核范数随机矩阵求解新方法及其 RPCA 应用

王 臻,杨 敏
(南京邮电大学 自动化学院,江苏 南京 210023)

摘 要:RPCA(稳健主成分分析)从原始观测数据中恢复低秩成分和稀疏成分。RPCA 常用交替方向法迭代求解,算法的效率取决于核范数优化求解,即 SVD 分解。而 RPCA 在计算机视觉应用中,图像和视频等巨大的数据量为大规模数据 SVD 分解带来了很大困难。采用随机矩阵算法对 SVD 分解进行改进,分别为计数缩略算法、标准随机 k -SVD 算法和快速随机 k -SVD 算法。主要是对原有大规模数据矩阵进行降维随机采样,使用随机投影算法得到原数据矩阵的一个近似,对这个近似矩阵进行 QR 分解,得到对应的酉矩阵。对酉矩阵进行相关操作,得到与原矩阵 SVD 相似的结果。算法的时间效率和存储空间得到极大改善。基于单张图像和视频前景检测等仿真实验,表明所提方法大大提高了 RPCA 迭代优化求解的效率。

关键词:稳健主成分分析;交替方向法;标准随机 k -SVD;快速随机 k -SVD
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2017)12-0071-06
doi:10.3969/j.issn.1673-629X.2017.12.016

A New Method for Solving Nuclear Norm with Random Matrix and Its Application in Robust Principal Component Analysis

WANG Zhen, YANG Min
(College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: RPCA (Robust Principal Component Analysis) recovers sparse and low rank components from the original observation data. It commonly uses ADM (Alternate Direction Method) for iterative solving, the efficiency of which depends on the nuclear norm optimization solution, that is SVD. The application of RPCA in computer vision, large amounts of data from images and video make it difficult for large-scale data SVD. Therefore, a random matrix algorithm is adopted to improve the SVD, respectively the algorithm of count sketch, the prototype randomized k -SVD and the faster randomized k -SVD. Its main idea is to reduce the size of the original large-scale data matrix and sample randomly. Using the random projection algorithm to obtain an approximation of the original matrix, and operating QR decomposition of this approximate matrix, the unitary matrix corresponding to it is obtained, and then the results which is similar to the SVD can be achieved through correlated operation of unitary matrix. The time and space of the algorithm have been greatly optimized. Simulation based on single image and video foreground detection shows that the proposed method can greatly improve the efficiency of RPCA iterative optimization.

Key words: RPCA; ADM; prototype randomized k -SVD; faster randomized k -SVD

0 引 言

PCA 在高维数据样本中寻找和挖掘低维特征空间。在较小的高斯随机噪声时,可通过奇异值分解准确求解。当不满足上述条件时,所得结果会有很大偏差,于是用 RPCA 来解决此种情况。RPCA 即低秩矩阵恢复[1],又称为稀疏与低秩矩阵分解[2]。

RPCA 模型(稳健主成分分析)[3]在视频前景提取、人脸识别图像预处理等计算机视觉领域中应用广泛。求解 RPCA 常用交替方向法[4],充分利用 RPCA 模型具有的很好的可分离结构特性,是对带有线性约束条件凸规划问题的增广拉格朗日乘子法的一种改进。算法每次迭代的主要计算量在于 SVD 分解[5]。像矩阵求逆、特征值分解、奇异值分解等这些矩阵计算,不但非常耗时,存储所需的空间也很大,因此,这些缺点限制了它的拓展和应用范围。为了对一个大规模的矩阵进行计算,随机矩阵近似技术[6]应运而生。随

收稿日期:2016-10-21 **修回日期**:2017-02-23 **网络出版时间**:2017-08-01
基金项目:国家自然科学基金资助项目(61271234)
作者简介:王 臻(1992-),男,硕士,研究方向为核范数极小化随机优化求解;杨 敏,博士,副教授,研究方向为计算机视觉与图像理解。

机矩阵计算已经在现代数据科学领域占有举足轻重的地位。特别在过去的十年里,随机数字线性代数取得了卓越进步,现在大规模的矩阵计算不再是一个不可完成的任务了。

文中将改进的 SVD 分解,即随机矩阵算法应用到视频图像的前景检测中。对于交替方向法中每次迭代时,对核范数的优化求解,即大规模 SVD 分解,进行优化改进,对于大规模的数据矩阵进行随机近似求解,大大提高了算法效率。数据表明,改进算法具有更快的计算速度和存储优势。

