Math behind NLP

Natural Language Processing

Piotr Fulmański

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
University of Lodz

Lecture goals

- Vector space (linear space).
- Linear transformation.
- Base and change of basis.
- Eigenvalues and eigenvectors.
- Eigendecomposition of a matrix.
- SVD - singular value decomposition.
- Dimensionality reduction with SVD.
- Variance and covariance.
- PCA - principal component analysis.
- Relationship between PCA and SVD.

Vector space

A linear space is a collection of elements (called vectors) that can be scaled and added

Linear space (vector space) - a set of elements (called ***vectors***) in which two actions are defined:

- **adding** vectors,
- **scaling** vectors, i.e. multiplication of vectors by numbers (called ***scalars***) from a fixed **field** (for example the field of real numbers).

As always in math, these actions must fulfill some additional axioms defined their properties.

Natural examples of linear spaces are two- and three-dimensional Euclidean spaces:

- vectors are identified with pairs and triples of real numbers, respectively,
- vectors are represented by geometric vectors (characterized by *direction* and *magnitude* (*value, size, length*), usually represented as arrows; in Polish we describe each vector by *kierunek*, *zwrot* i *moduł* (sometimes called *value*)).

The properties of geometric vectors provide a good intuitive model for vectors in more abstract linear spaces that have no geometric interpretation. Examples are:

- the set of all polynomials with real coefficients - the polynomial is a non-geometric vector;
- a set of square matrices of the same dimension - the matrix is a non-geometric vector.

Linear transformation

In nonelementary mathematics (linear algebra), a **linear map** (also called a **linear mapping**, **linear transformation** or, in some contexts, **linear function**) is a function (mapping) $f\colon\ U \to V$ between two vector spaces U and V that preserves the operations of addition and scalar multiplication:

- mapping the sum of vectors from one space to another is equal to the sum of the mappings of individual vectors of this sum,
- the mapping of the product of a vector by the scalar is equal to the product of the scalar by mapping the given vector.

A linear map is said to be *operation preserving* as *it does not matter* whether the linear map is applied *before* or *after* the operations of addition and scalar multiplication.

Linear transformation

Let U and V be *vector spaces* over the same field K (e.g. real or complex numbers). A function $f\colon U \to V$ is said to be a ***linear map*** if for any two vectors $\mathbf{u}, \mathbf{v} \in U$ and any scalar $c \in K$ the following two conditions are satisfied:

- additivity (preserves adding vectors)
  $f(\mathbf{u}+\mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$,
- homogeneity (preserves scalar multiplication)
  $f(c\mathbf{u}) = cf(\mathbf{u})$.

For example:

- For real numbers, the map $f(x) = x^2$ is not linear.
- For real numbers, the map $f(x) = x + 1$ is not linear (but is a linear equation, as the term is used in analytic geometry).

If U, V are linear spaces of a finite dimension, $\dim U = n$, $\dim V = m$ then the linear transformation between them can be represented by a matrix $m \times n$.

If A is a real $m \times n$ matrix, then A defines a linear map from R^n to R^m by sending the column vector $x \in R^n$ to the column vector $Ax \in R^m$. Conversely, any linear map between finite-dimensional vector spaces can be represented in this manner.

Linear transformation

Examples of linear transformation matrices

In two-dimensional space R^2 linear maps are described by 2×2 real matrices. These are some examples:

rotation
by 90 degrees counterclockwise:

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

by an angle Θ counterclockwise:

$$A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

reflection
through the X axis:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

through the Y axis:

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

through a line making an angle Θ with the origin:

$$A = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}$$

scaling
by 2 in all directions:

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

shear

horizontal:

$$A = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}$$

squeeze:

$$A = \begin{pmatrix} k & 0 \\ 0 & \frac{1}{k} \end{pmatrix}$$

projection
onto the Y axis:

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Vector space

Base - intuition

In mathematics, a set B of elements (vectors) in a vector space V is called a ***basis***, if every element of V may be written in a **unique** way as a (finite) linear combination of elements of B. The **coefficients of this linear combination** are referred to as ***components*** or ***coordinates*** on B of the vector. The elements of a basis are called *basis vectors*.

A vector space can have **several bases.**

All the bases have the same number of elements, called the *dimension of the vector space*.

Vector space

Base - more precisely

A basis B of a vector space V over a field F (such as the real numbers R) is a *linearly independent* subset of V that *spans* V. This means that a subset B of V is a basis if it satisfies the two following conditions:

- the **linear independence property**:
  for every finite subset $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ of B, if $c_1\mathbf{v}_1 + \ldots + c_m\mathbf{v}_m = 0$ for some $c_1, \ldots, c_m$ in F, then $c_1 = \ldots = c_m = 0$;
- the **spanning property**:
  for every vector $\mathbf{v}$ in V, one can choose $a_1, \ldots, a_n$ in F and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in B such that $\mathbf{v} = a_1\mathbf{v}_1 + \ldots + a_n\mathbf{v}_n$.

The scalars a_i are called the *coordinates* of the vector v with respect to the basis B, and by the first property they are uniquely determined.

For example, the coordinate vectors $\mathbf{e}_1 = (1,0,\ldots,0)$, $\mathbf{e}_2 = (0,1,\ldots,0)$, to $\mathbf{e}_n = (0,0,\ldots,1)$, form a basis of F^n, called the *standard basis*, since any vector $(x_1, x_2, \ldots, x_n)$ can be uniquely expressed as a linear combination of these vectors:

$$(x_1, x_2, \ldots, x_n) = x_1(1,0,\ldots,0) + x_2(0,1,\ldots,0) + \ldots + x_n(0,0,\ldots,0) = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \ldots + x_n\mathbf{e}_n$$

The corresponding coordinates $x_1, x_2, \ldots, x_n$ are just the Cartesian coordinates of the vector.

Vector space

Change of basis

Let V be a vector space of dimension n over a field F. Given two (ordered) bases $B_{\text{old}} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and $B_{\text{new}} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ of V, it is often useful to express the coordinates of a vector x which are given with respect to B_{old} in terms of the coordinates with respect to B_{new}.

Often the space V with base B_{old} is called **parent space**, while space V with base B_{new} is called **child space**.

It is useful to **describe the old coordinates in terms of the new ones**, because, in general, one has expressions involving the old coordinates, and one wants to obtain equivalent expressions in terms of the new coordinates. This is obtained by replacing the old coordinates by their expressions in terms of the new coordinates and can be done by the *change-of-basis* formula.

