

# Low-Rank Matrix Restoration

## 《矩阵分析》综合课题研究

杨敬轩, 董泽委

系统工程研究所, 智能交通实验室

2022 年 11 月 28 日



# 目录

① 引言

② 鲁棒主成分分析: 通用算  
法

③ 鲁棒主成分分析: 特殊算  
法

④ 矩阵补全

⑤ 实验结果分析

⑥ 代码说明

⑦ 贡献说明

⑧ 参考文献





# 引言

## 低秩矩阵恢复

- 在实际应用中, 若出现较高幅度的尖锐噪声或严重的离群点时, PCA 的性能会受到很大的影响.
- 当噪声矩阵  $\mathbf{S}$  足够稀疏时, 原始低秩矩阵仍然可能被恢复.
- 该任务可以建模为如下优化问题:

$$\begin{aligned} & \min_{\mathbf{L}, \mathbf{S}} \text{rank}(\mathbf{L}) + \lambda \|\mathbf{S}\|_0 \\ & \text{s.t. } \mathbf{M} = \mathbf{L} + \mathbf{S}, \end{aligned} \tag{2}$$

其中,  $\|\mathbf{S}\|_0$  指矩阵中非零元素的个数.

- 由于  $\text{rank}(\mathbf{L})$  和  $\|\mathbf{S}\|_0$  都是非凸的, 直接优化该问题非常困难, 因此需要对其进行凸松弛.



# 引言

## 主要贡献

- 鲁棒主成分分析: 通用算法
  - Gradient descent
  - Gradient descent with Adam
- 鲁棒主成分分析: 特殊算法
  - Singular value thresholding
  - Accelerated proximal gradient
  - Augmented Lagrange multipliers
- 矩阵补全
  - Singular value thresholding
  - Alternating direction method
  - Alternating least squares
  - Neural tangent kernels

# 引言

## GitHub Repository

- Low-Rank Matrix Restoration

<https://github.com/jingxuanyang/LowRankMatrixRestoration>

- 报告: doc/report.pdf
- 展示: pre/slide.pdf
- 说明: README.md

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

## 1 引言

## 2 鲁棒主成分分析: 通用算法

- Gradient descent
- Gradient descent with Adam

## 3 鲁棒主成分分析: 特殊算法

## 4 矩阵补全

## 5 实验结果分析

## 6 代码说明

## 7 贡献说明

## 8 参考文献

# 鲁棒主成分分析: 通用算法

## Gradient descent

- 梯度下降算法是一种求解优化问题的通用方法.
- 由式 (3) 给出的 RPCA 问题的优化目标中包含核范数和  $m_1$  范数两项, 而这两项均不可导, 则需使用其次梯度 (Subgradient) [3] 进行梯度下降.
- 对式 (3) 进行变形可得

$$\min_{\mathbf{L}} f(\mathbf{L}) = \|\mathbf{L}\|_* + \lambda \|\mathbf{M} - \mathbf{L}\|_{m_1}. \quad (4)$$



# 鲁棒主成分分析: 通用算法

梯度下降算法如 Algorithm 1 所示.

---

## Algorithm 1: Gradient descent for RPCA.

---

**Input:** Data matrix  $M$ , learning rate  $\alpha$ , max iterations  $T$ , tolerance  $\tau$

**Output:** Low-rank matrix  $L$ , sparse noise matrix  $S$

1 initialize  $t = 0$ ,  $L = O$ , terminate = False;

2 **while** not terminate and  $t < T$  **do**

3      $t \leftarrow t + 1$ ;

4      $G \leftarrow \text{Subgradient}(L)$  by Eq. (7);

5      $L' \leftarrow L - \alpha G$ ;

6     terminate  $\leftarrow |f(L') - f(L)| < \tau$ ;

7      $L \leftarrow L'$ ;

8 **end**

9  $S \leftarrow M - L$ ;

10 **return**  $L, S$ ;

---

## 1 引言

## 2 鲁棒主成分分析: 通用算法

- Gradient descent
- Gradient descent with Adam

## 3 鲁棒主成分分析: 特殊算法

## 4 矩阵补全

## 5 实验结果分析

## 6 代码说明

## 7 贡献说明

## 8 参考文献

# 鲁棒主成分分析: 通用算法

采用 Adam [6] 算法进行优化, 如 Algorithm 2 所示.

---

**Algorithm 2:** Gradient descent with Adam for RPCA.

**Input:** Data matrix  $M$ , learning rate  $\alpha$ , max iterations  $T$ , tolerance  $\tau$

**Output:** Low-rank matrix  $L$ , sparse noise matrix  $S$

```
1 set  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ ;  
2 initialize  $t = 0$ ,  $L = O$ , terminate = False,  $W_0 = V_0 = O$ ;  
3 while not terminate and  $t < T$  do  
4    $t \leftarrow t + 1$ ,  $G \leftarrow$  Subgradient( $L$ ) by Eq. (7);  
5    $W_t \leftarrow \beta_1 W_{t-1} + (1 - \beta_1)G$ ,  $V_t \leftarrow \beta_2 V_{t-1} + (1 - \beta_2)G \odot G$ ;  
6    $\hat{W}_t \leftarrow W_t / (1 - \beta_1^t)$ ,  $\hat{V}_t \leftarrow V_t / (1 - \beta_2^t)$ ;  
7    $L' \leftarrow L - \alpha \hat{W}_t \oslash (\sqrt{\hat{V}_t} + \epsilon)$  /*  $\sqrt{\cdot}$  applied element-wise */;  
8   terminate  $\leftarrow |f(L') - f(L)| < \tau$ ,  $L \leftarrow L'$ ;  
9 end  
10  $S \leftarrow M - L$ ;  
11 return  $L$ ,  $S$ ;
```

---

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

multipliers

1 引言

2 鲁棒主成分分析: 通用算法

3 鲁棒主成分分析: 特殊算法

• Preliminaries

- Singular value thresholding
- Accelerated proximal gradient
- Augmented Lagrange

4 矩阵补全

5 实验结果分析

6 代码说明

7 贡献说明

8 参考文献

# 鲁棒主成分分析: 特殊算法

## Preliminaries

- 设  $\mathbf{Q} \in \mathbb{R}^{m \times n}$ , 则优化问题

$$\min_{\mathbf{X}} \epsilon \|\mathbf{X}\|_{m_1} + \frac{1}{2} \|\mathbf{X} - \mathbf{Q}\|_F^2 \quad (8)$$

的最优解为  $\mathbf{X}^* = \mathcal{S}_\epsilon(\mathbf{Q})$  [2].