1 RPCA 模型及交替方向法求解算法

1.1 RPCA 模型

RPCA 问题可以抽象描述[7]为:已知观测数据矩阵 $\boldsymbol{M}$,且 $\boldsymbol{M}=\boldsymbol{L}+\boldsymbol{S}$,而 $\boldsymbol{L}$ 和 $\boldsymbol{S}$ 是未知的,但是已知 $\boldsymbol{L}$ 为低秩,$\boldsymbol{S}$ 为稀疏且非零元素可以任意大,要求恢复 $\boldsymbol{L}$。基于问题的描述,首先想到的解决方法是寻求观测数据中主成分 $\boldsymbol{L}$ 的最小秩,且干扰误差矩阵 $\boldsymbol{S}$ 是稀疏的,即非零元素个数较少。于是形成了如下优化问题:

$$\begin{cases}\min\limits_{L,S}\operatorname{rank}(\boldsymbol{L})+\lambda\|\boldsymbol{S}\|_0\\ \text{s.t. } \boldsymbol{L}+\boldsymbol{S}=\boldsymbol{M}\end{cases}\tag{1}$$

其中,$\|\cdot\|_0$ 表示 $\boldsymbol{S}$ 中非零元的个数。

通过对式(1)做松弛变化,可以把问题转化为一个易于解决的凸优化问题,即用 1 范数代替 0 范数,用 $\boldsymbol{L}$ 的核范数代替 $\boldsymbol{L}$ 的秩。式(1)转化为:

$$\begin{cases}\min\limits_{L,S}\|\boldsymbol{L}\|_*+\lambda\|\boldsymbol{S}\|_1\\ \text{s.t. } \boldsymbol{L}+\boldsymbol{S}=\boldsymbol{M}\end{cases}\tag{2}$$

这个凸优化问题,在一定条件下,可以求解并能得到理想的恢复结果($\boldsymbol{L},\boldsymbol{S}$),且解是唯一的。

1.2 交替方向法

交替方向法面向目标函数可分解[8](即为两个不同变量凸函数的和)、带线性约束和凸集约束的问题,它是 Lagrange 乘子法的一种变形。先构造问题的增广 Lagrangian 函数,然后通过交替极小化增广 Lagrangian 函数来更新两个变量,最后更新 Lagrange 乘子,如此迭代。其好处是更新变量子问题要比原问题简单,甚至有闭解。

对于稳健 PCA 问题(式(2)),易得其增广拉格朗日函数为:

$$\Gamma(\boldsymbol{L},\boldsymbol{S},\boldsymbol{Y})=\|\boldsymbol{L}\|_*+\lambda\|\boldsymbol{S}\|_1+\langle\boldsymbol{Y},\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\rangle+\frac{\mu}{2}\|\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\|_F^2\tag{3}$$

其中,$\boldsymbol{Y}\in R^{m\times n}$ 为线性约束乘子;$\mu>0$ 为标量,是违背线性约束的惩罚参数;$\langle\cdot\rangle$ 表示标准内积;$\|\cdot\|_F$ 表示 Frobenius 范数。

很明显,可以直接运用经典增广拉格朗日乘子法求解,其迭代主题是:

$$\begin{cases}(\boldsymbol{L}_{K+1},\boldsymbol{S}_{K+1})=\arg\min\limits_{L,S}\|\boldsymbol{L}\|_*+\lambda\|\boldsymbol{S}\|_1+\\ \qquad\langle\boldsymbol{Y}_K,\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\rangle+\frac{\mu}{2}\|\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\|_F^2\\ \boldsymbol{Y}_{K+1}=\boldsymbol{Y}_K+\mu(\boldsymbol{L}_K+\boldsymbol{S}_K-\boldsymbol{M})\end{cases}\tag{4}$$

每个迭代是一个大型的非光滑优化问题。经典的增广拉格朗日乘子法把式(2)看作一个一般的最小化问题,而忽视了它在目标函数和约束条件中都体现出的很好的可分离结构特性,即式(4)是同时最小化 $\boldsymbol{L}$ 和 $\boldsymbol{S}$ 的。

而对于 $\boldsymbol{S}$ 的极小化是非常容易得到的:

$$\arg\min_S\Gamma(\boldsymbol{L},\boldsymbol{S},\boldsymbol{Y})=\arg\min_S\|\boldsymbol{L}\|_*+\lambda\|\boldsymbol{S}\|_1+\langle\boldsymbol{Y},\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\rangle+\frac{\mu}{2}\|\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}\|_F^2=$$

$$\arg\min_S\lambda\|\boldsymbol{S}\|_1+\frac{\mu}{2}\|\boldsymbol{S}-(\boldsymbol{M}-\boldsymbol{L}-\frac{1}{\mu}\boldsymbol{Y})\|_F^2+$$

$$\varphi(\boldsymbol{L},\boldsymbol{M},\boldsymbol{Y})=\operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}(\boldsymbol{M}-\boldsymbol{L}-\mu^{-1}\boldsymbol{Y})\tag{5}$$