Vector space

Change of basis

Let $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are the coordinates of a vector $\mathbf{x}$ over the old ($B_{\text{old}} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$) and the new ($B_{\text{new}} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$) basis respectively. Then we have

(1) $\mathbf{x} = \sum_{i=1}^{n} x_i \mathbf{v}_i,$

and

(2) $\mathbf{x} = \sum_{i=1}^{n} y_i \mathbf{w}_i.$

Typically, the new basis vectors are given by their coordinates over the old basis, that is

(3) $\mathbf{w}_j = \sum_{i=1}^{n} a_{i,j} \mathbf{v}_i$

for $i = 1, \ldots, n$. The the change-of-basis formula is $\mathbf{x} = A\mathbf{y}$:

(4) $x_i = \sum_{j=1}^{n} a_{i,j} y_j$

for $i = 1, \ldots, n$ - we express the old coordinates x_i in terms of the new ones y_j. This formula is a simple consequence of the following

$$\mathbf{x} = \sum_{j=1}^{n} y_j \mathbf{w}_j = \sum_{j=1}^{n} y_j \left(\sum_{i=1}^{n} a_{i,j} \mathbf{v}_i \right) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} a_{i,j} y_j \right) \mathbf{v}_i.$$

From the above and (2) we have

$$\sum_{i=1}^{n} x_i \mathbf{v}_i = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} a_{i,j} y_j \right) \mathbf{v}_i$$

so we have $x_i = \sum_{j=1}^{n} a_{i,j} y_j.$

Vector space

Transform from child space to parent space

When we want to go from some new coordinate system (child space) with basis vectors $\mathbf{u}$, $\mathbf{v}$ to the old coordinates (parent space) with basis vectors $\mathbf{e_1}$, $\mathbf{e_2}$, where all vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{e_1}$, $\mathbf{e_2}$ are

- unit vectors,
- mutually perpendicular

and vectors $\mathbf{u}$ and $\mathbf{v}$ are expressed in parent space then the transformation matrix A is one whose **columns** are $\mathbf{u}$ and $\mathbf{v}$. (Why? See next slide.)

For example, we can define new basis over the old basis using formula (3) as follow:

$$\mathbf{w}_1 = \mathbf{u} = \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix} = \frac{3}{5}\mathbf{e}_1 + \frac{4}{5}\mathbf{e}_2 = \frac{3}{5}\mathbf{v}_1 + \frac{4}{5}\mathbf{v}_2 \longrightarrow a_{11} = \frac{3}{5}, a_{21} = \frac{4}{5}$$

$$\mathbf{w}_2 = \mathbf{v} = \begin{bmatrix} -\frac{4}{5} \\ \frac{3}{5} \end{bmatrix} = -\frac{4}{5}\mathbf{e}_1 + \frac{3}{5}\mathbf{e}_2 = -\frac{4}{5}\mathbf{v}_1 + \frac{3}{5}\mathbf{v}_2 \longrightarrow a_{12} = -\frac{4}{5}, a_{22} = \frac{3}{5}$$

In consequence matrix A takes the form:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

$\mathbf{u}$ $\mathbf{v}$

Vector space

Transform from child space to parent space - why is A matrix of the form it is?

From (4) we have:

$$x_i = \sum_{j=1}^{n} a_{i,j} y_j$$

Than the point $p_C = (y_1, y_2)$ from the $\mathbf{uv}$ (child) space expressed in $\mathbf{e}_1\mathbf{e}_2$ parent coordinates as $p_P = (x_1, x_2)$ is:

$$p_P = \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{2} a_{1,j} y_j \\ \sum_{j=1}^{2} a_{2,j} y_j \end{bmatrix} = \begin{bmatrix} a_{11}y_1 & a_{12}y_2 \\ a_{21}y_1 & a_{22}y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = A \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = A\mathbf{y}$$

In consequence matrix A takes the form (values a_{11}, a_{12}, a_{21}, a_{22} are defined in previous slide as a consequence od new base definition):

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

Vector space

Transform from child space to parent space - final calculation

When we want to go from child space with basis vectors $\mathbf{u}$, $\mathbf{v}$ on a vector to its coordinates in the parent space with basis vectors $\mathbf{e_1}$, $\mathbf{e_2}$ the transformation matrix is one whose **columns** are $\mathbf{u}$ and $\mathbf{v}$

For example, if

$$\mathbf{u} = \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix}$$

and

$$\mathbf{v} = \begin{bmatrix} -\frac{4}{5} \\ \frac{3}{5} \end{bmatrix}$$

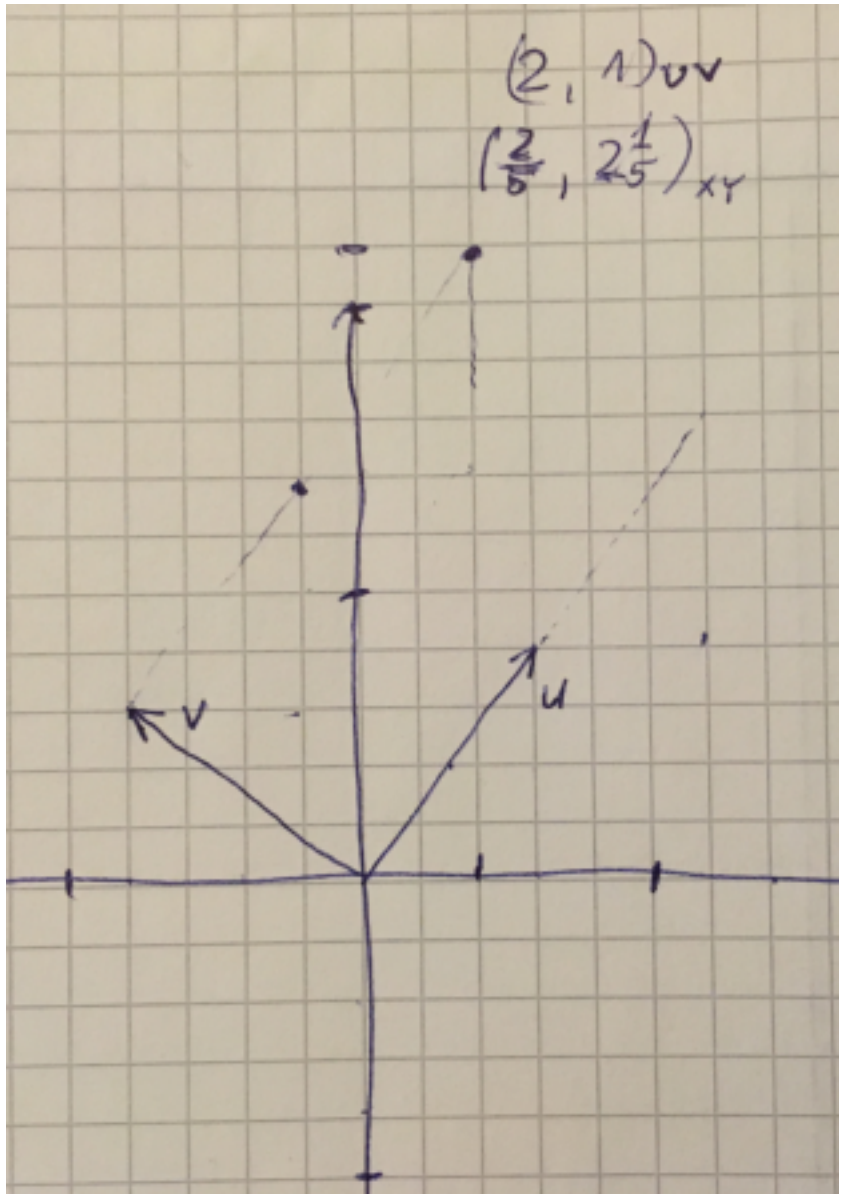
then the point p_C from the $\mathbf{uv}$ (child) space

