矩阵分析与应用大作业-附录文档

Xueke Cheng,　2021210956
Shuhan Du,　2021210969

1　技术路线

CSSP 问题描述：对于给定的矩阵 $A \in \mathbb{R}^{m \times n}$ 从其列向量集合 $\{a1, a2, \ldots, an\}$ 中选择一个 k 元子集 S，从而最小化 $Er_A(S) := ||A - P_S A||_F^2$。其中 $||\cdot||_F$ 是 Frobenius 范数，P_s 是从 A 的列空间到 span(S) 的投影矩阵。

1.1　符号说明

1. H_k^i 表示除 H_{ii} 为 0，其余对角元均为 1 的 $k \times k$ 维对角阵。
2. C^+ 表示矩阵 C 的 Moore-Penrose 逆。
3. 对于矩阵 A 和序列 R，A_R 表示由 A 中的 R 列所构成的子集。
4. $A_{i:}$ 表示 A 的第 i 行。
5. $A_{:i}$ 表示 A 的第 i 列。
6. $A \backslash i$ 表示去掉 A 的第 i 列所得到的 $m \times (n-1)$ 维矩阵。
7. $R[i]$ 表示序列 R 的第 i 个元素。
8. f_i 表示向量 f 的第 i 个分量。
9. $\circ$ 表示取 Hardamard 积。
10. 对于两个矩阵 A 和 B，$(A|B)$ 表示矩阵 $[A\ B]$。
11. e_i 是指定维度的标准基的第 i 列。

1.2 基本思路

不少已有算法基于以下两种思路来解决 CSSP 问题：从 S 为空集开始，以最小化 $Er_A(S)$ 为目标从给定矩阵的列向量集合中依次选出 S 的 k 项元素；或者设计指标比较每个列向量的表现，直接选取最优的 k 项构成 S。而我们对于文献 [1] 中的快速迭代算法做出复现，该算法采取了一种新的思路：首先随机生成一个 k 元子集 C，以最小化 $Er_A(C)$ 为目标依次替换 C 的每一列，替换完成后的 C 即为所求。该过程如算法 1所示，首先从 $\{1,\ldots,n\}$ 中均匀随机不放回地抽取 k 个数构成序列集合 R，C 表示由 A 中的 R 列所构成的子集，$\tilde{C}$ 则表示将 C 的第 i 列置零所得到的矩阵。设置收敛条件为 $Er_A(C)$ 不再减小，即找到了最优替换方案。

Algorithm 1 Prototype

$R \leftarrow$ a random initial subset of k column numbers
$C \leftarrow A_R$
while not converged **do**
  for $i = 1, \ldots, k$ **do**
    $\tilde{C} \leftarrow CH^i{}_k$
    $w \leftarrow \arg\min_j ||A - P_{(\tilde{C}|A_{:j})}A||_F$
    $C \leftarrow (\tilde{C}|A_{:w})\backslash i$
    $R[i] \leftarrow w$
    $C \leftarrow A_R$
  end for
end while
output R

接下来，基于一些理论推导对算法 1进行优化，使其同时适用于低维和高维矩阵。

1.3 理论推导

优化的要点在于加速 C^+ 的计算（也即投影矩阵的计算），这里给出一些关键性的结论，证明略。

命题 1. 令 $\rho = ((C^+)i:)^T$，有：

$$\tilde{C}^+ = C^+ - ||\rho||_2^{-2} C^+ \rho\rho^T. \tag{1}$$

命题 2. 令 $E = A - P_C A = A - CC^+A$，$\tilde{E} = A - P_{\tilde{C}}A = A - \tilde{C}\tilde{C}^+A$，有：

$$\tilde{E} = E + C_{:i}\rho^T A + ||\rho||_2^{-2}\tilde{C}C^+\rho\rho^T A. \tag{2}$$

