Matrix Analysis and Applications (Autumn 2022) **Final Project**

综合课题研究

Lecturer: Feng Chen chenfeng@mail.tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Zewei Dong dzw22@mails.tsinghua.edu.cn

摘 要

图像数据的低秩矩阵恢复是计算机视觉与模式识别任务的重要研究问题. 本文针对低秩矩阵恢复问题进行探究, 主要研究了鲁棒主成分分析算法以及矩阵补全算法. 针对鲁棒主成分分析算法, 给出了基于梯度下降算法的通用求解方法, 并基于 Adam 算法进行优化, 然后给出了 3 种特殊优化算法: SVT, APG 与 ALM. 针对矩阵补全问题, 给出了矩阵补全问题的数学模型, 以及 4 种具体求解算法: SVT, ADMM, ALS, NTK. 本文对上述算法在图像数据集上的效果进行了测试, 结果表明 PCA 算法的求解时间最短, 但是效果比较一般, 鲁棒主成分分析算法的效果更好, 但是耗时普遍较长, 其中 SVT 算法的用时较短且取得了相对更好的效果; 矩阵补全算法也可以实现低秩矩阵恢复, ADMM 和 SVT 算法的效果均比较好但用时较长, CNTK 算法与 EigenPro 联合使用既用时少又效果好. 本文还对不同超参数的实验结果进行了测试和分析. 代码开源地址: https://github.com/jingxuanyang/LowRankMatrixRestoration.

关键词: 低秩矩阵恢复; 鲁棒主成分分析; 矩阵补全

目录

1 引言

在计算机视觉与模式识别任务中, 通常都假设观测数据近似存在于一个低维的子空间下. 为找出这样的子空间, 经典的主成分分析 (Principal Component Analysis, PCA) [1] 方法假定数据受到较小的高斯噪声污染, 即数据 $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ 由一个低秩矩阵 $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ 和一个独立同分布的高斯噪声矩阵 $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ 构成. 当给定超参数 $r \in \mathbb{N}_{>0}$ 时, 该任务可以建模为如下的优化问题:

$$\begin{aligned} &\min_{\boldsymbol{L},\boldsymbol{S}} \ \|\boldsymbol{S}\|_{\mathrm{F}} \\ &\text{s.t. } \operatorname{rank}(\boldsymbol{L}) \leqslant r, \\ &\qquad \boldsymbol{M} = \boldsymbol{L} + \boldsymbol{S}. \end{aligned} \tag{1}$$

然而在实际应用中, 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. 而另一方面, 当噪声矩阵 $\boldsymbol{S}$ 足够稀疏时, 原始的低秩矩阵仍然可能被恢复 [2]. 该任务可以建模为如下的优化问题:

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

其中, $\|\boldsymbol{S}\|_0$ 指矩阵中非零元素的个数. 由于 $\operatorname{rank}(\boldsymbol{L})$ 和 $\|\boldsymbol{S}\|_0$ 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.

鲁棒主成分分析 (Robust Principal Component Analysis, RPCA) [3] 提出将 $\operatorname{rank}(\boldsymbol{L})$ 用核范数 $\|\boldsymbol{L}\|_* = \operatorname{tr}(\sqrt{\boldsymbol{L}^\top\boldsymbol{L}})$ 进行松弛, 并将 $\|\boldsymbol{S}\|_0$ 用矩阵 m_1 范数 $\|\boldsymbol{S}\|_{m_1} = \sum_{i,j}|s_{i,j}|$ 进行松弛, 从而有

$$\begin{aligned} &\min_{\boldsymbol{L},\boldsymbol{S}} \ \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_{m_1} \\ &\text{s.t. } \ \boldsymbol{M} = \boldsymbol{L} + \boldsymbol{S}. \end{aligned} \tag{3}$$

本研究首先针对 RPCA 问题给出基于梯度下降算法的通用求解方法, 然后给出几个典型的特殊优化算法. 除 RPCA 以外, 还有许多低秩矩阵恢复模型能解决类似问题, 如矩阵补全 (Matrix Completion) [4], 低秩表示 (Low-Rank Representation) [5] 等. 本研究选择矩阵补全进行求解, 给出几种典型优化算法. 而后, 对上述算法在测试数据集上的效果进行对比分析, 对算法的超参数进行分析, 并对所编写代码进行说明. 最后, 给出本研究的结论.

Related works

鲁棒主成分分析方法已被广泛的应用于图像去噪, 该方法主要通过最小化核范数和 m_1 范数的组合问题, 将观测到的低秩矩阵分解为低秩部分和稀疏的噪声部分. 但是核范数为了保持凸性,

赋予每一个奇异值的权重都是一样的, 忽略了矩阵奇异值的先验知识, 对此提出了加权核范数鲁棒主成分分析模型 [6]. 加权核范数在奇异值分解过程中, 给奇异值分配了不同的权值, 大的奇异值通常表示数据的主要成分, 所以大的奇异值需要分配一个小的权值, 以减少阈值收缩, 小的奇异值分配大的权值, 使矩阵获得更低的秩. 但当观测数据被两种以上的混合噪声污染时, 小的噪声会被大的噪声掩盖, 为此文献 [7] 提出了广义加权鲁棒主成分分析模型, 通过加权核范数, ℓ_1 范数和 Frobenius 范数的组合问题, 从观测矩阵中分离出低秩部分, 稀疏大噪声部分和稠密小噪声部分, 并用随机排序的交替方向乘子法求解.

为了进一步提高图像恢复效果, 文献 [8] 提出了一种新的二次函数近似秩函数, 利用数值实验分析了二次函数与核范数, 对秩函数的近似效果. 当 $\|\boldsymbol{L}\|_{\mathrm{F}}^2 \leqslant 1$ 时, 二次函数对秩函数近似效果更好, 为此引入了收缩参数保障二次函数对秩函数在全域上具有好的近似效果. 基于上述分析, 给出了秩极小化低秩矩阵恢复问题的二次近似模型, 并设计了求解低秩矩阵恢复问题的算法.