$$p_C = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

expressed in $\mathbf{e}_1\mathbf{e}_2$ parent coordinates is

$$p_P = Ap_C = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{6}{5} - \frac{4}{5} \\ \frac{8}{5} + \frac{3}{5} \end{bmatrix} = \begin{bmatrix} \frac{2}{5} \\ \frac{11}{5} \end{bmatrix}.$$

$\mathbf{u}$ $\mathbf{v}$



Vector space

Transform from parent space to child space

Conversely, when we want to go from parent space on a vector to its coordinates in the child space with basis vectors $\mathbf{u}$, $\mathbf{v}$, the transformation matrix is one whose **rows** are $\mathbf{u}$ and $\mathbf{v}$

For example, if

$$\mathbf{u} = \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix}$$

and

$$\mathbf{v} = \begin{bmatrix} -\frac{4}{5} \\ \frac{3}{5} \end{bmatrix}$$

then the point p_P from the $\mathbf{e}_1\mathbf{e}_2$ (parent) space

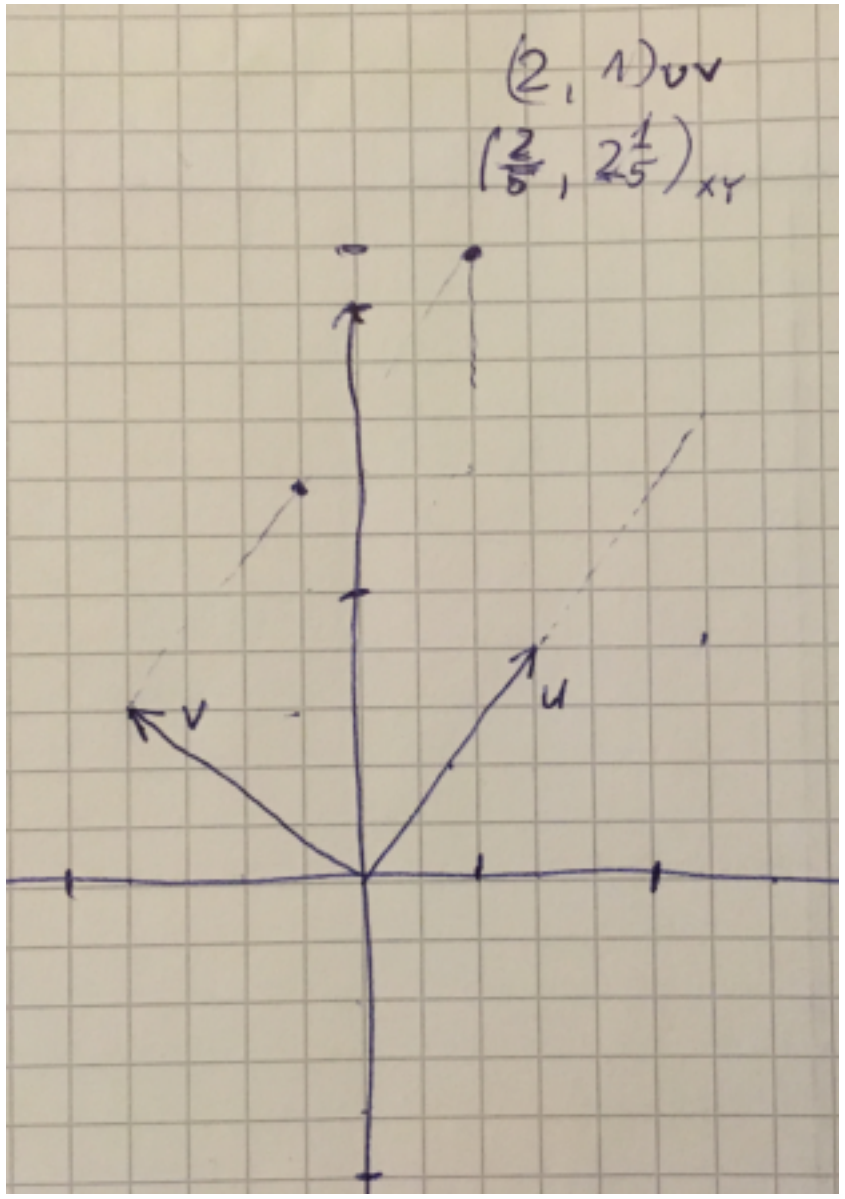
$$p_P = \begin{bmatrix} \frac{2}{5} \\ \frac{11}{5} \end{bmatrix}$$

expressed in $\mathbf{uv}$ child coordinates is

$\mathbf{u} \rightarrow$

$$p_C = Ap_P = \begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ -\frac{4}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} \frac{2}{5} \\ \frac{11}{5} \end{bmatrix} = \begin{bmatrix} \frac{6}{25} + \frac{44}{25} \\ -\frac{8}{25} + \frac{33}{25} \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

$\mathbf{v} \rightarrow$



Eigenvalues and eigenvectors

Definition

If f is a linear transformation from a vector space V over a field F into itself and $\mathbf{v}$ is a nonzero vector in V, then $\mathbf{v}$ is an eigenvector of f if

$$f(\mathbf{v}) = \lambda \mathbf{v}$$

where λ is a scalar in F, known as the ***eigenvalue***, ***characteristic value***, or ***characteristic root*** associated with $\mathbf{v}$.

