Matrix Decomposition Algorithms For Data Mining and Machine Learning

2019. 1. 4

Troy University

Ingyu Lee

(Contents are based on Petros Drineas, Saara Hyvonen and Rainer Gemulla's **Materials**)

Content

Part 0: Modern data

Facts:

- Computers make it easy to collect and store data.
- Costs of storage are very low and are dropping very fast.
  (most laptops have a storage capacity of more than 1TB..)

When it comes to storing data:

- The current policy typically is "store everything in case it is needed later" instead of deciding what could be deleted.

Data Mining:

- Extract useful information from the massive amount of available data.
- "Data mining is the process of discovering knowledge or patterns from massive amounts of data."

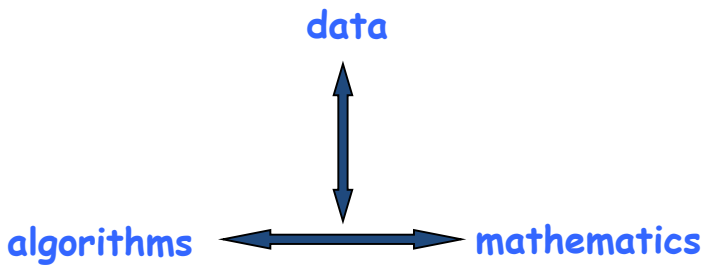
Part 0: About the tutorial

Tools:

- Introduce matrix algorithms and matrix decompositions for data mining applications.

Goals:

- Learn a model for the underlying "physical" system generating the dataset.



Math is necessary to design and analyze principled algorithmic techniques to data-mine the massive datasets that have become ubiquitous in scientific research.

Part 0: References

- **Information Retrieval and Data Mining: A Linear Algebraic Perspective**,
  http://www.cs.rpi.edu/~drinep/talks.html
  (or https://www.ipam.ucla.edu/schedule.aspx?pc=setut Podcast is available)
  Petros Drineas, Rensselaer Polytechnic Institute

- **Data Mining and Matrices,**
  Rainer Gemulla, and Pauli Miettinen, Max Planck Institute

- **Linear Algebra Methods for Data Mining**,
  http://www.uta.edu/faculty/rcli/Teaching/math5392/s2010/
  Saara Hyvonen, University of Helsinki

- **Data Mining with Graphs and Matrices**,
  http://feiwang03.googlepages.com/sdm-tutorial
  Fei Wang, Tao Li and Chris Ding, University of Texas at Arlington

Part 0: Suggested Readings

- Lars Elden
  Matrix Methods in Data Mining and Pattern Recognition, SIAM, 2007.

- David Skillicorn
  Understanding Complex Datasets: Data Mining with Matrix Decompositions, Chapman and Hall, 2007.

- Gene H. Golub and Charles F. Van Loan
  Matrix Computations, 3rd ed. Johns Hopkins University Press, 1996

- Carl Meyer
  Matrix Analysis and Applied Linear Algebra, SIAM, 2000.
  http://www.matrixanalysis.com

Part 0: Why linear (or multilinear) algebra?

Data are represented by matrices:

- Numerous modern datasets are in matrix form.

Data are represented by tensors:

- Data in the form of tensors (multi-mode arrays) are becoming very common in the data mining and information retrieval literature in the last few years.

- Linear algebra (and numerical analysis) provide the fundamental mathematical and algorithmic tools to deal with matrix and tensor computations.

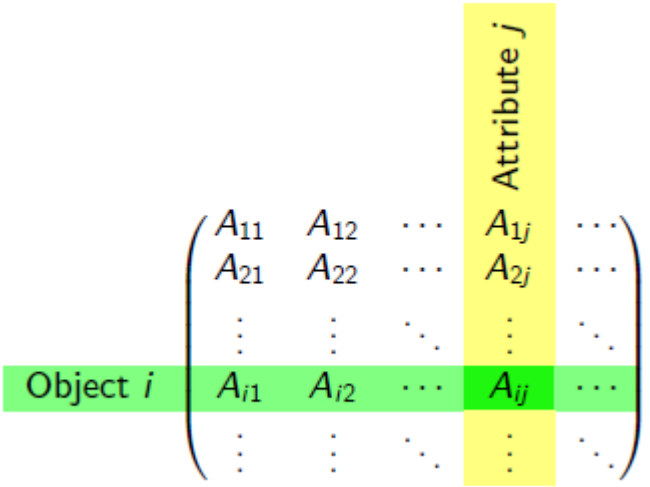
Part 0: What is a matrix?

- A means to describe computation
  - Rotation
  - Rescaling
  - Permutation
  - Projection
  - …

  } Linear operators

- A means to describe data

Rows	Columns	Entries
Objects	*Attributes*	*Values*
Equations	Variables	Coefficients
Data points	Axes	Coordinates
Vertices	Vertices	Edges
⋮	⋮	⋮



*** In data mining, we make use of both viewpoints simultaneously**

Part 0: Datasets in the form of matrices

We are given ***m*** objects and ***n*** features describing the objects.
(Each object has ***n*** numeric values describing it.)

Dataset:

- An *m-by-n* matrix **A**, $\mathbf{A}_{ij}$ shows the "importance" of feature *j* for object *i*.
- Every row of **A** represents an object.

Goal:

- We seek to **understand the structure of the data**, e.g. the underlying process generating the data.

	Bread	Butter	Beer
Anna	1	1	0
Bob	1	1	1
Charlie	0	1	1

Customer transactions

	Data	Matrix	Mining
Book 1	5	0	3
Book 2	0	0	7
Book 3	4	6	5

Document-term matrix

	Avatar	The Matrix	Up
Alice		4	2
Bob	3	2	
Charlie	5		3

Incomplete rating matrix

	Jan	Jun	Sep
Saarbrücken	1	11	10
Helsinki	6.5	10.9	8.7
Cape Town	15.7	7.8	8.7

Cities and monthly temperatures

Part 0: Market basket matrices

- Common representation for association rule mining.

n products
(e.g., milk, bread, wine, etc.)

m customers $\begin{pmatrix} & A & \end{pmatrix}$

A_{ij} = quantity of j-th product purchased by the i-th customer

Data mining tasks

- Find association rules

E.g., customers who buy product x buy product y with probability 89%.

- Such rules are used to make item display decisions, advertising decisions, etc.

Market Basket Example

Image source: deepclimate.org

Part 0: Document-term matrices

- A collection of documents is represented by an m-by-n matrix (bag-of-words model).

m documents

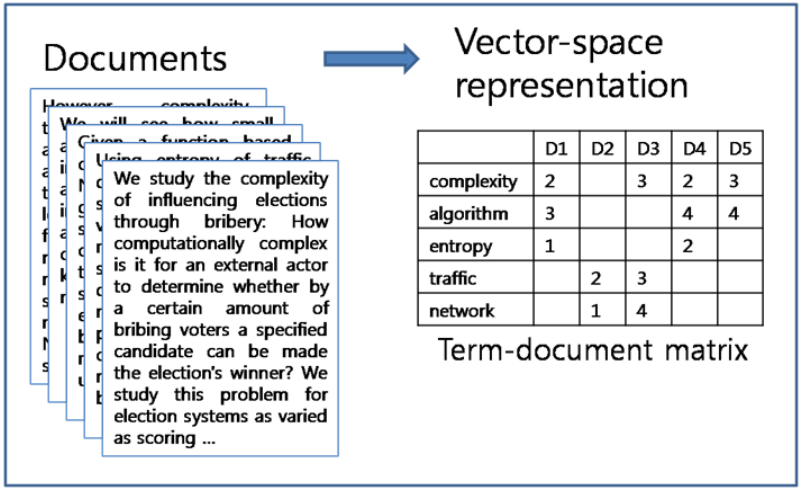
n terms (words)

A

A_{ij} = frequency of j-th term in i-th document

Data mining tasks

- Cluster or classify documents
- Find "nearest neighbors"
- Feature selection: find a subset of terms that (accurately) clusters or classifies documents.



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Part 0: Social networks (e-mail graph)

- Represent the email communications between groups of users.

n users

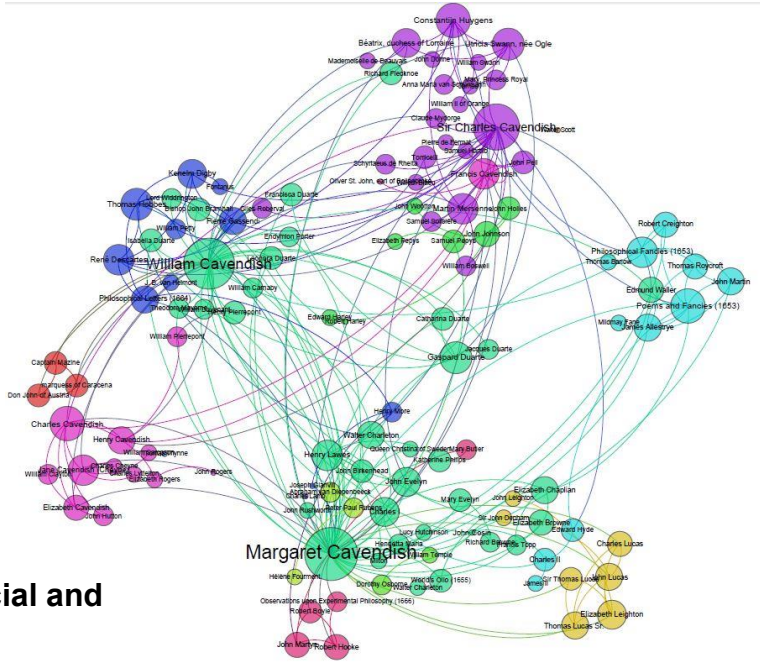
n users

$$A$$

A_{ij} = number of emails exchanged between users i and j during a certain time period

Data mining tasks

- cluster the users
- identify "dense" networks of users (dense subgraphs)



Margaret Cavendish's social and some textual networks.

Part 0: Recommendation systems

- The m-by-n matrix A represents *m* customers and *n* products.

customers

products

A

A_{ij} = utility of j-th product to i-th customer

Data mining task

- Given a few samples from A, recommend high utility products to customers.

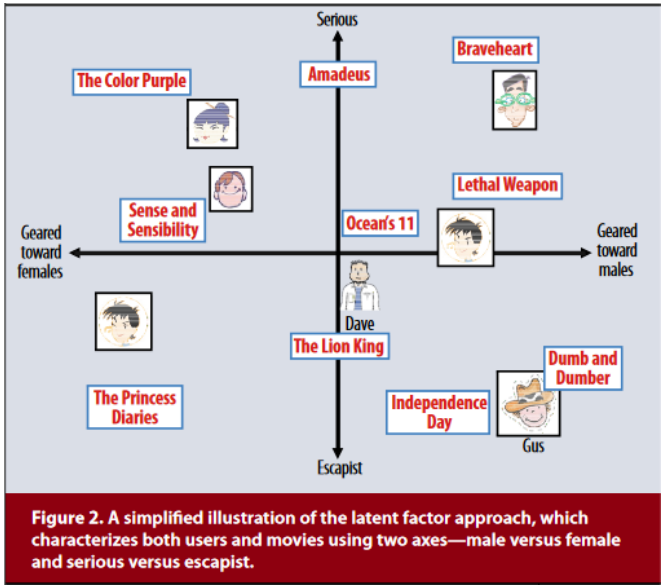


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

Yehuda Koren, Yahoo Research
Robert Bell and Chris Volinsky, AT&T Labs—Research

Part 0: Why matrix decompositions?

Matrix decompositions:
(e.g. SVD, QR, SDD, CX and CUR, NMF, ICA, etc.)

- They use the relationships between the available data in order to identify components of the underlying physical system generating the data.

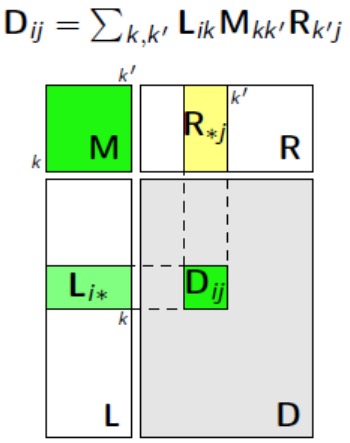
- Some assumptions on the relationships between the underlying components are necessary.
- Very active area of research: some matrix decompositions are more than one century old, whereas others are very recent.

- A matrix decomposition of a data matrix **D** is given by three matrices **L**, **M**, **R** such that

  D = LMR,

  where
  - **D** is an *m x n* data matrix,
  - **L** is an *m x r* matrix,
  - **M** is an *r x r* matrix,
  - **R** is an *r x n* matrix, and
  - **r** is an integer >= 1

There are many different kinds of matrix decompositions, each putting certain constraints on matrices **L**, **M**, **R** (which may not be easy to find).



Part 0: What can we do with matrix decompositions?

- Separate data from multiple processes
- Remove noise from the data
- Remove redundancy from the data
- Reveal latent structure and similarities in the data
- Fill in missing entries
- Find local patterns
- Reduce space consumption
- Reduce computational cost
- Aid visualization

Matrix decompositions can make data mining algorithms more effective. They may also provide insight into the data by themselves.

Part 0: Factor interpretation of matrix decompositions?

- Assume that M is diagonal. Consider object *i*

Row of **R** = part (or piece), called *latent factor* ("latent object")

Entry of **M** = weight of corresponding part

Row of **MR** = weighted part

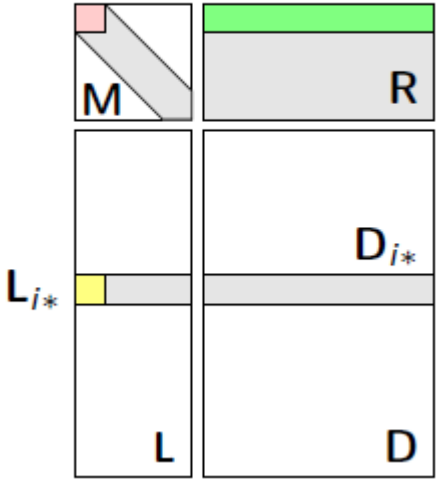
Row of **L** = "view" of corresponding row of **D** in terms of the weighted parts (r pieces of information)

r forces "compactness" (often $r < n$)

Each object can be viewed as a combination of r (weighted) "latent objects" (or "prototypical objects"). Similarly, each attribute can be viewed as a combination of r (weighted) "latent attributes."