当数据矩阵 $\boldsymbol{M}$ 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, 称此恢复过程为矩阵补全. 一般来说, 核范数正则化问题可以被看作为半定规划问题, 但是这种策略仅对维度较低的矩阵有效. 为了有效地求解矩阵补全问题, 大量的一阶梯度算法被提出. 比如 SVT [9] 和 APG [10], 在通常情况下能够得到较为满意的解且有严格的理论保证, 但是它们在每次迭代过程中需要进行费时的 SVD (Singular Value Decomposition) 分解, 这限制了它们在大规模矩阵上的应用.

为了缓解这一缺陷, 研究者提出了 FPCA (Fixed Point Continuation with Approximate) [11] 算法求解和 APG 算法相同的优化问题. 在 FPCA 算法中引入了快速蒙特卡罗算法来近似求解 SVD, 它的效率得到了极大的提升. 针对求解过程存在 “低秩＋稀疏” 的特性, 提出了 Soft-Impute 算法 [12], 同时结合 SVT 算法, 达到在每次迭代中快速进行 SVD 分解的目的. 最近, 文献 [13] 通过引入 Nesterov 优化理论, 使得 Soft-Impute 算法的计算时间复杂度由 $\mathcal{O}(1/T)$ 提升为 $\mathcal{O}(1/T^2)$, 其中 T 为迭代次数.

核范数正则化方法作为一个强大的工具已广泛地应用于求解低秩矩阵补全问题, 然而该方法同等对待目标矩阵中的所有奇异值, 从而导致过度惩罚较大奇异值. 换句话说, 核范数正则化方法在大多数时得到的解严重偏离真实解. 为了使得较大奇异值得到更少的惩罚, 大量的非凸函数被提出用于替代秩函数. 在文献 [14] 中, 一个快速、连续和精确的算法被提出用于求解高维线性回归问题. 另一类在低秩矩阵补全领域得到广泛关注的策略是, 利用 Schatten-q ($0 < q \leqslant 1$) 拟范数来逼近目标矩阵的秩函数. 然而, 基于该策略得到的非凸 ℓ_q 正则化优化问题较难被求解. ℓ_qPG 算法被提出用于求解该非凸 ℓ_q 正则化优化问题. 在算法的每次迭代过程中包含不精确的近似求解和费时的 SVD, 这严重制约了算法的应用范围. 基于此, 文献 [15] 和文献 [16] 分别提出利用 Schatten-$\frac{1}{2}$ 和 Schatten-$\frac{2}{3}$ 拟范数来逼近目标矩阵的秩函数. 同时, 提出基于半阈值算子和 $\frac{2}{3}$ 阈值算子的不动点算法. 对比 ℓ_qPG 算法, 基于 Schatten-$\frac{1}{2}$ 和 Schatten-$\frac{2}{3}$ 拟范数的阈值函数具有闭式解, 因此得到

的解更精确.

基于非凸正则化方法的优良性能, 文献 [17] 对基于加权核范数正则化模型进行进一步的分析和探讨, 并借鉴 Soft-Impute 算法的思想提出一个拥有更快收敛速率和能够得到更高精度解的低秩矩阵补全算法 WNNM-Impute (Weighted Nuclear Norm Minimization Impute). WNNM-Impute 算法引入不精确的近邻算子降低时间复杂度, 从而使得算法收敛更快. 同时, 在算法中引入 Nesterov 加速策略, 使得算法的总体迭代次数进一步减少.

2 鲁棒主成分分析: 通用算法

2.1 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) [18] 进行梯度下降.

对式 (3) 进行变形可得

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

令 $\boldsymbol{L}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ 为矩阵 $\boldsymbol{L}$ 的 SVD 分解, 则核范数的次梯度 [19, 20] 为

$$\partial\|\boldsymbol{L}\|_* = \{\boldsymbol{U}\boldsymbol{V}^\top + \boldsymbol{W} : \boldsymbol{U}^\top\boldsymbol{W}=\boldsymbol{O}, \boldsymbol{W}\boldsymbol{V}=\boldsymbol{O}, \|\boldsymbol{W}\|_2 \leqslant 1\}. \tag{5}$$

矩阵 m_1 范数的次梯度可取为

$$-\operatorname{sgn}(\boldsymbol{M}-\boldsymbol{L}) \in \partial\|\boldsymbol{M}-\boldsymbol{L}\|_{m_1}. \tag{6}$$

则目标函数 $f(\boldsymbol{L})$ 的次梯度可取为

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{V}^\top - \operatorname{sgn}(\boldsymbol{M}-\boldsymbol{L}). \tag{7}$$

从而, 梯度下降算法如 Algorithm 1 所示.

2.2 Gradient descent with Adam

梯度下降算法的效率基本上由梯度更新的步骤决定, 而 Adam [21] 算法是深度学习算法中常用的优化器, 因此本节采用 Adam 算法对梯度更新的过程进行优化, 具体算法如 Algorithm 2 所示.

Algorithm 1: Gradient descent for RPCA.

Input: Data matrix $\boldsymbol{M}$, learning rate α, max iterations T, tolerance τ

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

initialize $t = 0$, $\boldsymbol{L} = \boldsymbol{O}$, terminate = False;
while *not terminate and* $t < T$ **do**
  $t \leftarrow t + 1$;
  $\boldsymbol{G} \leftarrow$ Subgradient($\boldsymbol{L}$) by Eq. (7);
  $\boldsymbol{L}' \leftarrow \boldsymbol{L} - \alpha \boldsymbol{G}$;
  terminate $\leftarrow |f(\boldsymbol{L}') - f(\boldsymbol{L})| < \tau$;
  $\boldsymbol{L} \leftarrow \boldsymbol{L}'$;
end
$\boldsymbol{S} \leftarrow \boldsymbol{M} - \boldsymbol{L}$;
return $\boldsymbol{L}$, $\boldsymbol{S}$;

Algorithm 2: Gradient descent with Adam for RPCA.