对于 $\boldsymbol{L}$ 的极小化也很容易得到:

$$\arg\min_L\Gamma(\boldsymbol{L},\boldsymbol{S},\boldsymbol{Y})=\operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}(\boldsymbol{M}-\boldsymbol{S}-\mu^{-1}\boldsymbol{Y})\tag{6}$$

交替更新,可以得到结果:

$$\begin{cases}\boldsymbol{L}_{K+1}=\operatorname{argmin}\Gamma(\boldsymbol{L},\boldsymbol{S}_K,\boldsymbol{Y}_K)=\\ \qquad\operatorname{prox}_{\mu^{-1}\|\cdot\|_*}(\boldsymbol{M}-\boldsymbol{S}_K-\mu^{-1}\boldsymbol{Y}_K)\\ \boldsymbol{S}_{K+1}=\operatorname{argmin}\Gamma(\boldsymbol{L}_{K+1},\boldsymbol{S},\boldsymbol{Y}_K)=\\ \qquad\operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}(\boldsymbol{M}-\boldsymbol{L}_{K+1}-\mu^{-1}\boldsymbol{Y}_K)\\ \boldsymbol{Y}_{K+1}=\boldsymbol{Y}_K+\mu(\boldsymbol{L}_{K+1}+\boldsymbol{S}_{K+1}-\boldsymbol{M})\end{cases}\tag{7}$$

其中,prox 是在如下空间的欧几里得投影:

$$\lambda\mu^{-1}\|\cdot\|_1:=\{X\in R^{n\times n}\mid-\frac{\lambda}{\mu}\leqslant X_{ij}\leqslant\frac{\lambda}{\mu}\}\tag{8}$$

惩罚参数比较适合于动态调整,从相关文献[9-11]可以了解具有动态变化参数的交替方向法的收敛性以及一些具体有效的调整参数的方法。

交替方向法求解 RPCA 问题(式(4))时,算法效率取决于每一次迭代的核范数优化求解,即大规模的 SVD 分解。因此,SVD 算法的好坏对于交替方向法处理大规模矩阵的稀疏和低秩矩阵恢复问题有很重要的影响。

2 随机矩阵算法

在对矩阵进行操作前,通常会对大规模矩阵进行降维操作。矩阵的降维方式一般有随机投影和列选择两种方式[12]。假设给定一个矩阵 $\boldsymbol{A}\in R^{m\times n}$,而 $\boldsymbol{S}\in$

$R^{n\times s}$ 是一个缩略矩阵,比如是一个随机投影或者是列选择矩阵[13], $\boldsymbol{C}=\boldsymbol{AS}\in R^{m\times s}$ 是 $\boldsymbol{A}$ 的一个缩略。矩阵 $\boldsymbol{C}$ 的大小是远小于矩阵 $\boldsymbol{A}$ 的,但是 $\boldsymbol{C}$ 却保存了矩阵 $\boldsymbol{A}$ 的一些重要性质。文中使用的降维方式主要是使用随机投影中的计数缩略算法[14]。

算法 1:计数缩略算法。

输入矩阵 $\boldsymbol{A}\in R^{m\times n}$ 及缩略矩阵的列数 s;

初始化 $\boldsymbol{C}$ 为一个 $m\times s$ 的全零矩阵;

For$i=1,2,\cdots,n$ do;

(1)随机均匀采样矩阵 $\boldsymbol{C}$ 的 l 个列向量;

(2)随机均匀采样 g 个+1 或者-1;

(3)通过将矩阵 $\boldsymbol{C}$ 的第 l 个列的各元素加上 $g\times$矩阵 $\boldsymbol{A}$ 的第 i 个列向量来更新矩阵 $\boldsymbol{C}$ 的第 l 个列;

End For

得到缩略矩阵 $\boldsymbol{C}\in R^{m\times s}$。

该算法有以下一些性质:

(1)时间复杂度为 $O(nnz(\boldsymbol{A}))$

(2)空间复杂度为 $O(ms)$。当矩阵 $\boldsymbol{A}$ 不适合存储时,该算法将矩阵 $\boldsymbol{C}$ 保存在内存中,并且只经历一次矩阵 $\boldsymbol{A}$ 的各列向量。

(3)理论保证

①当 $s=\tilde{O}(\frac{m^2}{\varepsilon^2\delta})$,这个子空间植入特性保留了 $\gamma=1+\varepsilon$ 误差率和 δ 的可能性;

②当 $s=\frac{2k}{\varepsilon\delta}$,这个低秩近似特性保留了 $\eta=1+\varepsilon$ 的相对误差率和 δ 的可能性。

(4)计数矩阵缩略算法在矩阵 $\boldsymbol{A}$ 是稀疏矩阵时特别有效。

3　改进的 SVD 算法

3.1　标准随机 k–SVD 算法

本节描述标准随机 k –SVD 算法[15],它计算矩阵 $\boldsymbol{A}$ 的 k –SVD 分解并且到达了 $1+\varepsilon$ 的 F 范数相对误差。算法描述如下:

算法 2:标准随机 k –SVD 算法。

输入:目标秩 k 和矩阵 $\boldsymbol{A}\in R^{m\times n}$;

$\boldsymbol{S}\in R^{n\times s}$, $\boldsymbol{C}=\boldsymbol{AS}\in R^{m\times s}$

利用计数缩略算法构造缩略矩阵 $\boldsymbol{C}\in R^{m\times s}$,这里 $s=O(\frac{k}{\varepsilon})$;

对矩阵 $\boldsymbol{C}$ 进行 QR 分解, $[\underbrace{\boldsymbol{Q}_C}_{m\times s},\boldsymbol{R}_C]=\mathrm{qr}(\underset{m\times s}{\boldsymbol{C}})$;

$[\underset{s\times k}{\tilde{\boldsymbol{U}}},\underset{k\times k}{\tilde{\boldsymbol{\Sigma}}},\underset{n\times k}{\tilde{\boldsymbol{V}}}]=\mathrm{svd}(\underbrace{\boldsymbol{Q}_C^{\mathrm{T}}\boldsymbol{A}}_{s\times n},k)$: $\boldsymbol{Q}_C^{\mathrm{T}}\boldsymbol{A}$ 进行奇异值分解并且只取前 k 个向量得到 $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Sigma}}$;

$\tilde{\boldsymbol{U}}=\boldsymbol{Q}_C\tilde{\boldsymbol{U}}\in R^{m\times k}$;

得出 $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\mathrm{T}}\approx\boldsymbol{A}_k$

假如 $\boldsymbol{C}=\boldsymbol{AS}\in R^{m\times s}$ 是 $\boldsymbol{A}$ 的一个比较好的缩略矩阵, $\boldsymbol{C}$ 的列空间应该可以大致地包含 $\boldsymbol{A}_k$ 的列—这就是低秩近似特性[16]。假如 $\boldsymbol{S}\in R^{n\times s}$ 是一个高斯投影矩阵或是计数缩略矩阵,并且 $s=O(\frac{k}{\varepsilon})$ 时,那么低秩近似特性约束期望为:

$$\min_{\mathrm{rank}(\boldsymbol{Z})\leqslant k}\|\boldsymbol{CZ}-\boldsymbol{A}\|_F^2\leqslant(1+\varepsilon)\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2 \quad (9)$$

算法推导:

由于 $\boldsymbol{C}$ 的列空间和 $\boldsymbol{Q}_C$ 的列空间是一样的,式(9)的极小化问题就可以被等价转换为:

$$\boldsymbol{X}^*=\min_{\mathrm{rank}(\boldsymbol{X})\leqslant k}\|\underbrace{\boldsymbol{Q}_C}_{m\times s}\underset{s\times n}{\boldsymbol{X}}-\underset{m\times n}{\boldsymbol{A}}\|_F^2=(\boldsymbol{Q}_C^{\mathrm{T}}\boldsymbol{A})_k \quad (10)$$

这里第二个等式是已知事实。矩阵 $\boldsymbol{A}_k$ 是由 $\tilde{\boldsymbol{A}}_k:=\boldsymbol{Q}_C\boldsymbol{X}^*$ 得到的非常好的近似,所以只需要找出 $\tilde{\boldsymbol{A}}_k$ 的 k –SVD 分解:

$$\tilde{\boldsymbol{A}}_k:=\underbrace{\boldsymbol{Q}_C}_{m\times s}\underbrace{\boldsymbol{X}^*}_{s\times n}=\boldsymbol{Q}_C\underbrace{(\boldsymbol{Q}_C^{\mathrm{T}}\boldsymbol{A})_k}_{:=\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\mathrm{T}}}=\underbrace{\boldsymbol{Q}_C\tilde{\boldsymbol{U}}}_{:=\tilde{\boldsymbol{U}}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\mathrm{T}}=\underbrace{\tilde{\boldsymbol{U}}}_{m\times k}\underbrace{\tilde{\boldsymbol{\Sigma}}}_{k\times k}\underbrace{\tilde{\boldsymbol{V}}^{\mathrm{T}}}_{k\times n} \quad (11)$$

非常容易能够检验出 $\tilde{\boldsymbol{U}}$ 和 $\tilde{\boldsymbol{V}}$ 是标准正交矩阵,而 $\tilde{\boldsymbol{\Sigma}}$ 是对角矩阵。需要注意到,随机 k –SVD 分解的精度只依赖于缩略矩阵 $\boldsymbol{C}$ 的质量。

该算法有以下几个特性:

(1)该算法只有 2 次经过 $\boldsymbol{A}$;

(2)算法只在内存中保存了一个 $m\times\mathrm{O}(\frac{k}{\varepsilon})$ 大小的缩略矩阵 $\boldsymbol{C}$;

