音视频编码技术期末报告 – 图像去雾

刘昭炀
22215608
liuzhy86@mail2.sysu.edu.cn

摘要

Python 版源码[1]

1 Introduction

In this report, we reproduce a matrix completion algorithm called truncated nuclear norm regularization (TNNR) and three optimization schemes proposed by Hu et al[1]: the alternating direction method of multipliers[2] (ADMM), the accelerated proximal gradient line search method[3] (APGL) and the Alternating Direction Method of Multipliers with Adaptive Penalty (ADMMAP). Unlike other nuclear norm methods, TNNR only minimize the smallest $\min(m,n)-r$ singular values since the rank of a matrix only corresponds to the r non-zero singular values.

In Section 2, we briefly introduce three kinds of work related to matrix completion. In Section 3, we introduce the TNNR algorithm and three optimization schemes. In Section 4, we conduct experiments similar to the original paper and give the experimental results. In Section 5, we summarize what we have accomplished.

Notations 1 *Let* $\mathbf{X}=(\mathbf{x}_1,\cdots,\mathbf{x}_n)$ *be an* $m\times n$ *matrix,* $\in\{1,\cdots,m\}\times\{1,\cdots,n\}$ *denote the indices of the observed entries of* $\mathbf{X}$*, and let* $_c$ *denote the indices of the missing entries. The Forbenius norm of* $\mathbf{X}$ *is defined as* $\|\mathbf{X}\|_F=\sqrt{\sum_{(i,j)}\mathbf{X}_{i,j}^2}$*. Let* $\mathbf{X}=\mathbf{U}\Sigma\mathbf{V}^T$ *be the singular value decomposition for* $\mathbf{X}$*, where* $\Sigma=diag(\sigma_i)$*,* $1\le i\le\min\{m,n\}$*, and* σ_i *is the ith largest singular value of* $\mathbf{X}$*. The nuclear norm of* $\mathbf{X}$ *is denoted as* $\|\mathbf{X}\|_*=\sum_{i=1}^{\min(m,n)}\sigma_i$ *. Let* $\mathcal{P}$ *be the orthogonal projection operator onto the span of matrices vanishing outside of so that*

$$(\mathcal{P}(\mathbf{X}))_{ij}=\begin{cases}\mathbf{X}_{ij} & \text{if } (i,j)\in\\ 0 & \text{if } (i,j)\in\ ^c\end{cases}$$

[1]Source code available at: `https://github.com/AnnLIU15/matrixTheory`

The inner product of the matrix space is defined as $\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i,j} = \mathbf{X}_{ij}\mathbf{Y}_{ij}$.

Definition 1.1 *Consider the SVD of a matrix* $\mathbf{X} \in \mathbb{R}^{m\times n}$*:*

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T,\ \Sigma = diag(\{\sigma_i\}_{1\leq \min(m,n)}). \tag{1}$$

Define the singular value shrinkage operator $\mathcal{D}_\tau$*[4] as follows:*

$$\mathcal{D}_\tau(\mathbf{X}) = \mathbf{U}\mathcal{D}_\tau(\Sigma)\mathbf{V}^T,\ \mathcal{D}_\tau(\Sigma) = diag(\max\{\sigma_i - \tau, 0\}). \tag{2}$$

We have the following useful theorem:

Theorem 1.1 *For each* $\tau \geq 0$ *and* $\mathbf{Y} \in \mathbb{R}^{m\times n}$*, we have*

$$\mathcal{D}_\tau(\mathbf{X}) = \arg\min_{\mathbf{X}} \frac{1}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2 + \tau\|\mathbf{X}\|_*. \tag{3}$$

2 Related work

In this section, we introduce three methods for matrix completion: singular value thresholding (SVT)[4], singular value projection (SVP)[5], and Optspace[6]. These methods are also used to compare with the TNNR algorithm in subsequent experiments. In addition, ADMM and APGL is is briefly explained for better algorithm derivation in Section 3.

By approximating the rank function using the nuclear norm to solve the rank minimization problem (**??**), the matrix completion problem can be formulated[7] as follows:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* \\ \text{s.t.} \quad & \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}). \end{aligned} \tag{4}$$

For SVT, Cai et al. propose the SVT algorithm :

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* + \alpha\|\mathbf{X}\|_F^2 \\ \text{s.t.} \quad & \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}). \end{aligned} \tag{5}$$

Construct the Lagrangian function,

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X}\|_* + \alpha\|\mathbf{X}\|_F^2 + \langle \mathbf{Y}, P(\mathbf{M} - \mathbf{X})\rangle, \tag{6}$$

where Lagrangian multipliers $\mathbf{Y} \in \mathbb{R}^{m\times n}$. SVT solves problem (4) using alternating iterative methods,

$$\begin{aligned} \mathbf{X}_{k+1} &= D_\tau(\mathbf{Y}_k) \\ \mathbf{Y}_{k+1} &= \mathbf{Y}_k + \sigma_k\mathcal{P}(\mathbf{M} - \mathbf{X}_{k+1}). \end{aligned} \tag{7}$$

For SVP, rewrite (4) as:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \psi(\mathbf{X}) = \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - b\|_2^2 \\ \text{s.t.} \quad & \mathbf{X} \in \mathcal{C}(k) = \{\mathbf{X} \colon \operatorname{rank}(\mathbf{X}) \leq k\}, \end{aligned} \tag{8}$$

where $\mathcal{A}$ is affine transformations that satisfy a restricted isometry property. Construct the Lagrangian function,

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \min \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_F^2. \tag{9}$$

The iterative equation is

$$\begin{aligned} \mathbf{Y}_{k+1} &= \mathbf{X}_k - \gamma_k \mathcal{A}^*(\mathcal{A}(\mathbf{X}_k) - \mathbf{y}) \\ \mathbf{X}_{k+1} &= \text{Trancated SVD}_r(\mathbf{Y}_{k+1}). \end{aligned} \tag{10}$$

For OptSpace, the Lagrangian function of problem (4) is

$$L(\mathbf{X}, \mathbf{Y}) = \min_{\mathbf{S} \in \mathbb{R}^{r \times r}} L(\mathbf{X}, \mathbf{Y}, \mathbf{S}) = \frac{1}{2}\|\mathcal{P}\left(\mathbf{M} - \mathbf{X}\mathbf{S}\mathbf{Y}^T\right)\|_F^2 + \frac{\lambda}{2}\|\mathcal{P}_c(\mathbf{M} - \mathbf{X}\mathbf{S}\mathbf{Y}^T)\|_F^2. \tag{11}$$

ADMM algorithm is an important method for solving separable convex optimization problems, with fast processing speed and good convergence performance.The classic ADMM algorithm is suitable for solving the following 2-block (or N-block) convex optimization problems.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{w}} \quad & f(\mathbf{x}) + g(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{w} = \mathbf{b}, \end{aligned} \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^m$. And $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times m}$, $\mathbf{b} \in \mathbb{R}^p$ is convex set.f, g is convex function. Using an augmented Lagrangian with a square regularization term with coefficient $\frac{\beta}{2}$:

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{w}) = f(\mathbf{x}) + g(\mathbf{w}) + \mathbf{z}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{w} - \mathbf{b}) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{w} - \mathbf{b}\|_2^2. \tag{13}$$

By dual ascent,

$$\begin{aligned} (\mathbf{x}_{k+1}, \mathbf{w}_{k+1}) &:= \min_{\mathbf{x},\mathbf{w}} \mathcal{L}(\mathbf{x}, \mathbf{z}_k, \mathbf{w}) \\ \mathbf{z}_{k+1} &:= \mathbf{z}_k + \beta(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{w}_{k+1} - \mathbf{b}). \end{aligned} \tag{14}$$

Change $(\mathbf{x}, \mathbf{w})$ joint optimization to separate alternate iterations and let $\mathbf{y}_k = \frac{\mathbf{z}_k}{\beta}$,

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}_k, \mathbf{w}_k) \\ \mathbf{w}_{k+1} &= \arg\min_{\mathbf{w}} \mathcal{L}_\beta(\mathbf{x}_{k+1}, \mathbf{y}_k, \mathbf{w}) \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{y}_{k+1} - \mathbf{b}). \end{aligned} \tag{15}$$

APGL, also called as iterative shrinkage-thresholding algorithm (FISTA), solves problems like

$$\min_{\mathbf{X}} g(\mathbf{X}) + f(\mathbf{X}), \tag{16}$$

where g is closed, convex, possibly, nondifferentiable and f is a convex and differentiable function.

$$Q(\mathbf{X}, \mathbf{Y}) = f(\mathbf{Y}) + \langle \mathbf{X} - \mathbf{Y}, \nabla f(\mathbf{Y}) \rangle + \frac{1}{2t}\|\mathbf{X} - \mathbf{Y}\|_F^2 + g(\mathbf{X}). \tag{17}$$

Then APGL method solves optimization problem (16) by iteratively updating $\mathbf{X}$, $\mathbf{Y}$ and t. In the *k-th* iteration, we update $\mathbf{X}_{k+1}$ as the unique minimizer of $Q(\mathbf{X}, \mathbf{Y}_k)$:

$$\mathbf{X}_{k+1} = \arg\min_{\mathbf{X}} Q(\mathbf{X}, \mathbf{Y}_k) = \arg\min_{\mathbf{X}} g(\mathbf{X}) + \frac{1}{2t}\|\mathbf{X} - (\mathbf{Y}_k - t_k \nabla f(\mathbf{Y}_k))\|_F^2. \tag{18}$$

3 Truncated nuclear norm regularization

3.1 The approach proposed by Hu et al.

By using Truncated nuclear norm, this approach achieves a better approximation of the rank function than the nuclear norm.

Definition 3.1 *Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the truncated nuclear norm $\|\mathbf{X}\|_r$ is defined as the sum of $\min(m, n) - r$ minimum singular values, i.e., $\|\mathbf{X}\|_r = \sum_{i=r+1}^{\min(m,n)} \sigma_i(\mathbf{X})$.*

Thus, the objective function of this approach can be formulated as follows

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_r \\ \text{s.t.} \quad & \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}). \end{aligned} \tag{19}$$

Since $\|\mathbf{X}\|_r$ is non-convex, they propose the follow theorem[8]:

Theorem 3.1 *For any given matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, any matrices $\mathbf{A} \in \mathbb{R}^{r \times m}$, $\mathbf{B} \in \mathbb{R}^{r \times m}$ that $\mathbf{A}\mathbf{A}^T = I_{r \times r}$. For any nonnegative integer r ($r \leq \min(m, n)$), we have*

$$\mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \leq \sum_{i=1}^{r} \sigma_i(\mathbf{X}). \tag{20}$$

The proof is given in the Appendix.

Suppose, $\mathbf{U}\Sigma\mathbf{V}^T$ is the singular value decomposition of $\mathbf{X}$, where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_m) \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times n}$, and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_m) \in \mathbb{R}^{n \times n}$. The equality of holds when

$$\mathbf{A} = (\mathbf{u}_1, \ldots, \mathbf{u}_m)^T, \ \mathbf{B} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)^T. \tag{21}$$

This is because

$$\begin{aligned}\mathrm{Tr}((\mathbf{u}_1,\ldots,\mathbf{u}_m)^T X(\mathbf{v}_1,\ldots,\mathbf{v}_m)) &= \mathrm{Tr}((\mathbf{u}_1,\ldots,\mathbf{u}_m)^T U\Sigma\mathbf{V}^T(\mathbf{v}_1,\ldots,\mathbf{v}_m)) \\ &= \mathrm{Tr}(((\mathbf{u}_1,\ldots,\mathbf{u}_m)^T\mathbf{U})\Sigma(\mathbf{V}^T(\mathbf{v}_1,\ldots,\mathbf{v}_m))) \\ &= \mathrm{Tr}\left(\begin{bmatrix}\mathbf{I}_r & 0\\ 0 & 0\end{bmatrix}\Sigma\begin{bmatrix}\mathbf{I}_r & 0\\ 0 & 0\end{bmatrix}\right) \\ &= \mathrm{Tr}(\mathrm{diag}(\sigma_1(\mathbf{X}),\ldots,\sigma_r(\mathbf{X}),0,\ldots,0)) \\ &= \sum_{i=1}^{r}\sigma_i(\mathbf{X})\end{aligned} \tag{22}$$

Combining (20) and (22), we get

$$\max_{\mathbf{A}\mathbf{A}^T=\mathbf{I},\mathbf{B}\mathbf{B}^T=\mathbf{I}} \mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \le \sum_{i=1}^{r}\sigma_i(\mathbf{X}). \tag{23}$$

Then we have

$$\|\mathbf{X}\|_* - \max_{\mathbf{A}\mathbf{A}^T=\mathbf{I},\mathbf{B}\mathbf{B}^T=\mathbf{I}} \mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) = \sum_{i=1}^{\min(m,n)}\sigma_i(\mathbf{X}) - \sum_{i=1}^{r}\sigma_i(\mathbf{X}) = \|\mathbf{X}\|_r. \tag{24}$$

Thus, the optimization problem (19) can be rewritten as follows:

$$\begin{aligned}\min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* - \max_{\mathbf{A}\mathbf{A}^T=\mathbf{I},\mathbf{B}\mathbf{B}^T=\mathbf{I}} \mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \\ \text{s.t.} \quad & \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}),\end{aligned} \tag{25}$$

where $\mathbf{A} \in \mathbb{R}^{r\times m}$, $\mathbf{B} \in \mathbb{R}^{r\times m}$.

Based on (25), they design a simple but efficient iterative scheme, which is summarized in Algorithm 1. In the lth iteration, we first fix $\mathbf{X}_l$ and compute $\mathbf{A}_l$ and $\mathbf{B}_l$. And then we fix $\mathbf{A}_l$ and $\mathbf{B}_l$ ro update $\mathbf{X}_{l+1}$ by solving the following problem:

$$\begin{aligned}\min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* - \mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \\ \text{s.t.} \quad & \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}).\end{aligned} \tag{26}$$

3.2 The optimization using TNNR-ADMM

ADMM is a method for solving a decomposable convex optimization problem. It can equivalently decompose the objective function of the original problem into several solvable sub-problems, then solve each sub-problem in parallel, and finally coordinate the solutions of the sub-problems to obtain the global solution of the original problem.