命题 3. 令 $F = E^TE$，$\tilde{F} = \tilde{E}^T\tilde{E}$，$f = (||F_{:1}||_2^2, \ldots, ||F_{:n}||_2^2)$，$g = (F_{11}, \ldots, F_{nn})$，$\tilde{f} = (||\tilde{F}_{:1}||_2^2, \ldots, ||\tilde{F}_{:n}||_2^2)$，$g = (\tilde{F}_{11}, \ldots, \tilde{F}_{nn})$，$\delta = \tilde{E}_{:j}^T\tilde{E}$，$\gamma = E^TE\delta^T$ 有：

$$\tilde{f} = f + ||\delta||_2^2(\delta\circ\delta)\delta_j^{-2} + 2(\gamma\circ\delta)\delta_j^{-1}, \tag{3}$$

$$\tilde{g} = g + (\delta \circ \delta)\delta_j^{-1}. \tag{4}$$

命题 4. 令 $\omega = \tilde{E}_{:w}$，有：

$$C^+ = \tilde{C}^+ - ||\omega||_2^{-2}(\tilde{C}^+ A_{:w}\omega^T - e_i\omega^T). \tag{5}$$

由以上四个结论最终推出 E 的更新方式如下：

$$E = \tilde{E} - \omega\omega^T\tilde{E}||\omega||_2^{-2}. \tag{6}$$

现在来优化寻找 ω 的过程，也即寻找 w 的过程。由算法 1可知，$w = \arg\min_j ||A - P_{(\tilde{C}|A_{:j})}A||_F$，我们根据下面的定理对其进行简化：

定理 1.

$$\begin{aligned}\arg\min_i ||A - (C|A_{:i})(C|A_{:i})^T A||_F &= \arg\max_i \frac{||E^T E_{:i}||_2^2}{||E_{:i}||_2^2}, \\ &= \arg\max_i \frac{||E_{:i}||_2^2}{F_{ii}}, \\ &= \arg\max_i \frac{\tilde{f}_i}{\tilde{g}_i}. \end{aligned} \tag{7}$$

至此，对算法 1主体部分的优化已经完成，最后对矩阵 A 行数非常大 $(m \gg n)$ 或列数非常大 $(n \gg m)$ 的特殊情况进行优化，以使算法能更快地处理高维矩阵。改进步骤基于以下定理：

定理 2. 对于 $A \in \mathbb{R}^{m\times n}$，$m > n$，$rank(A) = n$，令 $U^T AV = \Sigma$ 为其奇异值分解，有

$$||A - CC^+A||_F^2 = ||\Sigma V^T - (U^T C)(U^T C)^+\Sigma V^T||_F^2, \tag{8}$$

定理 3. 对于 $E = A - CC^+A$，令 $\tilde{U}^T E\tilde{V} = \tilde{\Sigma}$ 为其奇异值分解，有

$$f_i = ||(\tilde{\Sigma}^2\tilde{V}^T)_{:i}||_2^2, \tag{9}$$

$$g_i = ||(\tilde{\Sigma}\tilde{V}^T)_{:i}||_2^2. \tag{10}$$

由此推出，当 $m \gg n$ 时，可用 ΣV^T 代替 A 进行后续计算；而当 $n \gg m$ 时，可用 $f_i = ||(\tilde{\Sigma}^2\tilde{V}^T)_{:i}||_2^2$ 代替 $f_i = ||E^T E_{:i}||_2^2$、$g_i = ||(\tilde{\Sigma}\tilde{V}^T)_{:i}||_2^2$ 代替 $g_i = ||E_{:i}||_2^2$ 进行后续计算。

