計算科学における情報圧縮

Information Compression in Computational Science

2017.10.19

#4:情報圧縮の数理2 （特異値分解と低ランク近似）

Singular value decomposition and low rank approximation

理学系研究科　物理学専攻　大久保　毅

Department of Physics, **Tsuyoshi Okubo**

Outline

- Singular value decomposition (SVD)
- Generalized inverse matrix
- Low rank approximation
  - Low rank approximation by SVD
  - Low "rank" approximation for tensor
- Application of low rank approximation to images

Singular value decomposition

Diagonalization

Diagonalizaiton（対角化）：

$$A : N \times N$$

(Square matrix)

$$P^{-1}AP = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix}$$

$$A\vec{v} = \lambda\vec{v} \qquad P = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_N)$$

$$(\vec{u}^*)^t A = \lambda(\vec{u}^*)^t \qquad (P^{-1})^t = (\vec{u}_1^*, \vec{u}_2^*, \cdots, \vec{u}_N^*)$$

- Eigenvalue problems and diagonalizations are defined for a square matrix.
- Even if A is a square matrix, it may not be diagonalized.
  - Normal or Hermitian matrices are always diagonalized by a unitary matrix

Spectral decomposition

(For a normal matrix A,)

Spectral decomposition
(スペクトル分解)

$$A = U \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix} U^\dagger$$

$$= \sum_{i=1}^{N} \lambda_i \underline{\vec{u}_i \vec{u}_i^\dagger}$$

$$\left(= \sum_{i=1}^{N} \lambda_i |u_i\rangle\langle u_i| \right)$$

Note:

$$\vec{u}_i \vec{u}_i^\dagger = \begin{pmatrix} u_1 u_1^* & u_1 u_2^* & \cdots & u_1 u_N^* \\ u_2 u_1^* & u_2 u_2^* & \cdots & u_2 u_N^* \\ \vdots & \vdots & \cdots & \vdots \\ u_N u_1^* & u_N u_2^* & \cdots & u_N u_N^* \end{pmatrix}$$

Matrix decomposition into a sum of projectors onto its eigen subspaces.

Projector:

$$P^2 = P$$

Singular value decomposition (SVD)

Singular value decomposition (特異値分解)

$$A : M \times N$$

$$A_{ij} \in \boldsymbol{C}$$

$$A = \underline{U}\Sigma\underline{V}^\dagger$$

$U : M \times M$ **Unitary**

$V : N \times N$ **Unitary**

$$\Sigma = \begin{pmatrix} \underline{\Sigma_{r\times r}} & 0_{r\times(N-r)} \\ 0_{(M-r)\times r} & 0_{(M-r)\times N-r} \end{pmatrix}$$

$$\Sigma_{r\times r} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{pmatrix}$$

Diagonal matrix with non-negative real elements

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

Singular values

Properties of SVD 1

1. Any matrices can be decomposed as SVD: $A = U\Sigma V^\dagger$

$$A : M \times N \Rightarrow A^\dagger A : N \times N$$

* $A^\dagger A$ is a Hermitian matrix.

$$(A^\dagger A)^\dagger = A^\dagger A$$

It can be diagonalized by
a unitary matrix V .

$$V^\dagger (A^\dagger A) V = \mathrm{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_N\}$$

$$V = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_N)$$

$\vec{v}_i$: eigenvector

* $A^\dagger A$ is a positive semi-definite matrix.
（半正定値、準正定値）

$$\vec{x}^* \cdot (A^\dagger A \vec{x}) = \|A\vec{x}\|^2 \geq 0 \Leftrightarrow$$

Its eigenvalues are
non-negative

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$$

Properties of SVD 1

1. Any matrices can be decomposed as SVD: $A = U\Sigma V^\dagger$

$$V^\dagger (A^\dagger A) V = \mathrm{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_N\}$$
$$V = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_N)$$

$\Leftrightarrow$

$$(A\vec{v}_i)^* \cdot (A\vec{v}_j) = \lambda_i \delta_{ij}$$
$$(\|A\vec{v}_i\|^2 = \lambda_i)$$

Suppose $\lambda_1 \geq \lambda_2 \geq \cdots \underline{\geq \lambda_r > 0} = \lambda_{r+1} = \cdots = \lambda_N$

(There are r positive eigenvalues.)

Make new orthonormal basis $U = (\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_M)$ in $\boldsymbol{C}^M$

For $(i = 1, 2, \ldots, r)$ $\quad \sigma_i = \sqrt{\lambda_i}, \vec{u}_i = \dfrac{1}{\sigma_i} A\vec{v}$

For $(i = r+1, \ldots, M)$ Any orthonormal basis orthogonal to $\vec{u}_i \quad (i = 1, 2, \ldots, r)$

$$\vec{u}_i^* \cdot (A\vec{v}_j) = \sigma_i \delta_{ij} \quad (i = 1, \ldots, M; j = 1, \ldots, N)$$

(For simplicity, we set $\sigma_i = 0$ for $i > r$.)

Properties of SVD 1

1. Any matrices can be decomposed as SVD: $A = U\Sigma V^\dagger$

We can perform same "proof" by using $AA^\dagger$.

$U = (\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_M)$ is the unitary matrix which diagonalize $AA^\dagger$ as

$$U^\dagger (AA^\dagger) U = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_r, \underbrace{0, \ldots, 0}_{M-r}\}$$

In summary,

- A matrix A can be decomposed as SVD: $A = U\Sigma V^\dagger$
- Singular values are related to the eigenvalues of $A^\dagger A$ and $AA^\dagger$ as

$$\sigma_i = \sqrt{\lambda_i} \, .$$

- V and U are eigenvectors of $A^\dagger A$ and $AA^\dagger$,respectively.