- 其中  $\mathcal{S}_\epsilon$  为 Shrinkage 算子,

$$\mathcal{S}_\epsilon(\mathbf{Q}) = \max(|\mathbf{Q}| - \epsilon, 0) \odot \text{sgn}(\mathbf{Q}). \quad (9)$$

- 上述  $\max()$ ,  $\text{sgn}()$  运算均为按元素进行 (element-wise) 的运算.

# 鲁棒主成分分析: 特殊算法

## Preliminaries

- 将  $m_1$  范数替换为核范数, 则有

$$\min_{\mathbf{X}} \epsilon \|\mathbf{X}\|_* + \frac{1}{2} \|\mathbf{X} - \mathbf{Q}\|_F^2. \quad (10)$$

- 其闭式解为  $\mathbf{X}^* = \mathcal{D}_\epsilon(\mathbf{Q})$  [7],  $\mathcal{D}_\epsilon$  为 Singular value thresholding 算子.
- 记  $\mathbf{Q} = \mathbf{U}\Sigma\mathbf{V}^\top$  为矩阵  $\mathbf{Q}$  的 SVD 分解, 则

$$\mathcal{D}_\epsilon(\mathbf{Q}) = \mathbf{U}\mathcal{S}_\epsilon(\Sigma)\mathbf{V}^\top. \quad (11)$$

multipliers

1 引言

2 鲁棒主成分分析: 通用算法

3 鲁棒主成分分析: 特殊算法

- Preliminaries
- Singular value thresholding
- Accelerated proximal gradient
- Augmented Lagrange

4 矩阵补全

5 实验结果分析

6 代码说明

7 贡献说明

8 参考文献

# 鲁棒主成分分析: 特殊算法

## Singular value thresholding

- Singular value thresholding (SVT) [7] 是一种迭代阈值算法.
- 对式 (3) 进行正则可得

$$\begin{aligned} & \min_{\mathbf{L}, \mathbf{S}} \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_{m_1} + \mu (\|\mathbf{L}\|_F^2 + \|\mathbf{S}\|_F^2) \\ \text{s.t. } & \mathbf{M} = \mathbf{L} + \mathbf{S}, \end{aligned} \tag{12}$$

其中,  $\mu > 0$  为较小的正数.

- 上式的 Lagrange 函数为

$$\mathcal{L}(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_{m_1} + \mu (\|\mathbf{L}\|_F^2 + \|\mathbf{S}\|_F^2) + \langle \mathbf{Y}, \mathbf{M} - \mathbf{L} - \mathbf{S} \rangle. \tag{13}$$

# 鲁棒主成分分析: 特殊算法

## Singular value thresholding

- SVT 算法交替更新矩阵  $L, S, Y$ . 当  $S = S_t, Y = Y_t$  时,

$$L_{t+1} = \operatorname{argmin}_L \mathcal{L}(L, S_t, Y_t) = \mathcal{D}_{1/\mu}(Y_t/\mu). \quad (14)$$

- 当  $L = L_{t+1}, Y = Y_t$  时,

$$S_{t+1} = \operatorname{argmin}_S \mathcal{L}(L_{t+1}, S, Y_t) = \mathcal{S}_{\lambda/\mu}(Y_t/\mu). \quad (15)$$

- 当  $L = L_{t+1}, S = S_{t+1}$  时,

$$Y_{t+1} = Y_t + \alpha(M - L_{t+1} - S_{t+1}), \quad (16)$$

其中  $\alpha \in (0, 1)$  为迭代步长.

# 鲁棒主成分分析: 特殊算法

SVT 具体算法如 Algorithm 3 所示.

---

### Algorithm 3: Singular value thresholding for RPCA.

---

**Input:** Data matrix  $\mathbf{M}$ , learning rate  $\alpha$ , max iterations  $T$ , tolerance  $\tau$ , regularization  $\lambda, \mu$

**Output:** Low-rank matrix  $\mathbf{L}$ , sparse noise matrix  $\mathbf{S}$

- 1 initialize  $t = 0$ ,  $\mathbf{L} = \mathbf{S} = \mathbf{Y} = \mathbf{O}$ , terminate = False;
- 2 **while** not terminate and  $t < T$  **do**
- 3      $t \leftarrow t + 1$ ;
- 4      $\mathbf{L} \leftarrow \mathcal{D}_{1/\mu}(\mathbf{Y}/\mu)$ ;
- 5      $\mathbf{S} \leftarrow \mathcal{S}_{\lambda/\mu}(\mathbf{Y}/\mu)$ ;
- 6      $\mathbf{Y} \leftarrow \mathbf{Y} + \alpha(\mathbf{M} - \mathbf{L} - \mathbf{S})$ ;
- 7     terminate  $\leftarrow \|\mathbf{M} - \mathbf{L} - \mathbf{S}\|_F < \tau$ ;
- 8 **end**
- 9 **return**  $\mathbf{L}, \mathbf{S}$ ;

---

multipliers

## 1 引言

## 2 鲁棒主成分分析: 通用算法

## 3 鲁棒主成分分析: 特殊算法

- Preliminaries
- Singular value thresholding
- Accelerated proximal gradient
- Augmented Lagrange

## 4 矩阵补全

## 5 实验结果分析

## 6 代码说明

## 7 贡献说明

## 8 参考文献





# 鲁棒主成分分析: 特殊算法

APG 具体算法如 Algorithm 4 所示.