2 算法实现

按照上述理论推导对算法 1进行优化后得到算法 2，即为实际实现的算法，即对文献 [1] 快速迭代算法的复现。

Algorithm 2 Optimized Version

$R \leftarrow$ a random initial subset of k column numbers
if m > n **then**
  $A \leftarrow \Sigma V^T$
end if
$C \leftarrow A_R$
$E \leftarrow A - CC^+A$
if n > m **then**
  $f_i \leftarrow ||(\tilde{\Sigma}^2 \tilde{V}^T)_{:i}||_2^2$; $g_i \leftarrow ||(\tilde{\Sigma}\tilde{V}^T)_{:i}||_2^2$
else
  $f_i \leftarrow ||E^T E_{:i}||_2^2$; $g_i \leftarrow ||E_{:i}||_2^2$
end if
while not converged **do**
  for $i = 1, \ldots, k$ **do**
    $j \leftarrow R[i]$
    $\tilde{C} \leftarrow C{H^i}_k$
    $\tilde{C}^+ \leftarrow C^+ - ||\rho||_2^{-2} C^+ \rho\rho^T$
    $\tilde{E} = E + C_{:i}\rho^T A + ||\rho||_2^{-2} \tilde{C} C^+ \rho\rho^T A$
    $\delta \leftarrow \tilde{E}_{:j}^T \tilde{E}$; $\gamma \leftarrow E^T E \delta^T$
    $\tilde{f} \leftarrow f + ||\delta||_2^2 (\delta \circ \delta)\delta_j^{-2} + 2(\gamma \circ \delta)\delta_j^{-1}$
    $\tilde{g} = g + (\delta \circ \delta)\delta_j^{-1}$
    $w \leftarrow \arg\min_h \frac{\tilde{f}_h}{\tilde{g}_h}$
    $\delta \leftarrow \tilde{E}_{:w}^T \tilde{E}$; $\gamma \leftarrow \tilde{E}^T \tilde{E} \delta^T$
    $f \leftarrow \tilde{f} + ||\delta||_2^2 (\delta \circ \delta)\delta_w^{-2} - 2(\gamma \circ \delta)\delta_w^{-1}$
    $g \leftarrow \tilde{g} - (\delta \circ \delta)\delta_w^{-1}$
    $C^+ \leftarrow \tilde{C}^+ - ||\omega||_2^{-2} (\tilde{C}^+ A_{:w}\omega^T - e_i\omega^T)$
    $E \leftarrow \tilde{E} - \omega\omega^T \tilde{E} ||\omega||_2^{-2}$
    $R[i] \leftarrow w$
    $C \leftarrow A_R$
  end for
end while
output R

3 算法分析与对比

3.1 复杂度

该算法的时间复杂度主要取决于循环的复杂度，循环中主要的步骤在于 C^+ 和 E 的更新，以及 δ 和 γ 的计算。

1. C^+ 的计算（一个循环内）
   由 C 直接计算 C^+ 伪逆的计算时间复杂度一般在 $O(mk^2)$，例如列递推法。
   算法中使用 $\tilde{C}^+ = C^+ - ||\rho||_2^{-2} C^+ \rho\rho^T$ 和 $C^+ = \tilde{C}^+ - ||\omega||_2^{-2}(\tilde{C}^+ A_{:w}\omega^T - e_i\omega^T)$ 优化后，时间复杂度减小到 $O(mk)$。

2. E、δ 和 γ 的计算（一个循环内）

$$\because \tilde{E} = E + C_{:i}\rho^T A + ||\rho||_2^{-2}\tilde{C}C^+\rho\rho^T A \therefore \text{复杂度为} O(mn).$$

$$\because \delta \leftarrow \tilde{E}_{:j}^T \tilde{E}; \gamma \leftarrow E^T E \delta^T \therefore \delta\text{和}\gamma\text{的计算时间复杂度是}O(mn).$$

$$\because \tilde{f} \leftarrow f + ||\delta||_2^2(\delta\circ\delta)_j^{-2} + 2(\gamma\circ\delta)_j^{-1}; \tilde{g} = g + (\delta\circ\delta)_j^{-1} \therefore \tilde{f}\text{和}\tilde{g}\text{的计算时间复杂度是}O(n).$$

$$\because w \leftarrow \arg\min_h \frac{\tilde{f}_h}{\tilde{g}_h}\text{是排序问题} \therefore \text{复杂度小于}O(n^2).$$

$$\because \delta \leftarrow \tilde{E}_{:w}^T \tilde{E}; \gamma \leftarrow \tilde{E}^T \tilde{E} \delta^T \therefore \delta\text{和}\gamma\text{的计算时间复杂度是}O(mn).$$

$$\because f \leftarrow \tilde{f} + ||\delta||_2^2(\delta\circ\delta)_w^{-2} - 2(\gamma\circ\delta)_w^{-1}; g \leftarrow \tilde{g} - (\delta\circ\delta)_w^{-1} \therefore f\text{和}g\text{的计算时间复杂度是}O(n).$$

$$\because E \leftarrow \tilde{E} - \omega\omega^T\tilde{E}||\omega||_2^{-2} \therefore \text{复杂度为}O(mn).$$