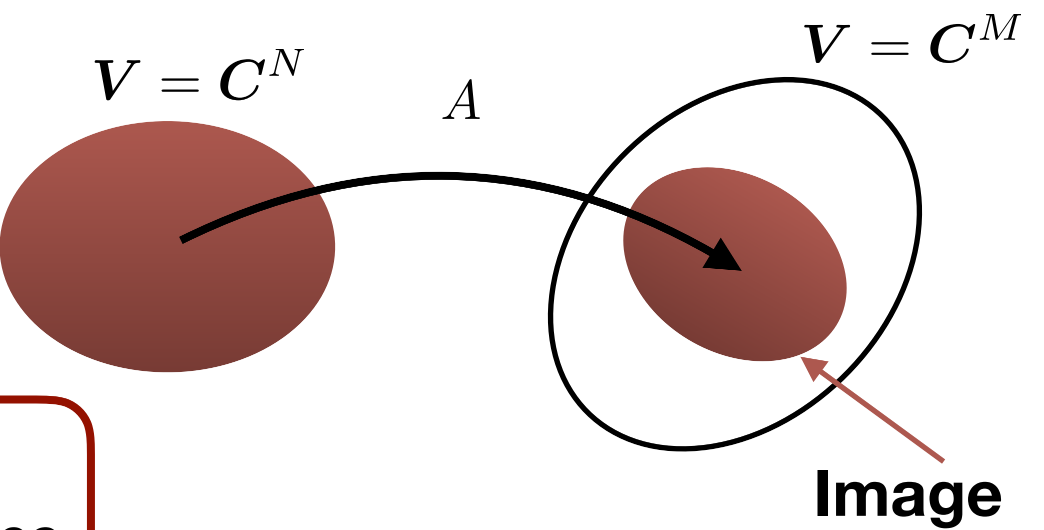
Properties of SVD 2

$$A = U\Sigma V^\dagger$$

2. # of positive singular values is identical with the rank.

$$A : N \times M \Rightarrow A : \boldsymbol{C}^N \to \boldsymbol{C}^M$$

$$\mathrm{rank}(A) \equiv \dim(\mathrm{img}(A))$$



Remember

The orthonormal basis $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_N\}$ satisfies

$$(A\vec{v}_i)^* \cdot (A\vec{v}_j) = \lambda_i \delta_{ij}$$

Here, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0 = \lambda_{r+1} = \cdots = \lambda_N$

and $\sigma_i = \sqrt{\lambda_i}$

$$\Rightarrow \mathrm{img}(A) = \mathrm{Span}\{A\vec{v}_1, A\vec{v}_2, \ldots, A\vec{v}_r\}$$

$$\Rightarrow \dim(\mathrm{img}(A)) = r$$ **=# of positive singular values**

Properties of SVD 3

$$A = U\Sigma V^\dagger$$

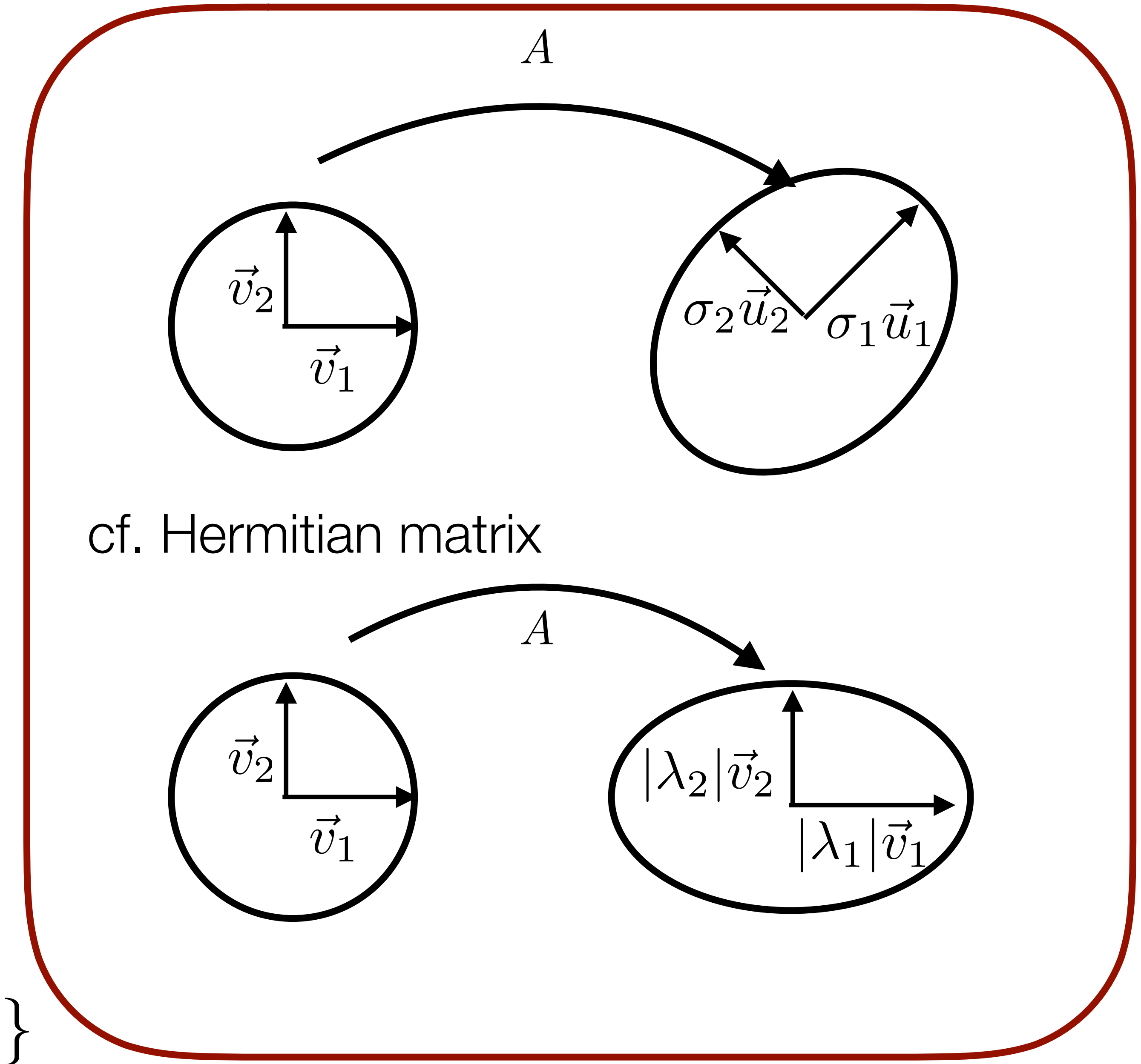
3. Singular vectors

$A : M \times N$ $\quad U = (\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_M)\,,\ V = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_N)$

For $i = 1, 2, \ldots, r$

$$A\vec{v}_i = \sigma_i \vec{u}_i\ ,\ A^\dagger \vec{u}_i = \sigma_i \vec{v}_i$$

$\vec{v}_i$: right singular vector
$\vec{u}_i$: left singular vector



Relation to image and kernel:

$$\mathrm{img}(A) = \mathrm{Span}\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\}$$
$$\ker(A) = \mathrm{Span}\{\vec{v}_{r+1}, \vec{v}_{r+2}, \ldots, \vec{v}_N\}$$

$$\mathrm{img}(A^\dagger) = \mathrm{Span}\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r\}$$
$$\ker(A^\dagger) = \mathrm{Span}\{\vec{u}_{r+1}, \vec{u}_{r+2}, \ldots, \vec{u}_M\}$$