---

## Algorithm 4: Accelerated proximal gradient for RPCA.

---

**Input:** Data matrix  $\mathbf{M}$ , max iterations  $T$ , tolerance  $\tau$

**Output:** Low-rank matrix  $\mathbf{L}$ , sparse noise matrix  $\mathbf{S}$

- 1 set  $L_f = 2$ ,  $\mu_0 = 0.99\|\mathbf{M}\|_2$ ,  $\hat{\mu} = 10^{-6}\mu_0$ ,  $\eta = 0.9$ ;
  - 2 initialize  $t = 0$ ,  $\mathbf{L}_t = \mathbf{S}_t = \mathbf{Y}_t = \mathbf{Y}'_t = \mathbf{O}$ ,  $\alpha_t = 1$ , terminate = False;
  - 3 **while** not terminate and  $t < T$  **do**
  - 4     $\mathbf{L}_{t+1} \leftarrow \mathcal{D}_{\mu_t/L_f}(\mathbf{Y}_t + (\mathbf{M} - \mathbf{Y}_t - \mathbf{Y}'_t)/L_f);$
  - 5     $\mathbf{S}_{t+1} \leftarrow \mathcal{S}_{\mu_t/L_f}(\mathbf{Y}'_t + (\mathbf{M} - \mathbf{Y}_t - \mathbf{Y}'_t)/L_f)$ ,  $\alpha_{t+1} \leftarrow \frac{1+\sqrt{1+4\alpha_t^2}}{2};$
  - 6     $\mathbf{Y}_{t+1} \leftarrow \mathbf{L}_t + \frac{\alpha_t-1}{\alpha_{t+1}}(\mathbf{L}_t - \mathbf{L}_{t+1})$ ,  $\mathbf{Y}'_{t+1} \leftarrow \mathbf{S}_t + \frac{\alpha_t-1}{\alpha_{t+1}}(\mathbf{S}_t - \mathbf{S}_{t+1});$
  - 7     $\mu_{t+1} \leftarrow \max(\eta\mu_t, \hat{\mu})$ ,  $t \leftarrow t + 1$ ;
  - 8    terminate  $\leftarrow \|\mathbf{M} - \mathbf{L}_t - \mathbf{S}_t\|_F < \tau$ ;
  - 9 **end**
  - 10 return  $\mathbf{L}_t$ ,  $\mathbf{S}_t$ ;
-

## multipliers

### 1 引言

### 2 鲁棒主成分分析: 通用算法

### 3 鲁棒主成分分析: 特殊算法

- Preliminaries
- Singular value thresholding
- Accelerated proximal gradient
- Augmented Lagrange

### 4 矩阵补全

### 5 实验结果分析

### 6 代码说明

### 7 贡献说明

### 8 参考文献

# 鲁棒主成分分析: 特殊算法

## Augmented Lagrange multipliers

- 对式 (3) 构造增广 Lagrange 函数为

$$\mathcal{L}(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_{m_1} + \frac{\mu}{2} \|\mathbf{M} - \mathbf{L} - \mathbf{S}\|_F^2 + \langle \mathbf{Y}, \mathbf{M} - \mathbf{L} - \mathbf{S} \rangle. \quad (23)$$

- Augmented Lagrange multipliers (ALM) [2, 9, 10] 算法交替更新矩阵  $\mathbf{L}$ ,  $\mathbf{S}$ ,  $\mathbf{Y}$ .

# 鲁棒主成分分析: 特殊算法

## Augmented Lagrange multipliers

- 当  $\mathbf{S} = \mathbf{S}_t$ ,  $\mathbf{Y} = \mathbf{Y}_t$  时,

$$\mathbf{L}_{t+1} = \mathcal{D}_{1/\mu}(\mathbf{M} - \mathbf{S}_t + \mathbf{Y}_t/\mu). \quad (24)$$

- 当  $\mathbf{L} = \mathbf{L}_{t+1}$ ,  $\mathbf{Y} = \mathbf{Y}_t$  时,

$$\mathbf{S}_{t+1} = \mathcal{S}_{\lambda/\mu}(\mathbf{M} - \mathbf{L}_{t+1} + \mathbf{Y}_t/\mu). \quad (25)$$

- 当  $\mathbf{L} = \mathbf{L}_{t+1}$ ,  $\mathbf{S} = \mathbf{S}_{t+1}$  时,

$$\mathbf{Y}_{t+1} = \mathbf{Y}_t + \mu(\mathbf{M} - \mathbf{L}_{t+1} - \mathbf{S}_{t+1}). \quad (26)$$

# 鲁棒主成分分析:特殊算法

ALM 具体算法如 Algorithm 5 所示.

---

## Algorithm 5: Augmented Lagrange multipliers for RPCA.

---

**Input:** Data matrix  $\mathbf{M}$ , max iterations  $T$ , tolerance  $\tau$

**Output:** Low-rank matrix  $\mathbf{L}$ , sparse noise matrix  $\mathbf{S}$

```
1 set  $\lambda = 1/\sqrt{\max(m, n)}$ ,  $\mu = mn/(4\|\mathbf{M}\|_1)$ ;  
2 initialize  $t = 0$ ,  $\mathbf{L} = \mathbf{S} = \mathbf{Y} = \mathbf{O}$ , terminate = False;  
3 while not terminate and  $t < T$  do  
4      $t \leftarrow t + 1$ ;  
5      $\mathbf{L} \leftarrow \mathcal{D}_{1/\mu}(\mathbf{M} - \mathbf{S} + \mathbf{Y}/\mu)$ ;  
6      $\mathbf{S} \leftarrow \mathcal{S}_{\lambda/\mu}(\mathbf{M} - \mathbf{L} + \mathbf{Y}/\mu)$ ;  
7      $\mathbf{Y} \leftarrow \mathbf{Y} + \mu(\mathbf{M} - \mathbf{L} - \mathbf{S})$ ;  
8     terminate  $\leftarrow \|\mathbf{M} - \mathbf{L} - \mathbf{S}\|_F < \tau$ ;  
9 end  
10 return  $\mathbf{L}, \mathbf{S}$ ;
```

---

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

# 矩阵补全

## Problem Formulation

- 当数据矩阵  $M$  含丢失元素时, 可根据矩阵的低秩结构来恢复矩阵的所有元素, 此恢复过程称为矩阵补全 (Matrix Completion) [11].
- 记  $\Omega$  为观测数据的指标集合, 则矩阵补全可以建模为

$$\begin{aligned} & \min_{\mathcal{L}} \operatorname{rank}(\mathcal{L}) \\ & \text{s.t. } \mathcal{P}_{\Omega}(\mathcal{L}) = \mathcal{P}_{\Omega}(M), \end{aligned} \tag{27}$$

其中  $\mathcal{P}_{\Omega}$  为投影算子, 对  $M = [m_{i,j}]$  有

$$\mathcal{P}_{\Omega}(m_{i,j}) = \begin{cases} m_{i,j}, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases} \tag{28}$$

# 矩阵补全

## Problem Formulation

- 令  $P_\Omega = \mathcal{P}_\Omega(\mathbf{1}\mathbf{1}^\top)$ , 则约束条件  $\mathcal{P}_\Omega(\mathbf{L}) = \mathcal{P}_\Omega(\mathbf{M})$  可以重写为  $P_\Omega \odot \mathbf{L} = P_\Omega \odot \mathbf{M}$ , 其中  $\odot$  表示矩阵按元素相乘.
- 将矩阵的秩松弛到矩阵的核范数, 即得到凸优化问题:

$$\begin{aligned} & \min_{\mathbf{L}} \|\mathbf{L}\|_* \\ \text{s.t. } & P_\Omega \odot \mathbf{L} = P_\Omega \odot \mathbf{M}. \end{aligned} \tag{29}$$