By using ADMM to solve (26), an optimization algorithm TNNR-ADMM is proposed. First, rewrite (26) as follows:

$$\begin{aligned}\min_{\mathbf{X},\mathbf{W}} \quad & \|\mathbf{X}\|_* - \mathrm{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T) \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{W},\ \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M}).\end{aligned} \tag{27}$$

Algorithm 1 The Proposed Two-Step Approach for Sovling (6)

Input: original incomplete data matrix $\mathbf{M}$, where is the indices of the observed entries, tolerance ϵ_0

Initialization: $\mathbf{X}_1 = \mathcal{P}(\mathbf{M})$.

repeat

step 1. Given $\mathbf{X}_l$, compute $[\mathbf{U}_l, \Sigma_l, \mathbf{V}_l] = \text{svd}(\mathbf{X}_l)$,

where $\mathbf{U}_l = (\mathbf{u}_1, \ldots, \mathbf{u}_m) \in \mathbb{R}^{m\times m}$, $\mathbf{V}_l = (\mathbf{v}_1, \ldots, \mathbf{v}_m) \in \mathbb{R}^{n\times n}$.

Compute $\mathbf{A}_l$ and $\mathbf{B}_l$, as $\mathbf{A}_l = (\mathbf{u}_1, \ldots, \mathbf{u}_m)^T$, $\mathbf{B}_l = (\mathbf{v}_1, \ldots, \mathbf{v}_m)^T$.

step 2. Solve $\mathbf{X}_{l+1} = \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* - \text{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \quad \text{s.t.} \quad \mathcal{P}(\mathbf{X}) = \mathcal{P}(\mathbf{M})$

until $\|\mathbf{X}_{l+1} - \mathbf{X}_l\|_F \le \epsilon_0$.

return the recovered matrix.

By (12), we have $f(\mathbf{X}) = \|\mathbf{X}\|_*$ and $g(\mathbf{W}) = \text{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T)$.

The augmented lagrange function of is

$$L(\mathbf{X}, \mathbf{Y}, \mathbf{W}, \beta) = \|\mathbf{X}\|_* - \text{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T) + \frac{\beta}{2}\|\mathbf{X} - \mathbf{W}\|_F^2 + \text{Tr}(\mathbf{Y}^T(\mathbf{X} - \mathbf{W})), \tag{28}$$

where $\beta > 0$ is the penalty parameter. Given the initial setting $\mathbf{X}_1 = \mathcal{P}(\mathbf{M})$, $\mathbf{W}_1 = \mathbf{X}_1$, and $\mathbf{Y}_1 = \mathbf{X}_1$, the optimization problem (27) can be solved via the following three steps.

Computing $\mathbf{X}_{k+1}$. Fix $\mathbf{W}_k$ and $\mathbf{Y}_k$, and minimize $L(\mathbf{X}, \mathbf{Y}_k, \mathbf{W}_k, \beta)$ for $\mathbf{X}_{k+1}$ as follows:

$$\begin{aligned}\mathbf{X}_{k+1} &= \arg\min_{\mathbf{X}} L(\mathbf{X}, \mathbf{Y}_k, \mathbf{W}_k, \beta) \\ &= \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* - \text{Tr}(\mathbf{A}_l\mathbf{W}_k\mathbf{B}_l^T) + \frac{\beta}{2}\|\mathbf{X} - \mathbf{W}_k\|_F^2 + \text{Tr}(\mathbf{Y}_k^T(\mathbf{X} - \mathbf{W}_k)).\end{aligned} \tag{29}$$

Ignoring constant terms, this can be rewritten as

$$\mathbf{X}_{k+1} = \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{\beta}{2}\|\mathbf{X} - \left(\mathbf{W}_k - \frac{1}{\beta}\mathbf{Y}_k\right)\|_F^2 \tag{30}$$

By Theorem 1.1 ,we can solve the above problem as

$$\mathbf{X}_{k+1} = \mathcal{D}_{\frac{1}{\rho}}(\mathbf{W}_k - \frac{1}{\rho}\mathbf{Y}_k). \tag{31}$$

Computing $\mathbf{W}_{k+1}$. Fix $\mathbf{X}_{k+1}$ and $\mathbf{Y}_k$ to calculate $\mathbf{W}_{k+1}$ as follows:

$$\begin{aligned}\mathbf{W}_{k+1} &= \arg\min_{\mathcal{P}(\mathbf{W})=\mathcal{P}(\mathbf{M})} L(\mathbf{X}_{k+1}, \mathbf{Y}_k, \mathbf{W}, \beta) \\ &= \arg\min_{\mathcal{P}(\mathbf{W})=\mathcal{P}(\mathbf{M})} \|\mathbf{X}_{k+1}\|_* - \text{Tr}(\mathbf{A}_l\mathbf{W}\mathbf{B}_l^T) + \frac{\beta}{2}\|\mathbf{X}_{k+1} - \mathbf{W}\|_F^2 + \text{Tr}(\mathbf{Y}_k^T(\mathbf{X}_{k+1} - \mathbf{W})).\end{aligned} \tag{32}$$

then we get

$$\mathbf{W}_{k+1} = \mathbf{X}_{k+1} + \frac{1}{\beta}(\mathbf{A}_l^T\mathbf{B}_l + \mathbf{Y}_k). \tag{33}$$

Algorithm 2 The Optimization using TNNR-ADMM

Input: $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{M}$, and tolerance ϵ are given.

Initialization: $\mathbf{X}_1 = \mathbf{M}$, $\mathbf{W}_1 = \mathbf{X}_1$, $\mathbf{Y}_1 = \mathbf{X}_1$, and $\beta = 1$.

repeat

step 1. $\mathbf{X}_{k+1} = \mathcal{D}_{\frac{1}{\rho}}(\mathbf{W}_k - \frac{1}{\rho}\mathbf{Y}_k)$.

step 2. $\mathbf{W}_{k+1} = \mathbf{X}_{k+1} + \frac{1}{\beta}(\mathbf{A}_l^T\mathbf{B}_l + \mathbf{Y}_k)$.

Fix values at observed entries, $\mathbf{W}_{k+1} = \mathbf{X}_{k+1} + \frac{1}{\beta}(\mathbf{A}_l^T\mathbf{B}_l + \mathbf{Y}_k)$.

step 3. $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \beta(\mathbf{X}_{k+1} - \mathbf{W}_{k+1})$.

until $\|\mathbf{X}_{k+1} - \mathbf{X}_k\|_F \leq \epsilon$.

fix the values at the observed entries

$$\mathbf{W}_{k+1} = \mathcal{P}^c(\mathbf{X}_{k+1}) + \mathcal{P}(\mathbf{M}). \tag{34}$$

Computing $\mathbf{Y}_{k+1}$. Fix $\mathbf{X}_{k+1}$ and $\mathbf{W}_{k+1}$ to calculate $\mathbf{Y}_{k+1}$ as follows:

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k + \beta(\mathbf{X}_{k+1} - \mathbf{W}_{k+1}). \tag{35}$$

The whole procedure of TNNR-ADMM is summarized in Algorithm 2.

