Introduction to Information Retrieval (Japanese translation) ver.0.0.2

Christopher D. Manning
Prabhakar Raghavan
Hinrich Schütze
Translated by Yoshihiko Suhara

Last update: 2012-03-26

第18章　Matrix decompositions and latent semantic indexing

123 ページにおいて，単語文書行列[1](term document matrix) の概念について紹介した．$M \times N$ 行列 C の各行が単語を表し，各列が文書を表す．それほど大きくない文書群についても，単語文書行列 C は何万という行や列になりがちである．18.1.1 では，まず，行列分解 (matrix decomposition) として知られる線形代数の演算を解説する．18.2 では，行列分解の特殊形を用いて単語文書行列の低階数近似 (low-rank approximation) を構築する．18.3 では，潜在意味インデクシング (latent semantic indexing) と呼ばれる，そのような低階数近似のインデクシングや検索を行う応用方法について述べる．

潜在意味インデクシングは，元々情報検索におけるスコアづけやランキングのために構築されたものではないが，テキスト文書群の分野をクラスタリングするための興味深いアプローチである．その可能性については，まだ活発な研究領域であることを理解して頂きたい．

線形代数の補習が必要ない読者は 18.1 を飛ばしてもよいが，本章で後ほど利用される固有値の特性を強調しているため，例 18.1 は特に推奨する．

18.1　線形代数の基礎 (Linear algebra view)

必要な線形代数の背景知識について簡潔に解説を行う．C を実数を成分とする M × N 行列とする．単語文書行列では，実際全ての成分が非負である．行列の階数 (rank) は，線形独立の行（または列）の数である．したがって，$rank(C) \leq \min\{M, N\}$ である．非対角成分が 0 である $r \times r$ の平方行列は対角行列 (diagonal matrix) と呼ばれる．その階数は 0 でない対角成分の数に等しい．もし，そのような対角行列の r 個の対角成分全てが 1 の場合，r 次元の単位行列 (indentity matrix) と呼ばれ，I_r で表される．

$M \times M$ の平方行列 C と，0 ではないベクトル $\vec{x}$ について，

$$C\vec{x} = \lambda\vec{x} \tag{18.1}$$

[1]索引語・文書行列とも呼ばれる．

を満たす λ の値を行列 C の固有値 (eigenvalue) と呼ぶ．式 (18.1) を満たす固有値 λ に対する N 次元ベクトル $\vec{x}$ が，対応する右固有ベクトル (right eigenvector) である．最大の固有値に対応する固有ベクトルを主固有ベクトル (principal eigenvalue) と呼ぶ．同様に，C の左固有ベクトル (left eigenvector) は，

$$\vec{y}^T C = \lambda \vec{y}^T \tag{18.2}$$

のような M 次元ベクトル y である．C の 0 でない固有値の数は，最大で $rank(C)$ となる．

行列の固有値は，式 (18.1) を変形した固有方程式[2] (characteristic equation) $(C - \lambda I_M)\vec{x} = 0$ を解くことで導くことができる．C の固有値は，平方行列 S の行列式 (determinant) を $|S|$ とした際，$|C - \lambda I_M| = 0$ の解となる．方程式 $|(C - \lambda I_M)| = 0$ は，λ についての M 次多項式となり，最大で M 個の，C の固有値となる解を持つ．(can have at most M roots.) たとえ C の全要素が実数であっても，これらの固有値は一般的に複素数になる．

ここでは 18.2 の特異値分解 (singular value decompositions) の中心となる概念を組み立てるため，固有値と固有ベクトルの特性についてもう少し詳しく述べる．まず，行列ベクトル乗算と固有ベクトルの関係に注目する．

例 18.1: 以下の行列を考える．

$$S = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

明らかに，この行列は階数 3 であり，3 個の 0 でない固有値 $\lambda_1 = 30$，$\lambda_2 = 20$，$\lambda_3 = 1$ と，対応する 3 つの固有ベクトル

$$\vec{x_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{x_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \vec{x_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