(3)时间复杂度为 $O(nnz(\boldsymbol{A})k/\varepsilon)$。

3.2　快速随机 k–SVD 算法

标准算法将大多数时间花费在解决式(10)上。假如式(10)可以更高效地解决,那么标准随机 k –SVD 可以变得更加迅速。可以注意到,可以通过不精确的最小二乘回归算法[17]解决式(10)的问题。

现在构造一个 $m\times p$ 大小的计数缩略(或者是高斯投影+计数缩略)矩阵 $\boldsymbol{P}$ 来解决这个问题:

$$\tilde{\boldsymbol{X}}=\min_{\mathrm{rank}(\boldsymbol{X})\leqslant k}\|\underbrace{\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Q}_C}_{p\times s}\underset{s\times n}{\boldsymbol{X}}-\underbrace{\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}}_{p\times n}\|_F^2 \quad (12)$$

让 $\boldsymbol{P}=\underbrace{\boldsymbol{P}_{cs}}_{m\times p_{cs}}\underbrace{\boldsymbol{P}_{srht}}_{p_{cs}\times p}$,其中 $p_{cs}=s^2\log^6\frac{s}{\varepsilon}+\frac{s}{\varepsilon}$, $p=\frac{s}{\varepsilon}\log\frac{s}{\varepsilon}$。计数缩略矩阵的子空间植入性质表明:

$$(1+\varepsilon)^{-1}\|\boldsymbol{Q}_C\tilde{\boldsymbol{X}}-\boldsymbol{A}\|_F^2\leqslant\|\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{Q}_C\tilde{\boldsymbol{X}}-\boldsymbol{A})\|_F^2\leqslant\|\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{Q}_C\boldsymbol{X}^*-\boldsymbol{A})\|_F^2\leqslant(1+\varepsilon)\|\boldsymbol{Q}_C\boldsymbol{X}^*-$$

$$\|\boldsymbol{A}\|_F^2 \Rightarrow \|\boldsymbol{Q}_C\tilde{\boldsymbol{X}} - \boldsymbol{A}\|_F^2 \leqslant (1+\varepsilon)^2 \|\boldsymbol{Q}_C\boldsymbol{X}^* - \boldsymbol{A}\|_F^2 \leqslant (1+\varepsilon)^3 \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \tag{13}$$

这里第二个不等式是来源于 $\tilde{\boldsymbol{X}}$ 的最优解,最后一个不等式是来源于缩略矩阵 $\boldsymbol{C} = \boldsymbol{AS}$ 的低秩近似特性。因此,通过求解式(13),可以得到一个达到 $1+O(\varepsilon)$ 的 F 范数相对误差。

算法描述如下:

算法 3 :快速随机 k –SVD 算法。

输入:目标秩 k 和矩阵 $\boldsymbol{A} \in R^{m\times n}$;

令参数 $s = \mathrm{O}(\frac{k}{\varepsilon})$, $p_{cs} = s^2 \log^6 \frac{s}{\varepsilon} + \frac{s}{\varepsilon}$, $p = \frac{s}{\varepsilon} \log \frac{s}{\varepsilon}$;

构造一个 $n \times s$ 计数缩略矩阵 $\boldsymbol{S}$,然后令 $\boldsymbol{C} = \boldsymbol{AS}$;

构造一个 $m \times p_{cs}$ 的计数缩略矩阵 $\boldsymbol{P}_{cs}$,还有一个 $p_{cs} \times p$ 的矩阵 $\boldsymbol{P}_{srht}$;

操作缩略矩阵 $\boldsymbol{D} = \boldsymbol{P}_{srht}^{\mathrm{T}}\boldsymbol{P}_{cs}\boldsymbol{C}^{\mathrm{T}} \in R^{p\times s}$ 和 $\boldsymbol{L} = \boldsymbol{P}_{srht}^{\mathrm{T}}\boldsymbol{P}_{cs}^{\mathrm{T}}\boldsymbol{A} \in R^{p\times n}$;

对矩阵 $\boldsymbol{D}$ 进行 QR 分解:$[\underbrace{\boldsymbol{Q}_D}_{p\times s}, \underbrace{\boldsymbol{R}_D}_{s\times s}] = \mathrm{qr}(\underset{p\times s}{\boldsymbol{D}})$;

$[\underset{s\times k}{\bar{\boldsymbol{U}}}, \underbrace{\bar{\boldsymbol{\Sigma}}}_{k\times k}, \underset{n\times k}{\bar{\boldsymbol{V}}}] = \mathrm{svds}(\underbrace{\boldsymbol{Q}_D^{\mathrm{T}}\boldsymbol{L}}_{s\times n}, k)$ 进行 K–SVD 分解;