If the dimension of V is finite, and a basis has been chosen, f and $\mathbf{v}$ may be represented, respectively, by a square matrix M and a column matrix (vertical vector) $\mathbf{u}$; the equation defining eigenvectors and eigenvalues becomes

$$M\mathbf{u} = \lambda \mathbf{u}.$$

Using the identity matrix I, whose entries are all zero, except those of the main diagonal, which are equal to one, this may be rewritten

$$(M - \lambda I)\mathbf{u} = 0.$$

As $\mathbf{u}$ is supposed to be nonzero, this means that $M - \lambda I$ is a singular matrix (is not invertible), and thus

$$\det(M - \lambda I) = 0.$$

The eigenvalues are thus the roots of the polynomial

$$g(x) = \det(xI - M).$$

An n-by-n square matrix A is called ***invertible*** (also ***nonsingular*** or ***nondegenerate***), if there exists an n-by-n square matrix B such that

$$AB = BA = I.$$

A **square matrix** that is ***not invertible*** is called ***singular*** or ***degenerate***. A square matrix is singular if and only if its determinant is zero.

Eigenvalues and eigenvectors

Interpretation

Loosely speaking, in a multidimensional vector space, the eigenvectors are vectors which are not changed (rotated) by the transformation - they stays the same (preserves the same direction) but may be scaled (stretched).

Geometrically, an eigenvector, corresponding to a real nonzero eigenvalue, points in the same direction in which it is stretched by the transformation as it points before applying the transformation. The eigenvalue is the factor by which it is stretched. If the eigenvalue is negative, the direction is reversed.

Eigenvalues and eigenvectors

Properties

- Eigenvectors corresponding to different eigenvalues are linearly independent.
- The matrix A is invertible if and only if every eigenvalue is nonzero.
- Determinant of matrix A is equal to product of eigenvalues of A as given below: $\det(A) = \lambda_1 \lambda_2 \ldots \lambda_n$.
- If A is an $n \times n$ triangular matrix (upper triangular, lower triangular, or diagonal), then the eigenvalues of A are entries of the main diagonal of A.
- If an $n \times n$ matrix A has n distinct eigenvalues, then A is diagonalizable.
- If a basis exists that consists only of eigenvectors, the matrix of transformation on this basis has a very simple structure: it is a diagonal matrix such that the entries on the main diagonal are eigenvalues, and the other entries are zero.

Eigenvalues and eigenvectors

Properties

If A is an $n \times n$ matrix, then the following are equivalent:

- A is invertible.
- $\lambda = 0$ is not an eigenvalue of A.
- $\det(A) \neq 0$.
- A is diagonalizable.
- A has n linearly independent eigenvectors.
- A is expressible as a product of elementary matrices.
- The column vectors of A are linearly independent.
- The row vectors of A are linearly independent.
- The column vectors of A span R^n
- The row vectors of A span R^n.
- The column vectors of A form a basis for R^n.
- The row vectors of A form a basis for R^n.

Eigenvalues and eigenvectors

Properties - diagonalizable matrices

In linear algebra, a square matrix A is called *diagonalizable* if it is similar to a diagonal matrix, i.e., if there exists an invertible matrix P and a diagonal matrix D such that $P^{-1}AP = D$, or equivalently $A = PDP^{-1}$ (such P, D are not unique).

An $n \times n$ matrix A over a field F is diagonalizable if and only if there exists a basis of F consisting of eigenvectors of A. If such a basis has been found, one can form the matrix P having these basis vectors as columns, and $P^{-1}AP$ will be a diagonal matrix whose diagonal entries are the eigenvalues of A.

Eigenvalues and eigenvectors

Diagonalizable matrices

An $n \times n$ matrix A over a field F is diagonalizable if and only if there exists a basis of F consisting of eigenvectors of A.

TODO: *How to diagonalize a matrix* in *Diagonalizable matrix*:

`https://en.wikipedia.org/wiki/Diagonalizable_matrix`

Eigenvalues and eigenvectors

Diagonalizable matrices

Diagonalize procedure is the way we can decompose some matrices.

If there exists a basis of F consisting of eigenvectors of A then an $n \times n$ matrix A over a field F is diagonalizable and in consequence we can decompose A according to the following formula

$$A = PDP^{-1}$$

Eigendecomposition of a matrix

Theory

Let A be a square $n \times n$ matrix with n linearly independent eigenvectors q_i, where $i = 1, \ldots, n$.

Then A can be factorized as

$$A = Q \Lambda Q^{-1}$$

where Q is the square $n \times n$ matrix whose i-th column is the eigenvector q_i of A, and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues: $\Lambda_{ii} = \lambda_i$.

This decomposition can be derived from the fundamental property of eigenvectors:

$$\begin{aligned} Av &= \lambda v \\ AQ &= Q\Lambda \\ A &= Q\Lambda Q^{-1} \end{aligned}$$

As it was stated before, only diagonalizable matrices can be factorized in this way.

The n eigenvectors q_i are usually normalized, but they need not be. A non-normalized set of n eigenvectors can also be used as the columns of Q. That can be understood by noting that the magnitude of the eigenvectors in Q gets canceled in the decomposition by the presence of matrix Q^{-1}.

Eigendecomposition of a matrix

Theory - matrix inverse via eigendecomposition

One of the most important question is: *Why we do this?* You can guess that not only for pleasure but for some profits. **A decomposition operation does not result in a compression of the matrix**; instead, it breaks it down into constituent parts to give valuable insights into the properties of the matrix and make certain operations on the matrix easier to perform or/and less error prone.

- If a matrix A can be eigendecomposed and if none of its eigenvalues are zero, then A is nonsingular and its inverse is given by

  $$A^{-1} = Q\Lambda^{-1}Q^{-1}$$

  If A is a symmetric matrix, since Q is formed from the eigenvectors of A it is guaranteed to be an orthogonal matrix, therefore $Q^{-1} = Q^{\mathrm{T}}$. Furthermore, because Λ is a diagonal matrix, its inverse is easy to calculate:

  $$\left[\Lambda^{-1}\right]_{ii} = \frac{1}{\lambda_i}.$$

  As we can see, if a matrix A can be eigendecomposed, calculating its inverse is very fast thanks to eigendecomposition.

- Also computing the power of the matrix, become much easier when we use the eigendecomposition of the matrix.

A real square matrix is an **orthogonal** matrix if $Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I$ or equivalently $Q^{-1} = Q^{\mathrm{T}}$ because I can be expressed as $QB = BQ = I$ where B is called the inverse of Q and is denoted as Q^{-1}.

Eigendecomposition of a matrix

Practice

- Calculation of eigendecomposition
  `lecture_04_01_01.py`
- Confirm an eigenvector and eigenvalue
  `lecture_04_01_02.py`
- Reconstruct original matrix
  `lecture_04_01_03.py`

SVD - singular value decomposition

The **singular value decomposition** (**SVD**) is a factorization of a real or complex matrix A that generalizes the eigendecomposition of a square matrix to any $m \times n$ matrix.

Specifically, it takes the form

$A = U\Sigma V^*$,

where

- U is an $m \times m$ real or complex unitary matrix,
- Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal,
- and V is an $n \times n$ real or complex unitary matrix.