を持つ．それぞれの固有ベクトルについて，S による乗算はあたかも固有ベクトルに単位ベクトルの倍数をかけたように振る舞う．乗算は各固有ベクトルごとに異なる．ここで，$\vec{v} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$ のような任意のベクトルを考える．$\vec{v}$ は，常に S の 3 つの固有ベクトルの線形結合で表現することができる．この例では，

[2]特性方程式とも呼ばれる

$$\vec{v} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = 2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}$$

$\vec{v}$ に S を掛けると，

$$\begin{aligned} S\vec{v} &= S(2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}) \\ &= 2S\vec{x_1} + 4S\vec{x_2} + 6S\vec{x_3} \\ &= 2\lambda_1\vec{x_1} + 4\lambda_2\vec{x_2} + 6\lambda_3\vec{x_3} \\ &= 60\vec{x_1} + 80\vec{x_2} + 6\vec{x_3} \end{aligned} \tag{18.3}$$

となる．

例 18.1 は，$\vec{v}$ が任意のベクトルであっても，S による乗算の効果は S の固有ベクトルと固有値によって決定されるということを示している．更に，$S\vec{v}$ の積が S の小さい固有値によって相対的に影響されにくいことが式 (18.3) より直感的に理解できる．例では，$\lambda_3 = 1$ のため，式 (18.3) の右式第 3 項の寄与が小さい．実際に，仮に $\lambda_3 = 1$ に対応する第 3 固有ベクトルによる寄与を完全に無視すると，$S\vec{v}$ の積は，正しい結果である $\begin{pmatrix} 60 \\ 80 \\ 6 \end{pmatrix}$ ではなく，$\begin{pmatrix} 60 \\ 80 \\ 0 \end{pmatrix}$ と計算される．これらふたつのベクトルは，計算可能なさまざまな（ベクトル差の長さのような）尺度を用いても比較的近い．

これは，行列ベクトル積において，小さな固有値（と固有ベクトル）の影響が小さいことを示している．

この直感は 18.2 において行列分解や，低階数近似を学ぶまで繰り越すことにする．その前に，特に関心が得られる特殊な行列の固有値と固有ベクトルについて述べる．

対称行列 (symmetrix matrix) S について，異なる固有値に対応する固有ベクトルは直交 (orthogonal) する．更に，S が実数かつ対称[3]である場合，固有値は全て実数となる．

例 18.2 実対称行列を考える．

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \tag{18.4}$$

[3] このような行列は実対称行列と呼ばれる．

固有方程式 $|S-\lambda I|=0$ より，二次方程式 $(2-\lambda)^2-1=0$ が導かれ，その解より，固有値 3 と 1 が得られる．対応する固有ベクトルは $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ と $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ は直交する．

18.1.1 行列分解 (Matrix decompositions)

本節では，平方行列がその固有ベクトルから得られる行列の積に分解する方法について述べる．この方法を行列分解と呼ぶ．本節と同様の行列分解は，18.3 の重要なテキスト解析手法の基本となる．そこでは平方でない単語文書行列の分解を行う．本節の平方行列の分解は，より単純で，分解が行われる様子を読者が理解するのに十分な数学的厳格さを持って扱うことができる．

特殊な形をした 3 つの行列の積について 2 つの定理から始める．最初の定理 18.1 は，実数平方行列の 3 つの要素への基本的な分解を与える．二つ目の定理 18.2 は，平方対称行列に適用され，定理 18.3 に記述されている特異値分解の基本となる．

定理 18.1. (対角化定理: matrix diagonalization theorem) S を M 個の線形独立な固有ベクトルを持つ $M\times M$ の実数平方行列とする．その際，固有分解 (eigen decomposition)

$$S = U\Lambda U^{-1} \tag{18.5}$$

が存在する．ここで U の列は S の固有ベクトル，Λ は対角成分が S の固有値を降順に並べた対角行列

$$\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_M \end{pmatrix}, \lambda_i \geq \lambda_{i+1} \tag{18.6}$$