Input: Data matrix $\boldsymbol{M}$, learning rate α, max iterations T, tolerance τ

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

set $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$;
initialize $t = 0$, $\boldsymbol{L} = \boldsymbol{O}$, terminate = False, $\boldsymbol{W}_0 = \boldsymbol{V}_0 = \boldsymbol{O}$;
while *not terminate and* $t < T$ **do**
  $t \leftarrow t + 1$;
  $\boldsymbol{G} \leftarrow$ Subgradient($\boldsymbol{L}$) by Eq. (7);
  $\boldsymbol{W}_t \leftarrow \beta_1 \boldsymbol{W}_{t-1} + (1 - \beta_1)\boldsymbol{G}$;
  $\boldsymbol{V}_t \leftarrow \beta_2 \boldsymbol{V}_{t-1} + (1 - \beta_2)\boldsymbol{G} \odot \boldsymbol{G}$;
  $\hat{\boldsymbol{W}}_t \leftarrow \boldsymbol{W}_t / (1 - \beta_1^t)$;
  $\hat{\boldsymbol{V}}_t \leftarrow \boldsymbol{V}_t / (1 - \beta_2^t)$;
  $\boldsymbol{L}' \leftarrow \boldsymbol{L} - \alpha \hat{\boldsymbol{W}}_t \oslash (\sqrt{\hat{\boldsymbol{V}}_t} + \epsilon)$ /* $\sqrt{\cdot}$ applied element-wise */;
  terminate $\leftarrow |f(\boldsymbol{L}') - f(\boldsymbol{L})| < \tau$;
  $\boldsymbol{L} \leftarrow \boldsymbol{L}'$;
end
$\boldsymbol{S} \leftarrow \boldsymbol{M} - \boldsymbol{L}$;
return $\boldsymbol{L}$, $\boldsymbol{S}$;

3 鲁棒主成分分析: 特殊算法

本节对 RPCA 的特殊优化算法进行分析. 为简化推导过程, 下面首先介绍一些预备知识.

3.1 Preliminaries

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

$$\min_{\boldsymbol{X}} \ \epsilon\|\boldsymbol{X}\|_{m_1} + \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Q}\|_{\mathrm{F}}^2 \tag{8}$$

的最优解为 $\boldsymbol{X}^* = \mathcal{S}_\epsilon(\boldsymbol{Q})$ [3], 其中 $\mathcal{S}_\epsilon$ 为 Shrinkage 算子,

$$\mathcal{S}_\epsilon(\boldsymbol{Q}) = \max(|\boldsymbol{Q}|-\epsilon, 0)\odot \operatorname{sgn}(\boldsymbol{Q}). \tag{9}$$

注意, 上述 max(), sgn() 运算均为按元素进行 (element-wise) 的运算.

将 m_1 范数替换为核范数, 则有

$$\min_{\boldsymbol{X}} \ \epsilon\|\boldsymbol{X}\|_{*} + \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Q}\|_{\mathrm{F}}^2, \tag{10}$$

其闭式解为 $\boldsymbol{X}^* = \mathcal{D}_\epsilon(\boldsymbol{Q})$ [9], $\mathcal{D}_\epsilon$ 为 Singular value thresholding 算子. 记 $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ 为矩阵 $\boldsymbol{Q}$ 的 SVD 分解, 则

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

3.2 Singular value thresholding

Singular value thresholding (SVT) [9] 是一种迭代阈值算法. 对式 (3) 进行正则可得

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

其中, $\mu > 0$ 为较小的正数. 上式的 Lagrange 函数为

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

SVT 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,

$$\boldsymbol{L}_{t+1} = \operatorname*{argmin}_{\boldsymbol{L}} \mathcal{L}(\boldsymbol{L},\boldsymbol{S}_t,\boldsymbol{Y}_t) = \operatorname*{argmin}_{\boldsymbol{L}} \frac{1}{\mu}\|\boldsymbol{L}\|_* + \frac{1}{2}\|\boldsymbol{L}-\boldsymbol{Y}_t/\mu\|_{\mathrm{F}}^2 = \mathcal{D}_{1/\mu}(\boldsymbol{Y}_t/\mu). \tag{14}$$

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

$$\boldsymbol{S}_{t+1}=\underset{\boldsymbol{S}}{\operatorname{argmin}}\,\mathcal{L}(\boldsymbol{L}_{t+1},\boldsymbol{S},\boldsymbol{Y}_t)=\underset{\boldsymbol{S}}{\operatorname{argmin}}\,\frac{\lambda}{\mu}\|\boldsymbol{S}\|_{m_1}+\frac{1}{2}\|\boldsymbol{S}-\boldsymbol{Y}_t/\mu\|_{\mathrm{F}}^2=\mathcal{S}_{\lambda/\mu}(\boldsymbol{Y}_t/\mu). \tag{15}$$

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

$$\boldsymbol{Y}_{t+1}=\boldsymbol{Y}_t+\alpha(\boldsymbol{M}-\boldsymbol{L}_{t+1}-\boldsymbol{S}_{t+1}), \tag{16}$$

其中 $\alpha\in(0,1)$ 为迭代步长.

SVT 具体算法如 Algorithm 3 所示.

Algorithm 3: Singular value thresholding for RPCA.

Input: Data matrix $\boldsymbol{M}$, learning rate α, max iterations T, tolerance τ, regularization λ,μ

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

initialize $t=0$, $\boldsymbol{L}=\boldsymbol{S}=\boldsymbol{Y}=\boldsymbol{O}$, terminate = False;
while *not terminate and* $t<T$ **do**
  $t\leftarrow t+1$;
  $\boldsymbol{L}\leftarrow\mathcal{D}_{1/\mu}(\boldsymbol{Y}/\mu)$;
  $\boldsymbol{S}\leftarrow\mathcal{S}_{\lambda/\mu}(\boldsymbol{Y}/\mu)$;
  $\boldsymbol{Y}\leftarrow\boldsymbol{Y}+\alpha(\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S})$;
  terminate $\leftarrow\|\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S}\|_{\mathrm{F}}<\tau$;
end
return $\boldsymbol{L}$, $\boldsymbol{S}$;

3.3 Accelerated proximal gradient

将优化问题式 (3) 的等式约束松弛到目标函数中, 得到 Lagrange 函数为

$$\mathcal{L}(\boldsymbol{L},\boldsymbol{S},\mu)=\mu(\|\boldsymbol{L}\|_*+\lambda\|\boldsymbol{S}\|_{m_1})+\|\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S}\|_{\mathrm{F}}^2. \tag{17}$$

Accelerated proximal gradient (APG) [10] 算法通过对 Lagrange 函数做二次逼近来求解式 (3).