对 $[\underset{n\times k}{\tilde{\boldsymbol{U}}}, \underbrace{\tilde{\boldsymbol{\Sigma}}}_{k\times k}, \underset{k\times k}{\hat{\boldsymbol{V}}}] = \mathrm{svd}(\underbrace{\boldsymbol{CR}_D^{\dagger}\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}}_{s\times k})$ 进行 SVD 分解;

$\tilde{\boldsymbol{V}} = \underbrace{\bar{\boldsymbol{V}}}_{n\times k}\underbrace{\hat{\boldsymbol{V}}}_{k\times k}$;

最后得到 $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\mathrm{T}} \approx \boldsymbol{A}_k$

算法推导:

快速随机 k –SVD 算法解决了式(12)。为了获得秩为 k 的矩阵 $\tilde{\boldsymbol{X}} \in R^{c\times n}$,通过 $\boldsymbol{A}_k \approx \boldsymbol{C}\tilde{\boldsymbol{X}}$ 来近似 $\boldsymbol{A}_k$,定义 $\boldsymbol{D} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{C}$,$\boldsymbol{L} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}$,令 $\boldsymbol{Q}_D\boldsymbol{R}_D = \boldsymbol{D}$ 进行 QR 分解。然后式(12)式变形为:

$$\tilde{\boldsymbol{X}} = \min_{\mathrm{rank}(\boldsymbol{X})\leqslant k} \|\underset{p\times s}{\boldsymbol{D}}\underset{s\times n}{\boldsymbol{X}} - \underset{p\times n}{\boldsymbol{L}}\|_F^2 = \boldsymbol{D}^{\dagger}\boldsymbol{Q}_D(\boldsymbol{Q}_D^{\mathrm{T}}\boldsymbol{L})_k = \underbrace{\boldsymbol{R}_D^{\dagger}}_{s\times s}\underbrace{(\boldsymbol{Q}_D^{\mathrm{T}}\boldsymbol{L})_k}_{s\times n} \tag{14}$$

基于此,通过式(15)进行分解。

$$\boldsymbol{A}_k \approx \boldsymbol{C}\tilde{\boldsymbol{X}} = \boldsymbol{CR}_D^{\dagger}(\boldsymbol{Q}_D^{\mathrm{T}}\boldsymbol{L})_k = \boldsymbol{CR}_D^{\dagger}\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{V}}^{\mathrm{T}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^{\mathrm{T}}\bar{\boldsymbol{V}}^{\mathrm{T}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}} \tag{15}$$

这里需要注意:

(1)该算法只经过 2 次矩阵 $\boldsymbol{A}$;

(2)算法的时间复杂度为 $O(nnz(\boldsymbol{A}) + (m+n)\mathrm{poly}(k/\varepsilon))$;

(3)参数应该满足 $k < s < p_{cs} < p < m, n$;

(4)算法中的缩略算法也可以使用其他的矩阵缩略方式;

(5)矩阵 $\boldsymbol{A}$,矩阵 sketch,矩阵 $\boldsymbol{L}$ 这些都是整个系统中存储花销最大的,但是幸运的是,它们只需要 1 次或 2 次扫描。假如矩阵 $\boldsymbol{A}$,缩略矩阵,矩阵 $\boldsymbol{L}$ 不适合随机存储,那么它们应该被存储到硬盘中,再逐个部分被加载到内存中进行计算。

4 仿真实验

4.1 单张图像分解仿真

测试图像为 MATLAB 自带图片,其大小为 512 * 512。如图 1 所示,前 10 个奇异值中,奇异值的衰减速度相当快,并且逐渐趋向于 0,图像表现为低秩矩阵。现分别使用 SVD 算法、标准随机 k –SVD 算法以及快速随机 k –SVD 算法进行低秩矩阵重建仿真,比较算法之间的运行效率、恢复精度,以及与目标秩之间的关系。