If matrix A is real then U and $V^{\mathrm{T}} = V^*$ are real orthogonal matrices.

The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are known as the **singular values** of A. The number of non-zero singular values is equal to the rank of A. The columns of U and the columns of V are called the **left-singular** vectors and **right-singular** vectors of A, respectively.

The SVD is not unique. It is always possible to choose the decomposition so that the singular values Σ_{ii} are in descending order. In this case Σ (but not always U and V) is uniquely determined by A.

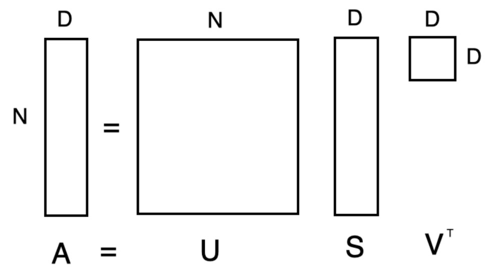
Every rectangular matrix has a singular value decomposition, although the resulting matrices may contain complex numbers and the limitations of floating point arithmetic may cause some matrices to fail to decompose neatly.

- A complex square matrix U is ***unitary*** if its conjugate transpose U^* is also its inverse, that is, if $U^*U = UU^* = I$ where I is the identity matrix.
- The number of linearly independent rows or columns is called the ***rank*** of matrix.

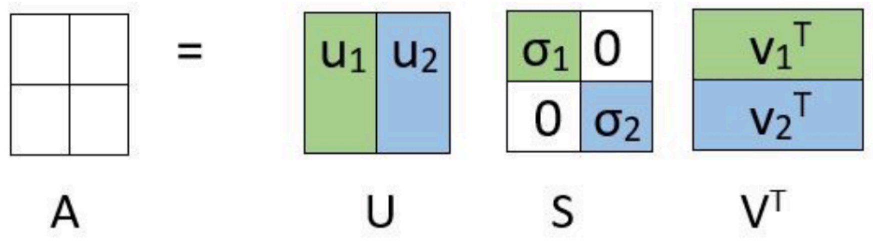
Dimensionality reduction with SVD

How to reduce dimension with SVD

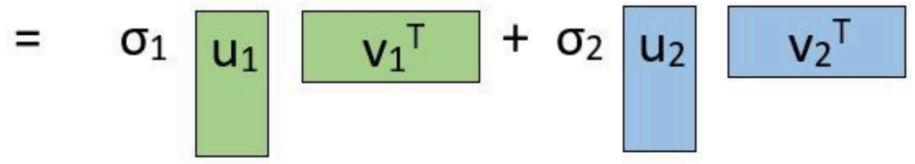
As we know SVD decomposition takes a form $M = U\Sigma V^*$:



Analyzing carefully this representation



we discover alternative representation



So, if $A_{m\times n}$ is decomposed as $U_{m\times m}\Sigma_{m\times n}\left[V^{\mathsf{T}}\right]_{n\times n}$ we can write it equivalently as

$A_{m\times n} = \sigma_1 \mathrm{u}_1[\mathrm{v}^{\mathsf{T}}]_1 + \ldots + \sigma_q \mathrm{u}_q[\mathrm{v}^{\mathsf{T}}]_q$

where $q = \min(m, n)$.

Dimensionality reduction with SVD

How to reduce dimension with SVD

So, if $A_{m\times n}$ is decomposed as $U_{m\times m}\Sigma_{m\times n}\left[V^{\mathsf{T}}\right]_{n\times n}$ we can write it equivalently as

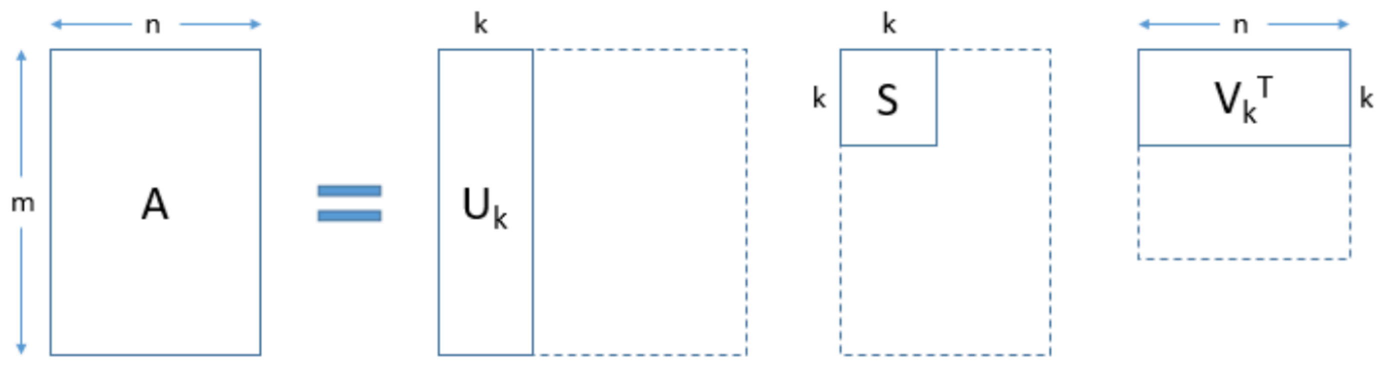
$$A_{m\times n} = \sigma_1 \mathrm{u}_1[\mathrm{v}^{\mathsf{T}}]_1 + \ldots + \sigma_q \mathrm{u}_q[\mathrm{v}^{\mathsf{T}}]_q$$

where $q = \min(m, n)$.

Data with a large number of features, such as more features (columns) than observations (rows) **may be reduced to a smaller subset of features that are most relevant to the prediction problem**.

In a practical application, you will observe that only the first few, say k, singular values are large. The rest of the singular values approach zero. As a result, terms except the first few can be ignored without losing much of the information

$$A_{m\times n} \simeq \sigma_1 \mathrm{u}_1[\mathrm{v}^{\mathsf{T}}]_1 + \ldots + \sigma_k \mathrm{u}_k[\mathrm{v}^{\mathsf{T}}]_k$$



This way we obtain a matrix $B_{m\times n} = \sigma_1 \mathrm{u}_1[\mathrm{v}^{\mathsf{T}}]_1 + \ldots + \sigma_k \mathrm{u}_k[\mathrm{v}^{\mathsf{T}}]_k$ which approximate given matrix A.

How close this approximation is?
Eckat-Young-Mirsky theorem for matrix methods describes about the Low-Rank-Approximation. It states that A_k, which is addition of singular matrices up to k^{th} largest singular value, is the closest matrix of rank k for the matrix A.

In natural language processing, this approach can be used on matrices of word occurrences or word frequencies in documents and is called *Latent Semantic Analysis* or *Latent Semantic Indexing*.