である．固有値がそれぞれ異なる場合，この分解は一意に定まる．

定理 18.1 がどのように働くのか理解するため，U は S の固有ベクトルを列として持つことにする．

$$U = (\vec{u_1}\vec{u_2}\cdots\vec{u_M}) \tag{18.7}$$

すると以下が得られる．

$$
\begin{aligned}
SU &= S(\vec{u_1}\vec{u_2}\cdots\vec{u_M}) \\
&= (\lambda_1\vec{u_1}\lambda_2\vec{u_2}\cdots\lambda_M\vec{u_M}) \\
&= (\vec{u_1}\vec{u_2}\cdots\vec{u_M})\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & \cdots & & \\ & & & \lambda_M \end{pmatrix}
\end{aligned}
$$

したがって，$SU = U\Lambda$ または $S = U\Lambda U^{-1}$ となる．

(p.373) 次に，深く関連のある平方対称行列をその固有ベクトルから導かれる行列の積によって分解する方法について述べる．これは，テキスト解析の主要な手法である特異値分解への道すじとなる (18.2).

定理 18.2. (対称対角化定理: symmetric diagonalization theorem) S を M 個の線形独立な固有ベクトルを持つ $M \times M$ の平方な実対称行列とする．その際，symmetric diagonal decomposition（対称対角分解？）

$$S = Q\Lambda Q^T \tag{18.8}$$

が存在する．

ここで Q の列は，直交かつ正規化された (単位長，実数である)S の固有ベクトル，Λ は，対角成分が S の固有値となる対角行列である．更に，Q の全ての成分は実数で，$Q^{-1} = Q^T$ となる．

この対称対角分解 (symmetric diagonal decomposition) を用いて単語文書行列に対する低階数近似を行う．

演習 18.1 以下の 3×3 対角行列の階数はいくつか？

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$

演習 18.2 $\lambda = 2$ が以下の行列の固有値であることを示せ．また，対応する固有ベクトルを求めよ．

$$C = \begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix}$$

演習 18.3 (18.4) の 2×2 行列の一意となる固有分解を計算せよ．

18.2 Term-document matrices and singular value decompositions

これまで学んできた分解は，平方行列に適用するものであった．しかし，我々が興味のある行列は，(めったにない偶然の一致を除けば)$M \neq N$ であるような $M \times N$ 単語文書行列 C である．更に言えば，C は対称になることはほとんどない．この目的を達成するため，まず特異値分解として知られる対象対角化分解の拡張について述べる．次に 18.3 において，これを用いて C の近似をどのように構築するかについて説明する．特異値分解の根底を成す全ての数学的取り扱いは，本書の範疇を超えてしまう．後に続く定理 18.3 の記述において，特異値分解を 18.1.1 の対称対角分解と関連づける．C が与えられた際，U を CC^T の直交する固有ベクトルを列とするような $M \times M$ 行列とし，V を C^TC の直交する固有ベクトルを列とするような $N \times N$ 行列とする．C^T は列 C の転置を意味する．

定理 18.3 r を $M \times N$ 行列 C の階数とする．その際，C の特異値分解 (以下 SVD と略) は以下の形式を取る．

$$C = U\Sigma V^T \tag{18.9}$$

ここで

1. CC^T の固有値 $\lambda_1, \ldots, \lambda_r$ は C^TC の固有値と等しい．
2. $1 \leq i \leq r$ について，$\sigma_i = \sqrt{\lambda_i}$ (ただし $\lambda_i \geq \lambda_{i+1}$) とする．その際 Σ は，$1 \leq i \leq r$ について $\Sigma_{ii} = \sigma_i$，それ以外は 0 となるような $M \times N$ 行列である．

σ_i の値は C の特異値 (singular value) と呼ばれる．定理 18.3 と定理 18.2 の関係を分析することは有益である．本書の範疇を超えてしまうため，ここでは定理 18.3 の一般的な証明から得られるものではない方法で行う．

式 (18.9) とその転置を掛けると，以下を得る．

$$CC^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T \tag{18.10}$$

式 (18.10) において，左辺は実対称平方行列，右辺は定理 18.2 で示した対称対角分解を表している．左辺の CC^T は何を表しているのだろうか？これは，行と列が対応する M 個の単語である平方行列である．行列の (i, j) 成分は，i 番目と j 番目の文書における共起に基づいた単語の重なりの尺度となる．正確な数学的意味づけは，単語重みづけに基づいて構築された C の方法に依存している．C が 3 ページの図 1.1 のような単語文書出現行列 (term-document incidence matrix) とすると，CC^T の (i,j) 成分は，単語 i と単語 j が共に出現する文書の数となる．