(e.g., latent attribute = "body size"; latent object relates body size to real attributes such as "height", "weight", "shoe size")

$$\mathbf{D}_{i*} = \sum_k \mathbf{L}_{ik}\mathbf{M}_{kk}\mathbf{R}_{k*}$$



Part 0: Lessons learned

- Data mining = from data to knowledge
  → Prediction, clustering, outlier detection, local patterns

- Many different data types can be represented with a matrix
  → Linear equations, data points, maps, graphs, relational data, …

- Common interpretation: rows = objects, columns = attributes

- Matrix decompositions reveal structure in the data
  → **D = LMR**

- Many different decompositions with different applications exist
  → QR, SVD, NMF, SDD, ICA, etc.

- Factor interpretation: objects described by “latent attributes”

Content

Part 1: Vector

- A vector is
  - a 1D array of numbers
  - a geometric entity with magnitude and direction
  - a matrix with exactly one row or column → row and column vectors

- A transpose $\mathbf{x}^T$ transposes a row vector into a column vector and vice versa

- The norm of vector defines its magnitude
  - 1-norm: $||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$
  - Euclidean or L_2 norm: $||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} {x_i}^2}$
  - General L_p: $||\mathbf{x}||_p = \left(\sum_{i=1}^{n} {x_i}^p\right)^{1/p}$

- Generally, a vector norm is a mapping $R^n \rightarrow R$ with the properties
  - $\left|\left|\mathbf{x}\right|\right| \geq 0 \; for \; all \; x,$
  - $\left|\left|\mathbf{x}\right|\right| = 0 \; if \; and \; only \; if \; x = 0,$
  - $\left|\left|\alpha\mathbf{x}\right|\right| = |\alpha|\left|\left|\mathbf{x}\right|\right|, for \; all \; \alpha \in R,$
  - $\left|\left|\mathbf{x} + \mathbf{y}\right|\right| \leq \left|\left|\mathbf{x}\right|\right| + \left|\left|\mathbf{y}\right|\right|, the \; triangular \; equality.$

Part 1: How to measure distance between vectors?

- Obvious answer: the distance between two vectors **x** and **y** is $||\mathrm{x} - \mathrm{y}||$, where $||\cdot||$ is some vector norm.

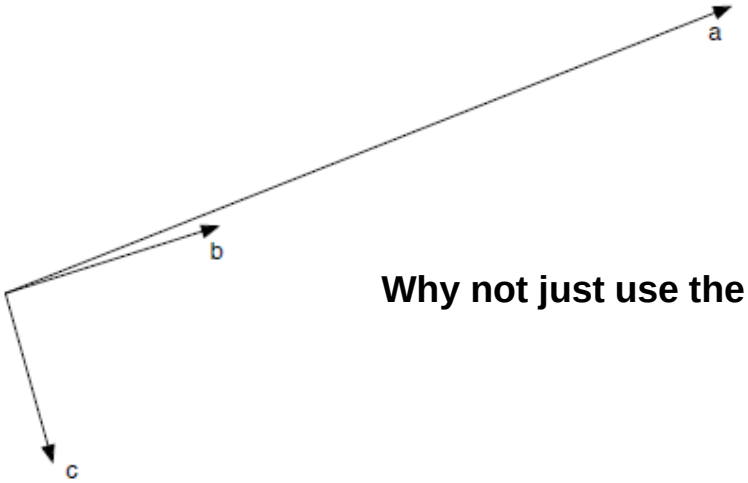
- Frequently one measures the distance by Euclidean norm $||\mathrm{x} - \mathrm{y}||_2$.
- So usually, if the index is dropped, this is what is meant.

- Alternative: use the angle between two vectors x and y to measure the distance between them.

- How to calculate the angle between two vectors?

Part 1: Angle between vectors

- The inner product between two vectors is defined by $(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$.
- This is associated with the Euclidean norm: $||\mathbf{x}||_2 = (\mathbf{x}^T\mathbf{x})^{1/2}$.
- The angle θ between two vectors **x** and **y** is $cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2}$.
- The cosine of the angle between two vectors **x** and **y** can be used to measure the similarity between the two vectors:
  - If **x** and **y** are close, the angle between them is small, and $cos\theta \approx 1$.
  - **x** and **y** are orthogonal, if $\theta = \frac{\pi}{2}, i.e. \mathbf{x}^T\mathbf{y} = 0$



Why not just use the Euclidean distance?

Part 1: Example: term-document matrix

- Each entry tells how many times a term appears in the document:

	Doc1	Doc2	Doc3
Term1	10	1	0
Term2	10	1	0
Term3	0	0	1

- Using the Euclidean distance Documents 1 and 2 look dissimilar, and Documents 2 and 3 look similar. This is just due to the length of the documents!!!

- Using the cosine of the angle between document vectors Documents 1 and 2 are similar to each other and dissimilar to Document 3.

Part 1: Matrix Norms

- Let $||\cdot||$ be a vector norm and $A \in R^{mxn}$.
- The corresponding matrix norm is $||\mathbf{A}|| = \mathbf{sup}_{\mathbf{x}\neq\mathbf{0}} \frac{||\mathbf{Ax}||}{||\mathbf{x}||}$.

- $||\mathbf{A}||_{\mathbf{2}} = (\max_{\mathbf{1}\leq \boldsymbol{i} \leq \boldsymbol{n}} \boldsymbol{\lambda_i}(\mathbf{A}^{\boldsymbol{T}}\mathbf{A}))^{\mathbf{1/2}}$ = square root of the largest eigenvalue of $\mathbf{A}^{\boldsymbol{T}}\mathbf{A}$. Heavy to compute!!!

- $||\mathbf{A}||_{\infty} = \max_{\mathbf{1}\leq \boldsymbol{i} \leq \boldsymbol{n}} \sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} |\boldsymbol{a}_{\boldsymbol{ij}}|$ (maximum over rows)

- $||\mathbf{A}||_{\mathbf{1}} = \max_{\mathbf{1}\leq \boldsymbol{j} \leq \boldsymbol{n}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} |\boldsymbol{a}_{\boldsymbol{ij}}|$ (maximum over columns)

- $||\mathbf{A}||_{\boldsymbol{F}} = \sqrt{\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{a}^{\mathbf{2}}{}_{\boldsymbol{ij}}}$ Frobenius norm: does not correspond to any vector norm. Still, related to Euclidean vector norm.

Part 1: Linear Independence

- Given a set of vectors $(\mathbf{v}_j)^n \ in\ R^m, m \geq n,$ consider the set of linear combinations $y = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$ for arbitrary coefficients α_j.

- The vectors $(\mathbf{v}_j)^n$ are linearly independent, if $\sum_{j=1}^{n} \alpha_j \mathbf{v}_j = 0$ if and only if $\alpha_j = 0$ for all $\mathrm{j} = 1, \cdots, n$.

- A set of *m* linearly independent vectors of R^m is called a basis in R^m: any vector in R^m can be expressed as a linear combination of the basis vectors.

- Example: The column vectors of the matrix are not linearly independent

$$[\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \mathbf{v}_4] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3 + \alpha_4\mathbf{v}_4 = 0$$

holds for $\alpha_1 = \alpha_3 = 1,\ \alpha_2 = \alpha_4 = -1.$

Part 1: Rank of a matrix

- The **rank** of a matrix is the maximum number of linearly independent column vectors.

- A square matrix $A \in R^{nxn}$ with rank *n* is called **nonsingular**, and it has an inverse $\mathbf{A}^{-1}$ satisfying $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

- The (outer product) matrix $\mathbf{x}\mathbf{y}^T$ has rank 1:
- All columns are linearly dependent (and so are all the rows).

- Example

The 4×4 matrix

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

has rank 3.

Part 1: Example

- Consider a $m \times n$ term-document matrix $\mathbf{A} = [a_1, a_2 \cdots a_n\ ]$, where $a_i \in R^m$ are the documents.

- If **A** has rank 3, then all the documents can be expressed as a linear combination of only three vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3 \in R^m$:

  $a_j = w_{1j}\mathbf{v}_1 + w_{2j}\mathbf{v}_2 + w_{3j}\mathbf{v}_3,\ j = 1, \cdots, n.$

- The term-document matrix can be written as

$$\mathbf{A} = \mathbf{VW}$$

  where $\mathbf{V} = (\mathbf{v}_1\mathbf{v}_2\mathbf{v_3}) \in R^{m\times 3}$ and $W = (w_{ij}) \in R^{3\times n}$.

Part 1: Orthogonality

- Two vectors **x** and **y** are orthogonal, if $\mathbf{x}^{\mathrm{T}}\mathbf{y}=0$.
- Let $q_j, j = 1, \cdots, n$ be orthogonal, i.e. $q^T q_j = 0, i \neq j$. Then, they are linearly independent.
- Let the set of orthogonal vectors $q_j, j = 1, \cdots, n \; in \; R^n$ be normalized, $||q|| = 1$. Then they are **orthonormal**, and constitute an **orthonormal basis** in R^n.
- A matrix $R^{m \times m} \ni Q = [q_1 q_2 \cdots q_m]$ with orthonormal column is called an **orthogonal matrix**.
- Example

$$\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \quad \text{to be} \quad \mathbf{v}_1 = \begin{pmatrix} 0.52 \\ 0.85 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} 0.85 \\ -0.52 \end{pmatrix}.$$

The vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are orthogonal:

$$\mathbf{v}_1^T \mathbf{v}_2 = 0.52 \cdot 0.85 + 0.85 \cdot (-0.52) = 0.$$

This is no coincidence!

Part 1: Why we like orthogonal matrices

- An orthogonal matrix $\mathbf{Q} \in R^{m \times m}$ has rank m (since its columns are linearly independent).
- $\mathbf{Q}^T\mathbf{Q}=\mathbf{I}$. $\mathbf{Q}\mathbf{Q}^T=\mathbf{I}$.
- The inverse of an orthogonal matrix **Q** is $\mathbf{Q}^{-1}=\mathbf{Q}^T$.
- The Euclidean length of a vector is invariant under an orthogonal transformation **Q**: $||\mathbf{Qx}||^2=(\mathbf{Qx})^T\mathbf{Qx}=\mathbf{x}^T\mathbf{x}=||\mathbf{x}||^2$.
- The product of two orthogonal matrices **Q** and **P** is orthogonal: $\mathbf{X}^T\mathbf{X}=(\mathbf{PQ})^T(\mathbf{PQ})=\mathbf{Q}^T\mathbf{P}^T\mathbf{PQ}=\mathbf{Q}^T\mathbf{Q}=\mathbf{I}$.

Part 1: Eigenvalues and eigenvectors of a symmetric matrix

- Let **A** be a *n* x *n* matrix. The vector **v** ≠ 0 that satisfies **Av**=**λv** for some scalar **λ** is called the eigenvector of **A** and **λ** is the eigenvalue corresponding to the eigenvector **v**.

- The eigenvectors of a symmetric matrix are mutually orthogonal and its eigenvalues are real.

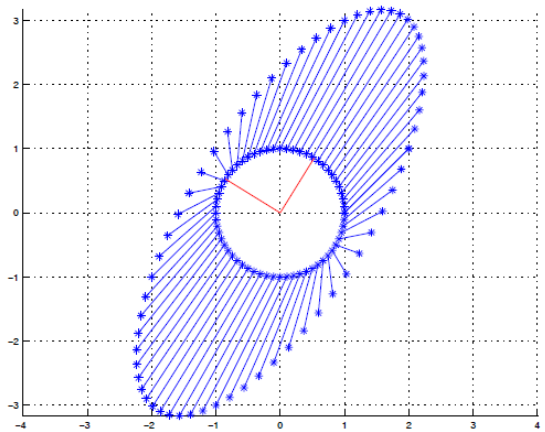
- A symmetric matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where the columns of **U** are the eigenvectors of **A** and $\mathbf{\Lambda}$ is a diagonal matrix, the diagonal elements of which are the corresponding eigenvalues of **A**. Note, that **U** is orthogonal. This is called the eigendecomposition or symmetric Schur decomposition of **A**.

$$\mathbf{A}\mathbf{v} = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \mathbf{v} = \lambda \mathbf{v}.$$

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix} = (2-\lambda)(3-\lambda) - 1 = 0$$

$$\lambda_1 = 3.62 \qquad \lambda_2 = 1.38$$

$$\mathbf{v}_1 = \begin{pmatrix} 0.52 \\ 0.85 \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} 0.85 \\ -0.52 \end{pmatrix}$$

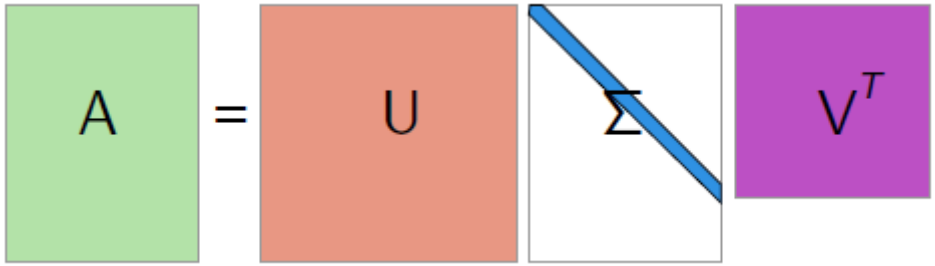


Part 1: What if the matrix is not symmetric?

- If **A** is symmetric, we can write $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where **U** and $\mathbf{\Lambda}$ contain the eigenvectors and corresponding eigenvalues of **A**.
- What if **A** Definitely not symmetric!!!
- Then we can use the singular value decomposition (SVD)

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, and $\mathbf{\Sigma}$ is diagonal.