SVD - singular value decomposition

Practice

For the case of simplicity we will focus on the SVD for real-valued matrices and ignore the case for complex numbers. In this case we will have simply

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

- Calculate singular-value decomposition
  `lecture_04_02_01.py`

- Reconstruct original matrix from SVD
  `lecture_04_02_02_01.py` - square case
  `lecture_04_02_02_02.py` - general (rectangular) case

- Dimensionality reduction with SVD
  `lecture_04_02_03_01.py` - calculations with the `svd` method
  `lecture_04_02_03_02.py` - calculations with the `TruncatedSVD` class
  `lecture_04_02_04.py` - SVD for image compression

Variance and covariance

The formula for ***variance*** is given by

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where n is the number of samples and $\bar{x}$ is the mean of the random variable x.

The ***covariance*** $\sigma(x, y)$ of two random variables x and y is given by

$$\sigma(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

with n samples. The variance σ_x^2 of a random variable x can be also expressed as the covariance with itself by $\sigma(x, x)$.

This can be written as

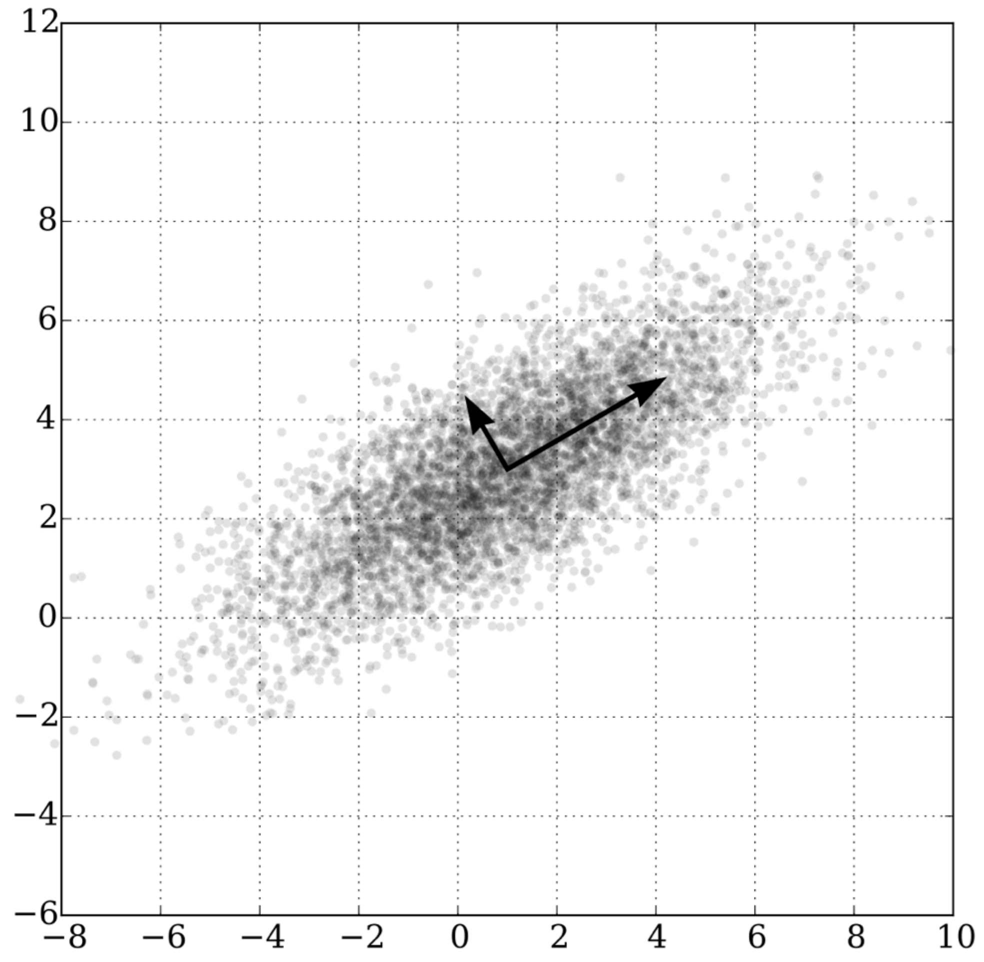
$$\mathrm{cov}(\mathbf{X}, \mathbf{Y}) = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{Y} - \mathrm{E}[\mathbf{Y}])^{\mathrm{T}}\right]$$

where the operator E denotes the expected value (mean) of its argument.

PCA - principal component analysis

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data.

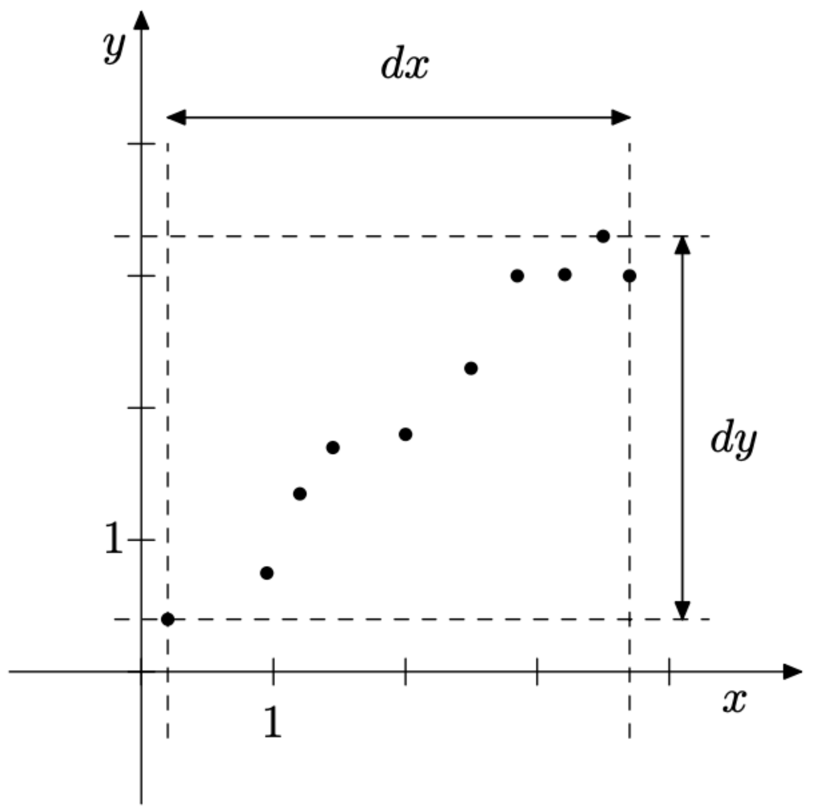
A data set consisting of N observations, each of which includes K variables, can be interpreted as a cloud of N points in a K-dimensional space. **The goal of PCA is to rotate the coordinate system** in such a way as to maximize first the variance of the first coordinate, then the variance of the second coordinate, etc. The coordinate values transformed in this way are called the principal components. In this way, a new observation space is constructed in which the most variability is explained by the initial factors.