3.3 The optimization using TNNR-APGL

Considering the noisy data in the real applications, it is beneficial to relax the constrained problem (26) into

$$\min \|\mathbf{X}\|_* - \mathrm{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T) + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{M}\|_F^2, \tag{36}$$

for some $\lambda > 0$.

According to (16), in problem (36), we choose

$$g(\mathbf{X}) = \|\mathbf{X}\|_*,$$

and

$$f(\mathbf{X}) = -\mathrm{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T) + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{M}\|_F^2.$$

By Theorem 1.1, we get

$$\begin{aligned}\mathbf{X}_{k+1} &= \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* + \frac{1}{2t}\|\mathbf{X} - (\mathbf{Y}_k - t_k\nabla f(\mathbf{Y}_k))\|_F^2 \\ &= \mathcal{D}_{t_k}(\mathbf{Y}_k - t_k\nabla f(\mathbf{Y}_k)) \\ &= \mathcal{D}_{t_k}(\mathbf{Y}_k - t_k(\mathbf{A}_l^T\mathbf{B}_l - \lambda(\mathcal{P}(\mathbf{Y}_k) - \mathcal{P}(\mathbf{M})))).\end{aligned} \tag{37}$$

Finally, t_{k+1} and $\mathbf{Y}_{k+1}$ are updated as:

$$\begin{aligned}t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \\ \mathbf{Y}_{k+1} &= \mathbf{X}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{X}_{k+1} - \mathbf{X}_k).\end{aligned} \tag{38}$$

Algorithm 3 The Optimization using TNNR-APGL

Input: $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{M}$, and tolerance ϵ are given.

Initialization: $t_1 = 1$, $\mathbf{X}_1 = \mathbf{M}$, $\mathbf{Y}_1 = \mathbf{X}_1$.

repeat

step 1. $\mathbf{X}_{k+1} = \mathcal{D}_{t_k}(\mathbf{Y}_k - t_k(\mathbf{A}_l^T\mathbf{B}_l - \lambda(\mathcal{P}(\mathbf{Y}_k) - \mathcal{P}(\mathbf{M}))))$.

step 2. $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$.

step 3. $\mathbf{Y}_{k+1} = \mathbf{X}_{k+1} + \frac{t_k-1}{t_{k+1}}(\mathbf{X}_{k+1} - \mathbf{X}_k)$.

until $\|\mathbf{X}_{k+1} - \mathbf{X}_k\|_F \leq \epsilon$.

The procedures of solving (26) are summarized in Algorithm 3.

3.4 The optimization using TNNR-ADMMAP

In Algorithm 2, the two constraints are considered separately, but the inconsistency of the two constraints may slow down the convergence. By combining the two constraints into a new constraint in a special way[9] and adding an adaptive penalty parameter[10] to speed up the convergence, a new approach by using the ADMM with adaptive penalty (TNNR-ADMMAP) is proposed.

3.4.1 The reformulation of problem

To deal with the two constraints simultaneously, in this section, we reformulate the problem 27 as follows:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* - \mathrm{Tr}(\mathbf{A}_l\mathbf{X}\mathbf{B}_l^T) \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{X}) + \mathcal{B}(\mathbf{M}) = \mathcal{C}, \end{aligned} \tag{39}$$

where $\mathcal{A}$ and $\mathcal{B} : \mathbb{R}^{m\times n} \to \mathbb{R}^{2m\times 2n}$ are linear opertors defined as follows:

$$\mathcal{A}(\mathbf{X}) = \begin{pmatrix} \mathbf{X} & 0 \\ 0 & 0 \end{pmatrix},\ \mathcal{B}(\mathbf{X}) = \begin{pmatrix} -\mathbf{W} & 0 \\ 0 & \mathcal{P}(\mathbf{X}) \end{pmatrix},\ \mathcal{C} = \begin{pmatrix} 0 & 0 \\ 0 & \mathcal{P}(\mathbf{M}) \end{pmatrix}.$$

Then discuss the properties of adjoint operators $\mathcal{A}$ and $\mathcal{B}$.

Suppose

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix},$$

where $\mathbf{Y}_{ij} \in \mathbb{R}^{m\times n}$, $i = 1, 2$ and $j = 1, 2$. Denote $\mathcal{A}^*$, $\mathcal{B}^* : \mathbb{R}^{2m\times 2n} \to \mathbb{R}^{\geqslant\times\ltimes}$ as the adjoint operators of $\mathcal{A}$ and $\mathcal{B}$ separately satisfying

$$\langle \mathcal{A}(\mathbf{X}), \mathbf{Y}\rangle = \langle \mathbf{X}, \mathcal{A}^*(\mathbf{Y})\rangle,\ \langle \mathcal{B}(\mathbf{X}), \mathbf{Y}\rangle = \langle \mathbf{X}, \mathcal{B}^*(\mathbf{Y}) \tag{40}$$

By the definition of operators $\mathcal{A}$ and $\mathcal{B}$, it is easy to verify that

$$\langle \mathcal{A}(\mathbf{X}), \mathbf{Y}\rangle = \mathrm{Tr}\begin{pmatrix} \mathbf{X} & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix}^T = \mathrm{Tr}(\mathbf{X}\mathbf{Y}_{11}^T) = \langle \mathbf{X}, \mathbf{Y}_{11}\rangle$$

and

$$\begin{aligned}\langle \mathcal{B}(\mathbf{W}), \mathbf{Y}\rangle &= \mathrm{Tr}\begin{pmatrix} -\mathbf{W} & 0 \\ 0 & \mathcal{P}(\mathbf{W}) \end{pmatrix}\begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix}^T \\ &= \mathrm{Tr}\begin{pmatrix} -\mathbf{W}\mathbf{Y}_{11}^T & -\mathbf{W}\mathbf{Y}_{21}^T \\ \mathcal{P}(\mathbf{W})\mathbf{Y}_{11}^T & \mathcal{P}(\mathbf{W})\mathbf{Y}_{22}^T \end{pmatrix} \\ &= \langle \mathbf{W}, -\mathbf{Y}_{11}\rangle + \langle \mathbf{W}, \mathcal{P}(\mathbf{Y}_{22})\rangle \\ &= \langle \mathbf{W}, -\mathbf{Y}_{11} + \mathcal{P}(\mathbf{Y}_{22})\rangle.\end{aligned}$$

By the definition of the adjoint operator (40), the adjoint operators $\mathcal{A}$ and $\mathcal{B}$ can be computed as

$$\begin{aligned}\mathcal{A}^*(\mathbf{Y}) &= \mathbf{Y}_{11}, \\ \mathcal{B}^*(\mathbf{Y}) &= -\mathbf{Y}_{11} + \mathcal{P}(\mathbf{Y}_{22}).\end{aligned} \tag{41}$$