Part 1: Example

customer \ day	Wed	Thu	Fri	Sat	Sun
ABC Inc.	1	1	1	0	0
CDE Co.	2	2	2	0	0
FGH Ltd.	1	1	1	0	0
NOP Inc.	5	5	5	0	0
Smith	0	0	0	2	2
Brown	0	0	0	3	3
Johnson	0	0	0	1	1

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{pmatrix} \times \begin{pmatrix} 9.64 & 0 \\ 0 & 5.29 \end{pmatrix} \times$$

$$\begin{pmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{pmatrix}$$

Part 1: Lessons learned

- Vector is
  - A 1D array of numbers, entity with magnitude and direction.

- Matrix is
  - A rectangular form representing observations.

- Orthogonal matrix
  - Has full rank
  - Has an inverse $\mathbf{Q}^T$
  - Is invariant

- Symmetric matrix has eigendecomposition as in $\mathbf{Av}=\boldsymbol{\lambda}\mathbf{v}$
  - Eigenvector $\mathbf{v}$, eigenvalue $\boldsymbol{\lambda}$

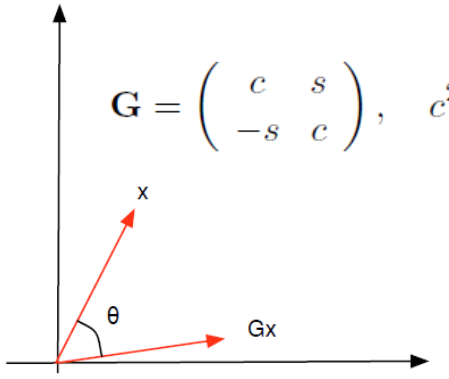
- Unsymmetric matrix has singular value and singular vectors as in $\mathbf{A} = \mathbf{U}\sum\mathbf{V}^T$
  - Left singular vector U
  - Right singular vector V
  - Singular value $\sum$

Content

Part 2: How to compute matrix decompositions?

- We have had a brief glimpse at eigenvalue decomposition and singular value decomposition.
- Before taking a closer look: how van we compute these?
- Answer 1: use LAPACK, MATLAB, Mathematica, …they are implemented everywhere!
- Answer 2: we need more linear algebra tools!!, Givens rotation, Householder transforms

$$\mathbf{G} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c^2 + s^2 = 1.$$



$$P = I - \frac{2}{v^T v} v v^T;$$

$$Px = x - \frac{2v^T x}{v^T v} v = x - v = y,$$

$$P_2 A^{(1)} = P_2 \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} =: A^{(2)}.$$

Part 2: Matrix decompositions

- We wish to decompose the matrix **A** by writing it as a product of two or more matrices

  $\mathbf{A}_{mxn} = \mathbf{B}_{mxk}\,\mathbf{C}_{kxn}$, $\mathbf{A}_{mxn} = \mathbf{B}_{mxk}\mathbf{C}_{kxr}\mathbf{D}_{rxn}$

- This is done in such a way that the **right side of the equation yields some useful information or insight** to the nature of the data matrix **A**.

- Or is in other ways useful for solving the problem at hand.

- There are numerous examples of useful matrix decompositions
  - **Eigendecomposition**: $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, **A** symmetric, **U** and $\boldsymbol{\Lambda}$ eigenvectors and eigenvalues.
  - **Singular value decomposition**: $\mathbf{A} = \mathbf{U}\sum\mathbf{V}^T$, **U**,**V** orthogonal, $\sum$ diagonal.
  - Matrix factorization is the same thing as matrix decomposition (e.g. **NMF** = nonnegative matrix factorization, **V**=**WH**, all elements nonnegative.)

Part 2: How to compute these?

- Roughly

(1) Manipulate **A** by multiplying it by intelligently chosen, fairly simple (orthogonal) matrices from both sides:

$$\mathbf{V}_k \ldots \mathbf{V}_2 \mathbf{V}_1 \mathbf{A} \mathbf{W}_1 \ldots \mathbf{W}_s = \mathbf{B},$$ until **B** is "nice".

(2) Denote $\mathbf{V}=\mathbf{V}_k \ldots \mathbf{V}_2 \mathbf{V}_1$, $\mathbf{W}=\mathbf{W}_1 \ldots \mathbf{W}_s$. Now $\mathbf{A}=\mathbf{V}\mathbf{B}\mathbf{W}^T$.

- But how to choose $\mathbf{V}_1, \ldots \mathbf{V}_k$, $\mathbf{W}_1, \ldots, \mathbf{W}_s$?

- Needed: linear algebra tools for transforming matrices in an orderly fashion: Givens!

Part 2: QR decomposition

- Any matrix $\mathbf{A} \in R^{m \times n}, m \geq n$, can be transformed to upper triangular form by an orthogonal matrix

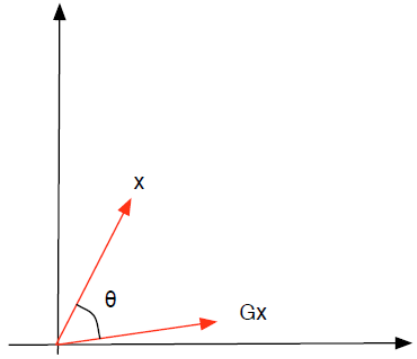
$$\mathbf{A} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix}$$

- If the columns of **A** are linearly independent, then **R** is non-singular.

- Needed: linear algebra tools for transforming matrices in an orderly fashion: Givens!

Let $\mathbf{x}$ be a vector. The parameters c and s, $c^2 + s^2 = 1$, can be chosen so that multiplication of $\mathbf{x}$ by

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & 0 & s \\ 0 & 0 & 1 & 0 \\ 0 & -s & 0 & c \end{pmatrix}$$



will zero the element 4 in vector $\mathbf{x}$ by a rotation in plane (2,4). How?

Part 2: What is QR good for?

- It will come up when we discuss the computation of some matrix decompositions
- It can also be used for solving the **least squares problem**.
- Other applications exist.

"Skinny" QR decomposition

Partition $\mathbf{Q} = (\mathbf{Q}_1\ \mathbf{Q}_2)$, where $\mathbf{Q}_1 \in \mathbb{R}^{m \times n}$:

$$\mathbf{A} = (\mathbf{Q}_1\ \mathbf{Q}_2) \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{R}_1.$$

Part 2: Least squares for linear regression

- Consider problems of the following form: given a number of measurements **X** =[$\mathbf{x}_1 \ldots \mathbf{x}_n$] and an outcome **y**, build a linear model

$$\hat{\mathbf{y}} = b_0 + \sum_{j=1}^{n} \mathbf{x_j} b_j,$$

- which uses the measurements **X** to predict the outcome **y**.
- **X** = clinical measures of patients, **y** = level of cancer specific antigen
- **X** = atmospheric measurements of each day, **y** = occurrence of spontaneous particle formation.
- The model $\hat{\mathbf{y}} = b_0 + \sum_{j=1}^{n} \mathbf{x_j} b_j$ can be written in the form
  - $\hat{y} = \mathbf{X}^{\mathrm{T}}\mathbf{b}$, $\mathbf{b}=(b_1 \ldots b_n b_0)^{\mathrm{T}}$.
  - To do this, one must append **X**=[$\mathbf{x}_1 \ldots \mathbf{x}_n \mathbf{x}_{n+1}$], where $\mathbf{x}_{n+1}$ is a vector of ones.
- Fitting a linear model to data is usually done using the method of least squares.
- Note. **X** need not be a square matrix !!!

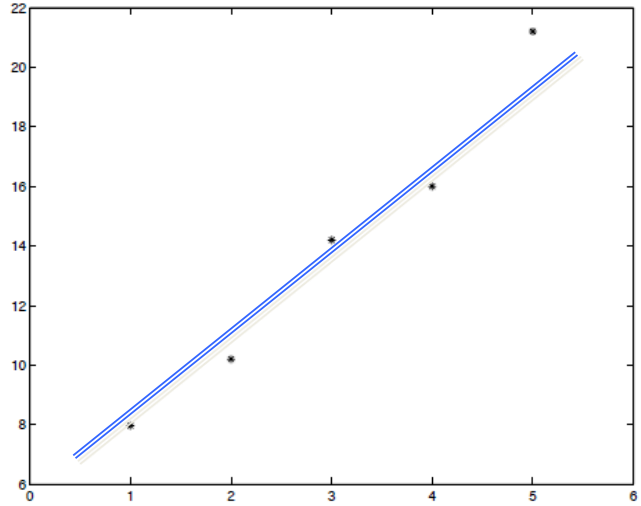
Part 2: Example

- We have measurement data

x	1	2	3	4	5
y	7.97	10.2	14.2	16.0	21.2

We wish to find α and β such that $\alpha x + \beta = y$. Thus

$$\begin{aligned}
\alpha + \beta &= 7.97 \\
2\alpha + \beta &= 10.2 \\
3\alpha + \beta &= 14.2 \\
4\alpha + \beta &= 16.0 \\
5\alpha + \beta &= 21.2
\end{aligned}$$



In matrix form:

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 7.97 \\ 10.2 \\ 14.2 \\ 16.0 \\ 21.2 \end{pmatrix}$$

Part 2: Example

- Make the residual vector
  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ as small as possible. But how?

- Make $\mathbf{r}$ orthogonal to the columns of $\mathbf{A}$:
  $\mathbf{r}^T(\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_n) = \mathbf{r}^T\mathbf{A} = 0.$

- Now write $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ to get the normal equations
  - $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$; solve for $\mathbf{x}$

- If the column vectors of $\mathbf{A}$ are linearly independent, then the **normal equations** $\mathbf{A}^T\mathbf{A}\mathbf{x}=\mathbf{A}^T\mathbf{b}$ are non-singular and have a unique solution.

- But the normal equations have two significant drawbacks:
  - Forming $\mathbf{A}^T\mathbf{A}$ leads to loss of information
  - The condition number of $\mathbf{A}^T\mathbf{A}$ is the square of that of $\mathbf{A}$

Part 2: Solving least squares problem using QR

- $||\mathbf{r}||^2 = ||\mathbf{b} - \mathbf{A}\mathbf{x}||^2 = ||\mathbf{b} - \mathbf{Q}\binom{\mathbf{R}}{\mathbf{0}}\mathbf{x}||^2 = ||\mathbf{Q}(\mathbf{Q}^\mathbf{T}\mathbf{b} - \binom{\mathbf{R}}{\mathbf{0}}\mathbf{x})||^2 = ||\mathbf{Q}^\mathbf{T}\mathbf{b} - \binom{\mathbf{R}}{\mathbf{0}}\mathbf{x}||^2$
- Partition **Q=(Q$_1$ Q$_2$),** where $\mathbf{Q_1} \in \boldsymbol{R^{m\times n}}$**,** and denote
  - $\mathbf{Q}^\mathbf{T}\mathbf{b} = \binom{\mathbf{b_1}}{\mathbf{b_2}} \coloneqq \binom{\mathbf{Q_1^T b}}{\mathbf{Q_2^T b}}$
- Then
  - $||\mathbf{r}||^2 = \left|\left|\binom{\mathbf{b_1}}{\mathbf{b_2}} - \binom{\mathbf{R}}{\mathbf{0}}\mathbf{x}\right)\right| |^2 = ||\mathbf{b_1} - \mathbf{R}\mathbf{x}||^2 + ||\mathbf{b_2}||^2$
- Minimize **||r||** by making the first term equal to zero: i.e. solve **Rx=b**$_1$
- Theorem. Let $\mathbf{A} \in \mathbf{R}^{m\times n}$ have full column rank and a thin QR-decomposition $\mathbf{A} = \mathbf{Q_1}\mathbf{R}$. Then the least squares problem

$$\min_x ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2$$

has the unique solution $\mathbf{x} = \mathbf{R}^{-1}{\mathbf{Q}_1}^T\mathbf{b}$.

Part 2: Updating the solution of the LS problem

Assume we have reduced the matrix and the right hand side

$$(\mathbf{A} \quad \mathbf{b}) \rightarrow \mathbf{Q}^T(\mathbf{A} \quad \mathbf{b}) = \begin{pmatrix} \mathbf{R} & \mathbf{Q}_1^T\mathbf{b} \\ \mathbf{0} & \mathbf{Q}_2^T\mathbf{b} \end{pmatrix}.$$

From this the solution of the LS problem is readily available.

Assume we have not saved $\mathbf{Q}$.

We then get a new observation $(\mathbf{a} \quad b)$, $\mathbf{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$.

Do we have to recompute the whole solution?

No. Instead, write the reduction on the previous slide as

$$\begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{a}^T & b \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{Q}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{a}^T & b \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{Q}_1^T\mathbf{b} \\ \mathbf{0} & \mathbf{Q}_2^T\mathbf{b} \\ \mathbf{a}^T & b \end{pmatrix}.$$

And reduce this to triangular form using plane rotations.

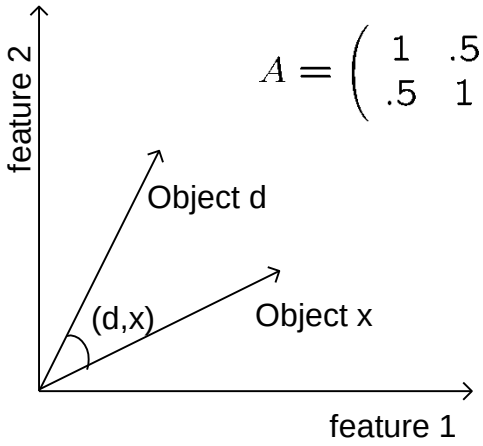
Part 2: The Singular Value Decomposition (SVD)

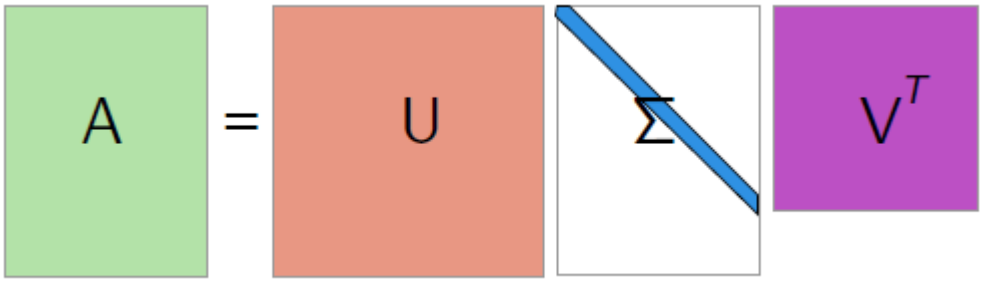
- Recall: data matrices have *m* rows (one for each object) and *n* columns (one for each feature).

