Exploratory Data Analysis

Clint P. George

University of Florida Informatics Institute

December 05, 2016

Outline

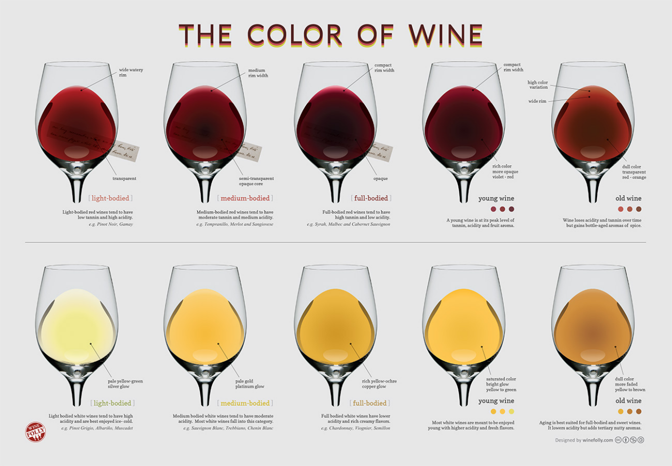
1. Principal Component Analysis (PCA)
2. Introduction to Document Modeling
3. Term-Frequency Inverse Document Frequency (TF-IDF)
4. Latent Semantic Analysis (LSA)

Principle Component Analysis: Goal

We wish to summarize datasets which may contain several redundant features (or characteristics).

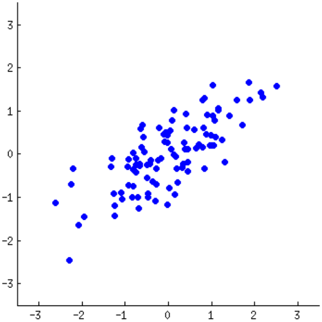
Principle Component Analysis: Goal

We wish to summarize datasets which may contain several redundant features (or characteristics).



Principle Component Analysis: Goal

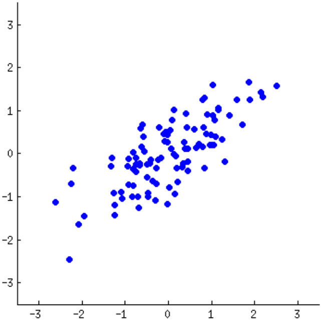
We wish to summarize datasets which may contain several redundant features (or characteristics).



synthetic data: x-axis - color intensity, y-axis - alcohol content

Principle Component Analysis: Goal

We wish to summarize datasets which may contain several redundant features (or characteristics).



synthetic data: x-axis - color intensity, y-axis - alcohol content

- look for some features that strongly differ across data points.
- look for the properties that would allow you to "reconstruct" well the original features

Eigenvectors and Eigenvalues: Overview

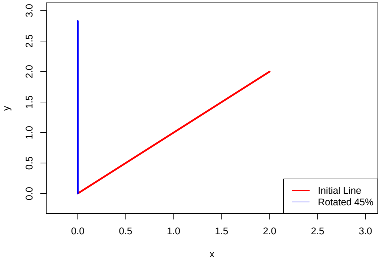
Let C be an $n \times n$ matrix and $\mathbf{u}$ is an $n \times 1$ vector.— $C\mathbf{u}$ is well-defined.

Typically, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{u}$

Eigenvectors and Eigenvalues: Overview

Let C be an $n \times n$ matrix and $\mathbf{u}$ is an $n \times 1$ vector.— $C\mathbf{u}$ is well-defined.

Typically, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{u}$



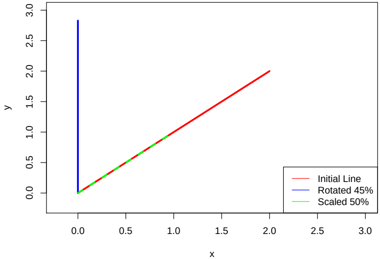
$$(x_1, y_1) = (0, 0)$$
$$(x_2, y_2) = (2, 2)$$

$$C = \begin{bmatrix} \cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) \\ -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix}$$

Eigenvectors and Eigenvalues: Overview

Let C be an $n \times n$ matrix and $\mathbf{u}$ is an $n \times 1$ vector.— $C\mathbf{u}$ is well-defined.