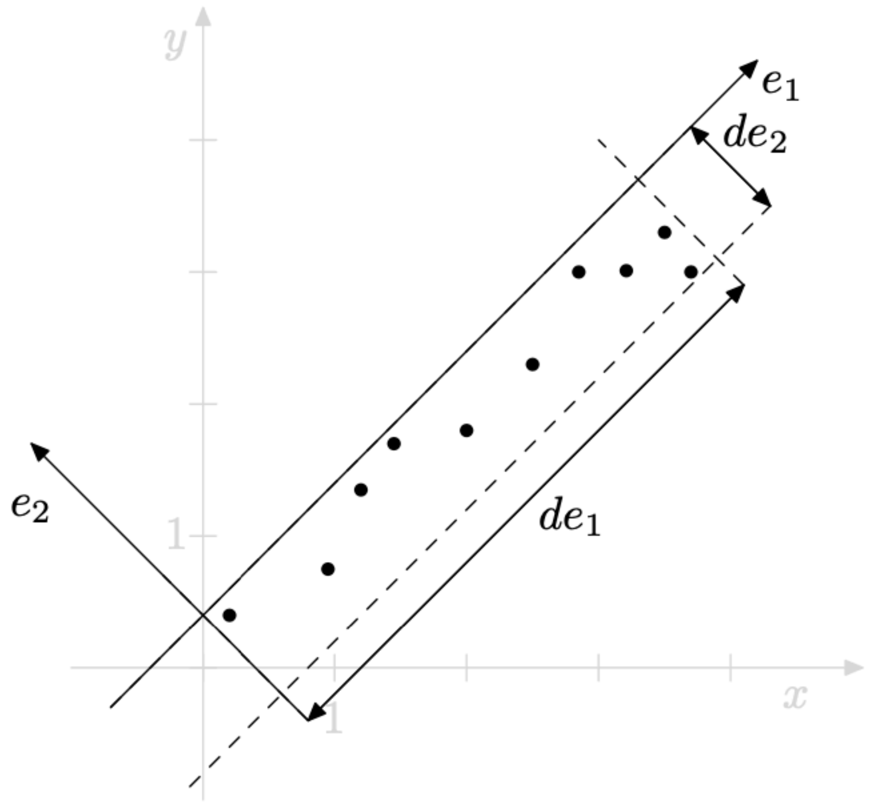


PCA - principal component analysis

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data.

A data set consisting of N observations, each of which includes K variables, can be interpreted as a cloud of N points in a K-dimensional space. **The goal of PCA is to rotate the coordinate system** in such a way as to maximize first the variance of the first coordinate, then the variance of the second coordinate, etc. The coordinate values transformed in this way are called the principal components. In this way, a new observation space is constructed in which the most variability is explained by the initial factors.





PCA - principal component analysis

PCA is often used to reduce the size of a statistical data set by discarding recent factors. You can also look for a substantive interpretation of the factors, depending on the type of data, which allows you to better understand the nature of the data, although it can be difficult with a greater number of variables. In signal processing, PCA is used e.g. for signal compression. Smetimes using only the first few principal components and ignoring the rest. PCA can be thought of as fitting a p-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small.

To find the axes of the ellipsoid, we must:

1. First subtract the mean of each variable from the dataset to center the data around the origin.

2. Then, we compute the covariance matrix of the data

3. and next calculate the eigenvalues and corresponding eigenvectors of this covariance matrix.

4. Next we normalize each of the orthogonal eigenvectors to turn them into unit vectors.

Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis (principal component) of the ellipsoid fitted to the data.

This *choice of basis* will **transform** our covariance matrix into a diagonalised form with the diagonal elements representing the variance of each axis.

PCA - principal component analysis

Example - method 1 (base)

TODO: Example from my book [FulGrzAI] - liczone z definicji

Rozdział 18.3.1 Analiza głównych składowych, z:
Piotr Fulmański, Marta Grzanek, *Sztuczna inteligencja. Podręcznik do wykładów i ćwiczeń*
`ai.pdf`

PCA - principal component analysis

So to make PCA we need eigenvectors of the covariance matrix C of the data matrix A.

If A is our data matrix with n with rows being the observations and with **mean-centered** columns, then

$$C = \frac{1}{n-1} A^{\mathrm{T}} A$$

is the covariance matrix. It is a symetric matrix, and therefore to find eigenvalues of C we can use **eigendecomposition**.

If we can use eigendecomposition, we can also use more general method, namely SVD.

PCA - principal component analysis

PCA and SVD

As we stated before: "*the goal of PCA is to rotate the coordinate system*". From the beginning of this lecture we know that any rotation can be expressed in matrix form. On the other hand, from material proceedings this part we know that any matrix A can be decomposed with SVD method:

$$A = U\Sigma V^{\mathrm{T}}.$$

Taking this into account, the matrix $A^{\mathrm{T}}A$ can be written

$$\begin{aligned} A^{\mathrm{T}}A &= V\Sigma^{\mathrm{T}}U^{\mathrm{T}}U\Sigma V^{\mathrm{T}} \\ &= V\Sigma^{\mathrm{T}}\Sigma V^{\mathrm{T}} \\ &= V\hat{\Sigma}^2 V^{\mathrm{T}} \end{aligned}$$

where $\hat{\Sigma}$ is the square diagonal matrix with the singular values of A and the excess zeros chopped off that satisfies $\hat{\Sigma}^2 = \Sigma^T\Sigma$.

PCA - principal component analysis

PCA and SVD

Making eigendecomposition of covariance matrix C we have (from the definition of eigendecomposition)

$$C = Q\Lambda Q^{-1}$$

On the other hand we know that

$$C = \frac{1}{n-1}A^{\mathrm{T}}A$$

and in consequence

$$Q\Lambda Q^{-1} = C = \frac{1}{n-1}A^{\mathrm{T}}A = V\frac{\hat{\Sigma}^2}{n-1}V^{\mathrm{T}}$$

Having in mind that $A = U\Sigma V^{\mathrm{T}}$, comparison with the eigenvector factorization of $A^{\mathrm{T}}A$ establishes that

- the right singular vectors V of A are equivalent to the eigenvectors of $A^{\mathrm{T}}A$,
- while the singular values σ_k of A are equal to the square-root of the eigenvalues λ_k of $A^{\mathrm{T}}A$:
  $$\Lambda = \frac{\hat{\Sigma}^2}{n-1}.$$