- 上述问题可以重新表示为

$$\begin{aligned} & \min_{\mathbf{L}} \|\mathbf{L}\|_* \\ \text{s.t. } & P_\Omega \odot \mathbf{S} = \mathbf{O}, \\ & \mathbf{M} = \mathbf{L} + \mathbf{S}. \end{aligned} \tag{30}$$

## 1 引言

method

- Alternating least squares
- Neural tangent kernels

## 2 鲁棒主成分分析: 通用算法

## 5 实验结果分析

## 3 鲁棒主成分分析: 特殊算法

## 6 代码说明

## 4 矩阵补全

- Singular value thresholding
- Alternating direction

## 7 贡献说明

## 8 参考文献

# 矩阵补全

## Singular value thresholding

- 第 3.2 小节中使用的 Singular value thresholding (SVT) [7] 算法也可以用于矩阵补全.
- 对式 (29) 进行正则可得

$$\begin{aligned} \min_{\mathbf{L}} \quad & \|\mathbf{L}\|_* + \frac{\mu}{2} \|\mathbf{L}\|_F^2 \\ \text{s.t.} \quad & \mathbf{P}_\Omega \odot (\mathbf{M} - \mathbf{L}) = \mathbf{O}, \end{aligned} \tag{31}$$

其中,  $\mu > 0$  为较小的正数.

- 上式的 Lagrange 函数为

$$\mathcal{L}(\mathbf{L}, \mathbf{Y}) = \|\mathbf{L}\|_* + \frac{\mu}{2} \|\mathbf{L}\|_F^2 + \langle \mathbf{Y}, \mathbf{P}_\Omega \odot (\mathbf{M} - \mathbf{L}) \rangle. \tag{32}$$

# 矩阵补全

## Singular value thresholding

- SVT 算法交替更新矩阵  $L$ ,  $Y$ .
- 当  $Y = Y_t$  时,

$$L_{t+1} = \mathcal{D}_{1/\mu}(Y_t). \quad (33)$$

- 当  $L = L_{t+1}$  时,

$$Y_{t+1} = Y_t + \alpha P_\Omega \odot (M - L_{t+1}), \quad (34)$$

其中  $\alpha \in (0, 1)$  为迭代步长.

# 矩阵补全

SVT 具体算法如 Algorithm 6 所示.

---

## Algorithm 6: Singular value thresholding for matrix completion.

---

**Input:** Data matrix  $M$ , mask matrix  $P_\Omega$ , learning rate  $\alpha$ , max iterations  $T$ , tolerance  $\tau$ , regularization  $\mu$

**Output:** Low-rank matrix  $L$ , sparse noise matrix  $S$

```
1 set  $\alpha = 1.2mn/\|P_\Omega\|_F^2$ ,  $\mu = 5(m+n)/2$ ;  
2 initialize  $t = 0$ ,  $L = Y = O$ , terminate = False;  
3 while not terminate and  $t < T$  do  
4      $t \leftarrow t + 1$ ;  
5      $L \leftarrow \mathcal{D}_{1/\mu}(Y)$ ;  
6      $Y \leftarrow Y + \alpha P_\Omega \odot (M - L)$ ;  
7     terminate  $\leftarrow \|P_\Omega \odot (M - L)\|_F / \|P_\Omega \odot M\|_F < \tau$ ;  
8 end  
9  $S = M - L$ ;  
10 return  $L, S$ ;
```

---

## 1 引言

### method

- Alternating least squares
- Neural tangent kernels

## 2 鲁棒主成分分析: 通用算法

## 5 实验结果分析

## 3 鲁棒主成分分析: 特殊算法

## 6 代码说明

## 4 矩阵补全

- Singular value thresholding
- Alternating direction

## 7 贡献说明

## 8 参考文献

# 矩阵补全

## Alternating direction method

- 对式 (30) 构造增广 Lagrange 函数为

$$\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \frac{\mu}{2} \|\boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S}\|_F^2 + \langle \boldsymbol{Y}, \boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S} \rangle. \quad (35)$$

- Alternating direction method (ADMM) [10] 算法交替更新矩阵  $\boldsymbol{L}$ ,  $\boldsymbol{S}$ ,  $\boldsymbol{Y}$ . 当  $\boldsymbol{S} = \boldsymbol{S}_t$ ,  $\boldsymbol{Y} = \boldsymbol{Y}_t$  时,

$$\boldsymbol{L}_{t+1} = \mathcal{D}_{1/\mu}(\boldsymbol{M} - \boldsymbol{S}_t + \boldsymbol{Y}_t/\mu). \quad (36)$$

- 当  $\boldsymbol{L} = \boldsymbol{L}_{t+1}$ ,  $\boldsymbol{Y} = \boldsymbol{Y}_t$  时,

$$\boldsymbol{S}_{t+1} = \mathcal{P}_\Omega \odot (\boldsymbol{M} - \boldsymbol{L}_{t+1} + \boldsymbol{Y}_t/\mu). \quad (37)$$

- 当  $\boldsymbol{L} = \boldsymbol{L}_{t+1}$ ,  $\boldsymbol{S} = \boldsymbol{S}_{t+1}$  时,  $\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t + \mu(\boldsymbol{M} - \boldsymbol{L}_{t+1} - \boldsymbol{S}_{t+1})$ .

# 矩阵补全

ADMM 具体算法如 Algorithm 7 所示.

---

**Algorithm 7:** Alternating direction method for matrix completion.

---

**Input:** Data matrix  $M$ , mask  $P_\Omega$ , max iterations  $T$ , tolerance  $\tau$

**Output:** Low-rank matrix  $L$ , sparse noise matrix  $S$

```
1 set  $\mu = mn/(4\|M\|_1)$ ;  
2 initialize  $t = 0$ ,  $L = S = Y = O$ , terminate = False;  
3 while not terminate and  $t < T$  do  
4      $t \leftarrow t + 1$ ;  
5      $L \leftarrow \mathcal{D}_{1/\mu}(M - S + Y/\mu)$ ;  
6      $S \leftarrow P_\Omega \odot (M - L + Y/\mu)$ ;  
7      $Y \leftarrow Y + \mu(M - L - S)$ ;  
8     terminate  $\leftarrow \|M - L - S\|_F < \tau$ ;  
9 end  
10 return  $L, S$ ;
```

---

## 1 引言

method

- Alternating least squares
- Neural tangent kernels

## 2 鲁棒主成分分析: 通用算法

## 5 实验结果分析

## 3 鲁棒主成分分析: 特殊算法

## 6 代码说明

## 4 矩阵补全

- Singular value thresholding
- Alternating direction

## 7 贡献说明

## 8 参考文献

# 矩阵补全

## Alternating least squares

- Alternating least squares (ALS) [12–14] 方法通过计算矩阵的近似低秩分解完成矩阵补全.
- 具体而言, 矩阵  $M \in \mathbb{R}^{m \times n}$  可以近似分解为  $\hat{M} = UV^\top$ , 其中  $U \in \mathbb{R}^{m \times r}$ ,  $V \in \mathbb{R}^{n \times r}$ , 且  $r \ll \min(m, n)$ .
- 矩阵  $U$  和  $V$  可通过分别求解最小二乘问题得到, 具体过程参见文献 [14].