Let us rewrite the augmented Lagrange function for the optimization problem (39) as

$$\mathcal{L}_{AP}(\mathbf{X}, \mathbf{W}, \mathbf{Y}, \beta) = -\mathrm{Tr}(\mathbf{A}_l\mathbf{W}\mathbf{B}_l^T) + \langle \mathbf{Y}, \mathcal{A}(\mathbf{X}) + \mathcal{B}(\mathbf{W}) - \mathcal{C}\rangle + \|\mathbf{X}\|_* + \frac{\beta}{2}\|\mathcal{A}(\mathbf{X}) + \mathcal{B}(\mathbf{W}) - \mathcal{C}\|_F^2, \tag{42}$$

where $\mathbf{Y}$ is the Lagrange multiplier matrix and $\beta > 0$ is the penalty parameter. The iterative scheme of ADMM for problem (39) is given as follows:

$$\begin{aligned}\mathbf{X}_{k+1} &= \arg\min_{\mathbf{X}} \mathcal{L}_{AP}(\mathbf{X}, \mathbf{W}_k, \mathbf{Y}_k, \beta) \\ &= \arg\min_{\mathbf{X}} \frac{\beta}{2}\|\mathcal{A}(\mathbf{X}) + \mathcal{B}(\mathbf{W}_k) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\|_F^2 + \|\mathbf{X}\|_*,\end{aligned} \tag{43}$$

$$\begin{aligned}\mathbf{W}_{k+1} &= \arg\min_{W} \mathcal{L}_{AP}(\mathbf{X}_{k+1}, \mathbf{W}, \mathbf{Y}_k, \beta) \\ &= \arg\min_{W} \frac{\beta}{2}\|\mathcal{A}(\mathbf{X}_{k+1}) + \mathcal{B}(\mathbf{W}) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\|_F^2 - \mathrm{Tr}(\mathbf{A}_l\mathbf{W}\mathbf{B}_l^T),\end{aligned} \tag{44}$$

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k + \beta[\mathcal{A}(\mathbf{X}_{k+1}) + \mathcal{B}(\mathbf{W}) - \mathcal{C}]. \tag{45}$$

3.4.2 The iterative scheme of TNNR-ADMMAP

Based on the iterative scheme (43)-(45), TNNR-ADMMAP algorithm is introduced.

Computing $\mathbf{X}_{k+1}$.

$$\begin{aligned}\mathbf{X}_{k+1} &= \arg\min_{\mathbf{X}} \frac{\beta}{2}\|\mathcal{A}(\mathbf{X}) + \mathcal{B}(\mathbf{W}_k) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\|_F^2 + \|\mathbf{X}\|_* \\ &= \arg\min_{\mathbf{X}} \frac{\beta}{2}\|\mathbf{P}\|_F^2 + \|\mathbf{X}\|_*,\end{aligned} \tag{46}$$

where

$$\mathbf{P} = \begin{pmatrix} \mathbf{X} - \mathbf{W}_k + \frac{1}{\beta} & \frac{1}{\beta}(\mathbf{Y}_k)_{12} \\ \frac{1}{\beta}(\mathbf{Y}_k)_{21} & \mathcal{P} + \frac{1}{\beta}(\mathbf{Y}_k)_{22} \end{pmatrix}.$$

By Theorem (1.1), we get

$$\mathbf{X}_{k+1} = \mathcal{D}_{\frac{1}{\beta}}\left(\mathbf{W}_k - \frac{1}{\beta}(\mathbf{Y}_k)_{11}\right). \tag{47}$$

Computing $\mathbf{W}_{k+1}$. Setting the first derivative of $\mathcal{L}_{AP}(\mathbf{X}_{k+1}, \mathbf{W}, \mathbf{Y}_k, \beta)$ to zero, we get

$$\beta\mathcal{B}^*\left[\mathcal{B}(\mathbf{W}) + \mathcal{A}(\mathbf{X}_{k+1}) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\right] - \mathbf{A}_l^T\mathbf{B}_l = 0,$$

which can be rewritten as

$$\mathcal{B}^*\mathcal{B}(\mathbf{W}) = \frac{1}{\beta}\mathbf{A}_l^T\mathbf{B}_l - \mathcal{B}^*\left[\mathcal{A}(\mathbf{X}_{k+1}) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\right]. \tag{48}$$

From the property of adjoint operator $\mathcal{B}^*$, the left-hand side of the above equation can be rewritten as

$$\mathcal{B}^*\mathcal{B}(\mathbf{W}) = \mathcal{B}^*\begin{pmatrix} -\mathbf{W} & 0 \\ 0 & \mathcal{P}(\mathbf{W}) \end{pmatrix} = \mathbf{W} + \mathcal{P}(\mathbf{W}). \tag{49}$$

Then we apply the orthogonal projection operator $\mathcal{P}$ on both sides of (49), and finally we get

$$\mathcal{P}(\mathbf{W}) = \frac{1}{2}\mathcal{P}(\mathcal{B}^*\mathcal{B}(\mathbf{W})). \tag{50}$$

From (49) and (50), we obtain

$$\mathbf{W}_{k+1} = \mathcal{B}^*\mathcal{B}(\mathbf{W}) - \mathcal{P}(\mathbf{W}) = \mathcal{B}^*\mathcal{B}(\mathbf{W}) - \frac{1}{2}\mathcal{P}(\mathcal{B}^*\mathcal{B}(\mathbf{W})).$$

Then compute $\mathcal{B}^*\mathcal{B}(\mathbf{W})$ directly as follows:

$$\begin{aligned}
\mathcal{B}^*\mathcal{B}(\mathbf{W}) &= \frac{1}{\beta}\mathbf{A}_l^T\mathbf{B}_l - \mathcal{B}^*\left[\mathcal{A}(\mathbf{X}_{k+1}) - \mathcal{C} + \frac{1}{\beta}\mathbf{Y}_k\right] \\
&= \frac{1}{\beta}\mathbf{A}_l^T\mathbf{B}_l - \mathcal{B}^*\left[\begin{pmatrix} \mathbf{X}_{k+1} & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & \mathcal{P} \end{pmatrix} + \frac{1}{\beta}\begin{pmatrix} (\mathbf{Y}_k)_{11} & (\mathbf{Y}_k)_{12} \\ (\mathbf{Y}_k)_{21} & (\mathbf{Y}_k)_{22} \end{pmatrix}\right] \\
&= -\mathcal{B}^*\begin{pmatrix} \frac{1}{\beta}(\mathbf{Y}_k)_{11} + \mathbf{X}_{k+1} & \frac{1}{\beta}(\mathbf{Y}_k)_{12} \\ \frac{1}{\beta}(\mathbf{Y}_k)_{21} & \frac{1}{\beta}(\mathbf{Y}_k)_{22} - \mathcal{P}(\mathbf{M}) \end{pmatrix} + \frac{1}{\beta}\mathbf{A}_l^T\mathbf{B}_l
\end{aligned}$$