SVD の数値を書き表す際，Σ を特異値を対角成分とする $r \times r$ 行列で表現するのが慣習である．なぜなら，これらの成分以外の部分行列は 0 であるためである．したがって，省略された Σ の行に対応する U の右端の $M-r$ 列についても省略する．同様に Σ の $N-r$ 列の 0 が掛けられる V^T の行に対応する，V の右端の $N-r$ 列についても同様に省略される．SVD のこの表現形式は reduced SVD や truncated SVD として知られ，演習 18.9 で再び扱う．これ以降，例や演習にはこの形式を用いることにする．

例 18.3: 階数 2 の 4×2 行列の特異値分解を説明する．特異値は，$\Sigma_{11}=2.236$, $\Sigma_{22}=1$ である．

$$C=\begin{pmatrix}1 & -1\\ 0 & 1\\ 1 & 0\\ -1 & 1\end{pmatrix}=\begin{pmatrix}-0.632 & 0.000\\ 0.316 & -0.707\\ -0.316 & -0.707\\ 0.632 & 0.000\end{pmatrix}\begin{pmatrix}2.236 & 0.000\\ 0.000 & 1.000\end{pmatrix}\begin{pmatrix}-0.707 & 0.707\\ -0.707 & -0.707\end{pmatrix} \tag{18.11}$$

18.1.1 で定義された行列分解のように，行列の特異値分解は様々なアルゴリズムで計算することができ，その多くはソフトウェア実装が公開されている．18.5 においてその手がかりについて述べる．

演習 18.4

$$C=\begin{pmatrix}1 & 1\\ 0 & 1\\ 1 & 0\end{pmatrix} \tag{18.12}$$

を文書群の単語文書出現行列とする．共起行列 CC^T を計算せよ．C が単語文書出現行列の際，CC^T の対角成分はどのような意味を持つか？

演習 18.5 式 (18.12) における行列の SVD が

$$U=\begin{pmatrix}-0.816 & 0.000\\ -0.408 & -0.707\\ -0.408 & 0.707\end{pmatrix},\Sigma=\begin{pmatrix}1.732 & 0.000\\ 0.000 & 1.000\end{pmatrix},V^T=\begin{pmatrix}-0.707 & -0.707\\ 0.707 & -0.707\end{pmatrix} \tag{18.13}$$

であることを定理 18.3 の記述の全ての性質の検証を行い，確認せよ．

演習 18.6 C を単語文書出現行列とする．C^TC の成分は何を表しているか？

演習 18.7

$$C = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 3 & 0 \\ 2 & 1 & 0 \end{pmatrix} \tag{18.14}$$

を成分が単語頻度であるような単語文書行列とする．したがって，単語 1 は，文書 2 に 2 回出現しており，文書 3 には 1 回出現している．CC^T を計算し，同一文書に共に出現する中で最も頻度の大きい 2 つの単語の成分が最大になることを観察せよ．

18.3 Low-rank approximation

次に，一見情報検索にあまり関係がなさそうに見える，行列近似の問題について述べる，この問題を SVD を用いて解く方法について説明を行い，情報検索への適用方法を述べる．

$M \times N$ 行列 C，正の整数 k が与えられた際に，以下で定義される行列の差 $X = C - C_k$ のフロベニウスノルム (Frobenius norm) を最小化するような最大で階数 k の $M \times N$ 行列 C_k を求めたい．

$$\|X\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} X_{ij}^2} \tag{18.15}$$

したがって，X のフロベニウスノルムは，C_k と C の違いの尺度であり，最大の階数 k という制約の下で，この違いを最小にする C_k を求めることが目的となる．r を C の階数とすると，$C_r = C$ は明らかで，差のフロベニウスノルムは 0 になる．k が r に比べて著しく小さい時に，C_k を低階数近似と呼ぶことにする．