Typically, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{u}$, unless the vector is *special* as in this transform $C\mathbf{u} = \lambda\mathbf{u}$, where λ is a scale that C performs over the vector $\mathbf{u}$.



$$C = \begin{bmatrix} .5 & .0 \\ .0 & .5 \end{bmatrix}$$

Eigenvectors and Eigenvalues: Overview

Let C be an $n \times n$ matrix and $\mathbf{u}$ is an $n \times 1$ vector.— $C\mathbf{u}$ is well-defined.

Typically, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{u}$, unless the vector is *special* as in this transform $C\mathbf{u} = \lambda\mathbf{u}$, where λ is a scale that C performs over the vector $\mathbf{u}$.

These special vectors and their corresponding λ's are called **eigenvectors** and **eigenvalues** of C.

Eigenvectors and Eigenvalues: Overview

Let C be an $n \times n$ matrix and $\mathbf{u}$ is an $n \times 1$ vector.— $C\mathbf{u}$ is well-defined.

Typically, multiplication by a matrix changes the direction of a *non-zero* vector $\mathbf{u}$, unless the vector is *special* as in this transform $C\mathbf{u} = \lambda\mathbf{u}$, where λ is a scale that C performs over the vector $\mathbf{u}$.

These special vectors and their corresponding λ's are called **eigenvectors** and **eigenvalues** of C. C can have upto n distinct eigenvalues.

Eigenvectors and Eigenvalues: Facts

Let U be an $n \times n$ matrix with n eigenvectors of C and Λ is the $n \times n$ diagonal matrix with the eigenvalues of C along its diagonal.

The column vectors of U are linearly independent, which gives

$$CU = U\Lambda \rightarrow C = U\Lambda U^{-1}.$$

This **diagonalizes** the matrix C.

If C is symmetric ($C = C^{\mathsf{T}}$), then its eigenvectors are perpendicular and we can have $U^{-1} = U^{\mathsf{T}}$ and

$$C = U\Lambda U^{\mathsf{T}}$$

Principle Component Analysis: Approach

Let X be a centered $m \times n$ data matrix.

We can write the $n \times n$ covariance matrix C as:

$$C = \frac{X^{\mathsf{T}} X}{n-1} = U \Lambda U^{\mathsf{T}},$$

where U is the matrix of eigenvectors $\mathbf{u}_i$ (each column is an eigenvector) and Λ is the diagonal matrix with eigenvalues λ_i on the diagonal.

Principle Component Analysis: Approach

Let X be a centered $m \times n$ data matrix.

We can write the $n \times n$ covariance matrix C as:

$$C = \frac{X^\mathsf{T} X}{n-1} = U \Lambda U^\mathsf{T},$$

where U is the matrix of eigenvectors $\mathbf{u}_i$ (each column is an eigenvector) and Λ is the diagonal matrix with eigenvalues λ_i on the diagonal.

PCA transformation: projections of the data X on the **principal components**, i.e. XU. One only needs to keep the most informative principal components.

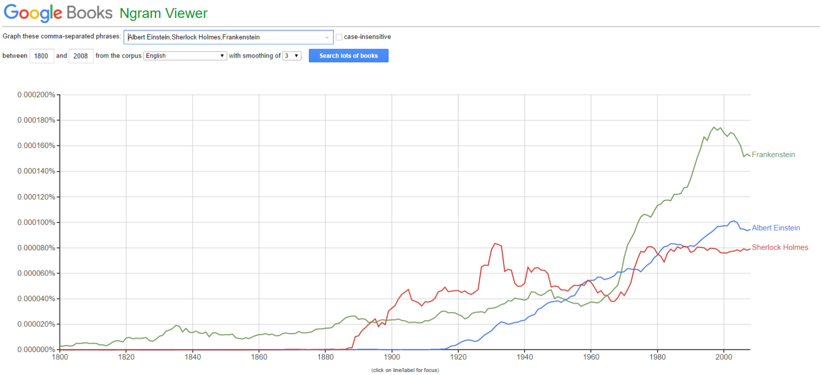
Text Corpus Exploration

We have a big pile of text documents (corpus).—What's going on inside?[1]

[1] PC: Olivia Harris, Reuters

Text Corpus Exploration

- Comparing document covariates—How do individual words correlate?