From property (41) of the adjoint operator $\mathcal{B}^*$, we can get

$$\mathcal{B}^*\begin{pmatrix} \frac{1}{\beta}(\mathbf{Y}_k)_{11} + \mathbf{X}_{k+1} & \frac{1}{\beta}(\mathbf{Y}_k)_{12} \\ \frac{1}{\beta}(\mathbf{Y}_k)_{21} & \frac{1}{\beta}(\mathbf{Y}_k)_{22} - \mathcal{P}(\mathbf{M}) \end{pmatrix} = -\left(\frac{1}{\beta}(\mathbf{Y}_k)_{11} + \mathbf{X}_{k+1}\right) + \mathcal{P}\left(\frac{1}{\beta}(\mathbf{Y}_k)_{22} - \mathbf{M}\right).$$

Thus,

$$\mathcal{B}^*\mathcal{B}(\mathbf{W}) = -\left[-\left(\frac{1}{\beta}(\mathbf{Y}_k)_{11} + \mathbf{X}_{k+1}\right) + \mathcal{P}\left(\frac{1}{\beta}(\mathbf{Y}_k)_{22} - \mathbf{M}\right)\right] + \frac{1}{\beta}\mathbf{A}_l^T\mathbf{B}_l. \tag{51}$$

Based on (50) and (51), we obtain

$$\mathbf{W}_{k+1} = \frac{1}{2\beta}\mathcal{P}\left[\beta(\mathbf{M}-\mathbf{X}_{k+1}) - (\mathbf{A}_l^T\mathbf{B}_l + (\mathbf{Y}_k)_{11} + (\mathbf{Y}_k)_{22})\right] + \mathbf{X}_{k+1} + \frac{1}{\beta}(\mathbf{A}_l^T\mathbf{B}_l + (\mathbf{Y}_k)_{11}). \tag{52}$$

Computing $\mathbf{Y}_{k+1}$. The update of Lagrange multipliers is the same as (45).

Algorithm 4 Inner Optimization by ADMMAP

Input: $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{M}$, and tolerance ϵ are given.

Initialization: $\mathbf{X}_1 = \mathbf{M}$, $\mathbf{W}_1 = \mathbf{X}_1$, $\kappa = 10^{-3}$, $\mathbf{Y}_1 = \mathcal{A}(\mathbf{X}_1)$.

repeat

step 1. $\mathbf{X}_{k+1} = \mathcal{D}_{\frac{1}{\beta}}\left(\mathbf{W}_k - \frac{1}{\beta}(\mathbf{Y}_k)_{11}\right)$.

step 2. $\mathbf{W}_{k+1} = \frac{1}{2\beta}\mathcal{P}\left[\beta(\mathbf{M} - \mathbf{X}_{k+1}) - (\mathbf{A}_l^T\mathbf{B}_l + (\mathbf{Y}_k)_{11} + (\mathbf{Y}_k)_{22})\right]$
$+\mathbf{X}_{k+1} + \frac{1}{\beta}(\mathbf{A}_l^T\mathbf{B}_l + (\mathbf{Y}_k)_{11})$.

step 3. $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \beta[\mathcal{A}(\mathbf{X}_{k+1}) + \mathcal{B}(\mathbf{W}) - \mathcal{C}]$.

step 4. Update β_k by (53) and (54)

until $\|\mathbf{X}_{k+1} - \mathbf{X}_k\|_F \leq \epsilon$.

3.4.3 Adapive penalty

For a fixed penalty parameter, if it is chosen to be too small or too large, the computational cost will increase significantly. At the same time, finding the optimal penalty parameter is difficult. Therefore, dynamic adjustment of penalty parameters may be better in practical applications. The adaptive update strategy adopted is as follows:

$$\beta_{k+1} = \min(\beta_{\max}, \rho\beta_k), \tag{53}$$

where $\beta_{\max}$ is an upper bound of β_k. The value of ρ is defined as

$$\rho = \begin{cases} \rho_0, & \text{if } \dfrac{\beta_k \max\{\|\mathbf{X}_{k+1} - \mathbf{X}_k\|_F\}}{\|\mathcal{C}\|_F} < \kappa \\ 1, & \text{otherwise,} \end{cases} \tag{54}$$

where $\rho > 1$ is a constant and $\kappa > 0$ is a threshold chosen in advance. When the difference between $(\mathbf{X}_{k+1}, \mathbf{W}_{k+1})$ and $(\mathbf{X}_k, \mathbf{W}_k)$ is small enough, β_{k+1} increases to $\rho_0\beta_k$ and the convergence rate is improved.

The procedures of TNNR-ADMMAP are summarized in Algorithm 4.

4 Experimental results

In this section, we conduct experiments on synthetic data and real visual data, and compare six matrix completion algorithms: SVT, SVP, OptSapce, TNNR-ADMM, TNNR-APGL, and TNNR-ADMMAP.

4.1 Experimental setup

Our project code runs in the experimental environment, software and hardware configurations shown in the table 1, and requires the installation of related dependencies.

[2] `https://code.google.com/archive/p/yamlmatlab/`

[3] `https://www.mathworks.com/matlabcentral/fileexchange/27991-tight_subplot-nh-nw-gap-marg_h-marg_w?s_tid=srchtitle_tight_subplot_1`

表 1: The experimental setup

Category	Name	Version
System	Windows 10	22H2
Hardware	CPU	Intel i7-11800H
Software	Matlab	R2019b
Dependencies	yamlmatlab[2]	0.4.3
	tight_subplot[3]	1.1.0.0

4.2 Synthetic data

We generate $m \times n$ matrices of rank r_0 by sampling two matrices, i.e., $\mathbf{M}_L \in \mathbb{R}^{m \times r_0}$ and $\mathbf{M}_R \in \mathbb{R}^{r_0 \times m}$, each having i.i.d. Gaussian entries, and setting $\mathbf{M} = \mathbf{M}_l \mathbf{M}_R$. The localtions of observed indices are sampled uniformly at random. Let p be the percentage of observed entries over $m \times n$. We generate synthetic data as follows:

$$\mathbf{B} = \mathbf{M} + \sigma \mathbf{Z},\ \mathbf{B}_{ij} = \mathbf{M}_{ij} + \sigma \mathbf{Z}_{ij},\ (i,j) \in\ ,$$

where $\mathbf{Z} \sim \mathcal{N}(0,1)$. Denote the full noiseless matrix as $\mathbf{X}_{full}$ and the solution given by an algorithm as $\mathbf{X}_{sol}$, define a commonly used criterion in matrix completion:

$$\text{total reconstruction error} = \|\mathcal{P}_c(\mathbf{X}_{sol} - \mathbf{X}_{full})\|_F.$$

Denote different matrix ranks as r_0 and differnet nosie levels as σ. We compare the reconstruction error of the six methods under different setting:

- the matrix size $m = 100$, $n = 200$, $r_0 = 10$. Fig. **??** shows the results under different noise levels and different observed ratios.
- the matrix size $m = 100$, $n = 200$, $\sigma = 0.5$. Fig. **??** shows the results under different ranks and different observed ratios.

As shown in Fig. **??**, under different noise levels and different observed ratios, compared with the original image:

- Our reproduced (abbreviated as our)SVT algorithm has better overall reconstruction error performance.