- Matrix rows: points (vectors) in a Euclidean space,
- e.g., given 2 objects (*x* & *d*), each described with respect to two features, we get a 2-by-2 matrix.

- Two objects are "close" if the angle between their corresponding vectors is small.

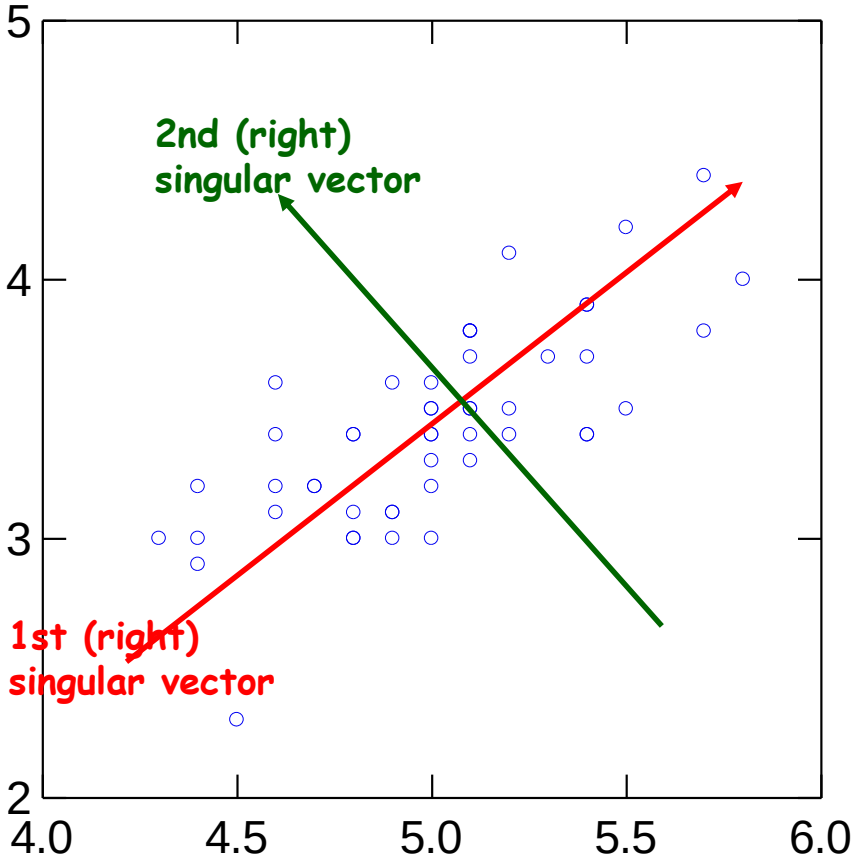
$$A = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix}$$





- I.e. every **A** has decomposition $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$
  - The singular value decomposition (SVD)
- The values σ_i are the singular values of **A**
- Columns of **U** are the left singular vectors and columns of **V** the right singular vectors of **A**

Part 2: Singular Value Decomposition (SVD) intuition



Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

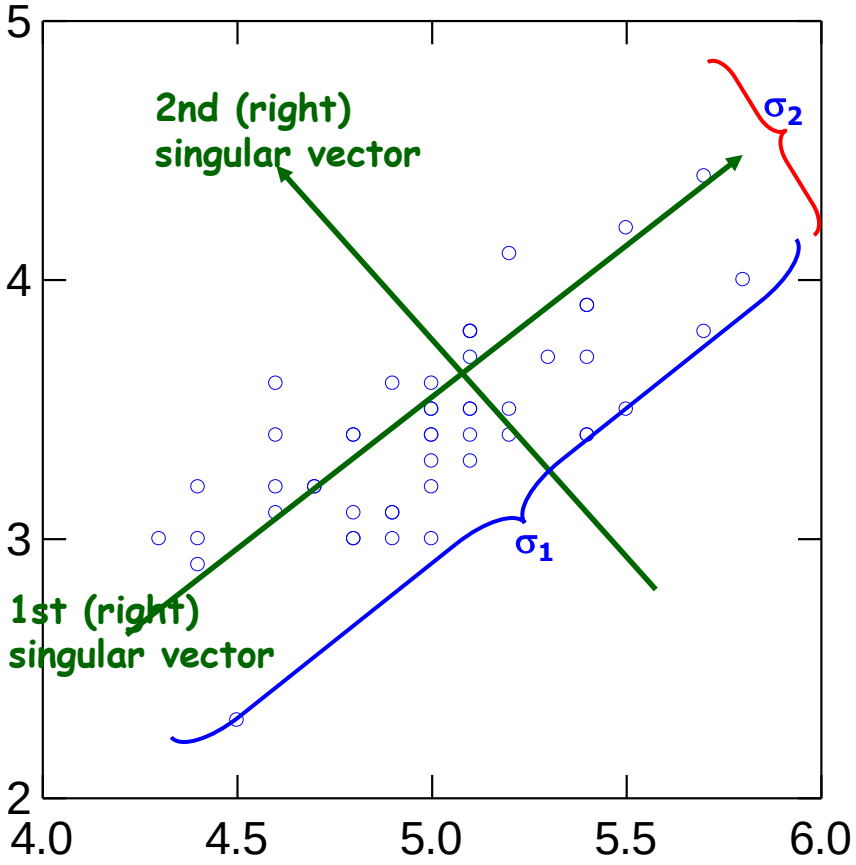
- 1st (right) singular vector:

direction of maximal variance,

- 2nd (right) singular vector:

direction of maximal variance, after removing the projection of the data along the first singular vector.

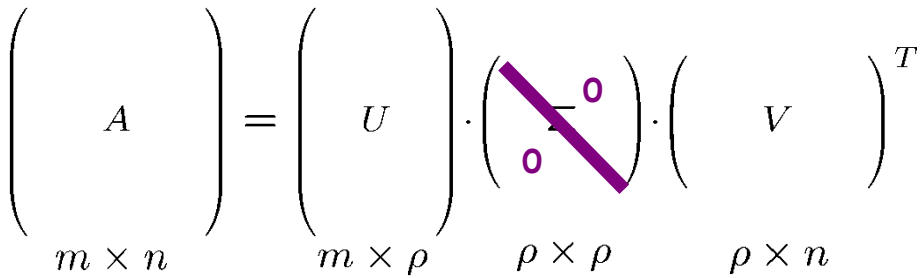
Part 2: Singular Values



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

Part 2: SVD: formal definition



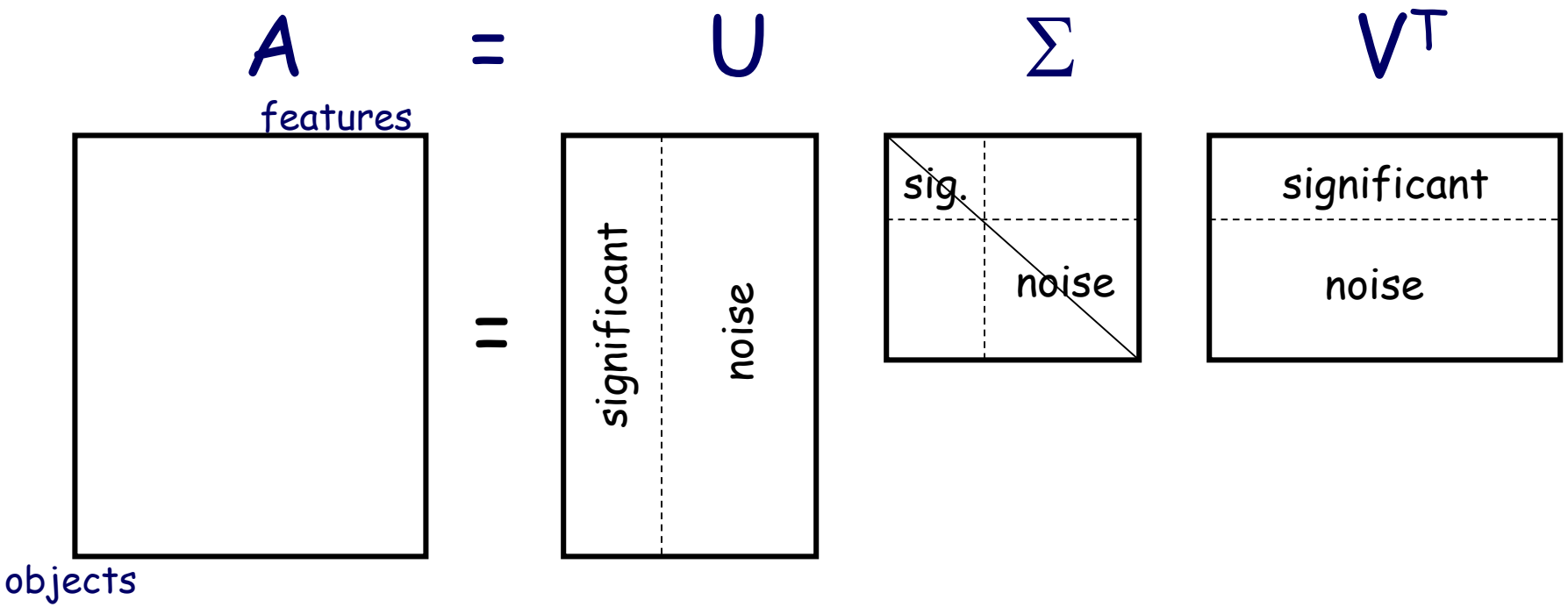
ρ: rank of **A**
U (V): orthogonal matrix containing the left (right) singular vectors of **A**.
Σ : diagonal matrix containing the singular values of **A**.

Let σ_1 , σ_2 , … , σ_ρ be the entries of Σ.

Exact computation of the SVD takes $O(\min\{mn^2 , m^2n\})$ time.

The top *k* left/right singular vectors/values can be computed faster using Lanczos/Arnoldi methods.

Part 2: Rank-k approximations via the SVD



$$\left(\begin{array}{c} A_k \\ \\ \\ \end{array}\right) = \left(\begin{array}{c} U_k \\ \\ \\ \end{array}\right) \cdot \left(\begin{array}{c} \Sigma_k \\ \\ \end{array}\right) \cdot \left(\begin{array}{c} V_k^T \\ \\ \\ \end{array}\right)$$

$m \times n \qquad m \times k \qquad k \times k \qquad k \times n$

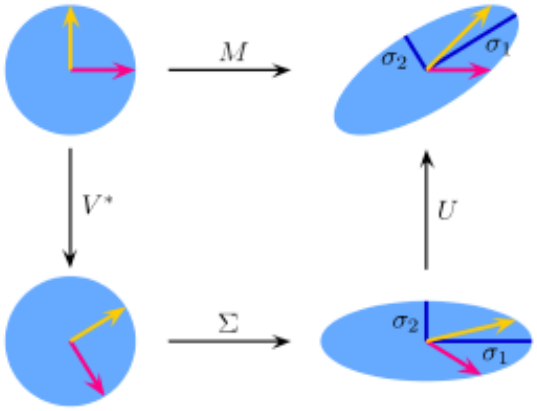
U_k (V_k): orthogonal matrix containing the top *k* left (right) singular vectors of **A**.
Σ_k: diagonal matrix containing the top *k* singular values of **A**.

Part 2: SVD is useful for

- Compression
- Noise reduction
- Finding “concepts” or “topics” (text mining/LSI)
- Data exploration and visualizing data (e.g. spatial data/PCA)
- Classification (of e.g. handwritten digits)
- SVD appears under different names:
  - Principal component analysis (PCA)
  - Latent semantic indexing (LSI) / Latent Semantic Analysis (LSA)
  - Karhunen-Loeve expansion / Hotelling transform (in image processing)

Part 2: Interpreting SVD

- The most common way to interpret SVD is to consider the columns of **U** (or **V**)
  - Let **A** be objects-by-attributes and $\mathbf{U}\sum\mathbf{V}^T$ its SVD
  - If two columns have similar values in a row of $\mathbf{V}^T$, these attributes are somehow similar (have strong correlation)
  - If two rows have similar values in a column of **U**, these uses are somehow similar
- Geometric interpretation
  - SVD shows that every linear mapping **y=Mx** can be considered as a **series of rotation, stretching, and rotation operations**
  - Matrix $\mathbf{V}^T$ performs the first rotation $\mathbf{y_1}=\mathbf{V}^T\mathbf{x}$
  - Matrix $\sum$ performs the stretching $\mathbf{y_2}=\sum \mathbf{y_1}$
  - Matrix **U** performs the second rotation $\mathbf{y}=\mathbf{U}\mathbf{y_2}$



$$M = U \cdot \Sigma \cdot V^*$$

- Dimension of largest variance
  - The singular vectors give the dimensions of the variance in the data
    - The first singular vector is the dimension of the largest variance
    - The second singular vector is the orthogonal dimension of the second…
- Component interpretations: sums of rank-1 layers
  - The first layer explains the most
  - The second corrects that by adding and removing smaller values

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sum_{i=1}^{r} \mathbf{A}_i$$

Part 2: Eigenvalue decomposition vs. SVD

- For symmetric **A** the singular value decomposition is closely related to the eigendecomposition $\mathbf{A}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, **U** and $\mathbf{\Lambda}$ eigenvectors and eigenvalues.