初始化 $\boldsymbol{L}_0=\boldsymbol{S}_0=\boldsymbol{Y}_0=\boldsymbol{Y}_0'=\boldsymbol{O}$, $\alpha_0=1$. 当 $\boldsymbol{S}=\boldsymbol{S}_t$, $\boldsymbol{Y}=\boldsymbol{Y}_t$, $\boldsymbol{Y}'=\boldsymbol{Y}_t'$, $\mu=\mu_t$ 时,

$$\boldsymbol{L}_{t+1}=\mathcal{D}_{\mu_t/L_f}(\boldsymbol{Y}_t+(\boldsymbol{M}-\boldsymbol{Y}_t-\boldsymbol{Y}_t')/L_f), \tag{18}$$

其中, $L_f=2$. 当 $\boldsymbol{L}=\boldsymbol{L}_{t+1}$, $\boldsymbol{Y}=\boldsymbol{Y}_t$, $\boldsymbol{Y}'=\boldsymbol{Y}_t'$, $\mu=\mu_t$ 时,

$$\boldsymbol{S}_{t+1}=\mathcal{S}_{\mu_t/L_f}(\boldsymbol{Y}_t'+(\boldsymbol{M}-\boldsymbol{Y}_t-\boldsymbol{Y}_t')/L_f). \tag{19}$$

令

$$\alpha_{t+1} = \frac{1+\sqrt{1+4\alpha_t^2}}{2}, \tag{20}$$

则

$$\boldsymbol{Y}_{t+1} = \boldsymbol{L}_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(\boldsymbol{L}_t - \boldsymbol{L}_{t+1}), \tag{21}$$

且

$$\boldsymbol{Y}'_{t+1} = \boldsymbol{S}_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(\boldsymbol{S}_t - \boldsymbol{S}_{t+1}). \tag{22}$$

APG 具体算法如 Algorithm 4 所示.

Algorithm 4: Accelerated proximal gradient for RPCA.

Input: Data matrix $\boldsymbol{M}$, max iterations T, tolerance τ

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

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

3.4 Augmented Lagrange multipliers

对式 (3) 构造增广 Lagrange 函数为

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

Augmented Lagrange multipliers (ALM) [3, 22, 23] 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,

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

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

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

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

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

ALM 具体算法如 Algorithm 5 所示.

Algorithm 5: Augmented Lagrange multipliers for RPCA.

Input: Data matrix $\boldsymbol{M}$, max iterations T, tolerance τ

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

set $\lambda=1/\sqrt{\max(m,n)}$, $\mu=mn/(4\|\boldsymbol{M}\|_1)$;
initialize $t=0$, $\boldsymbol{L}=\boldsymbol{S}=\boldsymbol{Y}=\boldsymbol{O}$, terminate = False;
while *not terminate and* $t<T$ **do**
  $t\leftarrow t+1$;
  $\boldsymbol{L}\leftarrow\mathcal{D}_{1/\mu}(\boldsymbol{M}-\boldsymbol{S}+\boldsymbol{Y}/\mu)$;
  $\boldsymbol{S}\leftarrow\mathcal{S}_{\lambda/\mu}(\boldsymbol{M}-\boldsymbol{L}+\boldsymbol{Y}/\mu)$;
  $\boldsymbol{Y}\leftarrow\boldsymbol{Y}+\mu(\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S})$;
  terminate $\leftarrow\|\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S}\|_{\mathrm{F}}<\tau$;
end
return $\boldsymbol{L}$, $\boldsymbol{S}$;

4 矩阵补全

当数据矩阵 $\boldsymbol{M}$ 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) [4]. Candès 等人 [4] 证明了当矩阵的奇异值及采样数目满足一定的条件时, 大多数低秩矩阵能够以较大的概率近乎完美地恢复.

记 Ω 为观测数据的指标集合, 则矩阵补全可以建模为

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

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

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

令 $\boldsymbol{P}_\Omega=\mathcal{P}_\Omega(\boldsymbol{1}\boldsymbol{1}^\top)$ 表示 Mask 矩阵, 则约束条件 $\mathcal{P}_\Omega(\boldsymbol{L})=\mathcal{P}_\Omega(\boldsymbol{M})$ 可以重写为

$$\boldsymbol{P}_\Omega\odot\boldsymbol{L}=\boldsymbol{P}_\Omega\odot\boldsymbol{M}, \tag{29}$$

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:

$$\begin{aligned} &\min_{\boldsymbol{L}}\ \|\boldsymbol{L}\|_*\\ &\text{s.t. } \boldsymbol{P}_\Omega\odot\boldsymbol{L}=\boldsymbol{P}_\Omega\odot\boldsymbol{M}. \end{aligned} \tag{30}$$

上述问题可以重新表示为

$$\begin{aligned} &\min_{\boldsymbol{L}}\ \|\boldsymbol{L}\|_*\\ &\text{s.t. } \boldsymbol{P}_\Omega\odot\boldsymbol{S}=\boldsymbol{O},\\ &\quad\ \boldsymbol{M}=\boldsymbol{L}+\boldsymbol{S}. \end{aligned} \tag{31}$$