## 1 引言

method

- Alternating least squares
- Neural tangent kernels

## 2 鲁棒主成分分析: 通用算法

## 5 实验结果分析

## 3 鲁棒主成分分析: 特殊算法

## 6 代码说明

## 4 矩阵补全

- Singular value thresholding
- Alternating direction

## 7 贡献说明

## 8 参考文献

# 矩阵补全

## Neural tangent kernels

- Neural tangent kernels (NTK) [15] 使用神经网络近似待补全矩阵, 并且利用无限宽神经网络与 NTK 的等价性给出一种简单快速的矩阵补全方法.
- 具体算法参见文献 [15].

## 1 引言

## 2 鲁棒主成分分析: 通用算 法

## 3 鲁棒主成分分析: 特殊算 法

## 4 矩阵补全

## 5 实验结果分析

## 6 代码说明

## 7 贡献说明

## 8 参考文献

## ① 引言

## ② 鲁棒主成分分析: 通用算法

## ③ 鲁棒主成分分析: 特殊算法

## ④ 矩阵补全

## ⑤ 实验结果分析

- 数据集与评价指标
- 实验结果

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

# 实验结果分析

## 数据集

实验所用数据集包括 3 张彩色图片, 如图 1 所示.



图 1: 实验数据集

# 实验结果分析

## 评价指标

- 相同迭代次数下的运行时间, 单位为秒.
- 低秩矩阵的秩  $\text{rank}(\mathbf{L})$ .
- 稀疏噪声矩阵的 0 范数  $\|\mathbf{S}\|_0$ .
- 目标函数值: 鲁棒主成分分析的目标函数值为

$$f = \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_{m_1}, \quad \lambda = \frac{1}{\sqrt{\max(m, n)}}, \quad (38)$$

矩阵补全的目标函数值为

$$f = \|\mathbf{L}\|_*. \quad (39)$$

## 1 引言

## 2 鲁棒主成分分析: 通用算法

## 3 鲁棒主成分分析: 特殊算法

## 4 矩阵补全

## 5 实验结果分析

### ● 数据集与评价指标

### ● 实验结果

- 主成分分析
- 鲁棒主成分分析
- 矩阵补全

## 6 代码说明

## 7 贡献说明

## 8 参考文献

# 实验结果

## 主成分分析

- 主成分分析的实验结果如图 2 所示.
- 第 1 行展示了原始图片与添加噪声后的图像, 其余行分别展示了 6 种不同超参数 ( $r_0-r_5$  分别对应  $\text{rank}(\mathcal{L}) = [1, 2, 3, 5, 10, 50]$ ) 下 PCA 算法的实验结果.
- 由图 2 可知, 随着低秩矩阵秩的增加, 恢复出的低秩图像从仅有模糊背景到较清晰图像再到包含噪声图像逐渐转变, 而噪声图像中的元素逐渐减少趋于空白.
- 主成分分析算法的性能指标如表 1 所示.
- 由表 1 可知, 主成分分析算法所需的运行时间很短, 仅为 0.3 秒; 随着低秩矩阵秩的增加目标函数值先降低再增加.

# 实验结果

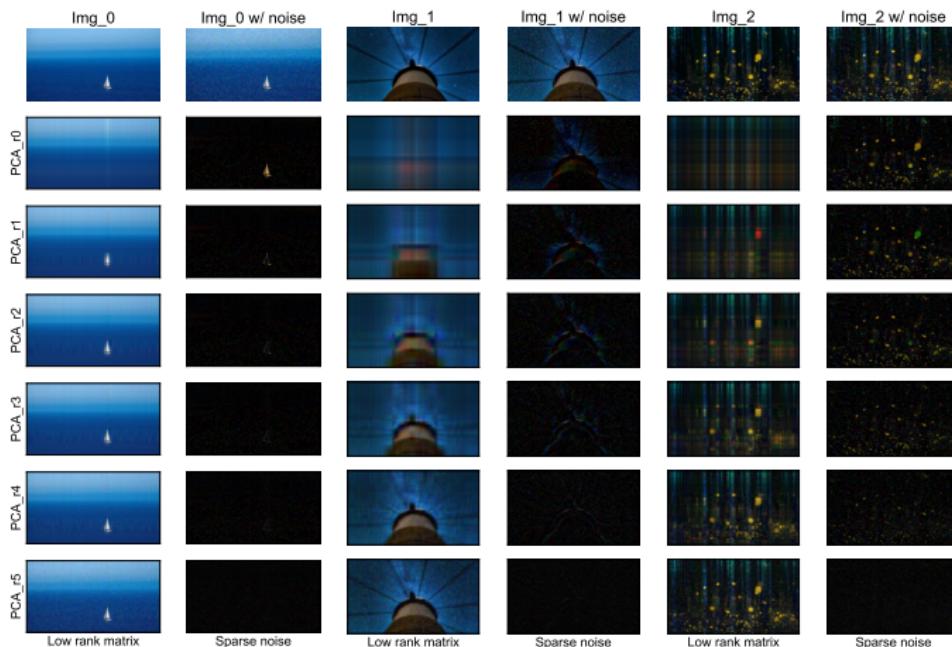


图 2: 主成分分析实验结果

# 实验结果

表 1: 主成分分析算法性能指标

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _* + \lambda \ S\ _{m_1}$
0	PCA r0	0.3	1.0	116500.0	398.6
0	PCA r1	0.3	2.0	116500.0	398.1
0	PCA r2	0.3	3.0	116500.0	408.5
0	PCA r3	0.3	5.0	116500.0	428.2
0	PCA r4	0.3	10.0	116500.0	476.6
0	PCA r5	0.3	50.0	116500.0	679.1
1	PCA r0	0.3	1.0	111000.0	570.8
1	PCA r1	0.3	2.0	111000.0	486.9
1	PCA r2	0.3	3.0	111000.0	478.5

# 实验结果

(续表 1: 主成分分析算法性能指标)

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _* + \lambda \ S\ _{m_1}$
1	PCA r3	0.3	5.0	111000.0	475.1
1	PCA r4	0.3	10.0	111000.0	498.5
1	PCA r5	0.3	50.0	111000.0	679.6
2	PCA r0	0.3	1.0	111000.0	514.5
2	PCA r1	0.3	2.0	111000.0	501.9
2	PCA r2	0.3	3.0	111000.0	496.9
2	PCA r3	0.3	5.0	111000.0	502.6
2	PCA r4	0.3	10.0	111000.0	528.8
2	PCA r5	0.3	50.0	111000.0	717.7

# 实验结果

## 鲁棒主成分分析

- 鲁棒主成分分析的实验结果如图 3 所示.
- 第 1 行展示了原始图片与添加噪声后的图像, 其余行分别展示了 Gradient descent (CGD), Gradient descent with Adam (CGD Adam), Augmented Lagrange multipliers (ALM), Singular value thresholding (SVT), Accelerated proximal gradient (APG) 算法的实验结果.
- 由图 3 可知, CGD, SVT 和 APG 算法的效果都比较好, ALM 算法在不同超参数下的效果差异比较明显.