PCA - principal component analysis

PCA and SVD

Making eigendecomposition of covariance matrix C we have (from the definition of eigendecomposition)

$$C = Q\Lambda Q^{-1}$$

On the other hand we know that

$$C = \frac{1}{n-1}A^{\mathrm{T}}A$$

and in consequence

$$Q\Lambda Q^{-1} = C = \frac{1}{n-1}A^{\mathrm{T}}A = V\frac{\hat{\Sigma}^2}{n-1}V^{\mathrm{T}}$$

Having in mind that $A = U\Sigma V^{\mathrm{T}}$, comparison with the eigenvector factorization of $A^{\mathrm{T}}A$ establishes that

- the right singular vectors V of A are equivalent to the eigenvectors of $A^{\mathrm{T}}A$,
- while the singular values σ_k of A are equal to the square-root of the eigenvalues λ_k of $A^{\mathrm{T}}A$:
  $$\Lambda = \frac{\hat{\Sigma}^2}{n-1}.$$

PCA - principal component analysis

PCA and SVD

Making eigendecomposition of covariance matrix C we have (from the definition of eigendecomposition)

$$C = Q\Lambda Q^{-1}$$

On the other hand we know that

$$C = \frac{1}{n-1}A^{\mathrm{T}}A$$

and in consequence

$$Q\Lambda Q^{-1} = C = \frac{1}{n-1}A^{\mathrm{T}}A = V\frac{\hat{\Sigma}^2}{n-1}V^{\mathrm{T}}$$

Having in mind that $A = U\Sigma V^{\mathrm{T}}$, comparison with the eigenvector factorization of $A^{\mathrm{T}}A$ establishes that

- the right singular vectors V of A are equivalent to the eigenvectors of $A^{\mathrm{T}}A$,
- while the singular values σ_k of A are equal to the square-root of the eigenvalues λ_k of $A^{\mathrm{T}}A$:

$$\Lambda = \frac{\hat{\Sigma}^2}{n-1}.$$

PCA - principal component analysis

Thus, the principal components are often computed

- either by eigendecomposition of the data covariance matrix
- or singular value decomposition of the data matrix (because the right singular vectors V of A are equivalent to the eigenvectors of $A^{\mathrm{T}}A$ which in turn is equal to covariance matrix multiplied by $n-1$).

Have in mind that **PCA can be interpreted as the SVD of a data matrix when the columns have first been centered by their means.**

We can say, that today computing the SVD is the standard way to calculate a principal components analysis from a data matrix.

PCA - principal component analysis

Example - method 2 (eigendecomposition of $A^{\mathrm{T}}A$)

TODO: Example from my book [FulGrzAI] ale jako XTX, na bazie przykładu 11.2

z

http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

PCA - principal component analysis

Example - method 3 (with SVD)

TODO: Example from my book [FulGrzAI] ale jako XTX, na bazie przykładu 11.2

z

http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

Teraz liczone z SVD

PCA - principal component analysis

Practice

- Calculate PCA decomposition
  `lecture_04_03_01_01.py` - calculations with method 1: base method
  `lecture_04_03_01_02.py` - calculations with method 2: eigendecomposition of $A^{\mathrm{T}}A$
  `lecture_04_03_01_03.py` - calculations with method 3: with SVD
  `lecture_04_03_01_04.py` - calculations with the `PCA` class

Relationship between PCA and SVD

PCA refers to data analysis technique, while the SVD is a general operation defined on all matrices.

For example, it doesn't really make sense to talk about "applying PCA" to a matrix A unless the rows of A have clear semantics - typically, as data points $x_1, \ldots, x_n$ in R^n. By contrast, the SVD is well defined for every matrix A, whatever the semantics for A. **In the particular case where A is a matrix where the rows represent data points, the SVD can be interpreted as performing the calculations required by PCA.**

Bibliography

- https://pl.wikipedia.org/wiki/Wektor Wektor
- https://en.wikipedia.org/wiki/Vector_space Vector space
- https://pl.wikipedia.org/wiki/Przestrzeń_liniowa Przestrzeń liniowa
- https://pl.wikipedia.org/wiki/Przekształcenie_liniowe Przekształcenie liniowe
- https://en.wikipedia.org/wiki/Diagonalizable_matrix#How_to_diagonalize_a_matrix Diagonalizable matrix: How to diagonalize a matrix
- https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix Eigendecomposition of a matrix
- https://machinelearningmastery.com/introduction-to-eigendecomposition-eigenvalues-and-eigenvectors/ Gentle Introduction to Eigenvalues and Eigenvectors for Machine Learning
- https://en.wikipedia.org/wiki/Singular_value_decomposition Singular value decomposition
- https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/ How to Calculate the SVD from Scratch with Python
- https://towardsdatascience.com/singular-value-decomposition-example-in-python-dab2507d85a0 Singular Value Decomposition Example In Python
- https://medium.com/analytics-vidhya/master-dimensionality-reduction-with-these-5-must-know-applications-of-singular-value-777299940b89 Master Dimensionality Reduction with these 5 Must-Know Applications of Singular Value Decomposition (SVD) in Data Science
- https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf Singular Value Decomposition (SVD)
- http://infolab.stanford.edu/~ullman/mmds/ch11.pdf Chapter 11, Dimensionality Reduction, 11.2 Principal-Component Analysis
- http://theory.stanford.edu/~tim/s15/l/l9.pdf CS168: The Modern Algorithmic Toolbox Lecture #9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations, Tim Roughgarden & Gregory Valiant April 27, 2015
- https://en.wikipedia.org/wiki/Principal_component_analysis Principal component analysis
- https://pl.wikipedia.org/wiki/Analiza_głównych_składowych Analiza głównych składowych
- https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd Singular Value Decomposition and its applications in Principal Component Analysis
- https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/ How to Calculate Principal Component Analysis (PCA) from Scratch in Python

Bibliography

- https://mlfromscratch.com/principal-component-analysis-pca-svd/ Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- https://arxiv.org/pdf/1404.1100.pdf A Tutorial on Principal Component Analysis, Jonathon Shlens
- https://en.wikipedia.org/wiki/Principal_component_analysis Principal component analysis
- https://pl.wikipedia.org/wiki/Analiza_głównych_składowych Analiza głównych składowych
- https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd Singular Value Decomposition and its applications in Principal Component Analysis
- https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/ How to Calculate Principal Component Analysis (PCA) from Scratch in Python
- https://mlfromscratch.com/principal-component-analysis-pca-svd/ Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- https://arxiv.org/pdf/1404.1100.pdf A Tutorial on Principal Component Analysis, Jonathon Shlens
- [FulGrzAI] Piotr Fulmański, Marta Grzanek, *Sztuczna inteligencja. Podręcznik do wykładów i ćwiczeń* `ai.pdf`