Properties of SVD 4

$$A = U\Sigma V^\dagger$$

4. Min-max theorem (Courant-Fischer theorem)

$A : N \times N$, Hermitian matrix

Suppose its eigenvalues are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$.

$$\lambda_k = \min_{\boldsymbol{S};\dim(\boldsymbol{S})\leq k-1} \max_{\vec{x}\in \boldsymbol{S}^\perp;\|\vec{x}\|=1} \vec{x}^* \cdot A\vec{x}$$

$$\boldsymbol{S}^\perp = \{\vec{x} : \vec{x}^* \cdot \vec{y} = 0, \vec{y} \in \boldsymbol{S}\}$$

Orthonormal complement（直交補空間）

We can prove this by considering vector subspace spanned by eigenvectors. (see references)

Intuitive examples:

Maximum appears for the eigenvector.

$$A\vec{u}_i = \lambda_i \vec{u}_i$$

$$\lambda_1 = \max_{\vec{x}\in \boldsymbol{C}^N;\|\vec{x}\|=1} \vec{x}^* \cdot (A\vec{x}) \quad \Rightarrow \quad \vec{x} = \vec{u}_1$$

$$\lambda_2 = \max_{\vec{x}\in \boldsymbol{C}^N;\vec{x}\perp\vec{u}_1,\|\vec{x}\|=1} \vec{x}^* \cdot (A\vec{x}) = \min_{\boldsymbol{S};\dim(\boldsymbol{S})\leq 1} \max_{\vec{x}\in \boldsymbol{S}^\perp;\|\vec{x}\|=1} \vec{x}^* \cdot (A\vec{x}) \quad \Rightarrow \quad \vec{x} = \vec{u}_2$$

Properties of SVD 4

$$A = U\Sigma V^\dagger$$

4. Min-max theorem (Courant-Fischer theorem)

$A : M \times N$

Suppose its singular values are $\sigma_1 \geq \sigma_2 \geq \cdots$

$$\sigma_k = \min_{\boldsymbol{S};\dim(\boldsymbol{S})\leq k-1} \max_{\vec{x}\in\boldsymbol{S}^\perp;\|\vec{x}\|=1} \|A\vec{x}\|$$

By setting k=1,

$$\sigma_1 = \max_{\vec{x}\in\boldsymbol{C}^N,\|\vec{x}\|=1} \|A\vec{x}\|$$

which means

$$\|A\vec{x}\| \leq \sigma_1 \|\vec{x}\|$$

for $\vec{x} \in \boldsymbol{C}^N$

We can easily prove this by using

$A^\dagger A$: Hermitian

$$A^\dagger A \vec{v}_i = \lambda_i$$

$$\sigma_i = \sqrt{\lambda_i}$$

Properties of SVD 5

$$A = U\Sigma V^\dagger$$

5. Singular values for multiplication and addition

$\sigma_i(A)$: singular value of matrix A
(for $i > \mathrm{rank}(A)$, we set $\sigma_i = 0$)

*Following properties can be proven by using min-max theorem.

Multiplication: $A : M \times L, B : L \times N$

$$\sigma_k(AB) \leq \sigma_1(A)\sigma_k(B) \qquad (k = 1, 2, \dots)$$

$$(\sigma_k(AB) \leq \sigma_k(A)\sigma_1(B))$$

$\Rightarrow$ $\mathrm{rank}(AB) \leq \min(\mathrm{rank}(A), \mathrm{rank}(B))$

Addition: $A, B : M \times N$

$$\sigma_{k+j-1}(A+B) \leq \sigma_k(A) + \sigma_j(B) \qquad (k, j = 1, 2, \dots)$$

$$(\sigma_{k+j-1}(A+B) \leq \sigma_j(A) + \sigma_k(B))$$

$\Rightarrow$ If $\mathrm{rank}(B) \leq r$,

$$\sigma_{k+r}(A+B) \leq \sigma_k(A)$$

Libraries for SVD

There are **LAPACK** routines for SVD.

DGESDD, ZGESDD

DGESVD, ZGESVD

(For dense matrices)

*Linear Algebra PACKage

At *netlib.org* (reference implementations)

+

A lot of vender implementations

- Intel MKL
- Apple Accelerate Framework
- Fujitsu SSLII
- ...

numpy and **scipy** modules in python have routines for SVD.

numpy.linalg.svd

scipy.linalg.svd

(For dense matrices)

scipy.sparse.linalg.svds

(For sparse matrices or calculation of partial singular values)

Computational cost

For a $M \times N$ matrix ($M \leqq N$):

Full SVD: $O(NM^2)$

Partial SVD: $O(NMk)$

k : # of singular values to be calculated

Generalized inverse matrix

Regular matrix and its inverse matrix

A square matrix A is a **regular matrix** （正則）if a matrix X satisfying

$$AX = XA = I$$

exists. The matrix X is called inverse matrix（逆行列） of A and it is written as $X = A^{-1}$.

Properties: A^{-1} is unique.

$$(A^{-1})^{-1} = A$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

A is a regular matrix $\Longleftrightarrow$ $\mathrm{rank}(A) = N$

Can we consider an "inverse matrix" of a non-regular matrix (including a rectangular matrix) ?

Generalized inverse matrix

Generalized inverse matrix（一般化逆行列）：

For $A : M \times N$, a matrix $A^{-} : N \times M$ satisfying

$$AA^{-}A = A$$

is called generalized inverse matrix.

Properties:

- Generalized matrix is not unique.
  - At least one generalized matrix exists.

- If A is a regular matrix, $A^{-} = A^{-1}$

➡ A^{-} is a generalization of inverse matrix.

Moore-Penrose pseudo inverse

Moore-Penrose pseudo inverse matrix（擬似逆行列）:

For $A : M \times N$, a matrix $A^+ : N \times M$ satisfying

(1) $AA^+A = A$ (2) $A^+AA^+ = A^+$

(3) $(AA^+)^\dagger = AA^+$ (4) $(A^+A)^\dagger = A^+A$

is called (Moore-Penrose) pseudo inverse matrix.