因此，在一个循环内，主要操作的时间复杂度为 $O(mn)$（因为 k 小于 n）。另一方面，由于候选子集的每个列都需要经过循环内的操作，因此循环次数为 k，因此每次子集迭代的复杂度为 $O(mnk)$。

当 m 远大于 n 时，根据定理 2，可以将循环的复杂度减少到 $O(n^2k)$，但在算法初始阶段，时间复杂度为 $O(mn^2)$ 操作。由于现有的奇异值分解高效且通常为并行，因此 E^TE 计算的复杂度为 $O(mn^2)$，在 $n > m$ 时，根据定理 3，可以减小到 $O(m^2n)$。

3.2 精度

这里讨论算法 2（快速迭代算法）给出的近似结果 C（也即 A_R）对应的 $Er_A(C) := ||A - P_C A||_F^2 = ||A - CC^+A||_F^2$ 的上界，也对应 $||CC^+A||_F^2$ 的下界，在文献 [2] 中有所证明。根据文献 [3]，可以进行如下证明。引理 1 给出了 **Greedy** 算法中每次迭代时 $||CC^+A||_F^2$ 增量的下界。

引理 1. 设矩阵 B 为 CSSP 问题的精确最优解，C 为 A 的列向量集合的子集所构成的矩阵，那么有 $||CC^+A||_F^2 \leqslant ||BB^+A||_F^2$。不失一般性，对 B 进行归一化，使其列向量均为单位向量，对于 B 中的某一列 v，若 $X=(C|v)$，有：

$$||XX^+A||_F^2 - ||CC^+A||_F^2 \geqslant \sigma_k^2(B)\frac{(||BB^+A||_F^2 - ||CC^+A||_F^2)^2}{4k||BB^+A||_F^2}, \tag{11}$$

其中 $\sigma_k(B)$ 表示 B 的第 k 大的奇异值。

基于引理 1，得到如下定理：

定理 1. 设算法 2给出的近似结果为 C（也即 A_R），矩阵 B 为 CSSP 问题的精确最优解，对 B 进行归一化，对于整数 $k \geqslant 2$，有：

$$||CC^+A||_F^2 \geqslant \sigma_k^2(B)\frac{\sigma_k^4(BB^+A)}{4||BB^+A||_F^2} + \sum_{i=n-k+2}^{n}\sigma_i^2(A), \tag{12}$$

下面给出证明。

从第一次迭代开始，对于随机生成的 $m \times k$ 维矩阵 C，令 $\bar{C}_i = C \backslash i$，那么 CC^+A 可以表示为 $CC^+A = \check{C}_i + \hat{C}_i$，其中 $\check{C}_i = \bar{C}_i\bar{C}_i^+A$，表示 CC^+A 在 $\bar{C}_i$ 的列空间上的分量，而 $\hat{C}_i = CC^+A - \bar{C}_i\bar{C}_i^+A$ 表示与之正交的部分，那么 $||\hat{C}_i||_F^2$ 就是将 C 的第 i 列去掉时 $||CC^+A||_F^2$ 所损失的量，因此有：