- Computation of both the eigendecomposition and the **SVD** follow the same pattern

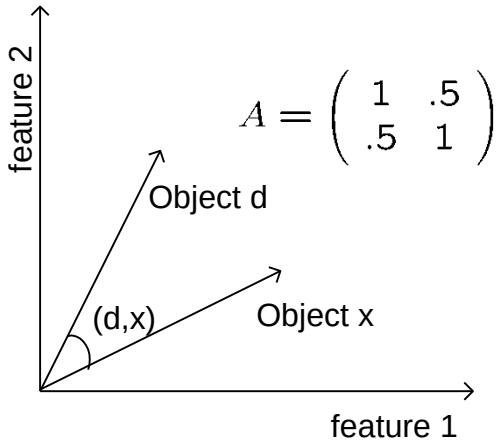
- Computation of the eigenvalue decomposition
  - Use givens to transform **A** into tridiagonal form
  - Use QR iteration with Wilkinson shift $\boldsymbol{\mu}$ to transform tridiagonal form to diagonal form: Repeat until converged
    - $\mathbf{QR} = \mathbf{T}_k - \boldsymbol{\mu}_k \mathbf{I}$
    - $\mathbf{T}_{k+1} = \mathbf{RQ} + \boldsymbol{\mu}_k \mathbf{I}$
  - Once an eigenvalue is found, forget it, reduce (deflate) the problem and repeat the same step.

- Computing the **SVD**
  - Use Householder transformations to transform **A** into bidiagonal form **B**. Now $\mathbf{B}^T\mathbf{B}$ is tridiagonal
  - Use **QR** iteration with Wilkinson shift to transform tridiagonal **B** to diagonal form (without forming $\mathbf{B}^T\mathbf{B}$ implicitly!)

Part 2: PCA and SVD

Principal Components Analysis (PCA) essentially amounts to the computation of the Singular Value Decomposition (SVD) of a covariance matrix.
SVD is the algorithmic tool behind MultiDimensional Scaling (MDS) and Factor Analysis.

SVD is "the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra."*
*Dianne O'Leary, MMDS '06



- **PCA is SVD done on centered data.**
- PCA looks for such a direction that the data projected onto it has maximal variance.
- When found, PCA continues by seeking the next direction, which is orthogonal to all the previously found directions, and which explains as much of the remaining variance in the data as possible.
- Principal components are uncorrelated.

- PCA is useful for
  - Data exploration, Visualizing data, Compressing data, Outlier detection, Ratio rules.

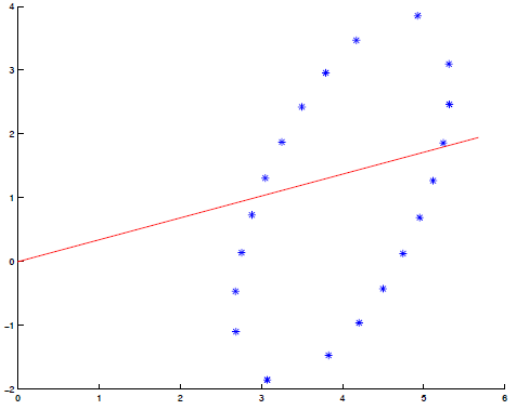
Part 2: How to compute the PCA

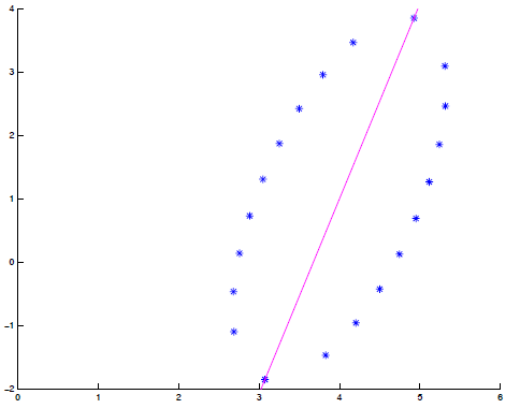
- Data matrix **A**, rows=data points, columns=variables (attributes, parameters).

1. Center the data by subtracting the mean of each column.
2. Compute the SVD of the centered matrix $\widehat{\mathbf{A}}$ (or the *k* first singular values and vectors)

$$\widehat{\mathbf{A}} = \mathbf{U}\sum\mathbf{V}^{\mathsf{T}}$$

3. The principal components are the columns of **V**, the coordinates of the data in the basis defined by the principal components are $\mathbf{U}\sum$.

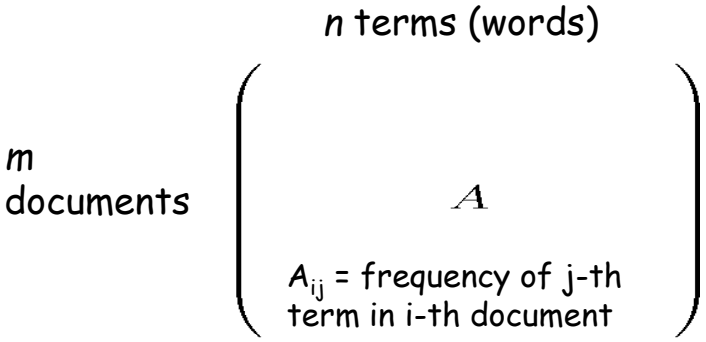




Part 2: LSI: A_k for document-term matrices

Latent Semantic Indexing (LSI)

Replace **A** by $\mathbf{A_k}$; apply clustering/classification algorithms on $\mathbf{A_k}$.



Pros

- Less storage for small *k*.
  O(km+kn) vs. O(mn)
- Improved performance.
  Documents are represented in a "concept" space.

Cons

- A_k destroys sparsity.
- Interpretation is difficult.
- Choosing a good *k* is tough.

- The data: a term-document matrix **A**
  - The values are (weighted) term frequencies
  - Typically TF/IDF values
- The truncated SVD $\mathbf{A_k}=\mathbf{U_k}\sum_k\mathbf{V_k}^T$ of **A** is computed
  - Matrix $\mathbf{U_k}$ associates documents to topics
  - Matrix $\mathbf{V_k}$ associates topics to terms
  - If two rows of $\mathbf{U_k}$ are similar, the corresponding documents "talk about the same things"
- A query q can be answered by considering its term vector q
  - **q** is projected to $\mathbf{q_k} = \mathbf{qV}\sum^{-1}$
  - $\mathbf{q_k}$ is compared to rows of **U** and most similar rows are returned

Part 2: A_k and k-means clustering

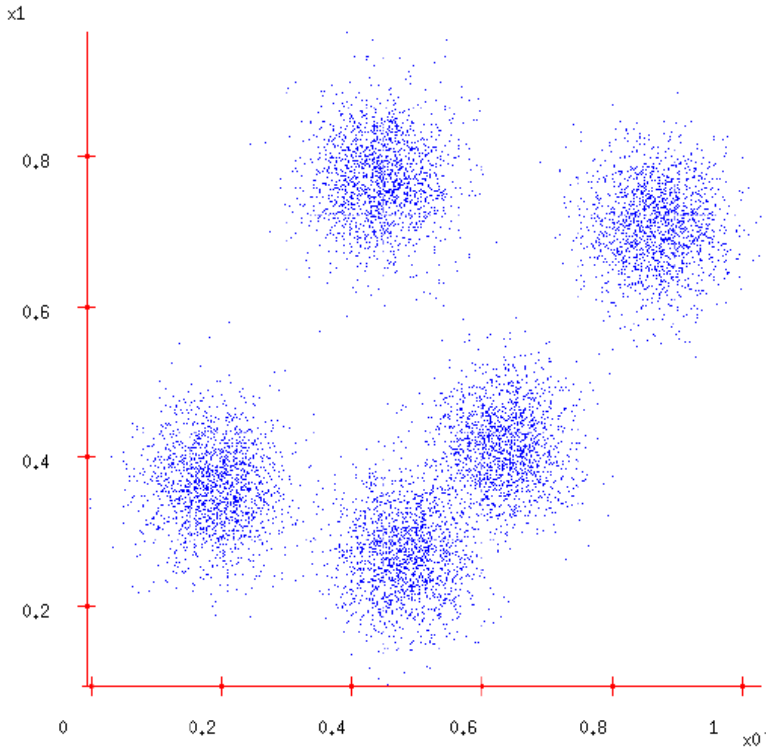
k-means clustering

A standard objective function that measures cluster quality.
(Often denotes an iterative algorithm that attempts to optimize the *k*-means objective function.)

k-means objective

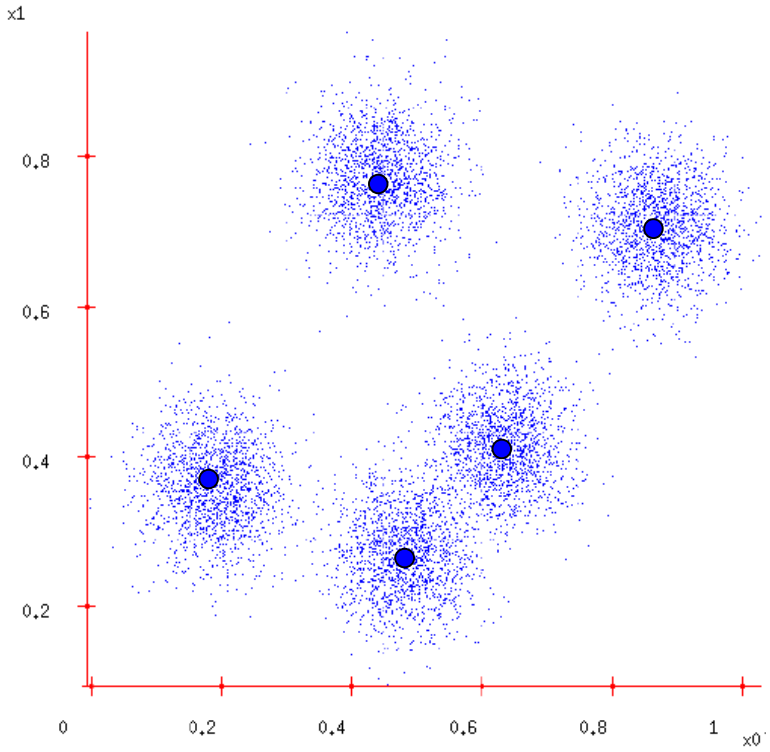
Input: set of *m* points in R^n, positive integer *k*
Output: a partition of the *m* points to *k* clusters
Partition the *m* points to *k* clusters in order to minimize the sum of the squared Euclidean distances from each point to its cluster centroid.

Part 2: k-means, cont'd



We seek to split the input points in 5 clusters.

Part 2: k-means, cont'd



We seek to split the input points in 5 clusters.

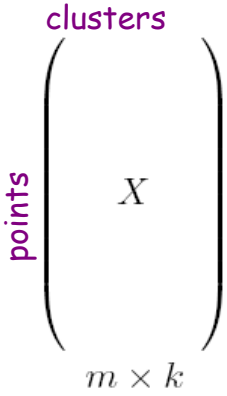
The cluster centroid is the "average" of all the points in the cluster.

Part 2: k-means: a matrix formulation

Let A be the *m*-by-*n* matrix representing *m* points in R^n. Then, we seek to

$$\min_{X \in \mathbb{R}^{m \times k}} \|A\|_F^2 - \left\|X^T A\right\|_F^2 \qquad \text{or} \qquad \max_{X \in \mathbb{R}^{m \times k}} \left\|X^T A\right\|_F^2$$

X is a special "cluster membership" matrix: X_{ij} denotes if the *i-th* point belongs to the *j-th* cluster.



- Columns of X are normalized to have unit length.
(We divide each column by the square root of the number of points in the cluster.)
- Every row of X has at most one non-zero element.
(Each element belongs to at most one cluster.)
- X is an orthogonal matrix, i.e., $X^TX = I$.

Part 2: SVD and *k*-means

If we only require that X is an orthogonal matrix and remove the condition on the number of non-zero entries per row of X, then

$$\min_{X \in \mathbb{R}^{m \times k}} \|A\|_F^2 - \left\|X^T A\right\|_F^2 \qquad \text{or} \qquad \max_{X \in \mathbb{R}^{m \times k}} \left\|X^T A\right\|_F^2$$

is easy to minimize! The solution is $X = U_k$.

Using SVD to solve *k*-means

• We can get a 2-approximation algorithm for k-means.
(Drineas, Frieze, Kannan, Vempala, and Vinay '99, '04)

• We can get heuristic schemes to assign points to clusters.
(Zha, He, Ding, Simon, and Gu '01)

• There exist PTAS (based on random projections) for the *k*-means problem.
(Ostrovsky and Rabani '00, '02)

• Deeper connections between SVD and clustering in Kannan, Vempala, and Vetta '00, '04.

Part 2: A_k and Kleinberg's HITS algorithm (Kleinberg '98, '99)

Hypertext Induced Topic Selection (HITS)

A link analysis algorithm that rates Web pages for their authority and hub scores.
Authority score: an estimate of the value of the content of the page.
Hub score: an estimate of the value of the links from this page to other pages.
These values can be used to rank Web search results.

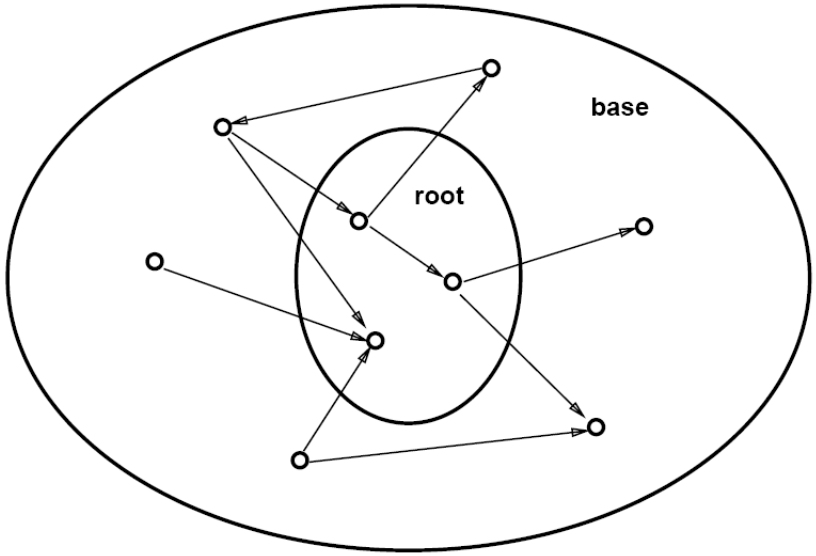
Authority: a page that is pointed to by many pages with high hub scores.
Hub: a page pointing to many pages that are good authorities.