Relation to SVD

- Pseudo inverse is unique and calculated from SVD.

$$A = U\Sigma V^\dagger = U \begin{pmatrix} \Sigma_{r\times r} & 0_{r\times(N-r)} \\ 0_{(M-r)\times r} & 0_{(M-r)\times N-r} \end{pmatrix} V^\dagger$$

$$A^+ = V \begin{pmatrix} \Sigma^{-1}_{r\times r} & 0_{r\times(M-r)} \\ 0_{(N-r)\times r} & 0_{(N-r)\times M-r} \end{pmatrix} U^\dagger$$

$$A^+A = V \begin{pmatrix} \Sigma^{-1}_{r\times r} & 0_{r\times(M-r)} \\ 0_{(N-r)\times r} & 0_{(N-r)\times M-r} \end{pmatrix} U^\dagger U \begin{pmatrix} \Sigma_{r\times r} & 0_{r\times(N-r)} \\ 0_{(M-r)\times r} & 0_{(M-r)\times N-r} \end{pmatrix} V^\dagger$$

$$= \sum_{i=1}^{r} \vec{v}_i \vec{v}_i^\dagger \left(= \sum_{i=1}^{r} |v_i\rangle\langle v_i|\right)$$

A^+A is a projector onto img($A^\dagger$).
(AA^+ is a projector onto img(A).)

Simultaneous linear equation

Simultaneous linear equation（連立一次方程式）

$$A\vec{x} = \vec{b} \qquad A : M \times N, \vec{x} \in \boldsymbol{C}^N, \vec{b} \in \boldsymbol{C}^M$$

Two situations:

(1) There are solutions. $\Longleftrightarrow$ $\vec{b} \in \mathrm{img}(A)$

(i) There is the unique solution.

$$\mathrm{rank}(A) = N$$

(ii) There are infinite solutions.

$$\mathrm{rank}(A) < N$$

If A is a regular matrix,
$\vec{x} = A^{-1}\vec{b}$

(2) There is no solution. $\Longleftrightarrow$ $\vec{b} \notin \mathrm{img}(A)$

Pseudo inverse and simultaneous linear equation

Simultaneous linear equation $A\vec{x} = \vec{b}$ $A : M \times N, \vec{x} \in \boldsymbol{C}^N, \vec{b} \in \boldsymbol{C}^M$

(1) There are solutions. $\Leftrightarrow$ $\vec{b} \in \mathrm{img}(A)$

- A vector defined by the pseudo inverse as

$$\vec{x}' \equiv A^+\vec{b}$$

is one of the solutions.

Because $\vec{b} \in \mathrm{img}(A)$,there exists $\vec{v} : A\vec{v} = \vec{b}$.

$$\Rightarrow \quad A\vec{x}' = AA^+\vec{b} = AA^+A\vec{v} = A\vec{v} = \vec{b}$$

- $\vec{x}'$ has the smallest norm $||\vec{x}'||$ among the solutions.

$$||\vec{x}|| \geq ||A^+A\vec{x}|| = ||A^+\vec{b}|| = ||\vec{x}'||$$

$\boxed{\textbf{The pseudo inverse gives us the smallest norm solution.}}$

Pseudo inverse and simultaneous linear equation

Simultaneous linear equation $A\vec{x} = \vec{b}$ $\quad A : M \times N, \vec{x} \in \boldsymbol{C}^N, \vec{b} \in \boldsymbol{C}^M$

(2) There is no solution. $\Longleftrightarrow$ $\vec{b} \notin \mathrm{img}(A)$

- A vector defined by the pseudo inverse as

$$\vec{x}' \equiv A^+ \vec{b}$$

minimizes the "distance" $\|A\vec{x}' - \vec{b}\|$.

$$\vec{y} = A\vec{c} \in \mathrm{img}(A), \quad \vec{c} \in \boldsymbol{C}^N$$

$$\|\vec{y} - \vec{b}\|^2 = \|\underbrace{\vec{y} - AA^+\vec{b}}_{\in\, \mathrm{img}(A)} - \underbrace{(I - AA^+)\vec{b}}_{\in\, \mathrm{img}(A)^\perp}\|^2$$

$$= \|\vec{y} - AA^+\vec{b}\|^2 + \|\vec{b} - AA^+\vec{b}\|^2$$

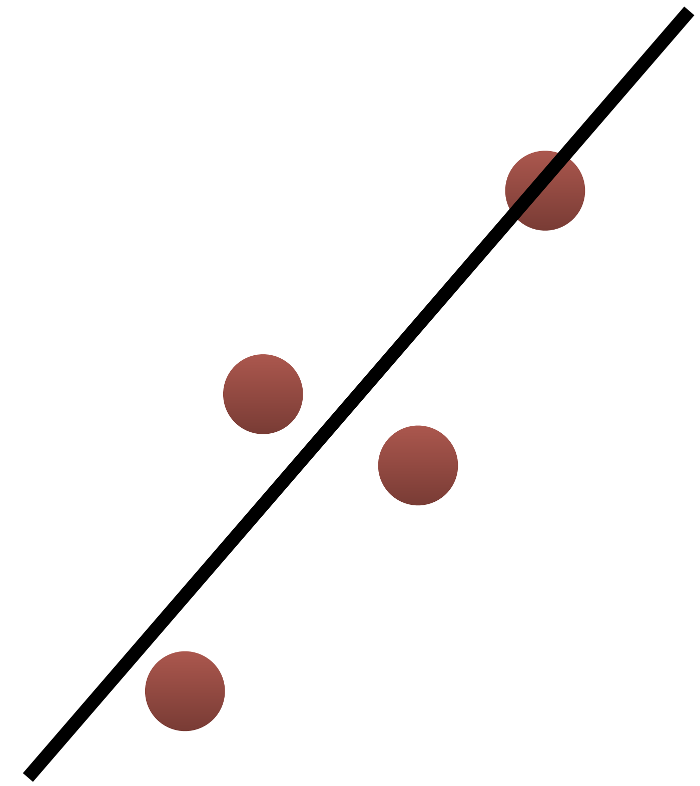
$$\geq \|\vec{b} - AA^+\vec{b}\|^2 = \|\vec{b} - A\vec{x}'\|^2$$

The pseudo inverse gives us approximate "least square solution".

Example of Least square solution problem

Fitting of a line to data points

$$y = ax + b$$



Data poins:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$$

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

$$A\vec{x} = \vec{b}$$

Least square fitting（最小二乗法） →

$$\begin{pmatrix} a \\ b \end{pmatrix} = A^+ \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Low rank approximation

Amount of data in SVD representation

$$A : M \times N$$

$$A = U\Sigma V^\dagger = U \begin{pmatrix} \Sigma_{r\times r} & 0_{r\times(N-r)} \\ 0_{(M-r)\times r} & 0_{(M-r)\times N-r} \end{pmatrix} V^\dagger$$

$$U = (\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_M)$$
$$V = (\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_N)$$

neglect zero singular values

$$= \bar{U}\Sigma_{r\times r}\bar{V}^\dagger$$

$$\bar{U} : M \times r, \bar{V}^\dagger : r \times N$$

$$\bar{U} = (\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r)$$
$$\bar{V} = (\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r)$$

If rank(A) is much smaller than M and N,

$$r \ll M, N$$

we can reduce the data to represent A.