4.1 Singular value thresholding

第 3.2 小节中使用的 Singular value thresholding (SVT) [9] 算法也可以用于矩阵补全. 对式 (30) 进行正则可得

$$\begin{aligned} &\min_{\boldsymbol{L}}\ \|\boldsymbol{L}\|_*+\frac{\mu}{2}\|\boldsymbol{L}\|_{\mathrm{F}}^2\\ &\text{s.t. }\ \boldsymbol{P}_\Omega\odot(\boldsymbol{M}-\boldsymbol{L})=\boldsymbol{O}, \end{aligned} \tag{32}$$

其中, $\mu>0$ 为较小的正数. 上式的 Lagrange 函数为

$$\mathcal{L}(\boldsymbol{L},\boldsymbol{Y})=\|\boldsymbol{L}\|_*+\frac{\mu}{2}\|\boldsymbol{L}\|_{\mathrm{F}}^2+\langle\boldsymbol{Y},\boldsymbol{P}_\Omega\odot(\boldsymbol{M}-\boldsymbol{L})\rangle. \tag{33}$$

SVT 算法交替更新矩阵 $\boldsymbol{L}$, $\boldsymbol{Y}$. 当 $\boldsymbol{Y}=\boldsymbol{Y}_t$ 时,

$$\boldsymbol{L}_{t+1}=\mathcal{D}_{1/\mu}(\boldsymbol{Y}_t). \tag{34}$$

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

$$\boldsymbol{Y}_{t+1}=\boldsymbol{Y}_t+\alpha\boldsymbol{P}_\Omega\odot(\boldsymbol{M}-\boldsymbol{L}_{t+1}), \tag{35}$$

其中 $\alpha\in(0,1)$ 为迭代步长.

SVT 具体算法如 Algorithm 6 所示.

Algorithm 6: Singular value thresholding for matrix completion.

Input: Data matrix $\boldsymbol{M}$, mask matrix $\boldsymbol{P}_\Omega$, learning rate α, max iterations T, tolerance τ, regularization μ

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

set $\alpha = 1.2mn/\|\boldsymbol{P}_\Omega\|_{\mathrm{F}}^2$, $\mu = 5(m+n)/2$;
initialize $t = 0$, $\boldsymbol{L} = \boldsymbol{Y} = \boldsymbol{O}$, terminate = False;
while *not terminate and* $t < T$ **do**
    $t \leftarrow t + 1$;
    $\boldsymbol{L} \leftarrow \mathcal{D}_{1/\mu}(\boldsymbol{Y})$;
    $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \alpha \boldsymbol{P}_\Omega \odot (\boldsymbol{M} - \boldsymbol{L})$;
    terminate $\leftarrow \|\boldsymbol{P}_\Omega \odot (\boldsymbol{M} - \boldsymbol{L})\|_{\mathrm{F}} / \|\boldsymbol{P}_\Omega \odot \boldsymbol{M}\|_{\mathrm{F}} < \tau$;
end
$\boldsymbol{S} = \boldsymbol{M} - \boldsymbol{L}$;
return $\boldsymbol{L}$, $\boldsymbol{S}$;

4.2 Alternating direction method

对式 (31) 构造增广 Lagrange 函数为

$$\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \frac{\mu}{2}\|\boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S}\|_{\mathrm{F}}^2 + \langle \boldsymbol{Y}, \boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S} \rangle. \tag{36}$$

Alternating direction method (ADMM) [23] 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,

$$\boldsymbol{L}_{t+1} = \mathcal{D}_{1/\mu}(\boldsymbol{M} - \boldsymbol{S}_t + \boldsymbol{Y}_t/\mu). \tag{37}$$

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

$$\boldsymbol{S}_{t+1} = \boldsymbol{P}_\Omega \odot (\boldsymbol{M} - \boldsymbol{L}_{t+1} + \boldsymbol{Y}_t/\mu). \tag{38}$$

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

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t + \mu(\boldsymbol{M} - \boldsymbol{L}_{t+1} - \boldsymbol{S}_{t+1}). \tag{39}$$

ADMM 具体算法如 Algorithm 7 所示.

4.3 Alternating least squares

Alternating least squares (ALS) [24–26] 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 $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ 可以近似分解为 $\hat{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^\top$, 其中 $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, 且 $r \ll$

Algorithm 7: Alternating direction method for matrix completion.

Input: Data matrix $\boldsymbol{M}$, mask matrix $\boldsymbol{P}_\Omega$, max iterations T, tolerance τ

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

set $\mu = mn/(4\|\boldsymbol{M}\|_1)$;
initialize $t = 0$, $\boldsymbol{L} = \boldsymbol{S} = \boldsymbol{Y} = \boldsymbol{O}$, terminate = False;
while *not terminate and* $t < T$ **do**
  $t \leftarrow t + 1$;
  $\boldsymbol{L} \leftarrow \mathcal{D}_{1/\mu}(\boldsymbol{M} - \boldsymbol{S} + \boldsymbol{Y}/\mu)$;
  $\boldsymbol{S} \leftarrow \boldsymbol{P}_\Omega \odot (\boldsymbol{M} - \boldsymbol{L} + \boldsymbol{Y}/\mu)$;
  $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \mu(\boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S})$;
  terminate $\leftarrow \|\boldsymbol{M} - \boldsymbol{L} - \boldsymbol{S}\|_{\mathrm{F}} < \tau$;
end
return $\boldsymbol{L}$, $\boldsymbol{S}$;

$\min(m, n)$. 矩阵 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 可通过分别求解最小二乘问题得到, 具体过程参见文献 [26].

4.4 Neural tangent kernels

Neural tangent kernels (NTK) [27] 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 [27].

5 实验结果分析

5.1 数据集与评价指标

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

实验评价指标为: 相同迭代次数下的运行时间, 低秩矩阵的秩, 稀疏噪声矩阵的 0 范数, 以及目标函数值. 具体而言, 鲁棒主成分分析的目标函数值为

$$f = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_{m_1}, \quad \lambda = \frac{1}{\sqrt{\max(m,n)}}, \tag{40}$$

矩阵补全的目标函数值为

$$f = \|\boldsymbol{L}\|_*. \tag{41}$$

5.2 实验结果分析

5.2.1 主成分分析

主成分分析的实验结果如图 2 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 6 种不同超参数 (r0-r5 分别对应 $\mathrm{rank}(\boldsymbol{L}) = [1, 2, 3, 5, 10, 50]$) 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.