- Our SVP and OptSpace performs better and has similar performance to TNNR-APGL.
- Our TNNR-ADMM performs similarly to the original image only at low noise levels. However, as the noise level increases, the performance deteriorates faster.
- Our TNNR-APGL has a similar performance to the original image.

As shown in Fig. **??**, under different observed ratios, as the matrix rank increases, the total reconstruction error becomes larger. Compared with the original image:

- Our SVT algorithm has better overall reconstruction error performance at low ranks, and our implementation of the SVT algorithm deteriorates more rapidly as the matrix rank increases.
- Our SVP and OptSpace performs better and has similar performance to TNNR-APGL.
- The total reconstruction error of our TNNR-ADMM and TNNR-ADMMAP is relatively large, and the results that are not close to the original work have been reached.
- Our TNNR-APGL has a similar performance to the original image.

4.2.1 Parameter setting

In our experiments, we readjusted the parameters of the six algorithms since the parameters given by the original author are incomplete and can not obtain good experimental results. Table 2 shows the parameters set by the original author and us.

4.2.2 Random mask

we randomly cover some pixels of an image (300×300), and then compare the recovery performance of the six algorithms.The results are shown in Fig. **??** and Fig. **??**. Naturally, the larger the observed ratio is the better recovery of the image that can be achieved. Form Fig. **??**, we can see that the TNNR-based algorithms can achieve higher PSNR values compared with the other three methods. Compare with the results in the original paper, our SVT recovers images better. And the results of TNNR-APGL are not as expected in the high observed ratio.

4.2.3 Text mask

Since the text is not random, it may cover important information in the picture, making the text more difficult to remove than pixels. Fig. **??** and Fig. **??** show the experimental results of the six matrix completion algorithms. Specifically, for the example image in

表 2: The parameters set by the original author and us

Parameter	Random mask	Text mask	Block mask	Paper set
p	0.87	0.8	0.8	
δ_{2k}	0.2	0.2	0.2	
τ	10^{-3}	10^{-3}	10^{-2}	
β	10^{-3}	10^{-3}	10^{-3}	10^{-3}
λ	10^{-2}	10^{-3}	10^{-3}	0.06
β_{max}	10^{10}	10^{10}	10^{10}	10^{10}
κ	10^{-3}	10^{-3}	10^{-3}	10^{-3}
ρ_0	1.1	1.01	1.01	1.9
ϵ_0	10^{-3}	10^{-3}	10^{-3}	10^{-3}
ϵ	10^{-4}	10^{-4}	10^{-4}	10^{-4}
rank	5	$3 \sim 20$	3	$1 \sim 20$
$\text{iter}_{\max}$	10^2	$10^2/10^3$	$10^2/10^3$	

Fig. **??**, the PSNR values for SVT, SVP, OptSpace, TNNR- ADMM, TNNR-APGL, and TNNR-ADMMAP are 23.47, 23.47, 22.43, 25.26, 25.26, and 25.27, respectively. And for the example image in Fig. **??**, the PSNR values for SVT, SVP, OptSpace, TNNR-ADMM, TNNR-APGL, and TNNR-ADMMAP are 21.61, 21.61, 21.95, 23.94, 23.16, and 23.95, respectively. From these results we can see that TNNR-ADMM and TNNR-ADMMAP algorithms achieve better performance than the other three matrix completion algorithms. Our TNNR-APGL cannot obtain the same image restoration effect as the original paper.

4.2.4 Block mask

In some situations, Some pixels in the image have been broken. We first mask all these pixels, then we use six matrix completion algorithms to recover these large missing blocks. Fig. **??** show the recovered images by using the six different matrix completion algorithms. Similar to the text mask, our TNNR-ADMM and TNNR-ADMMAP can recover the image better, but the the image recovered by our TNNR-APGl loses a lot of texture information.

表 3: The total time required for the six algorithms in text mask experiment

Algorithm	Fig. ??		Fig. ??	
	Total time (s)	Best rank	Total time (s)	Best rank
SVT	60.410	no need	25.128	no need
SVP	1856.893	15	253.012	11
OptSpace	1856.893	15	253.012	13
TNNR-ADMM	371.048	15	314.183	6
TNNR-APGL	1175.097	3	1509.657	3
TNNR-ADMMAP	759.241	15	570.614	6

4.3 Convergence rate and computational cost

In the experiments in Fig. **??**, we also tested the total number of iterations for the TNNR algorithm to recover images under different observed ratios. First, from Fig. **??**, we observe that in low observed ratio, the three TNNR-based algorithms have close PSNR again, while in high observed ratio, TNNR-APGL has lower PSNR. As shown in Fig. **??**, TNNR-ADMMAP converges much faster than TNNR-APGL and TNNR-ADMM. The total number of iterations needed for TNNR-ADMMAP to converge is roughly half of that for TNNR-APGL and TNNR-ADMM, while They obtain almost the same result.

Considering that the time cost of each iteration of the three algorithms is slightly different, in the text mask experiment, we tested the total time required for the six algorithms to recover the image, and the rank of the best PSNR , as shown in the Table 3. The time required for SVT is the shortest. Among the three TNNR-based algorithms, TNNR-ADMM takes the shortest time, and TNNR-APGL takes the longest time. Different from the original work, our implementation of the TNNR-ADMMAP algorithm does not show the convergence speed of the original paper, which is slower than the TNNR-ADMM algorithm, but faster than TNNR-APGL.

5 Conclusion

In this report, we replicate the paper by Hu et al.[1]. This paper proposes a new matrix completion method called truncated nuclear norm regularization. This approach tries to minimize the summation of the smallest $\min(m, n) - r$ singular values, where r is the matrix rank. Moreover, they also designed a two-step iterative solution, and performed three optimizations based on ADMM, APGL, and ADMMAP for the second iteration.

We conduct similar experiments on synthetic and real visual datasets and compare with the original results. The experimental results demonstrate the advantages of the TNNR-based algorithms in comparison with three matrix completion methods based on the nuclear norm or matrix factorization. However, there are some differences between our reproduction results and the results shown in the author's paper. Because the author's parameters cannot play well in this reproduction, the reason may be that our parameters are not adjusted to the best, or the author may have preprocessed the image.

In this coursework project, we have thoroughly studied and applied matrix theory for the first time, guided practice by theory, and further mastered knowledge. Moreover, through the reproduction of this paper, we generally have a deeper and clearer understanding of low-rank matrix approximation and matrix completion, and used two methods for solve separable convex optimization: ADMM and APGL. In the whole process of learning and practice, we have benefited a lot.

References

[1] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and Accurate Matrix Completion via Truncated Nuclear Norm Regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] Jian-Feng Cai, Emmanuel J. Candes, and Zuowei Shen. A Singular Value Thresholding Algorithm for Matrix Completion.

[5] Prateek Jain, Raghu Meka, and Inderjit Dhillon. Guaranteed Rank Minimization via Singular Value Projection. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[6] Raghunandan H. Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries. In *2009 IEEE International Symposium on Information Theory*, pages 324–328, 2009.

[7] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

[8] John Von Neumann. *Some matrix-inequalities and metrization of matric space*. Pergamon Press, 1937.

[9] Bingsheng He, Min Tao, and Xiaoming Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, 22(2):313–340, 2012.