(At this stage, no data loss)

Low rank approximation

Low rank approximation（低ランク近似）

Find an approximate matrix

$$A \simeq \tilde{A}$$

with lower rank:

$$\mathrm{rank}(A) > \mathrm{rank}(\tilde{A})$$

Through the low rank approximation,
we can reduce amount of the data.

An example of information compressions.

Notice! In order to quantify accuracy of the approximation,
we need a measure of distance between matrices.

Low rank approximation by SVD

Consider a matrix obtained by neglecting smaller singular values

$$A = \bar{U}\Sigma_{r\times r}\bar{V}^\dagger \quad \Longrightarrow \quad \tilde{A} = \tilde{U}\Sigma_{k\times k}\tilde{V}^\dagger \qquad (k < r)$$

$$\boxed{\begin{aligned}\Sigma_{r\times r} &= \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)\\ \bar{U} &= (\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r)\\ \bar{V} &= (\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r)\end{aligned}}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

$$\mathrm{rank}(A) = r$$

$$\boxed{\begin{aligned}\Sigma_{k\times k} &= \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)\\ \tilde{U} &= (\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k)\\ \tilde{V} &= (\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k)\end{aligned}}$$

Keep the largest k singular values
(and corresponding singular vectors).

$$\mathrm{rank}(\tilde{A}) = k < r$$

This approximation is one of the low rank approximation.

* For this approximation, we need $\mathrm{O}(MNk)$ calculations for SVD of a $M\times N$ matrix.

Norm of matrices $\|A\|$

There are two popular norms:

(1) **Frobenius norm** （フロベニウス　ノルム）

$$\|A\|_F = \sqrt{\sum_{i,j} |A_{ij}|^2} = \sqrt{\mathrm{Tr}(A^\dagger A)}$$

*Trace（対角和）

$$\mathrm{Tr}(X) = \sum_i X_{ii}$$

(2) **Operator norm**（作用素ノルム）

$$\|A\|_O = \inf\{c \geq 0; \|A\vec{x}\| \leq c\|\vec{x}\|\}$$
$$= \sigma_1(A)$$

*inf =infimum（下限）

*We define the norm for a vector as

$$\|\vec{x}\| = \sqrt{\sum_i |x_i|^2}$$

By using these norms, we define the distance between matrices:

$$\|A - \tilde{A}\|$$

Accuracy of low rank approximation by SVD

Theorem

For $A : M \times N$

$$\min\{\|A - B\|_F : \mathrm{rank}(B) = k\} = \sqrt{\sum_{i=k+1}^{\min(N,M)} \sigma_i^2}$$

$$\min\{\|A - B\|_O : \mathrm{rank}(B) = k\} = \sigma_{k+1}$$

Because the k-rank approximation by SVD gives

$$\|A - \tilde{A}\|_F = \sqrt{\sum_{i=k+1}^{\min(N,M)} \sigma_i^2}, \qquad \|A - \tilde{A}\|_O = \sigma_{k+1}$$

it is an "optimal" approximation with rank k.

Short proof of the theorem: Frobenius norm

*This proof is based on
"システム制御のための数学（１）"太田快人 著

From the inequality of singular values for matrix addition (property 5),

for j=1,... ,$\min(M,N)$ $\quad (\mathrm{rank}(B) = k)$

$$\sigma_{j+k}(A) = \sigma_{j+k}((A-B)+B) \leq \sigma_j(A-B)$$

Property 5

By taking a square and summing up them

$$\sum_{i=k+1}^{\min(M,N)} \sigma_i^2(A) \leq \sum_{j=1}^{\min(M,N)} \sigma_j^2(A-B) = \|A-B\|_F^2$$

*Note $\sigma_j(A) = 0 \quad (j > \mathrm{rank}(A))$

Short proof of the theorem: operator norm

*This proof is based on
"システム制御のための数学（１）"太田快人 著

From the min-max theorem of singular values (property 4),

$$(\mathrm{rank}(B) = k)$$

$$\sigma_{k+1}(A) \leq \max_{\vec{x} \in \ker(B), \|\vec{x}\|=1} \|A\vec{x}\| = \max_{\vec{x} \in \ker(B), \|\vec{x}\|=1} \|(A-B)\vec{x}\|$$