低階数近似問題を解くために SVD を用いることができる．単語文書行列を近似するための SVD の適用を以下の 3 つのステップで行う．

1. C が与えられた際，(18.9) に示されるような形式の SVD を構築する．よって $C = U\Sigma V^T$ となる．
2. Σ より $r - k$ 個の最小対角成分を 0 に置き換えることで行列 Σ_k を得る．
3. $C_k = U\Sigma_k V^T$ を計算し，結果を C の階数 k 近似とする．

C_k の階数は最大で k となる．これは Σ_k が最大で k 個の 0 でない要素を持つことからわかる．次に例 18.1 の直感的解釈を思い出そう．行列積における小さい固有値の影響は小さい．よって，小さい固有値を 0 に置き換えることで，積が大幅に変わることなく，C に “近い” ままにすることが妥当に思える．Eckart と Young による以下に述べる定理によって，最小のフロベニウス誤差 (Frobenius error) を持つ階数 k の行列を求めることができる．

定理 18.4.

$$min_{Z|rank(Z)=k}\|C-Z\|_F = \|C-C_k\|_F = \sqrt{\sum_{i=k+1}^{r}\sigma_i^2} \tag{18.16}$$

特異値は降順 $\sigma_1 \geq \sigma_2 \geq \cdots$ に並んでいることを思い出すと，定理 18.4 より C_k が C に対する，σ_{k+1} に等しい誤差 ($C - C_k$ のフロベニウスノルムによって計算される) を持つ最良の階数 k 近似であることがわかる．したがって k が大きくなればなるほど誤差は小さくなる (具体的には，$k = r$ の場合，誤差は 0 になる．なぜなら $\Sigma_r = \Sigma$; provided $r < M, N$, then $\sigma_{k+1} = 0$ and thus $C_r = C$).

Σ の最小の $r - k$ 個の特異値を切り捨てることによって小さい誤差の階数 k 近似を得られる理由について洞察を行うため，C_k の形式を観察する．

$$
\begin{aligned}
C_k &= U\Sigma_k V^T && (18.17)\\
&= U\begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0\\ 0 & \cdots & 0 & 0 & 0\\ 0 & 0 & \sigma_k & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdots \end{pmatrix} V^T && (18.18)\\
&= \sum_{i=1}^{k}\sigma_i \vec{u_i}\vec{v_i}^T && (18.19)
\end{aligned}
$$

ここで $\vec{u_i}$ と $\vec{v_i}$ はそれぞれ U と V の i 番目の列である．よって，$\vec{u_i}\vec{v_i}^T$ は階数 1 の行列となり，C_k を k 個の階数 1 行列をそれぞれの特異値によって重み付け和したもので表現を行っている．i が増加するにつれて，階数 1 の行列 $\vec{u_i}\vec{v_i}^T$ の寄与は，減少していく特異値 σ_i によって重み付けされる．

演習 18.8 例 18.12 における行列 C に対する階数 1 の近似 C_1 を演習 18.13 のように SVD を用いて計算せよ．この近似の誤差のフロベニウスノルムはいくつになるか？

演習 18.9 ここでは，演習 18.8 の計算について考える．図 18.2 の図にしたがうと，階数 1 近似によって σ_1 がスカラーとなることに気がつく．U_1 で U の 1 列目，V_1 で V の 1 列目を示す．その際，C の階数 1 近似は $U_1\sigma_1 V_1^T = \sigma_1 U_1 V_1^T$ とも書けることを示せ．

演習 18.10 演習 18.9 は，階数 k 近似に一般化することができる．U'_k と V'_k を，U と V それぞれの最初の k 列だけを保持することで得られる “縮小され

た" 行列とする．よって，U'_k は $M \times k$ 行列，V'^T_k は $k \times N$ 行列となる．すると，

$$C_k = U'_k \Sigma'_k V'^T_k \tag{18.20}$$

が得られる．ここで Σ'_k は特異値 $\sigma_1, \ldots, \sigma_k$ を対角成分に持つ Σ_k の $k \times k$ の平方な部分行列である．(18.20) を用いる一番の利点は，沢山の全てが 0 となる冗長な列を U と V から除去することである．それによって，低階数近似に影響しない列による乗算を明示的に取り除くことができる．この種類の SVD は，reduced SVD や truncated SVD として知られ，低階数近似を計算する上で，より簡単な表現である．
例 18.3 の行列 C について，Σ_2 と Σ'_2 を書き示せ．