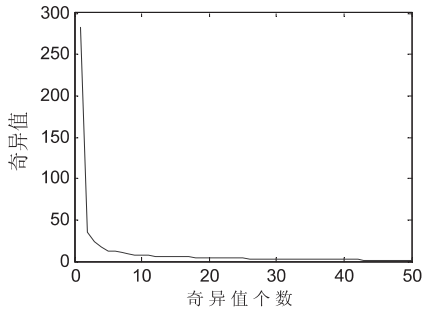

图 1 奇异值衰减图解

重建结果与原图的比较如图 2 所示。

SVD 标准随机 k-SVD 快速随机 k-SVD

图 2 三种 SVD 分解效果图

这是分别使用三种 SVD 算法,利用主成分累加和逼近原图,得到原图的低秩逼近效果图。可以看出,三种算法都很好地实现了对于原矩阵的处理。

图 3 为标准随机 k –SVD 算法和快速随机 k –SVD 算法的计算精度随目标秩变化的曲线。

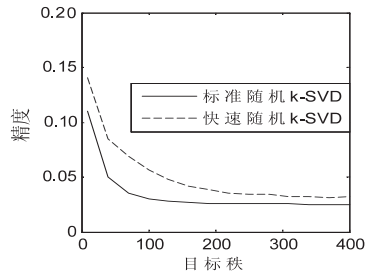

图 3 两种改进算法的精度比较

由图 3 可以看出,标准随机 k –SVD 算法的分解精度要比快速随机 k –SVD 算法高。随着目标秩的升高,

两种随机算法的精度都逐渐提高并趋向平稳。

图 4 为三种 SVD 算法的计算时间随目标秩的变化曲线。

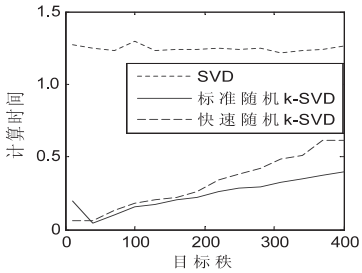

图 4　三种算法的计算时间比较

由图 4 可以看出,两种改进算法的计算时间都随目标秩的增加而增加。由于快速随机 k –SVD 每次计算都需要构造好几个缩略矩阵,所以在目标秩一定的情况下,标准随机 k –SVD 比快速随机 k –SVD 用时短。而 MATLAB 自带的 SVD 每次计算的时间几乎是不变的,比两种改进算法都要耗时。

4.2　视频前景提取仿真

待操作数据矩阵大小为 20 800 * 200,取视频中的 4 帧。取目标秩 k 为 50,并且同时构造出原矩阵的缩略矩阵 $\boldsymbol{C}$,其中 s 取 60。仿真结果如图 5 所示。

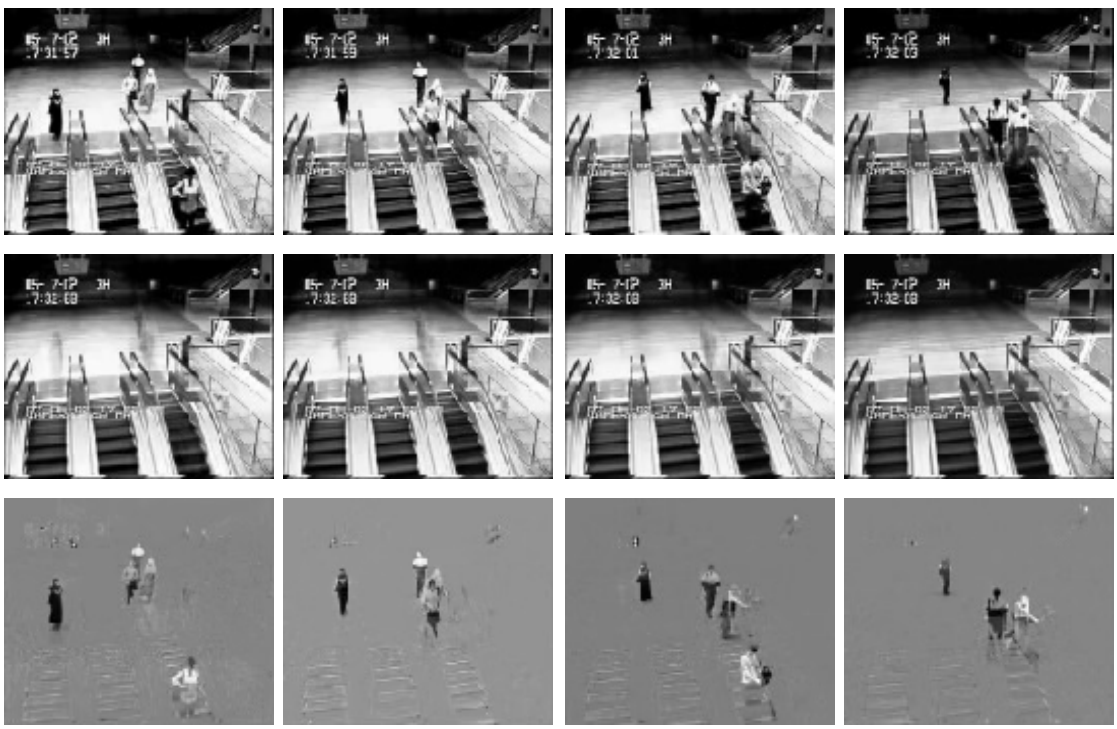

图 5　视频前景提取效果

对于快速随机 k –SVD 算法,同样先构造缩略矩阵 C , p 和 p_{cs} 分别取 100 和 80,取值条件需满足 $k < s < p_{cs} < p < m,n$ 。

如图 5 所示,第一行是视频中的 4 帧数据。第二行是文中改进算法对每帧数据分解得出的低秩部分,即背景。最后一行是分解得出的稀疏部分,即视频的运动前景。可以看出,改进算法都较好地实现了视频前景与背景的分离。