5.2.2 鲁棒主成分分析

鲁棒主成分分析的实验结果如图 3 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 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.

图 3 左侧命名带有 r0, r1, r2 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.

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.

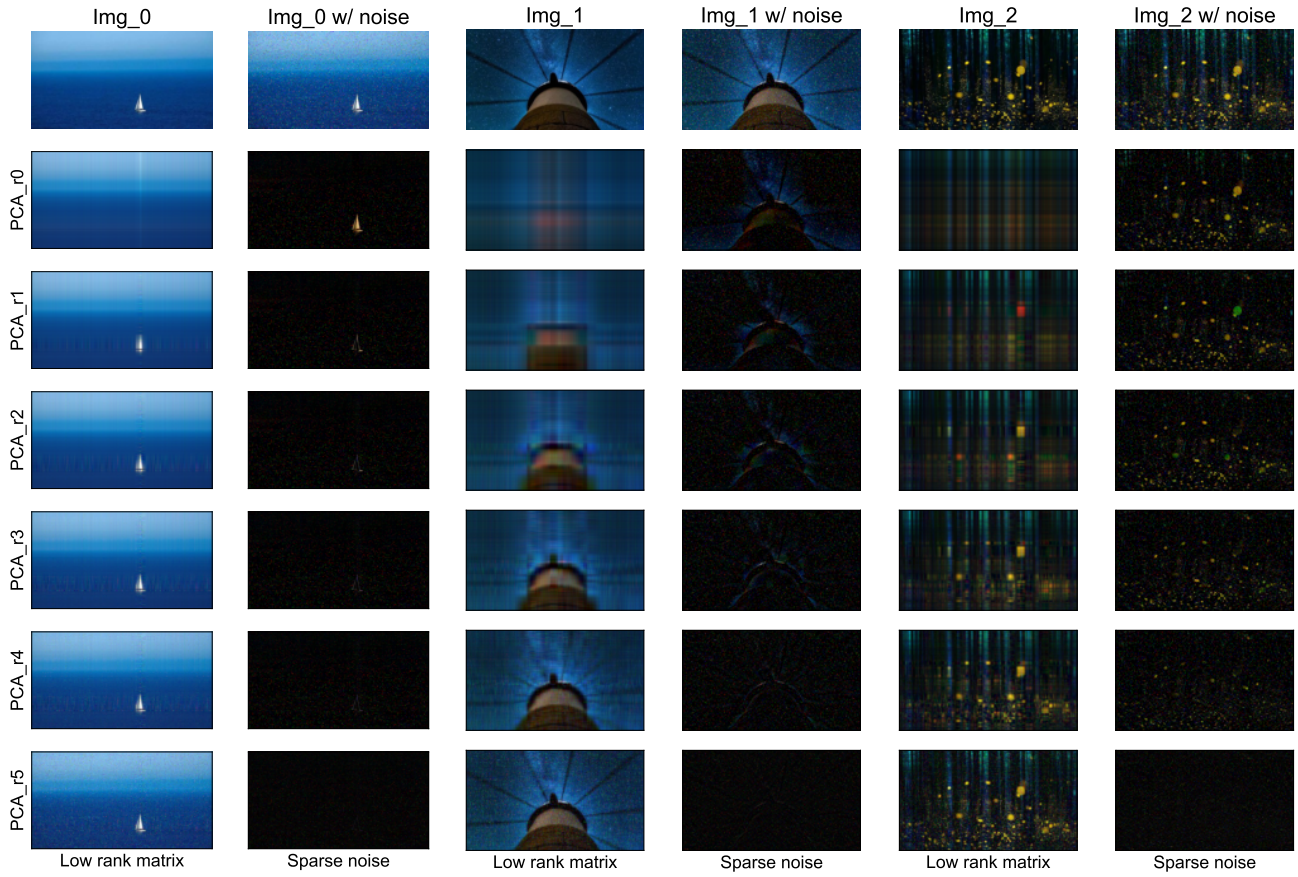

图 2: 主成分分析实验结果

表 1: 主成分分析算法性能指标

Image	Algorithm	Time (s)	rank($\boldsymbol{L}$)	$\|\boldsymbol{S}\|_0$	$\|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_{m_1}$
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
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

表 2: 鲁棒主成分分析各算法超参数设置

Algorithm	r0	r1	r2
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	$\mu = \alpha = 0.001$	$\mu = \alpha = 0.002$	$\mu = \alpha = 0.003$