Recursive definition; notice that each node has two scores.

Part 2: A_k and Kleinberg's HITS algorithm (Kleinberg '98, '99)

Phase 1:

Given a query term (e.g., "jaguar"), find all pages containing the query term (root set).
Expand the resulting graph by one move forward and backward (base set).



Part 2: A_k and Kleinberg's HITS algorithm (Kleinberg '98, '99)

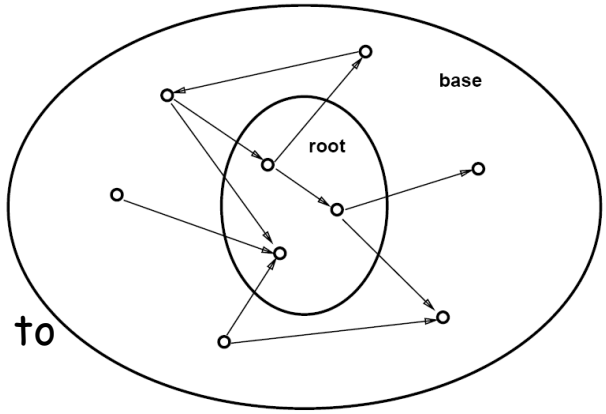
Phase 2:

Let A be the adjacency matrix of the (directed) graph of the base set.

Let h , $a \in R^n$ be the vectors of hub (authority) scores.

Then, $\boldsymbol{h = Aa}$ and $\boldsymbol{a = A^T h}$

$\boldsymbol{h = AA^T h}$ and $\boldsymbol{a = A^T Aa}$.



Thus, the top left (right) singular vector of A corresponds to hub (authority) scores.

What about the rest?

They provide a natural way to extract additional densely linked collections of hubs and authorities from the base set.

See the "jaguar" example in Kleinberg '99.

Part 2: SVD Lessons learned

- SVD is the Swiss Army knife of (numerical) linear algebra

→ ranks, kernels, norms, …

- SVD is also very useful in data analysis

→ noise removal, visualization, dimensionality reduction, …

- Selecting the correct rank for the truncated SVD is still a problem

Part 2: Non-Negative Datasets

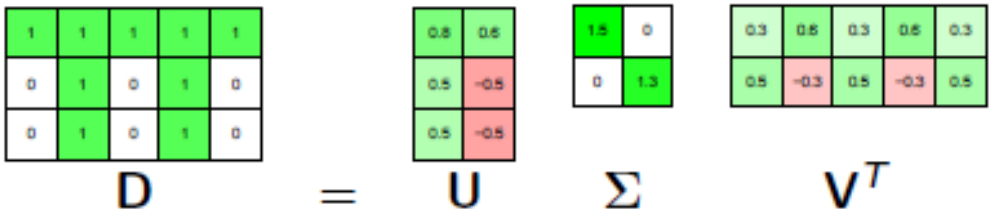
- Some datasets are intrinsically non-negative
  - Counters (e.g., no. occurrences of each word in a text document)
  - Quantities (e.g., amount of each ingredient in a chemical experiment)
  - Intensities (e.g., intensity of each color in an image)

- The corresponding data matrix **D** has only non-negative values
  - Decompositions such as **SVD** and **SDD** may involve negative values in factors and components
  - Negative values describe the absence of something
  - Often no natural interpretation

Can we find a decomposition that is more natural to non-negative data?

Part 2: Example (SVD)

Consider the following "bridge" matrix and its truncated SVD:

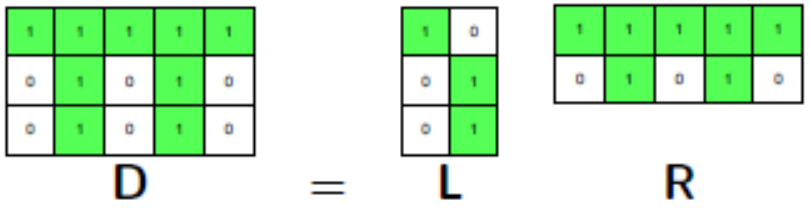


Here are the corresponding components:

Negative values make interpretation unnatural or difficult.

Part 2: Example (NMF)

Consider the following "bridge" matrix and its rank-2 NMF:



Here are the corresponding components:

Non-negative matrix decomposition encourage a more natural, part-based representation and (sometimes) sparsity.

Part 2: Examples

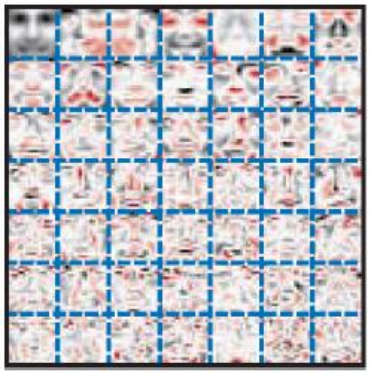
Decomposing faces (PCA)

D_{i*} (original)

$[LR]_{i*}$
$[U\Sigma V^T]_{i*}$

$=$
$=$

L_{i*}
$U_{i*}\Sigma$



R
V^T

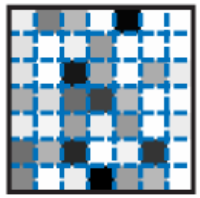
PCA factors are hard to interpret.

Decomposing faces (NMF)

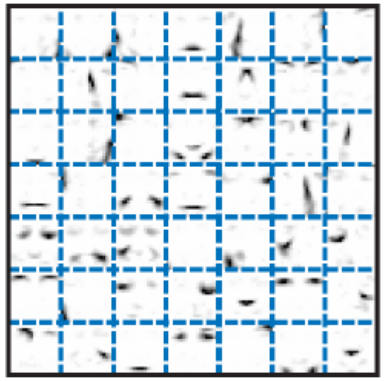
D_{i*} (original)

$[LR]_{i*}$

$=$



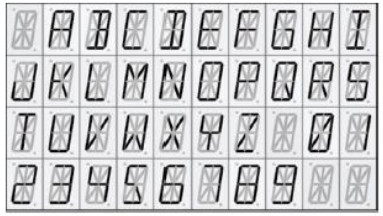
L_{i*}



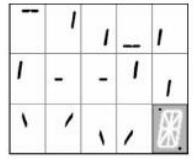
R

NMF factors correspond to parts of faces.

Decomposing digits (NMF)



D



R

NMF factors correspond to parts of digits and "background".

Part 2: Decompositions that respect the data

- Given a nonnegative matrix $\mathbf{A} \in \mathbf{R}^{m \times n}$, we wish to express the matrix as a product of two nonnegative matrices $\mathbf{W} \in \mathbf{R}^{m \times k}$ and $\mathbf{H} \in \mathbf{R}^{k \times n}$ such that

$$\mathbf{A} \approx \mathbf{WH}$$

- Why require nonnegativity?
  - Nonnegativity is natural in many applications
  - Term-document
  - Market basket
  - Etc.

$$\min_{\Phi \in \mathcal{R}^{m \times k}, X \in \mathcal{R}^{k \times n}} \left\| \underset{m \times n}{A} - \underset{m \times k}{\Phi} \cdot \underset{k \times n}{X} \right\|_F^2$$

Non-negative matrix factorization
(Lee and Seung '00)
Assume that the A_{ij} are non-negative for all *i,j*.

Constrain Φ and X to have only non-negative entries as well.
This should respect the structure of the data better than $A_k = U_k \Sigma_k V_k^T$ which introduces a lot of (difficult to interpret) negative entries.

Part 2: Example

3	2	4	0	0	0	0
1	0	1	0	0	0	0
0	1	2	0	0	0	0
0	0	0	1	1	1	1
0	0	0	1	0	1	1
0	0	0	3	2	2	3

Rows: customers, columns: products they buy.

PC=

-0.3751	0.4383	-0.8047
-0.2728	0.2821	0.4166
-0.5718	0.4372	0.4073
0.3940	0.4311	0.0413
0.2537	0.3341	0.0907
0.2883	0.2322	-0.0378
0.3940	0.4311	0.0413

W =

0	2.3515
0	0.5073
0	0.7955
0.8760	0
0.6937	0
2.4179	0

H' =

0	1.1206
0	0.7992
0	1.7347
1.1918	0
0.7532	0
0.8510	0
1.1918	0

Rows of $\mathbf{W}$ are customers, rows of $\mathbf{H}^T$ are products.

Principal components. Each column corresponds to a product.

Part 2: The Non-negative Matrix Factorization

$$\min_{\substack{\Phi \in \mathcal{R}^{m \times k}, X \in \mathcal{R}^{k \times n} \\ \forall i,j,\ \Phi_{ij}, X_{ij} \geq 0}} \left\| \underset{m \times n}{\begin{pmatrix} A \end{pmatrix}} - \underset{m \times k}{\begin{pmatrix} \Phi \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} X \end{pmatrix}} \right\|_F^2$$

It has been extensively applied to:

1. Image mining (Lee and Seung ’00)
2. Enron email collection (Berry and Brown ’05)
3. Other text mining tasks (Berry and Plemmons ’04)

Algorithms for NMF:

1. Multiplicative update rules (Lee and Seung ’00, Hoyer ’02)
2. Gradient descent (Hoyer ’04, Berry and Plemmons ’04)
3. Alternating least squares (dating back to Paatero ’94)

How to compute NMF: Multiplicative algorithm

```
W=rand(m,k);
H=rand(k,n);
for i=1:maxiter
        H=H.*(W'*A)./(W'*W*H+epsilon);
        W=W.*(A*H')./(W*H*H'+epsilon);
end
```

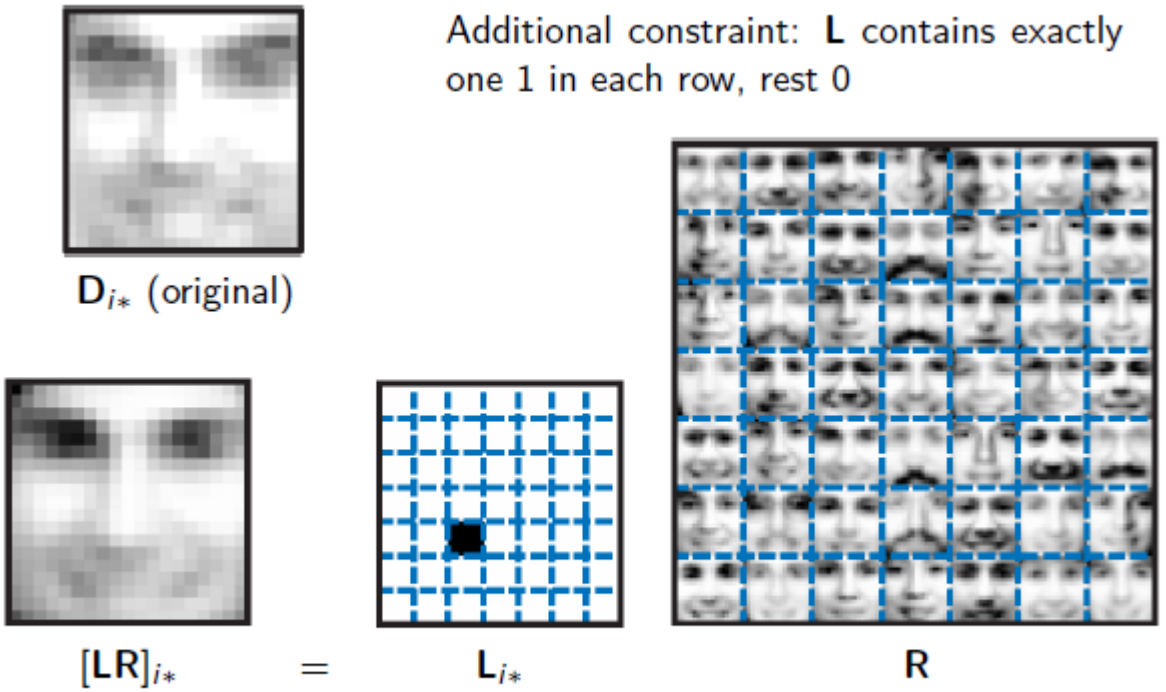
How to compute NMF: ALS

$\mathbf{W} = \text{rand}(m, k)$;
for i=1:maxiter

Solve for $\mathbf{H}$ in equation $\mathbf{W}^T\mathbf{W}\mathbf{H} = \mathbf{W}^T\mathbf{A}$
Set all negative elements in $\mathbf{H}$ to 0.
Solve for $\mathbf{W}$ in equation $\mathbf{H}\mathbf{H}^T\mathbf{W}^T = \mathbf{H}\mathbf{A}^T$.
Set all negative elements of W to zero.

end

Part 2: K-means can be seen as a variant of NMF



k-Means factors correspond to prototypical faces.

Part 2: Algorithmic challenges for NMF

$$\min_{\Phi \in \mathcal{R}^{m \times k}, X \in \mathcal{R}^{k \times n}} \left\| \underset{m \times n}{\begin{pmatrix} A \end{pmatrix}} - \underset{m \times k}{\begin{pmatrix} \Phi \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} X \end{pmatrix}} \right\|_F^2$$
$$\forall i, j, \ \Phi_{ij}, X_{ij} \geq 0$$

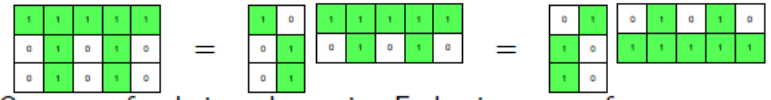
Algorithmic challenges for the NMF:

1. NMF (as stated above) is convex given Φ or X, but not if both are unknown.
2. No unique solution: many matrices Φ and X that minimize the error.
3. Other optimization objectives could be chosen (e.g., spectral norm, etc.)
4. NMF becomes harder if sparsity constraints are included (e.g., X has a small number of non-zeros).
5. For the multiplicative update rules there exists some theory proving that they converge to a fixed point; this might be a local optimum or a saddle point.
6. Little theory is known for the other algorithms.