$$\sum_{i=1}^{k}||\hat{C}_i||_F^2 \leqslant ||CC^+A||_F^2, \tag{13}$$

且 C 中至少有一列 u 满足：

$$||\hat{C}_u||_F^2 \leqslant \frac{||CC^+A||_F^2}{k}. \tag{14}$$

另一方面，考虑到将 C 的第 i 列去掉时，剩余的 $k-1$ 列（也即 $\bar{C}_i$）的范数保持不变，因此对于任意 i，有：

$$||\hat{C}_i||_F^2 \leqslant ||CC^+A||_F^2 - ||\bar{C}_i||_F^2. \tag{15}$$

根据奇异值的性质，对于 $i=1,\dots,k$，$\sigma_i(\bar{C}_i) \geqslant \sigma_{n-k+i+1}(A)$。再根据 Frobenius 范数的定义，得到 $||\bar{C}_i||_F^2 \geqslant \sum_{i=n-k+2}^{n}\sigma_i^2(A)$。因此，$C$ 中存在一列 u 满足：

$$||\hat{C}_u||_F^2 \leqslant \frac{1}{k}(||CC^+A||_F^2 - \sum_{i=n-k+2}^{n}\sigma_i^2(A)). \tag{16}$$

将上式与引理 1 相结合，可知 C 中至少有一列 u 满足：当它被替换后，$||CC^+A||_F^2$ 增加 $\sigma_k^2(B)\frac{(||BB^+A||_F^2 - ||\bar{C}_u\bar{C}_u^+A||_F^2)^2}{4k||BB^+A||_F^2} - \frac{1}{k}(||CC^+A||_F^2 - \sum_{i=n-k+2}^{n}\sigma_i^2(A))$。根据算法 2的设定，迭代停止时，该增量非负，因此有：

$$\frac{1}{k}||CC^+A||_F^2 \geqslant \sigma_k^2(B)\frac{(||BB^+A||_F^2 - ||\bar{C}_u\bar{C}_u^+A||_F^2)^2}{4k||BB^+A||_F^2} + \frac{1}{k}\sum_{i=n-k+2}^{n}\sigma_i^2(A). \tag{17}$$

最后，由于 $\bar{C}_u\bar{C}_u^+A$ 是 BB^+A 的 $k-1$ 维近似，推出：

$$\frac{1}{k}||CC^+A||_F^2 \geqslant \sigma_k^2(B)\frac{\sigma_k^4(BB^+A)}{4k||BB^+A||_F^2} + \frac{1}{k}\sum_{i=n-k+2}^{n}\sigma_i^2(A), \tag{18}$$

也即定理 1 所给出的结论。

3.3 与 SVD 的理论对比

根据课上所学, 对于给定矩阵 A, 可以利用奇异值分解对其进行低秩逼近。设 $A \in \mathbb{R}^{m\times n}$ 的奇异值分解为 $A = U\Sigma V^T$，且 $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_t \geq 0, r = rank(A)$。令

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T, k \leq r$$

则 $\min_{rank(X)\leq k} ||A - X||_F = ||A - A_k||_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_r^2}$，即 A_k 是 A 的最佳秩 k 逼近，在代码中被注释掉的部分可以证明等式的成立。

因此奇异值分解的输入为逼近矩阵的秩，而 CSSP 问题中，可能出现 $rank(A) < k$ 的情况，因此 CSSP 逼近的误差很可能大于 SVD 低秩逼近的误差。且奇异值低秩逼近法计算的步骤很少，只有奇异值分解和求和两步，因此其计算复杂度小于快速迭代法。

4 数值实验结果

4.1 精度验证

低秩情况下，遍历法可以在可接受时间内计算完毕，因此以遍历法为参照标准。可以看到本次作业复现的快速迭代法与遍历法相差较小。分别计算 $||BB^+A||_F^2$ $||CC^+A||_F^2$ 和 $\sigma_k^2(B)\frac{\sigma_k^4(BB^+A)}{4||BB^+A||_F^2} + \sum_{i=n-k+2}^{n} \sigma_i^2(A)$，可以得到算式 12 的不等关系。奇异值低秩逼近的误差小于该情况。下图为某次运算的结果，可以看到算式 12 的不等关系成立，奇异值低秩逼近的误差小于遍历法小于快速迭代法，且遍历法和快速迭代法的误差很相近。

```
E_traversal algorithm
   35.0964

E_fast iterative algorithm
   35.8676

E_SVD Echkart-Young
   19.5329

Lower bound
   75.9576

||CC+A||_F^2
  272.2018

||BB+A||_F^2
  273.2898
```