三种 SVD 算法的比较如表 1 所示。

表 1　三种 SVD 算法的比较

算法	算法运行时间/s	算法迭代次数/次
SVD	72.16	60
标准随机 k –SVD	32.02	71
快速随机 k –SVD	27.43	48

从表 1 的运行结果可以看出,两种改进的随机 k –SVD 分解与普通的 SVD 分解一样也实现了前景和背景的分离。标准随机 k –SVD 算法虽然迭代次数比普通的 SVD 算法要多一些,但是运行时间却大幅缩短,并且由于采用缩略矩阵算法,程序对于内存的占用也相对较少,很好地实现了算法的改进。快速随机 k –SVD 算法虽然每次迭代的时间比标准随机 k –SVD 算法要长一些,因为快速随机 k –SVD 算法中进行了多次的矩阵缩略降维,但是其迭代次数比普通的 SVD 算法和标准随机 k –SVD 算法都要少,运行时间也更短。虽然该算法中使用了多个缩略矩阵,比较占内存,但幸运的是,它们只需 1 次或 2 次的扫描就可以实现算法的功能。实验结果证明,该算法也有良好的收敛性,提高了核范数求解的效率。

5　结束语

针对 RPCA 模型中用来实现图像前景和背景分离的交替方向法,对矩阵算法进行了研究和改进。通过

分析,当用交替方向法求解 RPCA 问题时,每次计算量消耗最大的就在于核范数的优化求解,即大规模 SVD 的计算。于是介绍了随机矩阵算法,并提出了两种新的随机 SVD 算法。两种算法都大大提高了代码的运行效率,并且经过仿真实验表明,两种算法达到的实际分离效果和普通的 SVD 分解也相差无几。总之,两种改进的随机奇异值分解算法都加快了算法的收敛速度,很好地提高了算法效率,并且都不需要占用大量的内存空间。

参考文献:

[1] Candès E J,Li X,Ma Y,et al. Robust principal component analysis[J]. Journal of the ACM,2011,58(3):11-12.

[2] Lin Z,Chen M,Wu L,et al. The augmented Lagrange multiplier method for exact recovery of a corrupted low-rank matrices [R]. Illinois: University of Illinois at Urbana-Champaign, 2010.

[3] 江明阳,封举富. 基于鲁棒主成分分析的人脸子空间重构方法[J]. 计算机辅助设计与图形学学报,2012,24(6):761-765.

[4] Yuan X,Yang J. Sparse and low-rank matrix decomposition via alternating direction methods[R]. Hong Kong:Hong Kong Baptist University,2009.

[5] Wang Ruoxi, Li Yingzhou, Mahoney M W,et al. Structured block basis factorization for scalable kernel matrix evaluation [J]. Numerische Mathematik,2015,83:313-323.

[6] Mahoney M W. Randomized algorithms for matrices and data [J]. Foundations and Trends in Machine Learning,2013,3 (2):647-672.

[7] 戴琼海,付长军,季向阳. 压缩感知研究[J]. 计算机学报, 2011,34(3):425-434.

[8] 彭义刚,索津莉,戴琼海,等. 从压缩传感到低秩矩阵恢复: 理论与应用[J]. 自动化学报,2013,39(7):981-994.

[9] He B S,Liao L Z,Han D,et al. A new inexact alternating directions method for monontone variational inequalities[J]. Mathematical Programming,2002,92(1):103-118.

[10] He B S,Yang H. Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities[J]. Operations Research Letters,1998,23(1):151-161.

[11] Kontogiorgis S,Meyer R R. A variable-penalty alternating directions method for convex optimization [J]. Mathematical Programming,1998,83(1):29-53.

[12] Clarkson K,Woodruff D P. Low rank approximation and regression in input sparsity time[C]//Proceedings of annual ACM symposium on theory of computing. [s. l.]:ACM,2013: 121-128.

[13] Woodruff D P. Sketching as a tool for numerical linear algebra [J]. Foundations and Trends in Theoretical Computer Science,2014,10(1-2):1-157.

[14] Wang Shusen,Zhang Zhihua,Zhang Tong. Towards more efficient symmetric matrix sketching and cur matrix decomposition [J]. Journal of Machine Learning Research,2015,13:2729-2769.

[15] Halko N,Martinsson P G,Tropp J A. Finding structure with randomness:probabilistic algorithms for constructing approximate matrix decompositions[J]. SIAM Review,2011,53(2): 217-288.

[16] Gorinov S A,Tyrtyshnikov E E. The maximum-volume concept in approximation by low-rank matrices[J]. Contemp. Math. ,2001,280:47-51.

[17] Halko N. Randomized methods for computing low-rank approximations of matrices[D]. Colorado:University of Colorado,2012.