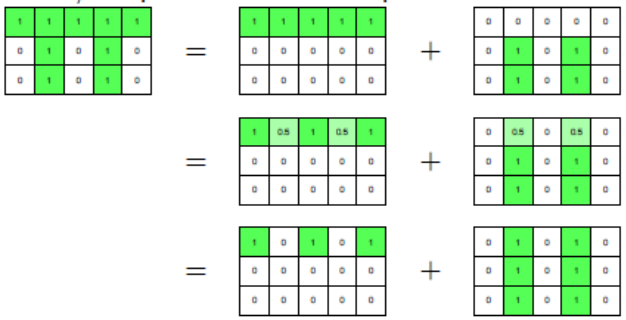
Part 2: NMF Challenges

NMF is not unique

- Factors are not "ordered"



- One way of ordering: decreasing Frobenius norm of components (i.e., order by $\|\mathbf{L}_{*k}\mathbf{R}_{k*}\|_F$)
- Factors/components are not unique



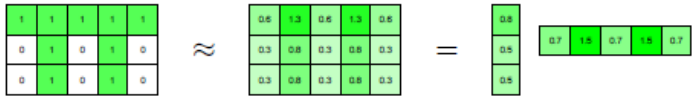
Additional constraints or regularization can encourage uniqueness.

General framework

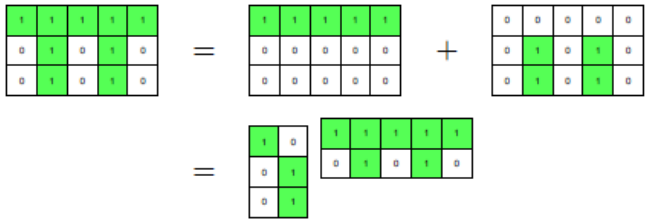
- Gradient descent generally slow
- Stochastic gradient descent inappropriate
- Key approach: alternating minimization
  1: Pick starting point $\mathbf{L}_0$ and $\mathbf{R}_0$
  2: **while** not converged **do**
  3: Keep $\mathbf{R}$ fixed, optimize $\mathbf{L}$
  4: Keep $\mathbf{L}$ fixed, optimize $\mathbf{R}$
  5: **end while**
- Update steps 3 and 4 easier than full problem

NMF is not hierarchical

- Rank-1 NMF



- Rank-2 NMF



- Best rank-k approximation may differ significantly from best rank-$(k-1)$ approximation
- Rank influences sparsity, interpretability, and statistical fidelity
- Optimum choice of rank is not well-studied (often requires experimentation)

Part 2: NMF Summary

- Given a nonnegative matrix $\mathbf{A} \in \mathbf{R}^{m \times n}$ find nonnegative matrices $\mathbf{W} \in \mathbf{R}^{m \times k}$ and $\mathbf{H} \in \mathbf{R}^{k \times n}$ so that

$$\min_{W,H} ||\mathbf{A} - \mathbf{WH}||$$

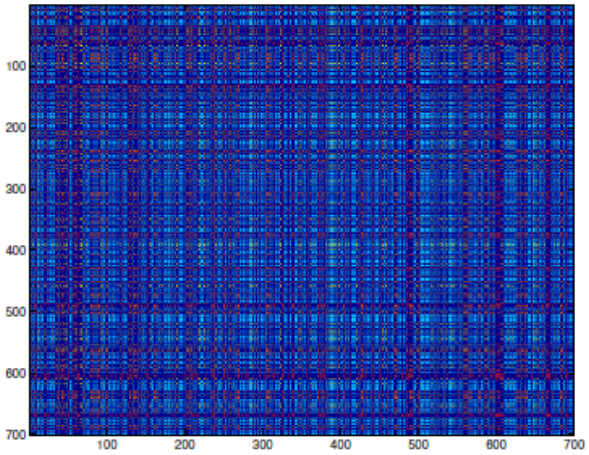
- Algorithms exist, both basic (easy to implement) and more advanced (implementing e.g. sparsity constraints)

- Interpretability

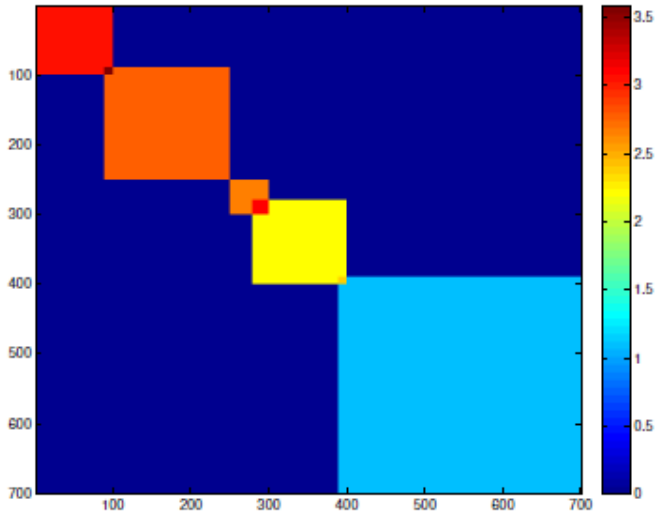
- Applications
  - Text mining
  - Email surveillance
  - Music transcription
  - Bioinformatics
  - Source separation
  - Spatial data analysis
  - Etc.

Part 2: SemiDiscrete Decomposition (SDD)

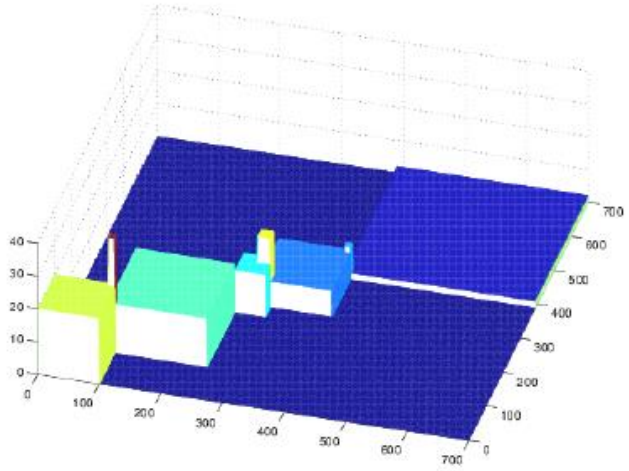
An example data



The data



The data after permuting rows and columns



The data in a 3D view

Can we find the bump in the picture automatically (from unpermuted data)?

What is a bump?
A submatrix of a matrix $A \in R^{m \times n}$ contains some rows of A and some columns of those row

$$\boldsymbol{A} = \begin{pmatrix} 3 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 2 & 3 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \boldsymbol{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \boldsymbol{A} \circ \boldsymbol{x}\boldsymbol{y}^T = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 0 & 0 \\ 3 & 0 & 3 \end{pmatrix}$$

Part 2: SemiDiscrete Decomposition (SDD)

$$\underset{m \times n}{A_k} = \underset{m \times k}{U_k} \cdot \underset{k \times k}{\Sigma_k} \cdot \underset{k \times n}{V_k^T}$$

$$\underset{m \times n}{A_{SDD}} = \underset{m \times k}{X_k} \cdot \underset{k \times k}{D_k} \cdot \underset{k \times n}{Y_k^T}$$

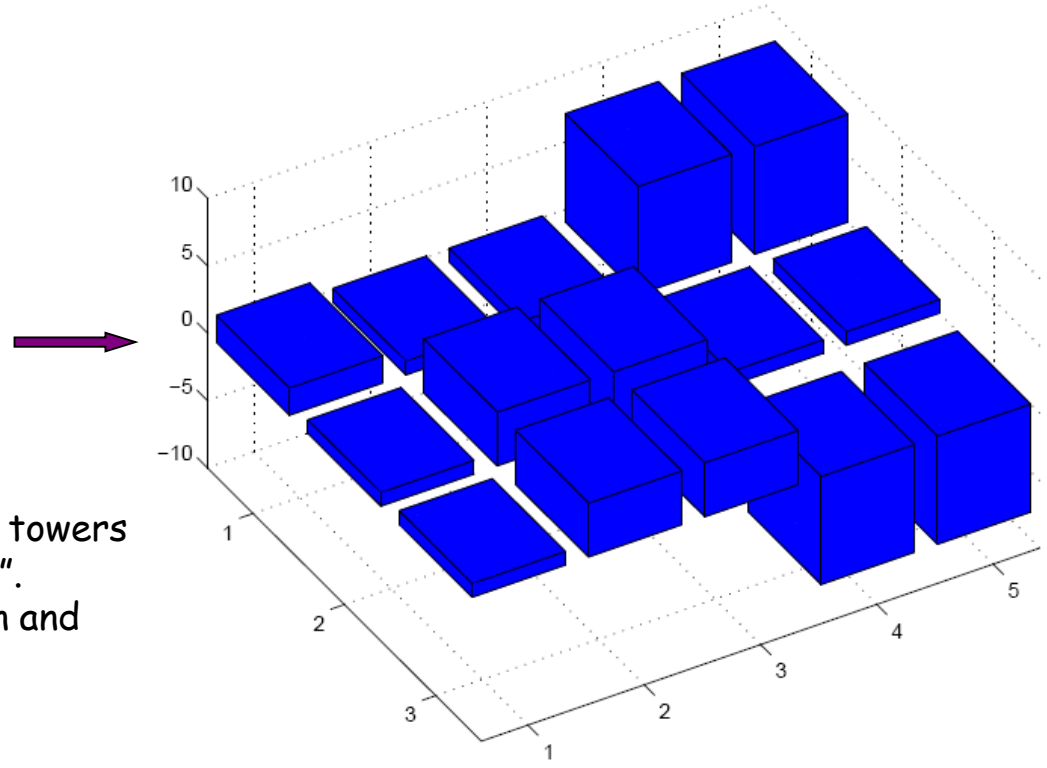
D_k: diagonal matrix
X_k, Y_k: all entries are in {-1,0,+1}

SDD identifies regions of the matrix that have homogeneous density.

- Assume we know how to find the largest bump of a matrix
- To find another bump, we can subtract the found bump from the matrix and find the largest bump of the residual matrix

Part 2: SemiDiscrete Decomposition (SDD)

$$\begin{bmatrix} 2 & 1 & 1 & 8 & 8 \\ 1 & 4 & 4 & 1 & 1 \\ 1 & 4 & 4 & -8 & -8 \end{bmatrix}$$



SDD looks for blocks of similar height towers and similar depth holes: "bump hunting". Applications include image compression and text mining.

O'Leary and Peleg '83, Kolda and O'Leary '98, '00, O'Leary and Roth '06
The figures are from D. Skillkorn's book on Data Mining with Matrix Decompositions.

Part 2: Properties of SDD

- The columns of $\mathbf{X_k}$ and $\mathbf{Y_k}$ do not need to be linearly independent
  - The same column can be even repeated multiple times
- The dimension *k* might need to be large for accurate approximation (compared to SVD)
- SDD factors are local
  - Only affect a certain submatrix, typically not every element
  - SVD factors typically change every value
- Storing an *k*-dimensional SDD takes less space than storing rank-*k* truncated SVD
  - $\mathbf{X_k}$ and $\mathbf{Y_k}$ are ternary and often sparse
- For every rank-1 layer of an SDD, all non-zero values in the layer have the same magnitude (d_{ii} for layer *i*)

Part 2: Interpretation of SDD

- The factor interpretation is not very useful as the factors are not independent
  - A later factor can change just a subset of values already changed by an earlier factor

- The SDD can be interpret as a form of bi-clustering
  - Every layer (bump) defines a group of rows and columns with homogeneous values in the residual matrix

- The component interpretation is natural to SDD
  - The SDD is a sum of local bumps
  - SDD does not model global phenomena (e.g. noise) well

Part 2: The outline of the SDD algorithm

1. **Input:** Matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, non-negative integer k
2. **Output:** k-dimensional SDD of $\boldsymbol{A}$, i.e. matrices $X_k \in \{-1, 0, 1\}^{m \times k}$, $\boldsymbol{Y}_k \in \{-1, 0, 1\}^{n \times k}$, and diagonal $\boldsymbol{D}_k \in \mathbb{R}_+^{k \times k}$
3. $\boldsymbol{R}_1 \leftarrow \boldsymbol{A}$
4. **for** $i = 1, \ldots, k$
    1. Select $\boldsymbol{y}_i \in \{-1, 0, 1\}^n$
    2. **while** not converged
        1. Compute $\boldsymbol{x}_i \in \{-1, 0, 1\}^m$ given $\boldsymbol{y}_i$ and $\boldsymbol{R}_i$
        2. Compute $\boldsymbol{y}_i$ given $\boldsymbol{x}$ and $\boldsymbol{R}_i$
    3. **end while**
    4. Set d_i to the average of $\boldsymbol{R}_i \circ \boldsymbol{x}_i \boldsymbol{y}_i^T$ over the non-zero locations of $\boldsymbol{x}\boldsymbol{y}^T$
    5. Set $\boldsymbol{x}_i$ as the ith column of $\boldsymbol{X}_i$, $\boldsymbol{y}_i$ the ith column of $\boldsymbol{Y}_i$, and d_i the ith value of $\boldsymbol{D}_i$
    6. $\boldsymbol{R}_{i+1} \leftarrow \boldsymbol{R}_i - d_i \boldsymbol{x}_i \boldsymbol{y}_i^T$
5. **end for**
6. **return** $\boldsymbol{X}_k$, $\boldsymbol{Y}_k$, and $\boldsymbol{D}_k$

