shen, Zuowei. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

Candès, Emmanuel J and Plan, Yaniv. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

Candès, Emmanuel J. and Tao, Terence. The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56 (5):2053–2080, 2010.

Chen, Yuxin and Chi, Yuejie. Spectral compressed sensing via structured matrix completion. In *ICML*, pp. 414–422, 2013.

Goldfarb, Donald, Ma, Shiqian, and Wen, Zaiwen. Solving low-rank matrix completion problems efficiently. In *Allerton conference on communication, control, and computing*, pp. 1013–1020. IEEE, 2009.

Halko, Nathan, Martinsson, Per-Gunnar, and Tropp, Joel A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

Jaggi, Martin and Sulovský, Marek. A simple algorithm for nuclear norm regularized problems. In *ICML*, pp. 471–478, 2010.

Lee, Jason., Recht, Benjamin, Salakhutdinov, Ruslan, Srebro, Nathan, and Tropp, Joel A. Practical large-scale optimization for max-norm regularization. In *NIPS*, pp. 1297–1305, December 2010.

Ma, Shiqian, Goldfarb, Donald, and Chen, Lifeng. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

Natarajan, Balas Kausik. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

Nesterov, Yurii. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Recht, Benjamin, Fazel, Maryam, and Parrilo, Pablo A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

Recht, Benjamin, Ré, Christopher, Wright, Stephen J., and Niu, Feng. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pp. 693–701, 2011.

Richtárik, Peter and Takáč, Martin. Parallel coordinate descent methods for big data optimization. *arXiv:1212.0873*, 2012.

Shalev-shwartz, Shai, Gonen, Alon, and Shamir, Ohad. Large-scale convex minimization with a low-rank constraint. In *ICML*, pp. 329–336, 2011.

Soyster, Allen L. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):pp. 1154–1157, 1973.

Srebro, Nathan. *Learning with matrix factorizations.* PhD thesis, MIT, 2004.

Srebro, Nathan, Rennie, Jason, and Jaakkola, Tommi S. Maximum-margin matrix factorization. In *NIPS*, pp. 1329–1336, 2004.

Tanner, Jared and Wei, Ke. Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5), 2013.

Tibshirani, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Tomioka, Ryota, Suzuki, Taiji, Sugiyama, Masashi, and Kashima, Hisashi. A fast augmented lagrangian algorithm for learning low-rank matrices. In *ICML*, pp. 1087–1094, 2010.

Tseng, Paul. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109 (3):475–494, 2001.

Yun, Hyokun, Yu, Hsiang-Fu, Hsieh, Cho-Jui, Vishwanathan, SVN, and Dhillon, Inderjit. NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix completion. *arXiv:1312.0193*, 2013.