Property4 with

$$\boldsymbol{S} = \mathrm{img}(B^\dagger)$$
$$\boldsymbol{S}^\perp = \ker(B)$$

$$B\vec{x} = 0 \quad (\vec{x} \in \ker(B))$$

$$\leq \max_{\|\vec{x}\|=1} \|(A-B)\vec{x}\| = \|A-B\|_O$$

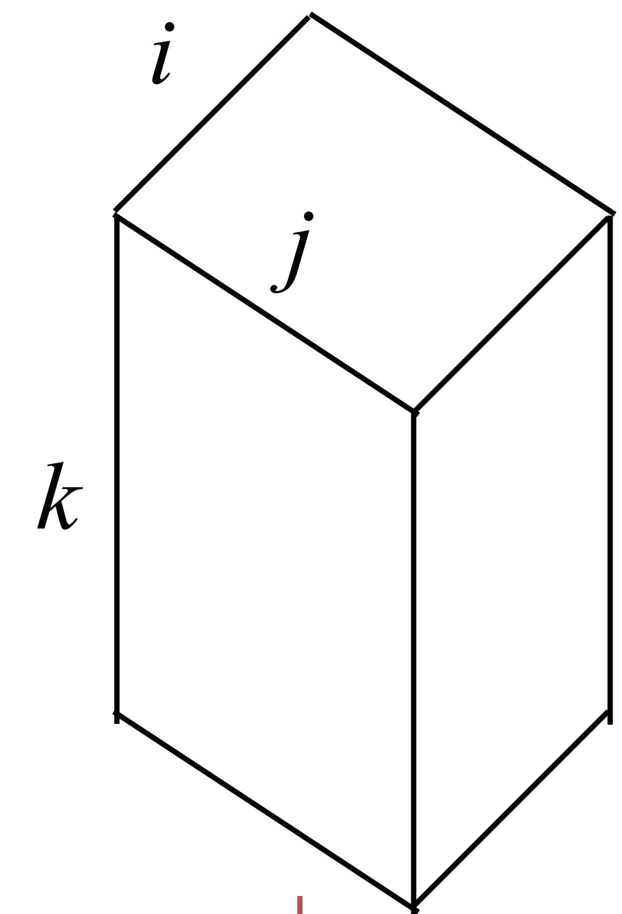
Expand the vector space

Definition of the operator norm

Generalization to tensor

Tensor: $T_{ijk\ldots}$

Higher dimensional "table" of numbers

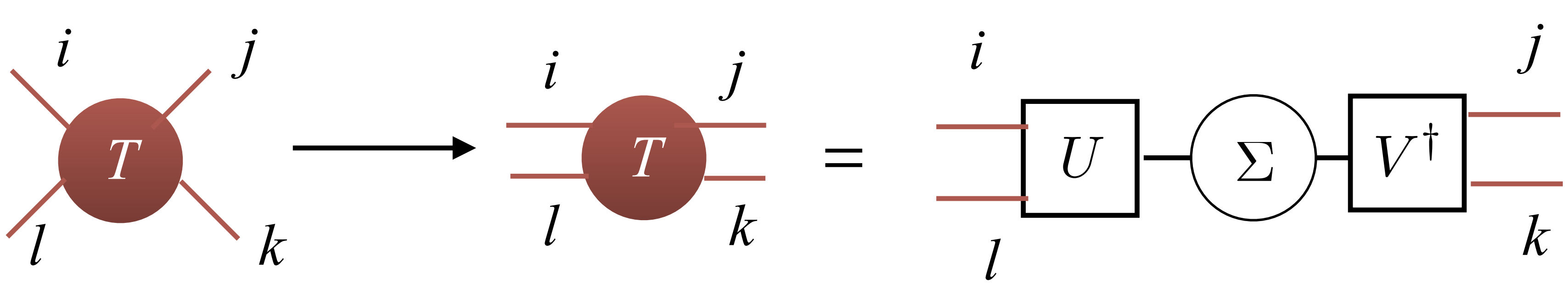


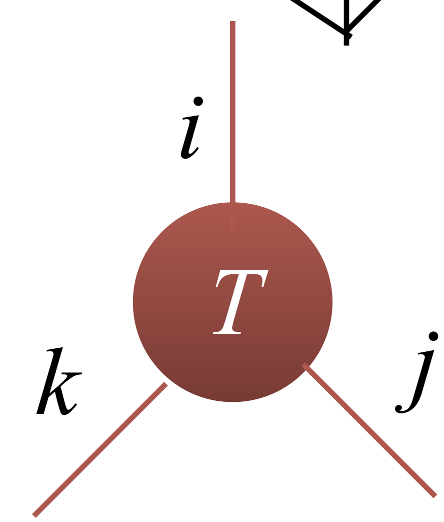
Naive application of SVD:

Make a matrix by dividing indices into two parts.

$$T_{ijkl} \to T_{(il),(jk)}$$

Then apply SVD (and low rank approximation).



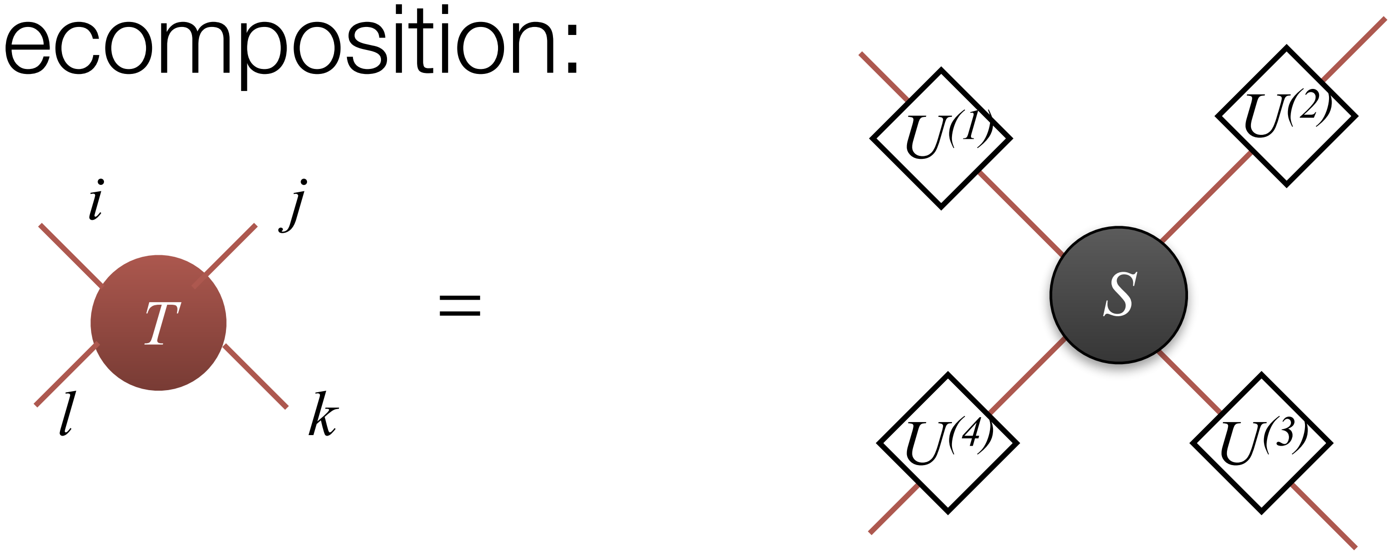


Note: The result depends on the initial mapping to a matrix.

Tucker decomposition: generalization of SVD

Review: T. G. Kolda et al, SIAM Review **51**, 455 (2006)

Tucker decomposition:



$U^{(i)}$: Factor matrix (usually unitary)

S :Core tensor

$$T_{ijkl} = \sum_{i'=1}^{I}\sum_{j'=1}^{J}\sum_{k'=1}^{K}\sum_{l'=1}^{L} S_{i'j'k'l'} U^{(1)}_{ii'} U^{(2)}_{jj'} U^{(3)}_{kk'} U^{(4)}_{ll'}$$

Low "rank" approximation

$$T_{ijkl} \simeq \sum_{i'=1}^{I'}\sum_{j'=1}^{J'}\sum_{k'=1}^{K'}\sum_{l'=1}^{L'} S_{i'j'k'l'} U^{(1)}_{ii'} U^{(2)}_{jj'} U^{(3)}_{kk'} U^{(4)}_{ll'}$$