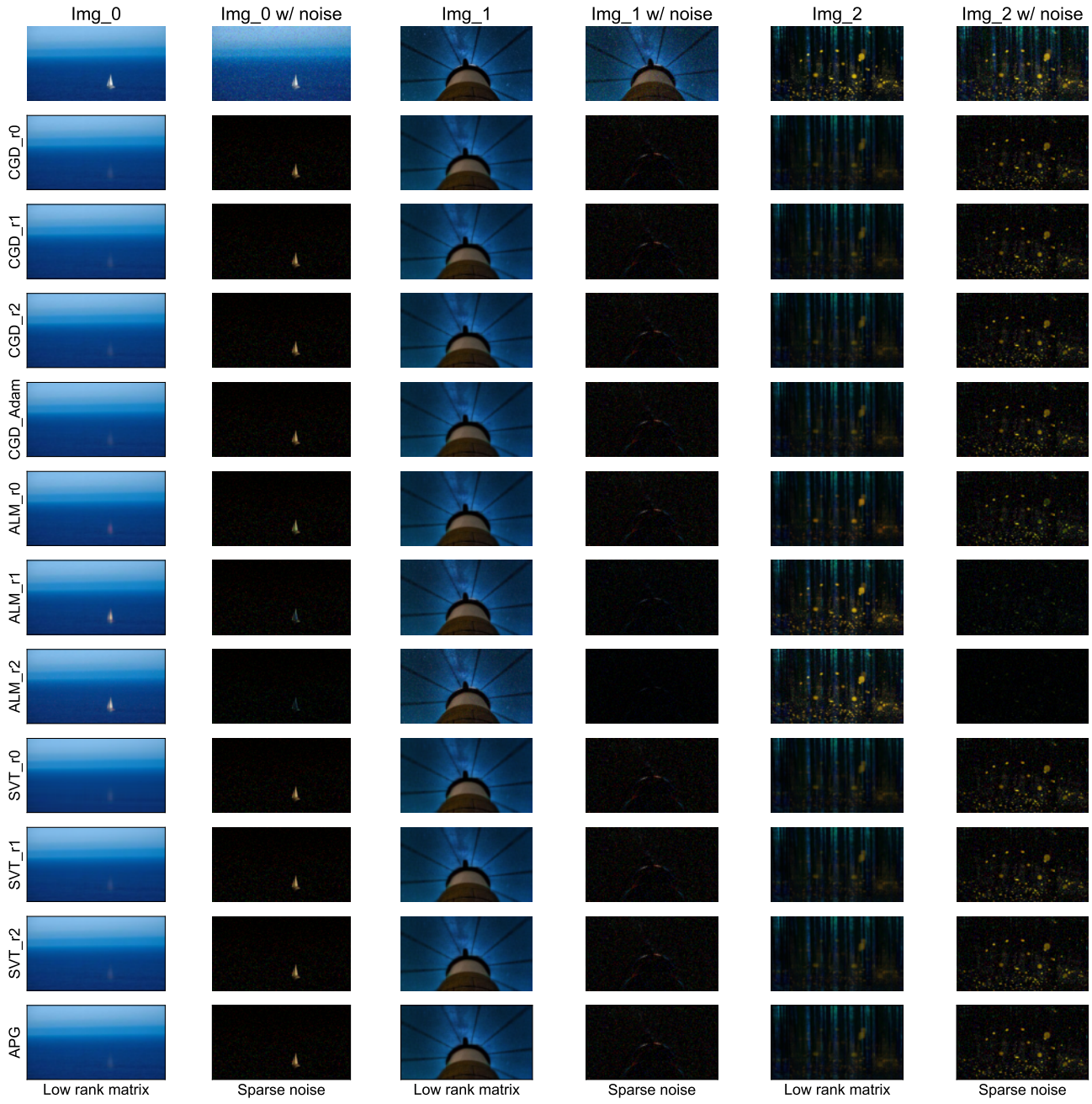

图 3: 鲁棒主成分分析实验结果

表 3: 鲁棒主成分分析各算法性能指标

Image	Algorithm	Time (s)	rank($\boldsymbol{L}$)	$\|\boldsymbol{S}\|_0$	$\|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_{m_1}$
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
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

5.2.3 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.

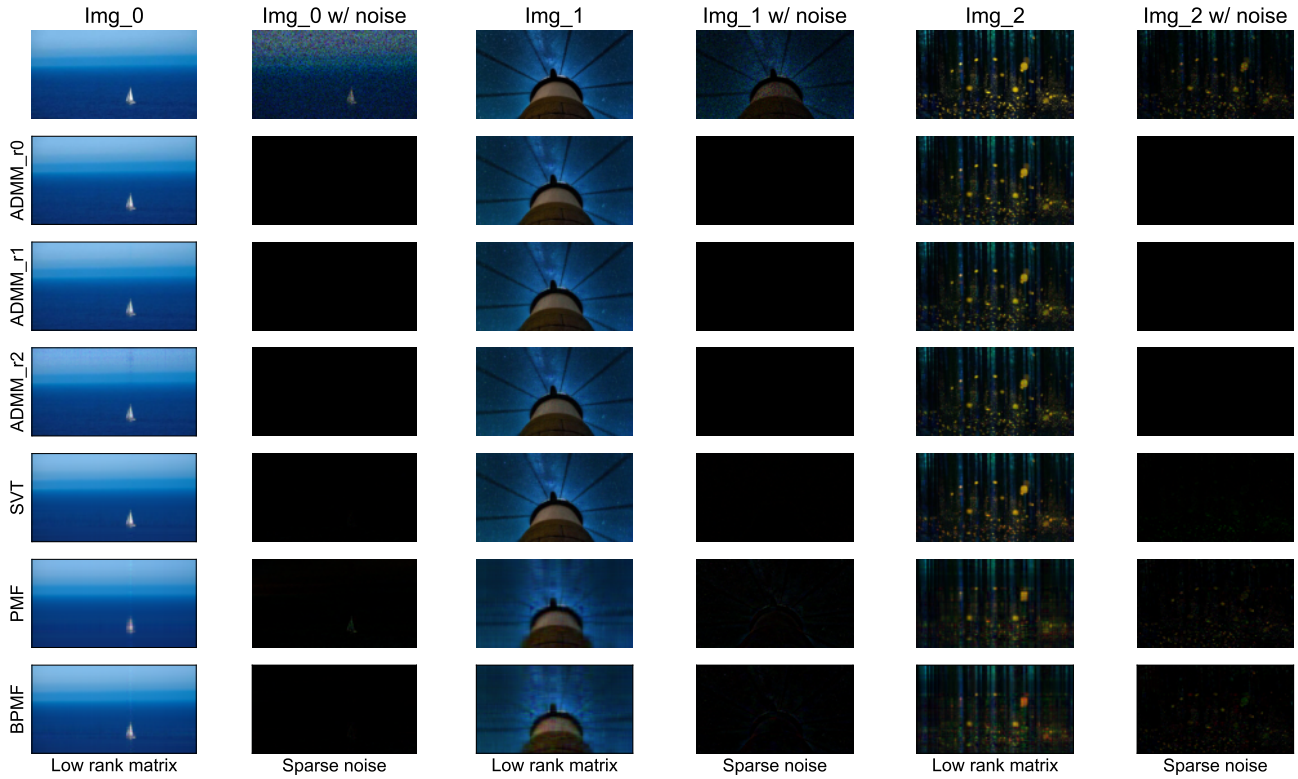

图 4: 矩阵补全实验结果

图 4 左侧命名带有 r0, r1, r2 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	r0	r1	r2
ADMM	$\mu \leftarrow 0.05\mu$	$\mu \leftarrow 0.1\mu$	$\mu \leftarrow 0.15\mu$

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.

