Vol.2018-MPS-119 No.12
2018/7/30

TRL法を用いた密行列のための部分特異値分解の改良

青木 雅奈[1,a)]　髙田 雅美[2,b)]　木村 欣司[3,c)]　中村 佳正[1,d)]

概要：本稿では, 密行列の部分特異値分解を行うための方法を提案する. 回帰分析では, 一般的に, QR 分解が行われる. しかしながら, いくつかの独立変数間に多重線形性が存在する場合, 回帰分析の代わりに, 大きいほうからいくつかの特異値とその特異ベクトルを用いて主成分回帰をするべきである. また, 回帰分析の対象となる観測データは, しばしば, 大規模密行列となる. ゆえに, 高速に部分特異値分解を行うための手法の開発が望まれている. そこで, 本稿では, thick-restart-Lanczos (TRL) 法を用いる. TRL 法は, TRLAN に実装されている方法で, 部分固有値分解を行うことができる. 疎行列を対象とした部分特異値分解を部分固有値分解に変換するための方法は, Takata らが提案している. この Takata 法は, 疎行列を用いた主成分分析のために開発された手法であり, Implicitly-restarted-Lanczos (IRL) 法に適用することを想定している. 本稿では, この Takata 法を密行列の特異値分解のために改良する. 実験では, 部分特異値分解を行う従来法が適用された TRL 法と提案法が適用された TRL 法を比較する.

Improvement of Computing Partial Singular Value Decomposition for Dense Matrix using TRL Method

Masana Aoki[1,a)]　Masami Takata[2,b)]　Kinji Kimura[3,c)]　Yoshimasa Nakamura[1,d)]

1. はじめに

回帰分析では, その問題の性質から大規模な密行列を対象とする場合が多い. いくつかの独立変数間に多重線形性が存在する場合には, QR 分解を用いて解くことのできる回帰分析の代わりに, 大きいほうからいくつかの特異値とその特異ベクトルを用いて主成分回帰をするべきである. ゆえに, 大規模な密行列の部分的な特異値分解が求められている.

大きいほうからいくつかの固有値とそれに対応する固有ベクトルを計算するための方法として, Implicitly-restarted-Lanczos (IRL) 法 [3] と Thick-restart-Lanczos (TRL) 法 [5] が知られている. IRL 法は, ARPACK [2] に, TRL 法は, TRLAN [6] に実装されている. ARPACK も TRLAN もどちらも長年利用されているため, バグが解消されている. ゆえに, ARPACK や TRLAN を用いて部分固有値分解を行うことは適切である.

特異値分解は, 与えられた行列とその転置行列を乗じることによって, 固有値分解に変換することができる. この際, 固有値分解で得られる固有値は, すべて非負の値となる. この変換式を利用して, Takata らは, 疎行列を対象とした, ARPACK を用いた部分特異値分解法を提案している [4]. すなわち, Takata 法では, ARPACK の IRL 法の改良を行っている. Takata 法は, 共有メモリを持つマルチコアプロセッサのキャッシュを考慮する. そのため, OpenMP を用いた実装において, CPU のキャッシュを効果的に活用できる方法である. その結果, 並列計算環境において, 高い性能を達成している. 一方, 本稿では, 回帰分析の 1 つである主成分回帰を対象としているため, 密行列を対象とする. そのため, 密行列を扱えるように Takata 法を改良する必要がある. さらに, IRL 法よりも高速な TRL 法を利用す

[1] 京都大学
Kyoto University, Kyoto, Kyoto 606–8501, JAPAN
[2] 奈良女子大学
Nara Women's University, Nara, Nara 630–8506, JAPAN
[3] サレジオ工業高等専門学校
Salesian Polytechnic, Machida, Tokyo 194–0215, JAPAN
[a)] aoki.masana.36e@st.kyoto-u.ac.jp
[b)] takata@ics.nara-wu.ac.jp
[c)] kimura.kinji.7z@kyoto-u.ac.jp
[d)] ynaka@i.kyoto-u.ac.jp

ることによって, より高速な実装を行う.

2 章では, 特異値分解を固有値分解に変換する方法を説明する. 3 章では, Takata らの方法を概説する. 4 章では, Takata らの方法を利用して, 密行列のための計算法を提案する. 5 章では, 従来法による TRLAN と提案法による TRLAN の実行結果を比較する.