# 实验结果

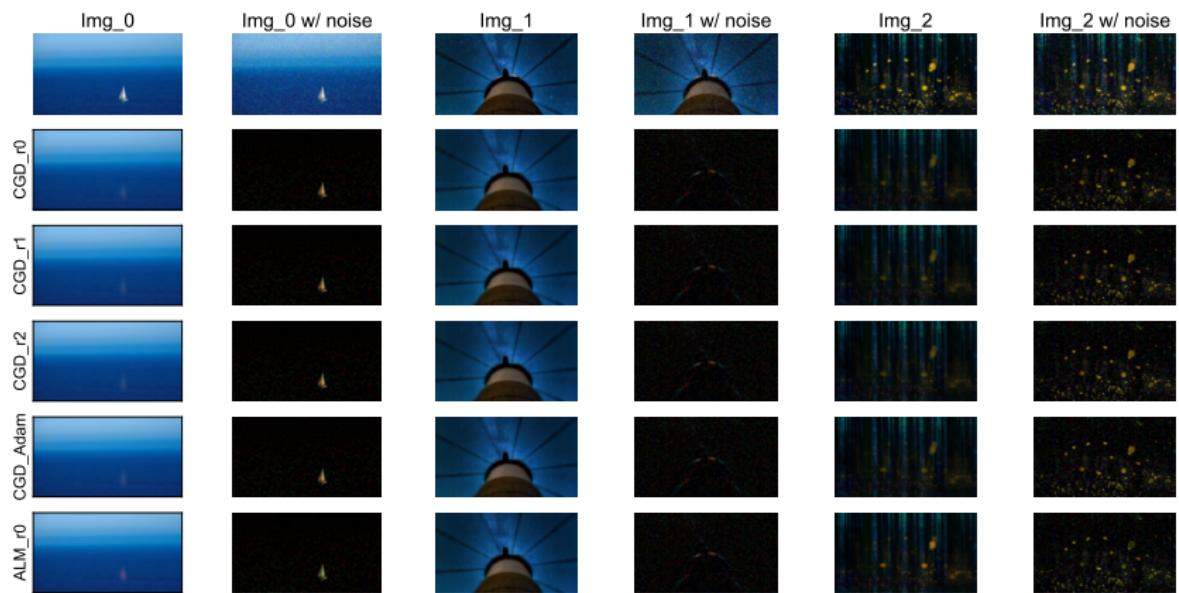
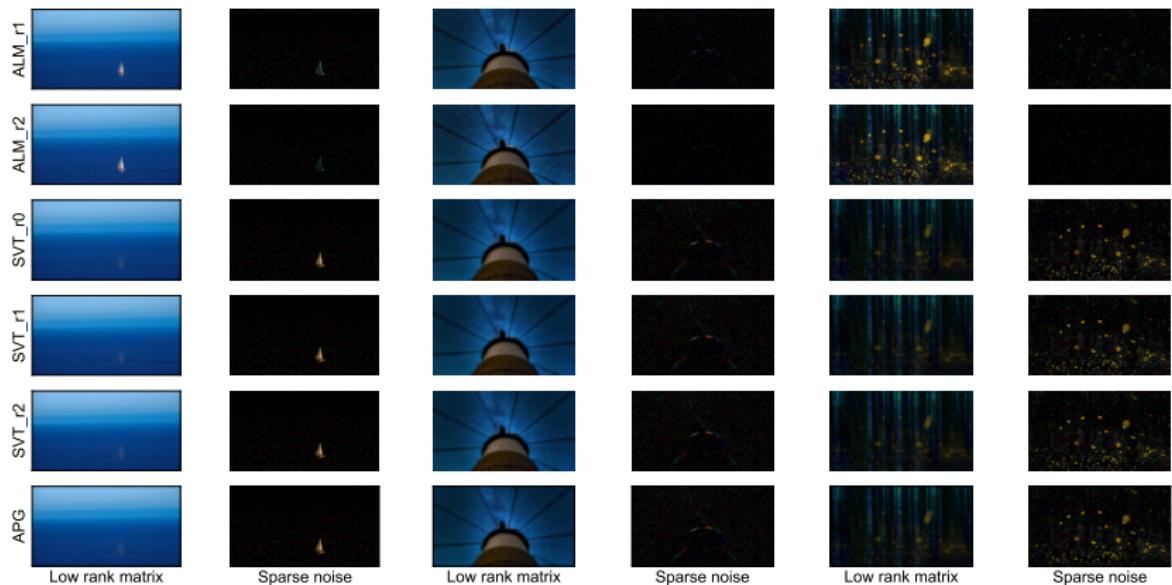


图 3: 鲁棒主成分分析实验结果

# 实验结果



(续图 3: 鲁棒主成分分析实验结果)

# 实验结果

## 超参数设置

图 3 左侧命名带有  $r_0$ ,  $r_1$ ,  $r_2$  的算法为不同超参数的实验结果, 具体超参数设置如表 2 所示.

表 2: 鲁棒主成分分析各算法超参数设置

Algorithm	$r_0$	$r_1$	$r_2$
CGD	$\mu \leftarrow 0.0005\mu$	$\mu \leftarrow 0.001\mu$	$\mu \leftarrow 0.005\mu$
ALM	$\mu \leftarrow 0.1\mu$	$\mu \leftarrow 0.5\mu$	$\mu \leftarrow \mu$
SVT	$\mu = \alpha = 0.001$	$\mu = \alpha = 0.002$	$\mu = \alpha = 0.003$

# 实验结果

## 性能指标

- 鲁棒主成分分析算法的性能指标如表 3 所示.
- 由表 3 可知, CGD 算法的运行时间显著高于其他 3 个算法, 而且几乎没有达到降秩的结果.
- 但是 CGD 算法得到的低秩图像和稀疏噪声图像的视觉效果还不错, 所以绝对意义上的降秩并不是达到预期效果的必要条件.
- SVT 算法的降秩效果最好, 而且噪声矩阵的稀疏性也最好, 因此其目标函数值也是最小的.