[10] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. *Advances in neural information processing systems*, 24, 2011.

A Appendix

proof Theorem 3.1

By Von Neumann's trace inequality, we get

$$\begin{aligned} \mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) &= \mathrm{Tr}(\mathbf{X}\mathbf{B}^T\mathbf{A}) \\ &\leq \sum_{i=1}^{\min(m,n)} \sigma_i(\mathbf{X})\sigma_i(\mathbf{B}^T\mathbf{A}), \end{aligned} \tag{55}$$

where $\sigma_1(\mathbf{X}) \geq \cdots \geq \sigma_{min(m,n)}(\mathbf{X}) \geq 0$. As $\mathrm{rank}(\mathbf{A}) = r$ and $\mathrm{rank}(\mathbf{B}) = r$, so $\mathrm{rank}(\mathbf{B}^T\mathbf{A}) = s \leq r$. For $i \leq s$, $\sigma_i(\mathbf{B}^T\mathbf{A}) \geq 0$ and $\sigma_i(\mathbf{B}^T\mathbf{A})$ is the ith eigenvalue of $\mathbf{B}\mathbf{B}^T = I$. Therefore, $\sigma_i(\mathbf{B}^T\mathbf{A}) = 1$, for $i = 1, 2, \ldots, s$, and the rest are all 0s. It follows that:

$$\begin{aligned} \sum_{i=1}^{\min(m,n)} \sigma_i(\mathbf{X})\sigma_i(\mathbf{B}^T\mathbf{A}) &= \sum_{i=1}^{s} \sigma_i(\mathbf{X})\sigma_i(\mathbf{B}^T\mathbf{A}) + \sum_{i=s+1}^{\min(m,n)} \sigma_i(\mathbf{X})\sigma_i(\mathbf{B}^T\mathbf{A}) \\ &= \sum_{i=1}^{s} \sigma_i(\mathbf{X}) \cdot 1 + \sum_{i=s+1}^{\min(m,n)} \sigma_i(\mathbf{X}) \cdot 0 \\ &= \sum_{i=1}^{s} \sigma_i(\mathbf{X}). \end{aligned} \tag{56}$$

Since $s \leq r$ and $\sigma_i(\mathbf{X}) \geq 0$:

$$\sum_{i=1}^{s} \sigma_i(\mathbf{X}) \leq \sum_{i=1}^{r} \sigma_i(\mathbf{X}).$$

Combining inequalities and ,we have

$$\mathrm{Tr}(\mathbf{A}\mathbf{X}\mathbf{B}^T) \leq \sum_{i=1}^{s} \sigma_i(\mathbf{X}) \leq \sum_{i=1}^{r} \sigma_i(\mathbf{X}). \tag{57}$$

proof Equation 31

Fix $\mathbf{W}_k$ and $\mathbf{Y}_k$,

$$\mathbf{X}_{k+1} = \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* - \mathrm{Tr}(\mathbf{A}_l\mathbf{W}_k\mathbf{B}_l^T) + \frac{\beta}{2}\|\mathbf{X} - \mathbf{W}_k\|_F^2 + \mathrm{Tr}(\mathbf{Y}_k(\mathbf{X} - \mathbf{W}_k)). \tag{58}$$

For the last two terms of (58),

$$\frac{\beta}{2}\|\mathbf{X} - \mathbf{W}_k\|_F^2 + \mathrm{Tr}(\mathbf{Y}_k^T(\mathbf{X} - \mathbf{W}_k)) = \frac{\beta}{2}\left(\|\mathbf{X} - \mathbf{W}_k\|_F^2 + \frac{2}{\beta}\mathrm{Tr}(\mathbf{Y}_k^T(\mathbf{X} - \mathbf{W}_k))\right). \tag{59}$$

Because

$$\begin{aligned}\|\mathbf{X}-\mathbf{W}_k+\frac{1}{\beta}\mathbf{Y}_k\|_F^2 &= \mathrm{Tr}\ (\mathbf{X}-\mathbf{W}_k+\frac{1}{\beta}\mathbf{Y}_k)(\mathbf{X}-\mathbf{W}_k+\frac{1}{\beta}\mathbf{Y}_k)^T \\ &= \mathrm{Tr}(\mathbf{X}-\mathbf{W}_k)(\mathbf{X}-\mathbf{W}_k)^T+2\ \mathrm{Tr}(\mathbf{X}-\mathbf{W}_k)\frac{1}{\beta}\mathbf{Y}_k^T+\frac{1}{\beta^2}\mathrm{Tr}(\mathbf{Y}_k Y_k^T) \\ &= \|\mathbf{X}-\mathbf{W}_k\|_F^2+\frac{1}{\beta^2}\|\mathbf{Y}_k\|_F^2+\frac{2}{\beta}\mathrm{Tr}(\mathbf{Y}_k^T(\mathbf{X}-\mathbf{W}_k)).\end{aligned}$$

For the latter term in parentheses of (59),

$$\begin{aligned}\frac{2}{\beta}\mathrm{Tr}(\mathbf{Y}_k^T(\mathbf{X}-\mathbf{W}_k)) &= 2\ \mathrm{Tr}(\frac{1}{\beta}\mathbf{Y}_k^T(\mathbf{X}-\mathbf{W}_k)) \\ &= \|\mathbf{X}-\mathbf{W}_k+\frac{1}{\beta}\mathbf{Y}_k\|_F^2-\frac{1}{\beta^2}\|\mathbf{Y}_k\|_F^2-\|\mathbf{X}-\mathbf{W}_k\|_F^2.\end{aligned} \tag{60}$$

Ignoring all constant terms,

$$\begin{aligned}\mathbf{X}_{k+1} &= \arg\min_{\mathbf{X}}\ L(\mathbf{X},\mathbf{Y}_k,\mathbf{W}_k,\beta) \\ &= \arg\min_{\mathbf{X}}\ \|\mathbf{X}\|_*-\mathrm{Tr}(\mathbf{A}_l\mathbf{W}_k\mathbf{B}_l^T)+\frac{\beta}{2}\|\mathbf{X}-\mathbf{W}_k\|_F^2+\mathrm{Tr}(\mathbf{Y}_k^T(\mathbf{X}-\mathbf{W}_k)) \\ &= \arg\min_{\mathbf{X}}\ \|\mathbf{X}\|_*+\frac{\beta}{2}\|\mathbf{X}-\left(\mathbf{W}_k-\frac{1}{\beta}\mathbf{Y}_k\right)\|_F^2-\frac{1}{2\beta}\|\mathbf{Y}_k\|_F^2 \\ &= \arg\min_{\mathbf{X}}\ \|\mathbf{X}\|_*+\frac{\beta}{2}\|\mathbf{X}-\left(\mathbf{W}_k-\frac{1}{\beta}\mathbf{Y}_k\right)\|_F^2.\end{aligned} \tag{61}$$

That is

$$\mathbf{X}_{k+1}=\mathcal{D}_{\frac{1}{\rho}}(\mathbf{W}_k-\frac{1}{\rho}\mathbf{Y}_k). \tag{62}$$