- Clustering and topic modeling
- Organizing and searching documents—information retrieval

An Information Retrieval Problem

Keyword-based search:

- searching for documents of interest
- e.g., keywords: computers, laptop, etc.

Implementing Keyword-based Search

An approach is via **Vector Space Modeling**

- Convert a corpus m documents and n vocabulary terms into a **term-document** $(n \times m)$ matrix
- Translate both documents and user keywords into vectors in vector space
- Define similarity between these vectors, e.g., via cosine similarity—small angle $\equiv$ large cosine $\equiv$ similar

TF-IDF

Term frequency inverse document frequency matrix (TF-IDF)[2]—a popular scheme

For each term t in document d, we compute

$$\text{tf-idf}_{dt} = \text{tf}_{dt} \times \log\left(\frac{n}{\text{df}_t}\right)$$

- tf_{dt} is the frequency of term t in document d
- df_t is the number of documents where term t appears

[2] Salton et al. (1975)

Hands-on Python: TF-IDF and document retrieval

Information Retrieval: Challenges

If we search for the keyword *computers*, we may miss documents that do not have *computers* and contain *PC*, *laptop*, *desktop*, etc.

Information Retrieval: Challenges

If we search for the keyword *computers*, we may miss documents that do not have *computers* and contain *PC*, *laptop*, *desktop*, etc.

- Problem #1: Synonymy—words with similar meaning

Information Retrieval: Challenges

If we search for the keyword *computers*, we may miss documents that do not have *computers* and contain *PC*, *laptop*, *desktop*, etc.

- Problem #1: Synonymy—words with similar meaning

Suppose, we search for the keyword *chair*, we may get documents that contain "the chair of the board" and "the chair maker"

Information Retrieval: Challenges

If we search for the keyword *computers*, we may miss documents that do not have *computers* and contain *PC*, *laptop*, *desktop*, etc.

- Problem #1: Synonymy—words with similar meaning

Suppose, we search for the keyword *chair*, we may get documents that contain "the chair of the board" and "the chair maker"

- Problem #2: Polysemy—words with multiple meanings

Information Retrieval: Challenges

If we search for the keyword *computers*, we may miss documents that do not have *computers* and contain *PC*, *laptop*, *desktop*, etc.

- Problem #1: Synonymy—words with similar meaning

Suppose, we search for the keyword *chair*, we may get documents that contain "the chair of the board" and "the chair maker"

- Problem #2: Polysemy—words with multiple meanings

One solution: search and explore documents based on the themes or **topics** that run through them.

TF-IDF

Term frequency inverse document frequency matrix (TF-IDF)[3]—a popular scheme

For each term t in document d, we compute

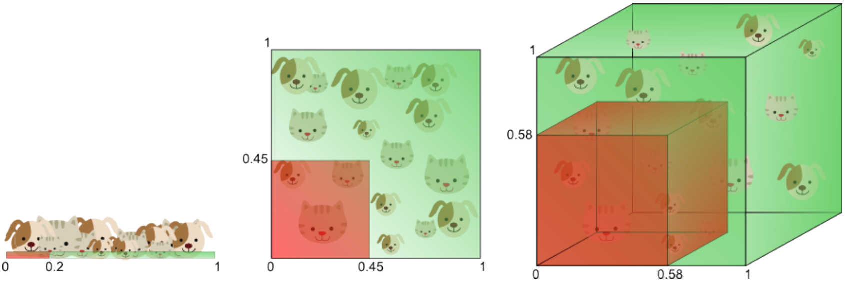
$$\text{tf-idf}_{dt} = \text{tf}_{dt} \times \log\left(\frac{n}{\text{df}_t}\right)$$

- tf_{dt} is the frequency of term t in document d
- df_t is the number of documents where term t appears

Problem #3: Working in the vocabulary space can cause computational challenges for large corpora.