2. 特異値分解を固有値分解に変換する方法

2.1 特異値分解

実行列 $B \in \mathbb{R}^{w\times n}$ は, $w \times w$ 直交行列 U と $n \times n$ 直交行列 V を用いて

$$B = U\Sigma V^\top \quad (\sigma_i > 0). \tag{1}$$

のように分解できる. ここで,

$$r_B := \operatorname{rank} B, \tag{2}$$

$$\Sigma := \begin{bmatrix} \sigma_1 & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & \sigma_{r_B} & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

$$(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r_B} > 0) \tag{3}$$

である. また, U と V は,

$$U^\top U = UU^\top = I_w, \tag{4}$$

$$V^\top V = VV^\top = I_n. \tag{5}$$

を満たす. I_w と I_n は, それぞれ, $w \times w$ 次と $n \times n$ 次の単位行列である. 正の実数 $\sigma_1, \ \ldots, \ \sigma_{r_B}$, U と V は, それぞれ, 特異値, 左特異ベクトルと右特異ベクトルを意味する. 式 (1) は, U と V の i 番目のベクトル $\boldsymbol{u}_i$ と $\boldsymbol{v}_i$ を用いて

$$B\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i, \tag{6}$$

$$B^\top \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i \quad (i = 1, \ \ldots, \ r_B), \tag{7}$$

に変換される.

2.2 固有値問題への変換

式 (6) と (7) によって,

$$B^{(r)\top} B^{(r)} \boldsymbol{v}_i^{(r)} = B^{(r)\top} \left(\sigma_i^{(r)} \boldsymbol{u}_i^{(r)} \right) = \sigma_i^{(r)} \left(B^{(r)\top} \boldsymbol{u}_i^{(r)} \right)$$
$$= {\sigma_i^{(r)}}^2 \boldsymbol{v}_j^{(r)}, \tag{8}$$

$$\boldsymbol{u}_i^{(r)} = \frac{B^{(r)} \boldsymbol{v}_i^{(r)}}{||B^{(r)} \boldsymbol{v}_i^{(r)}||_2} \quad (i = 1, \ \ldots, \ r_B^{(r)}), \tag{9}$$

Algorithm 1 Takata らの方法

```
1: r = 0;
2: #omp parallel for private(t) reduction(+:r)
3: for i = 1 to n do
4:     t = ⟨b_i, x⟩ (b_i = B(i,:));
5:     r = r + t b_i^⊤;
6: end for
7: #omp end parallel for
```

が満たされる. ここで, $B^{(r)} \in \mathbb{R}^{w\times n}$ $(w \geq n)$, $\sigma_i^{(r)}$, $\boldsymbol{u}_i^{(r)}$, $\boldsymbol{v}_i^{(r)}$, $r_B^{(r)}$ は, それぞれ, 実長方行列, i 番目の特異値, i 番目の左特異ベクトル, i 番目の右特異ベクトル, $\operatorname{rank} B^{(r)}$ である. 式 (8) を用いて, $B^{(r)}$ の特異値分解は, ${B^{(r)}}^\top B^{(r)}$ の固有値分解に変換することができる. この時, ${\sigma_i^{(r)}}^2$ は, ${B^{(r)}}^\top B^{(r)}$ の固有値と等しい. ゆえに, ${B^{(r)}}^\top B^{(r)}$ を TRL 法に入力することによって, $B^{(r)}$ の特異値 $\sigma_i^{(r)}$ と右特異ベクトル $\boldsymbol{v}_i^{(r)}$ を得ることができる. さらに, 式 (9) によって, 左特異ベクトル $\boldsymbol{u}_i^{(r)}$ が計算可能である.

長方行列 $B^{(c)} \in \mathbb{R}^{w\times n}$ $(w < n)$ に関して, ${B^{(c)}}^\top$ は $B^{(r)}$ と同じ形式の入力行列となる. よって, $\sigma_i^{(c)}$, $\boldsymbol{u}_i^{(c)}$, $\boldsymbol{v}_i^{(c)}$ を, それぞれ, $B^{(c)}$ の特異値, 左特異ベクトル, 右特異ベクトルとすると,

$$B^{(c)} {B^{(c)}}^\top \boldsymbol{u}_i^{(c)} = B^{(c)} \left(\sigma_i^{(c)} \boldsymbol{v}_i^{(c)} \right) = \left(B^{(c)} \boldsymbol{v}_i^{(c)} \right) \sigma_i^{(c)}$$
$$= \boldsymbol{u}_j^{(r)} {\sigma_i^{(c)}}^2, \tag{10}$$

$$\boldsymbol{v}_i^{(c)} = \frac{{B^{(c)}}^\top \boldsymbol{u}_i^{(c)}}{||{B^{(c)}}^\top \boldsymbol{u}_i^{(c)}||_2} \quad (i = 1, \ \ldots, \ r_B^{(c)}), \tag{11}$$