# 实验结果

表 3: 鲁棒主成分分析各算法性能指标

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _* + \lambda \ S\ _{m_1}$
0	ALM r0	330.7	136.3	73402.3	333.2
0	ALM r1	329.4	141.0	72262.0	335.3
0	ALM r2	327.6	184.0	51798.0	385.6
0	APG	328.0	135.7	73546.7	333.1
0	SVT r0	322.8	4.0	6249.3	303.3
0	SVT r1	327.1	14.7	10840.3	313.5
0	SVT r2	325.6	27.7	16095.7	320.0
0	CGD r0	535.9	250.0	116500.0	333.2
0	CGD r1	534.2	250.0	116500.0	333.3
0	CGD r2	537.1	250.0	116500.0	333.5
0	CGD Adam	532.9	250.0	116500.0	347.9
1	ALM r0	260.1	140.0	72664.0	388.1
1	ALM r1	263.1	207.0	51589.3	425.6
1	ALM r2	266.7	248.0	26648.7	533.6
1	APG	262.4	141.7	73975.7	388.1
1	SVT r0	261.9	28.3	20876.3	335.5
1	SVT r1	263.6	63.7	38295.0	367.9

# 实验结果

(续表 3：鲁棒主成分分析各算法性能指标)

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _* + \lambda \ S\ _{m_1}$
1	SVT r1	263.6	63.7	38295.0	367.9
1	SVT r2	265.8	85.7	48169.7	378.1
1	CGD r0	463.2	250.0	111000.0	388.2
1	CGD r1	464.9	250.0	111000.0	388.4
1	CGD r2	466.6	250.0	111000.0	388.6
1	CGD Adam	468.0	250.0	111000.0	404.4
2	ALM r0	261.9	141.7	72063.3	374.4
2	ALM r1	261.2	232.3	40029.3	462.0
2	ALM r2	268.4	250.0	19075.7	608.6
2	APG	266.9	141.7	74387.0	373.5
2	SVT r0	265.1	33.7	23243.7	331.7
2	SVT r1	263.1	58.7	36326.7	355.5
2	SVT r2	262.7	76.0	43697.3	363.4
2	CGD r0	463.7	250.0	111000.0	373.7
2	CGD r1	465.7	250.0	111000.0	373.9
2	CGD r2	466.2	250.0	111000.0	374.1
2	CGD Adam	470.6	250.0	111000.0	395.3

# 实验结果

## 矩阵补全

- 矩阵补全的实验结果如图 4 所示, 其中第 1 行展示了原始图片与添加噪声后的图像, 其余行分别展示了 Alternating direction method (ADMM), Singular value thresholding (SVT), Alternating least squares (PMF, BPMF) 算法的实验结果.
- 由图 4 可知, ADMM 和 SVT 算法的效果都比较好, PMF 和 BPMF 算法的效果则要差一些.
- 矩阵补全 CNTK 方法的实验结果如图 5 所示, 其中第 1 行展示了原始图片与添加噪声后的图像, 第 2 行是直接使用 CNTK 方法得到的结果, 第 3 行展示使用 CNTK 和 EigenPro 得到的结果.
- 由图 5 可知, CNTK 方法得到的结果都比较不错.