[3] Salton et al. (1975)

The Curse of Dimensionality[5]



Suppose the available data (documents) are fixed and we keep adding dimensions (words).[4]

The more features we use, the more sparse the data becomes

[4]Image source: `www.visiondummy.com`
[5]Bellman (1961)

Latent Semantic Analysis (LSA)

LSA (Deerwester et al. 1990) aims to explore the “semantics” underlying documents.

By factorizing the TF-IDF ($n \times m$) matrix—Singular Value Decomposition

Singular Value Decomposition (SVD)

Let A be an $m \times n$ matrix with real values and $m > n$. Let $B = A^{\mathsf{T}} A$ be an $n \times n$ matrix.—it's symmetric.

Singular Value Decomposition (SVD)

Let A be an $m \times n$ matrix with real values and $m > n$. Let $B = A^{\mathsf{T}} A$ be an $n \times n$ matrix.—it's symmetric.

The eigenvalues, $\sigma_1, \ldots, \sigma_n$, of such matrices are real non-negative numbers. We then can write: $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_n^2$.

Singular Value Decomposition (SVD)

Let A be an $m \times n$ matrix with real values and $m > n$. Let $B = A^{\mathsf{T}} A$ be an $n \times n$ matrix.—it's symmetric.

The eigenvalues, $\sigma_1, \ldots, \sigma_n$, of such matrices are real non-negative numbers. We then can write: $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_n^2$.

The corresponding eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are perpendicular. We normalize them to have length 1. Let

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \text{ and } V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

where we define $\mathbf{v}_i = \frac{1}{\sigma_i} A\mathbf{u}_i$.

Singular Value Decomposition (SVD)

Let A be an $m \times n$ matrix with real values and $m > n$. Let $B = A^{\mathsf{T}} A$ be an $n \times n$ matrix.—it's symmetric.

The eigenvalues, $\sigma_1, \ldots, \sigma_n$, of such matrices are real non-negative numbers. We then can write: $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_n^2$.

The corresponding eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are perpendicular. We normalize them to have length 1. Let

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \text{ and } V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

where we define $\mathbf{v}_i = \frac{1}{\sigma_i} A\mathbf{u}_i$.

We can easily show that $\mathbf{v}_i$'s are perpendicular m-dimensional vectors of length 1 (orthonormal vectors).

Singular Value Decomposition (SVD)

By construction, we have

$$\mathbf{v}_j^\mathsf{T} A\mathbf{u}_i = \mathbf{v}_j^\mathsf{T}(\sigma_i\mathbf{v}_i) = \sigma_i\mathbf{v}_j^\mathsf{T}\mathbf{v}_i = \begin{cases}\sigma_i, & \text{if } i = j\\ 0, & \text{otherwise}\end{cases}$$

Singular Value Decomposition (SVD)

By construction, we have

$$\mathbf{v}_j^\mathsf{T} A\mathbf{u}_i = \mathbf{v}_j^\mathsf{T}(\sigma_i \mathbf{v}_i) = \sigma_i \mathbf{v}_j^\mathsf{T}\mathbf{v}_i = \begin{cases} \sigma_i, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

We can then write the matrix form:

$$V^\mathsf{T} A U = \Sigma$$

where Σ is the diagonal $n \times n$ matrix with $\sigma_1, \ldots, \sigma_n$ along the diagonal.—singular values.

Singular Value Decomposition (SVD)

By construction, we have

$$\mathbf{v}_j^\mathsf{T} A\mathbf{u}_i = \mathbf{v}_j^\mathsf{T}(\sigma_i \mathbf{v}_i) = \sigma_i \mathbf{v}_j^\mathsf{T} \mathbf{v}_i = \begin{cases} \sigma_i, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

We can then write the matrix form:

$$V^\mathsf{T} A U = \Sigma$$

where Σ is the diagonal $n \times n$ matrix with $\sigma_1, \ldots, \sigma_n$ along the diagonal.—singular values.