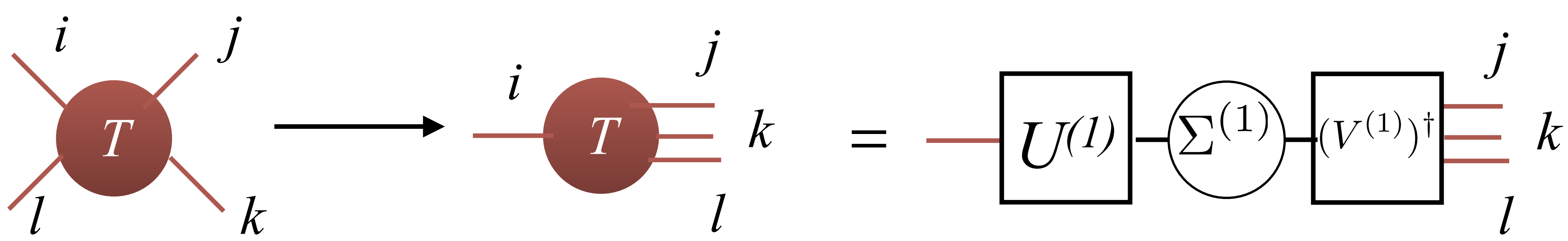
$I' < I, \quad J' < J, \quad K' < K, \quad L' < L$

rank-(I',J', K', L') approximation

Higher order SVD (HOSVD)

L. De Lathauwer et al, SIAM J. Matrix Anal. & Appl., **21**, 1253 (2000)

Define a factor matrix from matrix SVD:



Core tensor is calculated as

$$S_{i'j'k'l'} \equiv \sum_{ijkl} T_{ijkl} (U^{(1)})^{\dagger}_{i'i} (U^{(2)})^{\dagger}_{j'j} (U^{(3)})^{\dagger}_{k'k} (U^{(4)})^{\dagger}_{l'l}$$

Properties of the core tensor

$$S^{*}_{:,i_n=\alpha,:,:} \cdot S_{:,i_n=\beta,:,:} = \begin{cases} 0 & (\alpha \neq \beta) \\ (\sigma_{\alpha}^{(n)})^2 & (\alpha = \beta) \end{cases}$$

Dot product

$$A \cdot B \equiv \sum_{i,j,k,l} A_{ijkl} B_{ijkl}$$

Generalization of the diagonal matrix Σ in matrix SVD.

* Low-rank approximation based on HOSVD is not optimal.

Application of low rank approximation

Sample codes are uploaded uploaded on github and ITC-LMS.

SVD_sample.zip

(python2.7 + numpy + PIL)

Image compression: grayscale image

Image: 1024×768 pixels

768 x1024 matrix A

$$\mathrm{rank}(A) = 768$$

$\boxed{\text{Amount of data=786,432}}$

Perform SVD of A: $A = U\Sigma V^\dagger$

$\mathrm{rank}(\chi)$ approximation

$\boxed{\text{Amount of data=(768 +1024 + 1)}\times\chi}$

Image compression: grayscale image

Rank:	$\chi = 768$	$\chi = 100$	$\chi = 10$
Data:	786,432 (Original)	179,300	179,30

Sample code: image_svd.py

```
from PIL import Image ## Python Imaging Library
import numpy as np ## numpy

img = Image.open("./sample.jpg") ## load image
img_gray = img.convert("L") ## convert to grayscale
img_gray.show() ## show image
img_gray.save("./gray.png") ## save grayscale image
#img_gray.save("./gray.jpg") ## save grayscale image in jpg


array = np.array(img_gray) ## convert to ndarray

u,s,vt = np.linalg.svd(array,full_matrices=False) ## svd

#truncation
chi = 10
u = u[:,:chi]
vt = vt[:chi,:]
s = s[:chi]

array_truncated = np.dot(np.dot(u,np.diag(s)),vt) ## make truncated array

img_gray_truncated = Image.fromarray(np.uint8(array_truncated)) ## convert to grayscale image

img_gray_truncated.show() ## show image
img_gray_truncated.save("./gray_truncated.png") ## save compressed image
#img_gray_truncated.save("./gray_truncated.jpg") ## save compressed image in jpg
```

Image compression: color image

Image: 1024 × 768 pixels

768 x1024x3 tensor T

$\boxed{\text{Amount of data=2,359,296}}$

* Sub matrices for RGB colors

$$R_{ij} = T_{ij1}, \quad G_{ij} = T_{ij2}, \quad B_{ij} = T_{ij3}$$

Two image compressions:

Perform SVD for R, G, B → $\mathrm{rank}(\chi)$ approximation for RGB matrices

$\boxed{\text{Amount of data=}3\times(768 + 1024 + 1)\times\chi}$

Perform HOSVD for T → $\mathrm{rank}-(\chi', \chi', 3)$ approximation

$\boxed{\text{Amount of data= }(768 + 1024 + 3\chi)\times\chi}$

Image compression: color image

Original

SVD

$$\chi = 100$$

HOSVD

$$\chi' = 200$$

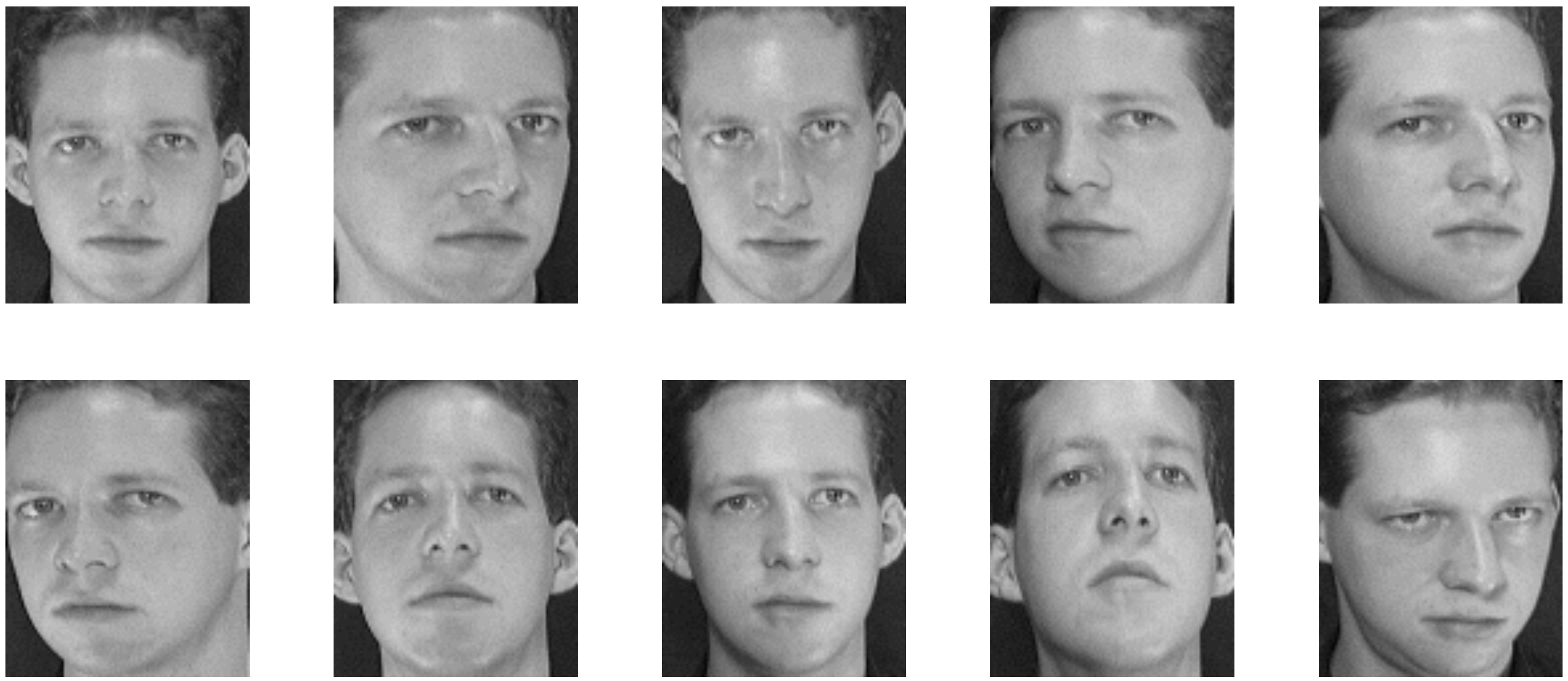
Data: 2,359,296 537,900 478,400

Sample code:
image_color_svd.py, image_color_hosvd.py

```python
img = Image.open("./sample_color.jpg") ## load image
img.show() ## show image
img.save("./img_original.png") ## save image

array = np.array(img) ## convert to ndarray
print "Array shape:", array.shape

array_truncated = np.zeros(array.shape)

## svd for each color

chi = 100
for i in range(3):
    u,s,vt = np.linalg.svd(array[:,:,i],full_matrices=False) ## svd

    #truncation
    u = u[:,:chi]
    vt = vt[:chi,:]
    s = s[:chi]

    array_truncated[:,:,i] = np.dot(np.dot(u,np.diag(s)),vt) ## make truncated array
```

```python
## row
matrix = np.reshape(array,(array.shape[0],array.shape[1]*array.shape[2]))
u,s,vt = np.linalg.svd(matrix[:,:],full_matrices=False) ## svd

#truncation
u1 = u[:,:chi]

## column
matrix = np.reshape(np.transpose(array,(1,0,2)),(array.shape[1],array.shape[0]*array.shape[2]))
u,s,vt = np.linalg.svd(matrix[:,:],full_matrices=False) ## svd

#truncation
u2 = u[:,:chi]

## for RGB we do not truncate
## make projectors
p1 = np.dot(u1,(u1.conj()).T)
p2 = np.dot(u2,(u2.conj()).T)

## make truncated array
array_truncated = np.tensordot(np.tensordot(array,p1,axes=(0,1)),p2,axes=(0,1)).transpose(1,2,0)
```

Image compression: multi images



92×122 pixels 10 images

Images were taken from ORL Database of Faces, AT&T Laboratories Cambridge

$92 \times 122 \times 10$ tensor T

$\boxed{\text{Amount of data}=112{,}240}$

by HOSVD

$\mathrm{rank}-(\chi, \chi, \chi')$ approximation

Amount of data=
$(92+122)\times\chi + 10\times\chi' + \chi^2\chi'$

Image compression: multi images

Data

Original

112,240

$\chi = 30$

$\chi' = 10$

15,520

$\chi = 30$

$\chi' = 9$

14,610

$\chi = 30$

$\chi' = 5$

10,970

Sample code: image_multi_hosvd.py

```
array=[]
for i in range(1,11):
    img = Image.open("./samples/"+repr(i)+".bmp") ## load image
    array.append(np.array(img)) ## convert to ndarray
array=np.array(array).transpose(1,2,0)

array_truncated = np.zeros(array.shape)

chi = 30
chi_p = 9

## row
matrix = np.reshape(array,(array.shape[0],array.shape[1]*array.shape[2]))
u,s,vt = np.linalg.svd(matrix[:,:],full_matrices=False) ## svd

#truncation
u1 = u[:,:chi]

## column
matrix = np.reshape(np.transpose(array,(1,0,2)),(array.shape[1],array.shape[0]*array.shape[2]))
u,s,vt = np.linalg.svd(matrix[:,:],full_matrices=False) ## svd

#truncation
u2 = u[:,:chi]

## layer
matrix = np.reshape(np.transpose(array,(2,0,1)),(array.shape[2],array.shape[0]*array.shape[1]))
u,s,vt = np.linalg.svd(matrix[:,:],full_matrices=False) ## svd

#truncation
u3 = u[:,:chi_p]

## make projectors
p1 = np.dot(u1,(u1.conj()).T)
p2 = np.dot(u2,(u2.conj()).T)
p3 = np.dot(u3,(u3.conj()).T)

## make truncated array
array_truncated = np.tensordot(np.tensordot(np.tensordot(array,p1,axes=(0,1)),p2,axes=(0,1)),p3,axes=(0,1))

for i in range(1,11):
    img_truncated = Image.fromarray(np.uint8(array_truncated[:,:,i-1])) ## convert to each image
    img_truncated.save("./outputs/"+repr(i)+".bmp") ## save compressed image
```

References:

- 齋藤正彦、「線形代数入門」東京大学出版会
- 太田快人、「システム制御のための数学（１）ー線形代数編ー」、コロナ社
- T. G. Kolda et al, SIAM Review **51**, 455 (2006).

Next week

第１回：　現代物理学における巨大なデータ
第２回：　情報圧縮と繰り込み
第３回：　情報圧縮の数理１　(線形代数の復習)
第４回：　情報圧縮の数理２　(特異値分解と低ランク近似)
第５回：　情報圧縮の数理３　(スパース・モデリングの基礎)

(Basics of sparse modeling) by Yamaji sensei

第６回：　情報圧縮の数理４　(クリロフ部分空間法の基礎)
第７回：　物質科学における情報圧縮
第８回：　スパース・モデリングの物質科学への応用
第９回：　クリロフ部分空間法の物質科学への応用
第１０回：　行列積表現の基礎
第１１回：　行列積表現の応用
第１２回：　テンソルネットワーク表現への発展
第１３回：　テンソルネットワーク繰り込みと低ランク近似の応用