# 实验结果

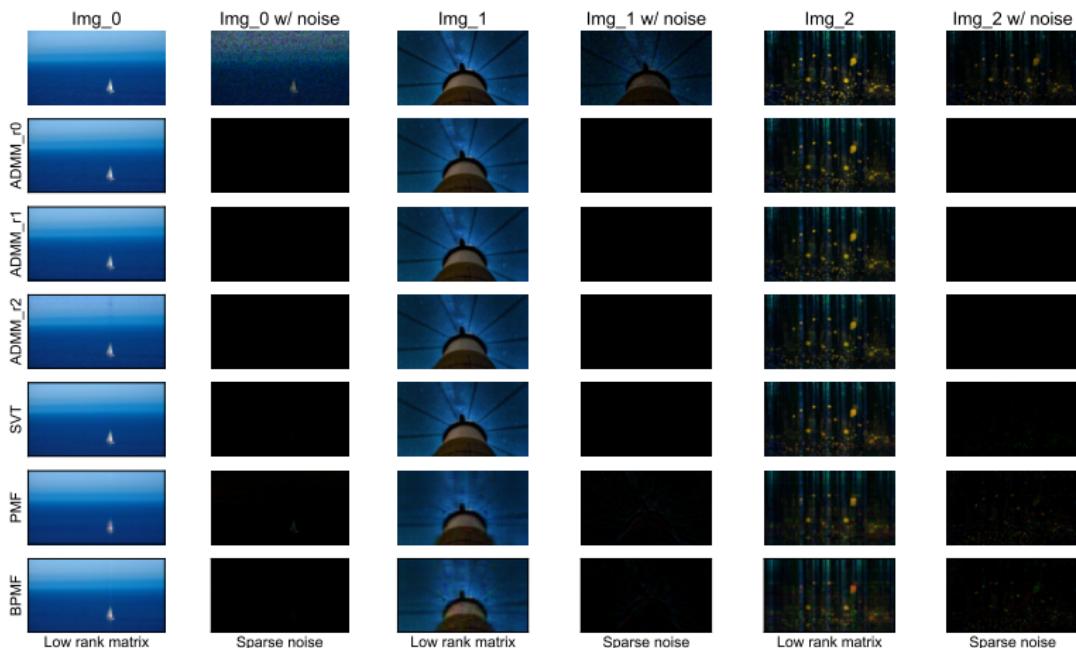


图 4: 矩阵补全实验结果

# 实验结果

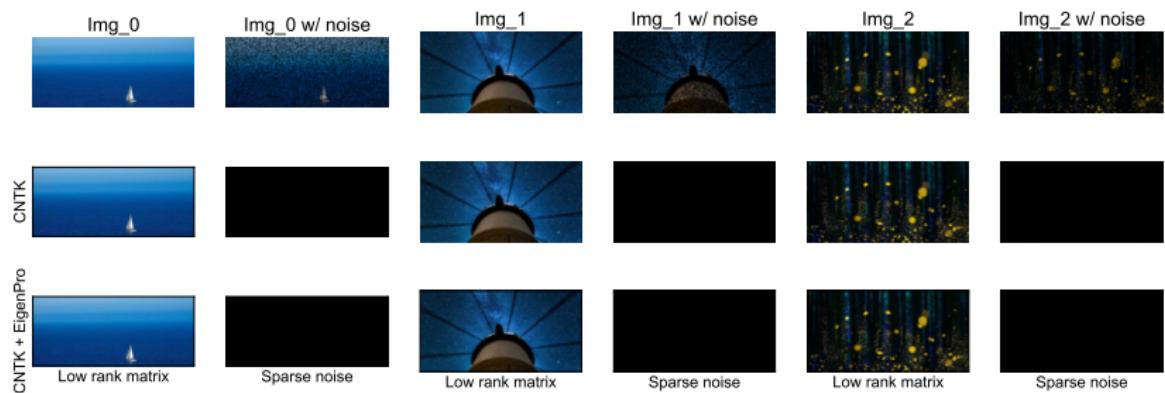


图 5: 矩阵补全 CNTK 方法实验结果

# 实验结果

## 超参数设置

图 4 左侧命名带有  $r_0, r_1, r_2$  的算法为不同超参数的实验结果, 具体超参数设置如表 4 所示.

表 4: 矩阵补全算法超参数设置

Algorithm	$r_0$	$r_1$	$r_2$
ADMM	$\mu \leftarrow 0.05\mu$	$\mu \leftarrow 0.1\mu$	$\mu \leftarrow 0.15\mu$

# 实验结果

## 性能指标

- 矩阵补全算法的性能指标如表 5 所示.
- 由表 5 可知, ADMM 和 CNTK 算法的运行时间显著多于其他算法, 使用了 EigenPro 的 CNTK 算法的运行时间显著减少.
- SVT, PMF 和 BPMF 算法得到的低秩矩阵的秩均比较小, 但是噪声矩阵的稀疏性较差.

# 实验结果

表 5: 矩阵补全各算法性能指标

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _*$
0	ADMM r0	322.6	191.7	57822.7	206.2
0	ADMM r1	330.4	192.0	57822.7	206.4
0	ADMM r2	325.9	198.0	57822.7	239.7
0	SVT	36.3	8.7	116500.0	178.0
0	PMF	2.0	10.0	116500.0	164.1
0	BPMF	2.0	12.0	116500.0	180.7
0	CNTK	340.7	192.0	43120.0	181.4
0	CNTK EigenPro	69.1	192.0	43120.0	180.0
1	ADMM r0	263.3	196.7	53239.3	262.9
1	ADMM r1	266.2	200.0	53239.3	263.7
1	ADMM r2	261.0	205.0	53239.3	269.6
1	SVT	297.5	58.7	111000.0	208.4

# 实验结果

(续表 5: 矩阵补全各算法性能指标)

Image	Algorithm	Time (s)	rank( $L$ )	$\ S\ _0$	$\ L\ _*$
1	PMF	1.8	10.0	111000.0	120.6
1	BPMF	1.9	12.0	111000.0	131.3
1	CNTK	340.7	192.0	36756.0	238.2
1	CNTK EigenPro	69.1	192.0	36755.7	225.8
2	ADMM r0	261.5	200.7	50828.0	282.4
2	ADMM r1	263.2	204.0	50828.0	283.7
2	ADMM r2	262.5	208.0	50828.0	286.6
2	SVT	730.6	83.3	111000.0	210.7
2	PMF	1.9	10.0	111000.0	80.4
2	BPMF	1.9	12.0	111000.0	95.2
2	CNTK	340.7	192.0	36746.0	253.3
2	CNTK EigenPro	69.1	192.0	36746.0	236.3

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

## 代码说明

### 参考代码

- 本文参考的代码如表 6 所示, 这些代码文件的位置为:  
`utils/`.
- `ntk/` 文件夹为文献 [15] 作者提供的 CNTK 方法的开源代码, 本文基于该开源代码进行了简单修改, 以用于本任务.

表 6: 参考代码及链接

file	function	link
<code>robust_pca.py</code>	<code>_augmented_Lagrange_multipliers()</code>	<code>Robust-PCA</code>
<code>matrix_completion.py</code>	<code>_singular_value_thresholding()</code>	<code>matrix-completion</code>
<code>matrix_completion.py</code>	<code>_probabilistic_matrix_factorization()</code>	<code>matrix-completion</code>
<code>matrix_completion.py</code>	<code>_biased_probabilistic_matrix_factorization()</code>	<code>matrix-completion</code>

## 代码说明

### 代码结构

代码结构如表 7 所示, 其中核心算法位于 `utils/` 文件夹.

表 7: 代码结构

folder	description
<code>bin/</code>	testing algorithms, analyzing results, and plotting figures
<code>data/</code>	source data and results
<code>doc/</code>	documentation
<code>ntk/</code>	the CNTK algorithm
<code>pre/</code>	slides
<code>utils/</code>	algorithms

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

## 贡献说明

本研究的分工如表 8 所示.

表 8: 贡献说明

工作内容	完成人
鲁棒主成分分析: 通用算法	杨敬轩
鲁棒主成分分析: 特殊算法	杨敬轩
矩阵补全算法	董泽委
实验结果分析	杨敬轩, 董泽委 (1: 0.9)
报告撰写	杨敬轩, 董泽委 (1: 0.7)
Beamer 制作	杨敬轩, 董泽委 (1: 0.5)

## ① 引言

## ② 鲁棒主成分分析: 通用算 法

## ③ 鲁棒主成分分析: 特殊算 法

## ④ 矩阵补全

## ⑤ 实验结果分析

## ⑥ 代码说明

## ⑦ 贡献说明

## ⑧ 参考文献

# 参考文献 |

- [1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.
- [2] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.
- [3] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.
- [4] G Alistair Watson. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170(0):33–45, 1992.
- [5] diadochos (<https://math.stackexchange.com/users/351390/diadochos>). Derivative of the nuclear norm. Mathematics Stack Exchange. URL:<https://math.stackexchange.com/q/2727651> (version: 2018-04-10).
- [6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

## 参考文献 II

- [7] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.
- [8] Zhouchen Lin, Arvind Ganesh, John Wright, Leqin Wu, Minming Chen, and Yi Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Coordinated Science Laboratory Report no. UILU-ENG-09-2214, DC-246*, 2009.
- [9] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12(2), 2009.
- [10] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.
- [11] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.
- [12] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

## 参考文献 III

- [13] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.
- [14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.
- [15] Adityanarayanan Radhakrishnan, George Stefanakis, Mikhail Belkin, and Caroline Uhler. Simple, fast, and flexible framework for matrix completion with infinite width neural networks. *Proceedings of the National Academy of Sciences*, 119(16):e2115064119, 2022.

*Thanks for your attention!*

Q & A