图 1: 不同方法精度结果

4.2 运行时间

对于同一低秩矩阵，重复运行算法 20 次计算总时间并取平均作为运行一次的时间。结果如图，可以看到本算法的时间复杂度远远小于遍历法，但大于奇异值低秩逼近，同时本算法与遍历法取到的列重复率较高。

```
t_fast iterative algorithm
    2.7200

t_fast iterative algorithm
    0.0200

t_SVD Echkart-Young
    0.0030

R_traversal algorithm
     1     2     4     6     7     9    10    12    13    15

R_fast iterative algorithm
     2     4     7     8     9    10    11    12    13    15
```

图 2: 不同方法运行时间结果

对于不同高维矩阵，改变矩阵的列数，保持选择的列数 k 不变，考察随着原始矩阵列数增大运行时间的变化。随机生成一个 200 行 250 列的矩阵，保持 $k = 100$，选取 A 矩阵，A 的列为该随机矩阵的前 n 列，n 从 100 取到 250。运行时间结果如下图，可以看到随着矩阵维数增大，运行时间增大。

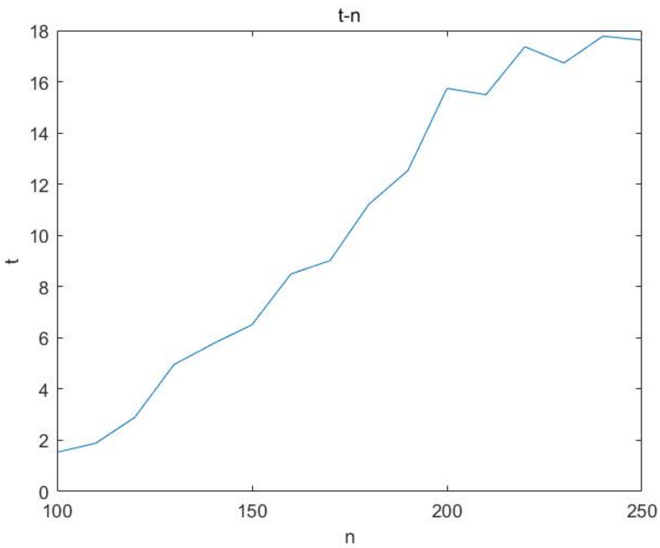

图 3: 运行时间随列数改变结果

5 代码自述

1. travelsal_CSSP.m：遍历法求解 CSSP 问题函数，输入为原始矩阵 A 和选择列数 k，输出为选中的列号。

2. approx_CSSP.m：快速迭代法（本次作业复现的算法）求解 CSSP 问题函数，输入为原始矩阵 A 和选择列数 k 以及重复迭代次数 ITER_NUM，输出为选中的列号。

3. SVD_EchkartYoung.m：奇异值分解低秩逼近函数，输入为原始矩阵 A 和低秩矩阵的秩 k，输出为秩为 k 的矩阵和误差（估计矩阵和原始矩阵差的 Frobeniues 范数的平方）。

4. main_low.m：直接运行文件，低秩情况下三种方法的对比（误差和运行时间）以及精度不等式的验算。

5. main_high.m：直接运行文件，高维情况下运行时间随着维数增加的变化，结果以折线图表示。

6 主要参考文献

[1] Ordozgoiti B , Canaval S G , Mozo A . A Fast Iterative Algorithm for Improved Unsupervised Feature Selection[C]// IEEE International Conference on Data Mining. IEEE, 2017.

[2] Altschuler J , Bhaskara A , Gang, et al. Greedy Column Subset Selection: New Bounds and Distributed Algorithms[J]. JMLR.org, 2016.

[3] Ordozgoiti B , Canaval S G , Mozo A . Iterative column subset selection[J]. Knowledge and Information Systems, 2017

注：文献 [1] 作者有公开代码，但是我们仅根据论文内容进行复现。