18.4 潜在意味インデクシング (Latent semantic indexing)

ここでは，単語文書行列 C を，SVD を用いて小さい階数の行列で近似する方法について論じる．C の低階数近似は，文書群のそれぞれの文書に新しい表現を与える．クエリも同様にこの低階数表現に書き換えることで，クエリと文書の類似度が計算可能になる．この処理は潜在意味インデクシング (latent semantic indexing) として知られる (一般的に LSI と表記される)．

その前にまず，そのような近似の動機付けを行う．6 章で紹介された文書とクエリのベクトル空間表現を思い出そう．このベクトル空間表現には，クエリと文書をベクトルとして一様に扱う，コサイン類似度に基づくスコア計算，単語に対して異なる重み付けを行うことができる，文書検索に留まらず，これらの拡張をクラスタリングや分類に適用する，といった数々の長所があった．しかしながら，ベクトル空間表現は同義性 (synonymy) と polysemy (多義性) という自然言語における典型的な問題を対処することできない．同義性は 2 つの異なる単語 (例えば car と automobile) が同じ意味を持つような状況を指す．ベクトル空間表現では，ベクトル空間において別の次元となるため，car と automobile のような類義語間の関係を捉えることができない．結果として，クエリ $\vec{q}$(例えば car) と car と automobile を含む文書 $\vec{d}$ の類似度 $\vec{q} \cdot \vec{d}$ は，ユーザが感じる真の類似度に比べて過小評価されてしまう．一方，多義性は，charge のような単語が複数の意味を持つ状況を指す．そのため，$\vec{q} \cdot \vec{d}$ がユーザが感じるよりも過大評価されてしまう．単語の潜在的な意味の関連性を捉え，これらの問題を緩和するために単語の共起 (例えば charge が steed を含む文書に出現することに対して，electron を含む文書にする) を使うことができるだろうか？

たとえ適度な大きさの文書群でも，単語文書行列は数万行，数万列という大きさになり，階数も同様に数万になってしまう．LSI (潜在意味解析 (latent semantic analysis; LSA) とも呼ばれる) では，SVD を用いて，C の元の階数よりはるかに小さい k について単語文書行列の低階数近似 C_k を求める．本節で後ほど引用される実験においては，k はおよそ 200 から 300 程度の値が選ばれている．したがって，それぞれの行／列 (それぞれが単語／文書に対応している) を，k 次元空間に写像する．この空間は CC^T と C^TC の k 個の (最大固有値に対応する) 主固有ベクトルによって張られる．行列 C_k 自身は k に関わらず $M \times N$ 行列であることに注意する．

次に，ベクトル間の類似度を求めるために，元の表現でそうしたように k 次元の LSI 表現を用いる．クエリベクトル $\vec{q}$ は，以下の変換によって LSI 空間の表現に写像される．

$$\vec{q_k} = \Sigma_k^{-1} U_k^T \vec{q} \tag{18.21}$$

これより，章 6 で述べたコサイン尺度を用いてクエリと文書，文書同士，または単語同士の類似度を計算することができる．式 (18.21) に $\vec{q}$ がクエリである必要はなく，単に単語集合の空間におけるベクトルであればよい，ということを述べておく．これは文書群の LSI 表現を持っていれば，文書群に含まれていなかった新しい文書を (18.21) 式を用いて LSI 表現に “たたみ込む” ことができることを意味している．これによって LSI 空間に文書を徐々に追加することができる．もちろん，そのような追加方法では，新しく加えられた文書の共起を捉えることができない (それだけではなく，追加された文書に含まれる新しい単語も無視してしまう)．そのため，文書が追加されるにつれて LSI 表現の質は低下していき，最終的には LSI 表現の再計算が必要になる．