Since U and V have orthonormal columns, we can have $A = V\Sigma U^\mathsf{T}$

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

$$A_{m\times n} = V_{m\times n}\,\Sigma_{n\times n}\,U^{\mathsf{T}}_{n\times n}$$

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

$$A_{m\times n} = V_{m\times n}\, \Sigma_{n\times n}\, U^{\mathsf{T}}_{n\times n}$$

where the columns of U are eigenvectors of $A^{\mathsf{T}}A$

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

$$A_{m\times n} = V_{m\times n}\,\Sigma_{n\times n}\,U_{n\times n}^{\mathsf{T}}$$

where the columns of U are eigenvectors of $A^{\mathsf{T}}A$ and Σ is a diagonal matrix containing the square roots of eigenvalues of $A^{\mathsf{T}}A$ in descending order.

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

$$A_{m\times n} = V_{m\times n}\,\Sigma_{n\times n}\,U_{n\times n}^{\mathsf{T}}$$

where the columns of U are eigenvectors of $A^{\mathsf{T}}A$ and Σ is a diagonal matrix containing the square roots of eigenvalues of $A^{\mathsf{T}}A$ in descending order.

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

$$A_{m \times n} = V_{m \times n} \, \Sigma_{n \times n} \, U_{n \times n}^{\mathsf{T}}$$

where the columns of U are eigenvectors of $A^{\mathsf{T}} A$ and Σ is a diagonal matrix containing the square roots of eigenvalues of $A^{\mathsf{T}} A$ in descending order.

In LSA, we set all but the $(K \ll n)$ highest singular values to 0, giving a $K \times n$ approximation matrix—the “semantic” space

Singular Value Decomposition (SVD): Summary

SVD factorizes an $m \times n$ data matrix A into:

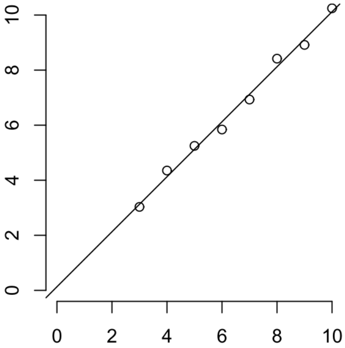
$$A_{m\times n} = V_{m\times n}\,\Sigma_{n\times n}\,U_{n\times n}^{\mathsf{T}}$$

where the columns of U are eigenvectors of $A^{\mathsf{T}}A$ and Σ is a diagonal matrix containing the square roots of eigenvalues of $A^{\mathsf{T}}A$ in descending order.

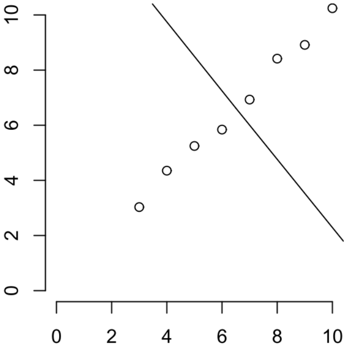
In LSA, we set all but the $(K \ll n)$ highest singular values to 0, giving a $K \times n$ approximation matrix—the "semantic" space

One can identify **similarities** between documents in this **semantic space**.