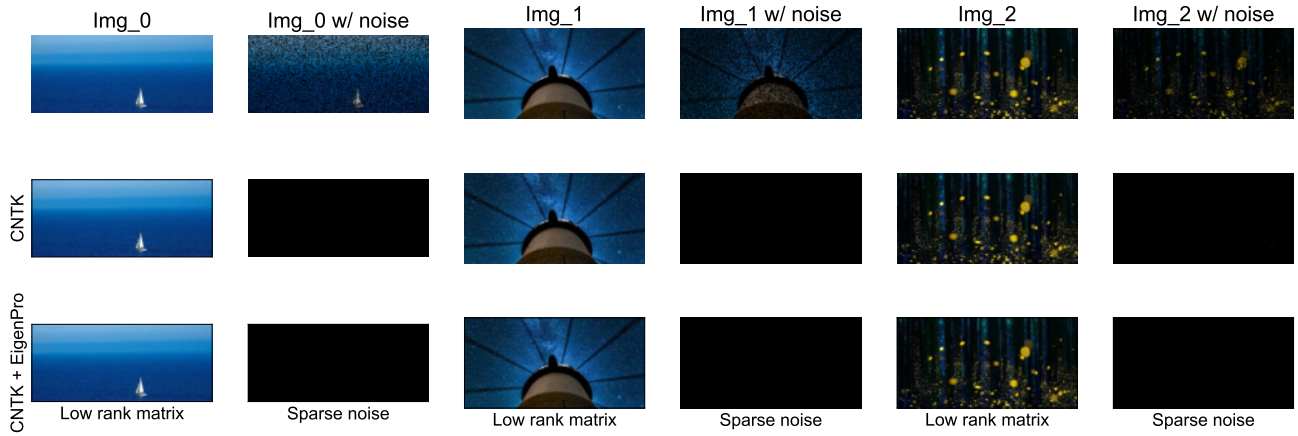

图 5: 矩阵补全 CNTK 方法实验结果

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.

表 5: 矩阵补全各算法性能指标

Image	Algorithm	Time (s)	rank($\boldsymbol{L}$)	$\|\boldsymbol{S}\|_0$	$\|\boldsymbol{L}\|_*$
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
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

6 代码说明

6.1 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/`. 另外, `ntk/` 文件夹为文献 [27] 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
`robust_pca.py`	`_augmented_Lagrange_multipliers()`	Robust-PCA
`matrix_completion.py`	`_singular_value_thresholding()`	matrix-completion
`matrix_completion.py`	`_probabilistic_matrix_factorization()`	matrix-completion
`matrix_completion.py`	`_biased_probabilistic_matrix_factorization()`	matrix-completion

6.2 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
`bin/`	testing algorithms, analyzing results, and plotting figures
`data/`	source data and results
`doc/`	documentation
`ntk/`	the CNTK algorithm
`pre/`	slides
`utils/`	algorithms

7 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)

8 结论

本文针对低秩矩阵恢复问题进行探究, 主要研究了鲁棒主成分分析算法以及矩阵补全算法. 针对鲁棒主成分分析算法, 给出了基于梯度下降算法的通用求解方法, 并基于 Adam 算法进行优化, 然后给出了 3 种特殊优化算法: SVT, APG 与 ALM. 针对矩阵补全问题, 给出了矩阵补全问题的数学模型, 以及 4 种具体求解算法: SVT, ADMM, ALS, CNTK. 本文对上述算法在图像数据集上的效果进行了测试, 结果表明 PCA 算法的求解时间最短, 但是效果比较一般, 鲁棒主成分分析算法的效果更好, 但是耗时普遍较长, 其中 SVT 算法的用时较短且取得了相对更好的效果; 矩阵补全算法也可以实现低秩矩阵恢复, ADMM 和 SVT 算法的效果均比较好但用时较长, CNTK 算法效果较好, 若与 EigenPro 联合使用可以达到用时少与效果好的双重目的. 本文还对不同超参数的实验结果进行了测试和分析.

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] 史加荣, 郑秀云, 魏宗田, 杨威. 低秩矩阵恢复算法综述. 计算机应用研究, 30(6):1601–1605, 2013.

[3] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[4] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[5] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery

of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012.

[6] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2862–2869, 2014.

[7] 王兴趣, 贾世会, 迟晓妮. 广义加权鲁棒主成分分析 (gwrpca) 的模型与算法. *系统科学与数学*, 41(12):3363, 2021.

[8] 郝运琪. 低秩矩阵恢复问题的qa-pgl算法及应用. *大连理工大学*, 2021.

[9] 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.

[10] 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.

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

[12] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[13] Quanming Yao and James T Kwok. Accelerated inexact soft-impute for fast large-scale matrix completion. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[14] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[15] Dingtao Peng, Naihua Xiu, and Jian Yu. S-1/2 regularization methods and fixed point algorithms for affine rank minimization problems. *Computational Optimization and Applications*, 67(3):543–569, 2017.

[16] Zhi Wang, Wendong Wang, Jianjun Wang, and Siqi Chen. Fast and efficient algorithm for matrix completion via closed-form 2/3-thresholding operator. *Neurocomputing*, 330:212–222, 2019.

[17] 冯伟, 谢冬秀. 基于加权核范数的低秩矩阵近似及其应用. *计算机应用*, 40:128–131, 2020.

[18] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

[19] G Alistair Watson. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170(0):33–45, 1992.

[20] 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).

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12(2), 2009.

[23] 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.

[24] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

[25] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[26] 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.

[27] 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.