C_k による C の近似が忠実であるため，コサイン尺度の相対値が保たれていることを期待してしまう．元空間においてクエリが文書に近い場合，k 次元空間においても相対的に近いだろう．しかし，疎なクエリベクトル $\vec{q}$ が低次元における密なクエリベクトル $\vec{q_k}$ に変換されることなどを考えると，それ自体が興味深いわけではない．元の形式で $\vec{q}$ を処理するコストに比べて，大きな計算コストがかかってしまう．

例 18.4 以下の単語文書行列 C を考える．

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	0	0	0	1	1	0
trip	0	0	0	1	0	1

この SVD は，以下の 3 つの行列の積となる．最初にこの例における U を示す．

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.14	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
voyage	−0.70	0.35	0.15	−0.58	0.16
trip	−0.26	0.65	−0.41	0.58	−0.09

SVD を単語文書行列に適用するとき，U は SVD 単語行列 (SVD term matrix) として知られる．特異値は，$\Sigma =$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

となる．最後に V^T を求める．これは単語文書行列の文脈では SVD 文書行列 (SVD document matrix) として知られる．

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Σ の最も大きい 2 つの特異値以外を “消去” することによって $\Sigma_2 =$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

を得る．これによって，$C_2 =$

	d_1	d_2	d_3	d_4	d_5	d_6
1	−1.62	−0.60	−0.44	−0.97	−0.70	−0.26
2	−0.46	−0.84	−0.30	1.00	0.35	0.65
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

が計算される．元の行列 C と違い，低階数近似は負の値を持つことがあることに注意する．

例 18.4 における C_2 と Σ_2 より，これらの行列それぞれの最後の 3 行が完全に 0 で置き換えられていることがわかる．これによって式 (18.18)SVD 積 $U\Sigma V^T$ は，Σ_2 と V^T の 2 行のみからによって計算できることがわかる．これらの行列を省略形の Σ_2' と $(V')^T$ で置き換える．例えば，この例における truncated SVD の文書行列 $(V')^T$ は，

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41

図 18.3 は $(V')^T$ の 2 次元における文書を表している．C_2 は C に深く関係がある．

一般的には，C の C_k による低階数近似を，C_k の階数最大値が k という制約のもとで，誤差 $C - C_k$ のフロベニウスノルムが小さい，C を形成する単語と文書の表現を探索する条件付き最適化問題 (constrained optimization problem) と見なすことができる．単語／文書を無理やり k 次元空間に縮小するとき，SVD は似たような共起をする単語をひとつにまとめる．この直感より，次元削減によって検索の質がそれほど影響を受けることがないが，むしろ**改善される**場合もあることがわかる．

Dumais (1993, 1995) は一般的に使われる Lanczos 法[4] を用いて SVD を計算することで，TREC 文書と課題に対して LSI の実験を行った．当時は 1990 年代初めで，数万文書の LSI 計算には 1 台のマシンでだいたい 1 日ほどかかった．これらの実験において，彼らは TREC 参加者の真ん中かそれ以上の精度を達成した．20%の TREC トピックについて，彼らのシステムはトップスコアを示し，大体 350 次元の LSI において，標準的なベクトル空間法よりも平均的に少しだけ良い結果が得られたと報告されている．彼らの研究，その後に続いた多くの実験などによって検証されたいくつかの結論を以下に述べる．

- SVD の計算量は膨大である．執筆時[5]において，100 万文書を超える実験の成功例は知られていない．これが LSI が広く普及するための最大の障壁となっている．この問題に対するアプローチのひとつとして，ランダムに選択された文書群の部分集合に対して LSI 表現を構築し，式 (18.21) に詳しく書かれているように，残った文書を “たたみ込む” というものがある．

- k を減少させるにつれて，期待するとおり再現率は向上する傾向にある．

[4]固有値問題の解法の一種．一般的にランチョスと読む．
[5]2008 年時

- 驚いたことに，k の値が 200 から 300 程度の場合，いくつかのクエリベンチマークにおいて精度が向上することが確認されている．これによって適切な k の値において，LSI は同義性の課題にいくらか対処していることが示唆される．