SVD: Geometric Interpretation

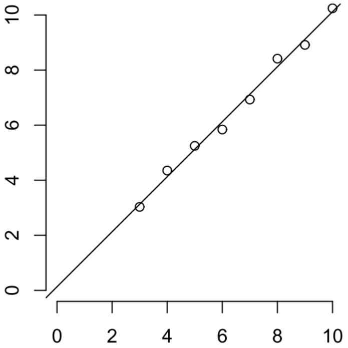


Regression line along 1st dimension

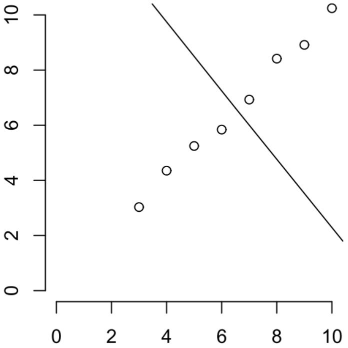


Regression line along 2nd dimension

SVD: Geometric Interpretation



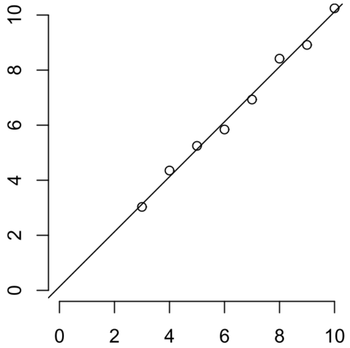
Regression line along 1st dimension



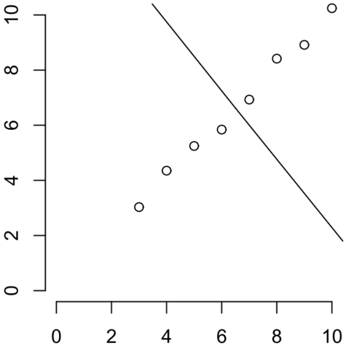
Regression line along 2nd dimension

The regression line on the 1st dimension (left) is the best approximation for the data—it is the line that minimizes the distance between each point and the line.

SVD: Geometric Interpretation



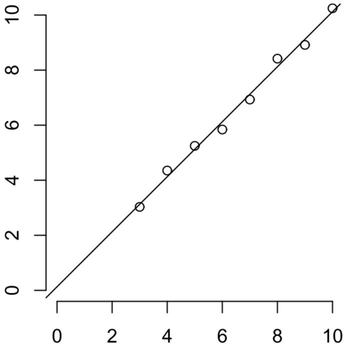
Regression line along 1st dimension



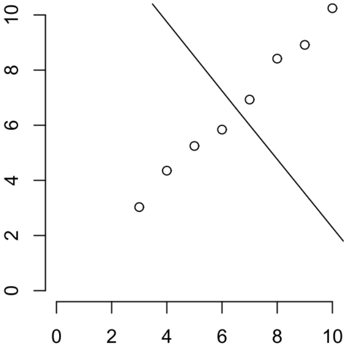
Regression line along 2nd dimension

The regression line on the 2nd dimension (right) does a poorer job of approximating the data, because it corresponds to a dimension exhibiting less variation

SVD: Geometric Interpretation



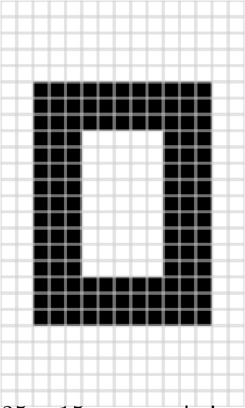
Regression line along 1st dimension



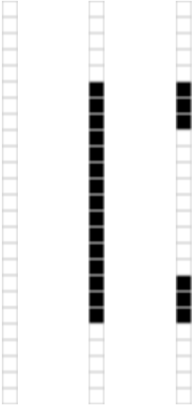
Regression line along 2nd dimension

SVD aims to find the dimensions along which data points exhibit the most variation.

Application of SVD: Data Compression



25×15 gray-scale image (375 cells)



Three types of columns/features

SVD on the matrix, A, gives three non-zero singular values.[6]

[6] Example from: http://www.ams.org/samplings/feature-column/fcarc-svd

Hands-on Python: Latent Semantic Analysis

PCA vs LSA

We assume a *centered* data matrix X. We write its covariance matrix C as[7]

$$\begin{aligned} C &= \frac{X^\intercal X}{n-1} && (1) \\ &= \frac{USV^\intercal VSU^\intercal}{n-1} \text{ (using SVD)} && (2) \\ &= U\frac{S^2}{n-1}U^\intercal && (3) \end{aligned}$$

[7] http://stats.stackexchange.com/questions/134282

PCA vs LSA

We assume a *centered* data matrix X. We write its covariance matrix C as[7]

$$C = \frac{X^{\mathsf{T}}X}{n-1} \quad (1)$$

$$= \frac{USV^{\mathsf{T}}VSU^{\mathsf{T}}}{n-1} \text{ (using SVD)} \quad (2)$$

$$= U\frac{S^2}{n-1}U^{\mathsf{T}} \quad (3)$$

Note: The right singular vectors U are principal axes, and singular values are related to the eigenvalues of the covariance matrix C via $\lambda_i = s_i^2/(n-1)$.

[7] `http://stats.stackexchange.com/questions/134282`

Questions?