が満たされる. すなわち, 式 (10) と (11) により, $B^{(r)}$ のように, $B^{(c)}$ の特異値分解が $B^{(c)} {B^{(c)}}^\top$ の固有値問題に帰着される.

3. Takata らの方法

$B^{(c)} {B^{(c)}}^\top$ と ${B^{(r)}}^\top B^{(r)}$ のデータ形式は, 同一のものである. ゆえに, 以下の議論では, $B^{(r)} \in \mathbb{R}^{w\times n}$ $(w \geq n)$ についてのみ述べる.

IRL 法を用いる場合, 各反復において, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ を計算するために, 行列とベクトルに対する操作を 2 回行わなければならない. Algorithm 1 は, Takata らの方法の疑似コードである. ここで, $B^{(r)}(i,:)$ は, $B^{(r)}$ の第 i 行を表している. ゆえに, Algorithm 1 において, $B^{(r)}(i,:)$ は, ${B^{(r)}}^\top(:,i)$ と同じデータである. その結果, $8n$ がキャッシュサイズよりも小さいとき, $\boldsymbol{b}_i$ がキャッシュに保持されるため, Algorithm 1 は, 有効に機能する. ゆえに, CPU のキャッシュが十分に大きい計算機において, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ は, 高速に計算される. より詳細には, Takata 法は, CRS 形式で保存された疎行列を対象としている. Algorithm 2 は, 疎行列に対する Takata らの方法の疑似コードである.

Algorithm 2 疎行列に対する Takata らの方法

```
 1: r = 0;
 2: #omp parallel for private(t) reduction(+:r)
 3: for i = 1 to n do
 4:     t = 0;
 5:     for j = row_ptr(i) to row_ptr(i + 1) − 1 do
 6:         t = t + val_R(j) × x(col_ind(j));
 7:     end for
 8:     for j = row_ptr(i) to row_ptr(i + 1) − 1 do
 9:         r(col_ind(j)) = r(col_ind(j)) + val_R(j) × t;
10:     end for
11: end for
12: #omp end parallel for
```

Algorithm 3 密行列のための従来法

```
 1: #omp parallel for;
 2: for i = 1 to n do
 3:     x̃_i = DDOT(b_i, x) (b_i = B(i, :));
 4: end for
 5: #omp end parallel for;
 6: r = 0;
 7: #omp parallel for reduction(+:r)
 8: for i = 1 to n do
 9:     CALL DAXPY for r = r + x̃_i b_i^⊤;
10: end for
11: #omp end parallel for
```

Algorithm 4 密行列のための提案法

```
1: r = 0;
2: #omp parallel for reduction(+:r)
3: for i = 1 to n do
4:     x̃_i = DDOT(b_i, x) (b_i = B(i, :));
5:     CALL DAXPY for r = r + x̃_i b_i^⊤;
6: end for
7: #omp end parallel for
```

4. 提案法

TRL 法を用いる場合, 各反復において, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ を計算するために, 行列とベクトルに対する操作を 2 回行わなければならない. Algorithm 3 は, 密行列を計算の対象として, OpenMP ならびに Intel Math Kernel Library [1] の DDOT と DAXPY を用いて実装された従来法の疑似コードである. 従来法では, $\tilde{\mathbf{x}} = B^{(r)}\mathbf{x}$ と $\mathbf{r} = {B^{(r)}}^\top\tilde{\mathbf{x}}$ の計算が別々に行われるため, 行列とベクトルに対する操作が 2 回必要となる. そのため, もし $B^{(r)}$ がキャッシュに入りきらない場合, 計算時間は長くなる.

本稿では, Takata 法に, 密行列を計算の対象とした改良を適用することによって, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ の計算を高速化する. Takata 法と同様に, 提案法では, ループ融合を行うことによって, 従来法の問題点を改善する. 提案法は, Takata 法と同様に, 高いキャッシュヒット率を実現する. ゆえに, CPU のキャッシュが十分に大きい計算機において, 高効率を達成する. Algorithm 4 は, 提案法の疑似コードである.