- LSI は，クエリと文書に重なりがほとんどないようなアプリケーションにおいて，最も良い働きをする．

実験では，LSI が伝統的なインデックスやスコア計算による精度を達成することができなかった場面についても記録されている．特に（そして明白かもしれないが），LSI はベクトル空間検索の基本的な 2 つの欠点,（german を含み，shepherd を含まない文書を見つけるといった）存在の否定を表現する良い方法がない，そしてブーリアン条件を用いることができない，を共有している．

LSI は，縮約された各次元をクラスタ，文書の各次元に対する値をそのクラスタに属する確率と解釈することで，ソフトクラスタリング (soft clustering) と見なすことができる．

演習 18.11 英語かスペイン語のいずれかで書かれた文書集合があるとする．文書群は図 18.4 の通りである．

図 18.5 は読者の理解のための英語とスペイン語の用語集である．この用語集は検索システムでは**利用できない**ものとする．

1. これらの文書から成る文書群を用いるため，適切な単語文書行列 C を作成せよ，簡単のため，正規化された TF-IDF 値ではなく，単語の頻度そのものを用いる．行列の各次元の役割を明確にすること．

2. 行列 U_2，Σ'_2，V_2 を書き出し，これらから階数 2 の近似 C_2 を導出せよ．

3. C^TC の (i,j) 成分が何を表現しているかを簡潔に述べよ．

4. $C_2^TC_2$ の (i,j) 成分が何を表現しているか，そして何故それが C^TC のそれと異なるかを簡潔に述べよ．

18.5 References and further reading

Strang (1986) は，特異値分解を含む行列分解の素晴らしい導入概説を書いている．定理 18.4 は，Eckart と Young (1936) によるものである．情報検索と単語文書行列の低階数近似のつながりは，Deerwester ら (1990)，それに続く Berry ら (1995) の結果のサーベイによって紹介された．Dumais (1993, 1995) は TREC ベンチマークにおける実験結果を説明し，少なくともいくつかのベンチマークにおいて，LSI が標準的なベクトル空間検索よりもより良い適合

率と再現率を得られることを述べた．Berry [6]や Telcordia Technologies [7] は，LSI の文献やソフトウェアについて広範囲の情報を提供している．Schütze と Silverstein (1997) は，LSI と，K-means クラスタリング (16.4 節) における重心の縮約表現 (truncated representation) の評価を行った．Bast と Majumdar (2005) は，LSI における縮約次元 k の役割と，k の値を変えることで，異なるペアの単語が結合していく様子の詳細を述べている．言語横断情報検索 (cross-language information retrieval) (2 つ以上の異なる言語で書かれた文書のインデックスを作成し，それらとは別の言語のクエリによってそれらの文書を対象に検索を行う) に対する LSI の適用は，Berry と Young (1995) や Littman ら (1998) によって発展された．LSI (より一般的な背景では LSA と呼ばれる) は，メモリ設計からコンピュータビジョンまで計算機科学のその他たくさんの問題に適用されてきた．

Hofmann (1999a, 1999b) は，基本的な LSI 手法に初めて確率的な拡張を行った．次元削減に対する確率的潜在意味変数モデルのさらに十分な形式的基準としては，Latent Dirichlet Allocation (LDA) モデルがある (Blei et. al. 2003)．これは生成モデルで，訓練セット以外の文書に対して確率を与える．このモデルの階層的クラスタリングに対する拡張が Rosen-Zvi ら (2004) によって行われた．Wei と Croft (2006) は，LDA について初めての大規模な評価を示し，12.2 節 (223 ページ) のクエリ尤度モデル (query likelihood model) より優位に優れているが，12.4 節 (230 ページ) で言及した関連性モデル (relevance model) ほど精度が高くないことがわかった．ただし後者については，LDA では行わない各クエリについて追加の処理が行われる．Teh ら (2006) は，階層ディリクレ過程 (Hierarchical Dirichlet process) を提案した．これは，トピックが文書にまたがって共有されるグループ（本書では文書）が潜在トピックの無限混合から取り出されるという確率モデルである．

[6] http://www.cs.utk.edu/ berry/lsi++/
[7] http://lsi.argreenhouse.com/lsi/LSIpapers.html

謝辞

本稿の誤りを指摘してくださった Redwood Shore さんに感謝申し上げます.