5. 実験

5.1 誤差解析

σ_i $(i = 1,\ \ldots,\ r_B)$ を行列 $B \in \mathbb{R}^{w\times n}$ $(w \geq n)$ の特異値とする. この際, B のランクを, $r_B = \operatorname{rank} B$ とする. B の拡張行列 $M \in \mathbb{R}^{(w+n)\times(w+n)}$ は, 次のように表される.

$$M = \begin{bmatrix} O & B \\ B^\top & O \end{bmatrix} \in \mathbb{R}^{(w+n)\times(w+n)}. \tag{12}$$

M の固有値 λ_i $(i = 1,\ \ldots,\ w+n)$ は,

$$\begin{aligned} &\lambda_1 = \sigma_1,\ \ldots,\ \lambda_{r_B} = \sigma_{r_B},\ \lambda_{r_B+1} = -\sigma_1,\ \ldots, \\ &\lambda_{2r_B} = -\sigma_{r_B},\ \lambda_{2r_B+1} = 0,\ \ldots,\ \lambda_{w+n} = 0, \end{aligned} \tag{13}$$

である. $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ $(i = 1,\ \ldots,\ r_B)$ を B の特異値分解を計算機上で行った際に得られた計算値とする. 左特異ベクトル $\tilde{\boldsymbol{u}}_i$ と右特異ベクトル $\tilde{\boldsymbol{v}}_i$ を用いると, $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^{w+n}$ は,

$$\tilde{\boldsymbol{x}}_i := \frac{1}{\sqrt{||\tilde{\boldsymbol{u}}_i||^2 + ||\tilde{\boldsymbol{v}}_i||^2}} \begin{bmatrix} \tilde{\boldsymbol{u}}_i \\ \tilde{\boldsymbol{v}}_i \end{bmatrix}. \tag{14}$$

と表される. ここで, $||\tilde{\boldsymbol{x}}_i|| = 1$ が満たされる. Wilkinson の定理より,

$$\begin{aligned} \min_j |\tilde{\sigma}_i - \lambda_j| &\leq ||M\tilde{\boldsymbol{x}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{x}}_i|| = \sqrt{||M\tilde{\boldsymbol{x}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{x}}_i||^2}, \\ &= \frac{\sqrt{||A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i||^2 + ||A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i||^2}}{\sqrt{2}}, \end{aligned} \tag{15}$$

が成り立つ. 式 (13) より, λ_j が B の特異値であることは保証されていない. 主成分回帰では, 大きい特異値のみ必要とされるため, $|\tilde{\sigma}_i - \lambda_j|$ を最小とする λ_j を B の特異値として扱うことができる. ゆえに, 式 (15) より, 誤差解析の指標として, $\sqrt{||A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i||^2 + ||A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i||^2}/\sqrt{2}$ を用いる.

5.2 実験環境

TRLAN を用いた実験では, Krylov 部分空間の基底の数 m を $m = 2l$ とする. ここで, l は, 所望の特異値の数である. テスト行列として, 指数関数的に特異値が減少する $100,000 \times 10,000$ 行列 B_1 と B_2 を用いる. それぞれの行列の特異値と条件数は, 以下のとおりである.

- B_1:

$$\sigma_i := \sqrt{\epsilon^{\frac{i-1}{10000-1}}}, \tag{16}$$

$$\text{条件数}: \frac{\sigma_1}{\sigma_{10000}} = 6.7108864 \times 10^7. \tag{17}$$

- B_2:

$$\sigma_i := \epsilon^{\frac{i-1}{10000-1}}, \tag{18}$$

$$\text{条件数}: \frac{\sigma_1}{\sigma_{10000}} = 4.5035996 \times 10^{15}. \tag{19}$$

表 1 B_1 に対する計算時間

l	従来法	提案法
10	55.686	35.819
50	74.437	50.550
100	104.914	72.036
500	281.183	215.116

表 2 B_2 に対する計算時間

l	従来法	提案法
10	39.246	25.690
50	56.579	39.150
100	78.7035	56.841
500	268.065	199.513

表 3 B_1 に対する精度

l	従来法	提案法
10	5.04E-15	4.37E-15
50	4.07E-15	4.43E-15
100	6.62E-15	6.62E-15
500	4.38E-15	4.38E-15

表 4 B_2 に対する精度

l	従来法	提案法
10	3.41E-15	4.41E-15
50	4.13E-15	2.70E-15
100	3.41E-15	3.41E-15
500	5.86E-15	5.72E-15

B_1 の条件数は Cholesky QR 分解の観点から小さく [7], B_2 の条件数は大きい. ϵ は, マシンイプシロンで, 倍精度計算機では, おおよそ $\epsilon = 2.220 \times 10^{-16}$ である.

計算機の構成は,

- CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz × 2
- Mem: 128GB
- OS: Ubuntu 16.04.3 LTS
- Compiler: icc version 18.0.1, ifort version 18.0.1
- Option: -qopenmp -ipo -xHOST -O3 -prec-sqrt -prec-div -fp-model precise
- Library: Intel MKL 2018 [1]

である.

5.3 結果と考察

従来方法を用いた TRLAN と提案法を用いた TRLAN の比較実験を行う. **表 1** と **表 2** は, B_1 と B_2 に対する計算時間である. また, **表 3** と **表 4** は, 誤差解析の指標の値を表す. 表 1 と 表 2 より, 提案法を用いた TRLAN の計算時間は, 従来法と比べ, 75% になる. 従来法の計算量は提案法の計算量とほぼ等しいにも関わらず, 計算時間は異なるため, 提案法を用いることによって, キャッシュヒット率が向上していることがわかる. 表 3 と 表 4 より, 計算精度の違いはあまりない. 加えて, 主成分回帰を行うために, 十分な精度を達成している.

6. まとめ

回帰分析では, 大規模な密行列を対象とする場合が多い. いくつかの独立変数間に多重線形性が存在する場合, 回帰分析の代わりに, 大きいほうからいくつかの特異値とその特異ベクトルを用いて主成分回帰をすべきである. そこで, TRLAN に実装されている TRL 法を, 部分特異値分解のために適用する. TRLAN は, 部分固有値分解のための効率の高いソフトウェアである. 特異値分解は, 固有値問題に変換可能であるため, 部分特異値分解を TRL 法を用いて行うことが可能である. 固有値問題に変換された行列を TRLAN で実行する場合, 各反復において, 行列とベクトルに対する操作を 2 回行わなければならない. 本稿で対象とする行列は密行列であるため, すべての要素をキャッシュに格納することが困難な場合が多い. そこで, 本稿では, 疎行列を対象とする Takata 法を, 密行列向けに改良する. その結果, キャッシュヒット率を向上させることができる. 数値実験の結果, 提案法を用いた TRLAN の計算時間は, 従来法を用いた TRLAN と比べ 75% になり, 高速化を達成している. また, 得られた計算精度は, 主成分回帰に対して十分な精度を達成している.

今後の課題として, 最小 2 乗問題の高精度計算を実現するために, 提案法をその計算の終了判定に適用することをあげる.

謝辞 本研究は JSPS 科研費 JP17H02858 と JP17H00167 の助成を受けている.

参考文献

[1] Intel Math Kernel Library, 入手先 ⟨https://software.intel.com/en-us/mkl⟩ (1995).
[2] Lehoucq, R. B., Sorensen, D. C., and Yang, C.: ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.
入手先 ⟨http://www.caam.rice.edu/software/ARPACK⟩ (1998).
[3] Sorensen, D. C., Calvetti, D., and Reichel, L.: *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Elect. Trans. Numer. Anal., vol.2, pp.1-21(1994).
[4] Takata, M., Araki, S., Kimura, K., Fujii, Y., and Nakamura, Y.: *Implementation of computing partial singular value decomposition for principal component analysis using ARPACK*, IPSJ Transactions on Mathematical Modeling and Its Applications, vol.11, no.1, pp.37-44(2018).
[5] Wu, K., and Simon, H.: *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. and Appl., 22(2), pp.602-616(2000).
[6] Wu, J.: Research on Eigenvalue Computations, 入手先 ⟨https://sdm.lbl.gov/ kewu/ps/trlan-ug.pdf⟩, (1999).
[7] Yamamoto, Y., Nakatsukasa, Y., Yanagisawa, Y., and Fukaya, T.: *Roundoff error analysis of the CholeskyQR2 algorithm in an oblique inner product*, JSIAM Letters, vol. 8, pp. 5-8(2015).