Part 2: SDD Clustering

- SDD performs a type of bi-clustering of the matrix
  - Every bump $\boldsymbol{dxy^T}$ gives a cluster of rows $x_i \neq 0$ and a cluster of columns $y_j \neq 0$, together with 'centroid' d
  - This is **not** a partition clustering: the clusters can overlap and not every row or column has to belong to some cluster

- We can impose an ordering of the bumps based on the values of d_i
  - The algorithm usually returns the bumps in that order

- This ordering can be used to obtain a hierarchical clustering
  - First column of **X** clusters the rows of **A** into three sets (-1,0,1) and same for the first column of **Y** and columns of **A**
  - The second column of **X** splits the previous clusters again into three sets
  - And so on and so forth

- Distance between two objects in the hierarchical clustering is not the usual dendrogram depth, but depends on whether we use the 0 or +/-1 branches

Part 2: SDD Other applications

- Image compression (O'Leary and Peleg, 1983)
  - A grayscale image is compressed using its SDD
  - The original application, modern image compression techniques are better

- Latent topic models (Kolda and O'Leary, 1998)
  - Used similarly as SVD is used to compute LSA
  - Compute the SDD of the term-document matrix

- SDD to correlation matrices
  - A bump in the correlation matrix $\mathbf{A}\mathbf{A}^{\mathbf{T}}$ corresponds to rows of $\mathbf{A}$ with similar values
  - A bump in $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ corresponds to columns with similar values

Part 2: SVD+SDD General approaches

- Most common way to combine SVD and SDD is to first use SVD to denoise the data and then to compute the SDD on the clean data
  - Clean data = truncated SVD($\mathbf{A_k}$)
  - SVD is good at finding global structure, SDD at finding local structure

- Another option is to first compute the $\mathbf{A_k}$ with SVD and then apply SDD to $\mathbf{A_k A_k^T}$ (or $\mathbf{A_k^T A_k}$)
  - SDD finds the objects with similar values

- The results can be visualized using the first 2-3 columns of **U** from SVD and the first layers of the hierarchical clustering from SDD

Part 2: SDD Lessons learned

- SDD finds the local areas of values with uniform magnitude

→ Easier interpretation, ‘orthogonal’ view to SVD

- Finding SDD is hard and requires a heuristic

- Together SVD and SDD provide a strong analysis toolset

Part 2: Independent Component Analysis

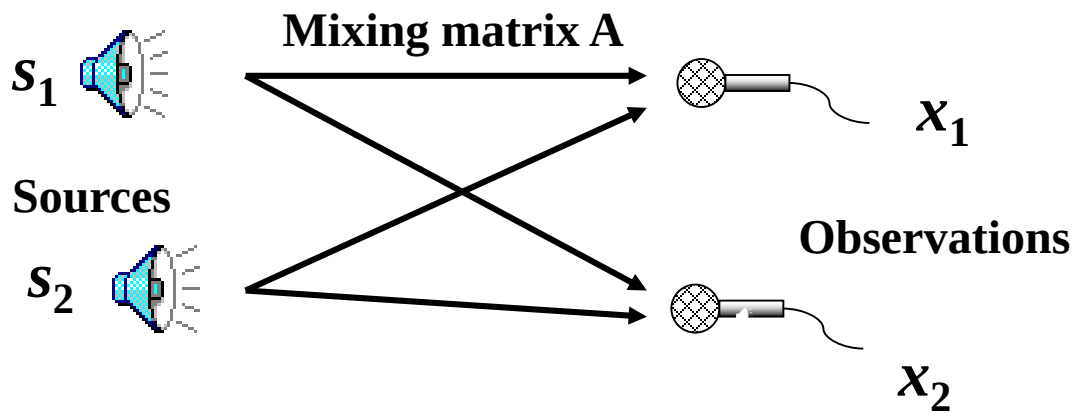
- Consider the cocktail-party problem: in a room, you have two people speaking simultaneously, and two microphones recording the mixture of these speech signals.

- Each recording is a weighted sum of speech signals $s_1(t)$ and $s_2(t)$

$$x_1(t) = a_{11}s_1 + a_{12}s_2$$
$$x_2(t) = a_{21}s_1 + a_{22}s_2$$

- where a_{ij} are some parameters depending on the distances of the microphones from the speakers.

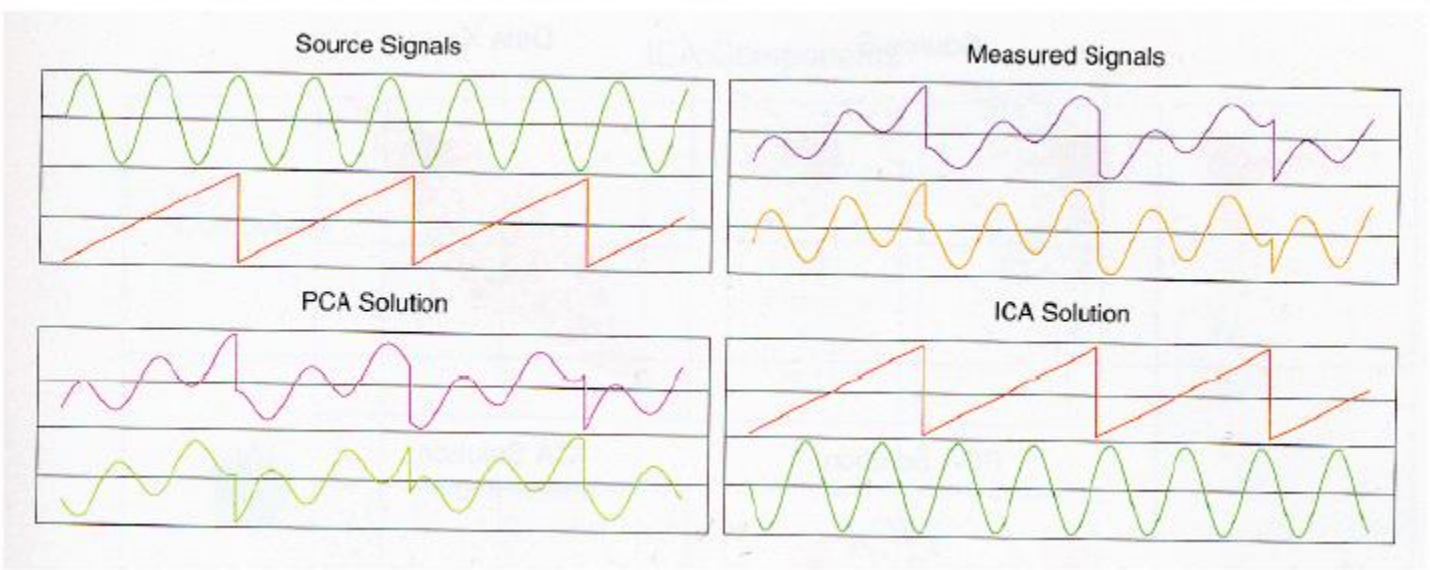
- How to recover the original signals s_1 and s_2 from the recorded signals x_1 and x_2?



x = As

Task: estimate A and s using only the observeable random vector x

Part 2: Independent Component Analysis



- PCA gives uncorrelated components. In the cocktail-party problem this is not the right answer.
- ICA gives statistically independent components.
- Variables y_1 and y_2 are independent, if information on the value of y_1 does not give any information on the value of y_2, and vice versa.
- Note: data must be nongaussian.

Part 2: Example: Image separation. Mixtures of images

Part 2: Example: Image separation. Mixtures of images

ICA produced the following images:

Part 2: Lessons learned

- SVD is the Swiss Army knife of (numerical) linear algebra
  → ranks, kernels, norms, …
- SVD is also very useful in data analysis
  → noise removal, visualization, dimensionality reduction, …
- Selecting the correct rank for truncated SVD is still a problem
- SDD finds the local areas of values with uniform magnitude
  → easier interpretation, ‘orthogonal’ view to SVD
- Finding SDD is hard and requires a heuristic
- Non-negative matrix factorization (NMF) appears natural for non-negative data
- NMF encourages parts-based decomposition, interpretability, and sparseness
- Usually solved via alternating minimization algorithms.

Content

Part 3: Problem definition and motivation

- In many applications (e.g., statistical data analysis and scientific computation), one has n observations of the form:

$$y_i = y(t_i), i = 1, \ldots, n$$

Model y(t) (unknown) as a linear combination of d basis functions:

$$y(t) \approx x_1\phi_1(t) + \cdots + x_d\phi_d(t)$$

A is an n x d "design matrix" (n >> d):

$$A_{ij} = \phi_j(t_i)$$

In matrix-vector notation,

$$y \approx Ax$$

Part 3: Least-norm approximation problems

- Recall a linear measurement model:

$$y = Ax + \varepsilon \quad \begin{cases} y \text{ are the measurements} \\ x \text{ is the unknown} \\ \varepsilon \text{ is an error process} \end{cases}$$

- In order to estimate x, solve:

$$\hat{x} = \arg\min \|y - Ax\|$$

- First application: Astronomy
  Predicting the orbit of the asteroid Ceres (in 1801!).
  Gauss (1809) -- see also Legendre (1805) and Adrain (1808).
  First application of "least squares optimization" and runs in $O(nd^2)$ time!

- Data analysis: Fit parameters of a biological, chemical, economical, physical (astronomical), social, internet, etc. model to experimental data.

Part 3: Norms of common interest

Let y = b and define the residual: $r = Ax - b \in R^n$

Least-squares approximation:

$$\text{minimize: } ||Ax - b||_2^2 = r_1^2 + r_2^2 + \cdots + r_n^2$$

Chebyshev or mini-max approximation:

$$\text{minimize: } ||Ax - b||_\infty = \max\{|r_1|, \ldots, |r_n|\}$$

Sum of absolute residuals approximation:

$$\text{minimize: } ||Ax - b||_1 = |r_1| + |r_2| + \cdots + |r_n|$$

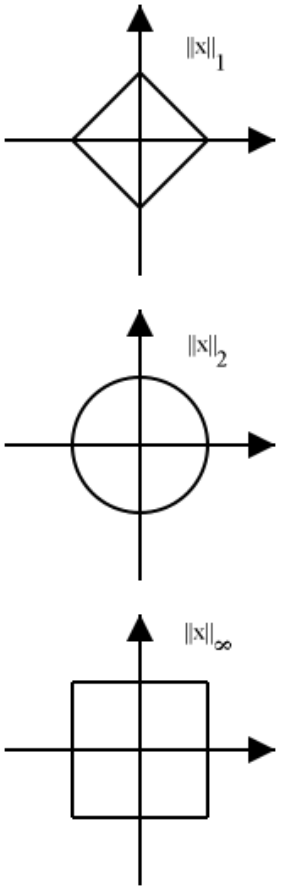
Part 3: L_p norms and their unit balls

Recall the L_p norm for $z \in R^n$

$$||z||_p = \left(\sum_{i=1}^{n} |z_i|^p \right)^{1/p} \quad , p \in [1, \infty)$$

$$||z||_\infty = \max_i |z_i|$$

$$||z||_2^2 = \sum_i z_i^2 = z^T z$$

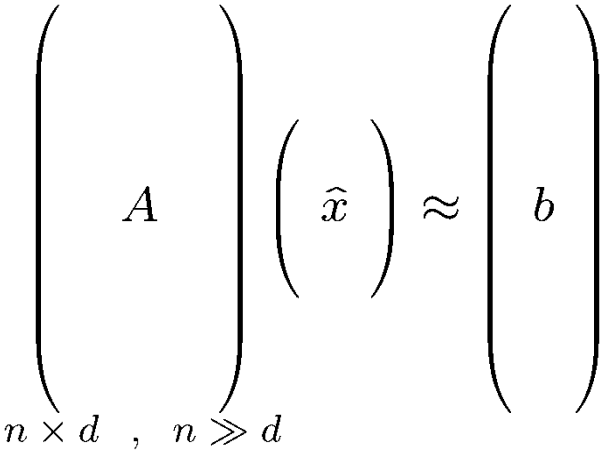


Some inequality relationships include:

$$\frac{1}{\sqrt{n}}||z||_2 \leq ||z||_\infty \leq ||z||_2 \leq ||z||_1 \leq \sqrt{n}||z||_2$$

Part 3: L_p regression problems

$$\begin{aligned} \mathcal{Z}_p &= \min_{x \in R^d} ||b - Ax||_p \\ &= ||b - A\hat{x}||_p \end{aligned}$$



We are interested in over-constrained L_p regression problems, $n >> d$.

- Typically, there is no x such that $Ax = b$.
- Want to find the "best" x such that $Ax \approx b$.

L_p regression problems are convex programs (or better).

- There exist poly-time algorithms.
- We want to solve them faster.

Part 3: Exact solution to L_2 regression

Cholesky Decomposition:

If A is full rank and well-conditioned,

decompose $A^TA = R^TR$, where R is upper triangular, and

solve the normal equations: $R^TRx = A^Tb$.

QR Decomposition:

Slower but numerically stable, esp. if A is rank-deficient.

Write A = QR, and solve $Rx = Q^Tb$.

Singular Value Decomposition:

Most expensive, but best if A is very ill-conditioned.

Write $A = U\Sigma V^T$, in which case: $\mathbf{x_{OPT} = A^+b = V\Sigma^{-1}U^Tb}$.

Complexity is $O(nd^2)$, but constant factors differ.

$$\begin{aligned} \mathcal{Z}_2 &= \min_{x \in R^d} ||b - Ax||_2 \\ &= ||b - A\hat{x}||_2 \end{aligned}$$

Projection of *b* on the subspace spanned by the columns of *A*

$$\begin{aligned} \mathcal{Z}_2^2 &= ||b||_2^2 - ||AA^+b||_2^2 \\ \hat{x} &= A^+b \end{aligned}$$

Pseudoinverse of A

Part 3: Questions ⋯

$$\mathcal{Z}_p = \min_{x \in R^d} ||b - Ax||_p = ||b - A\hat{x}||_p$$

Approximation algorithms:

Can we approximately solve L_p regression faster than “exact” methods?

Core-sets (or induced sub-problems):

Can we find a small set of constraints such that solving the L_p regression on those constraints gives an approximation to the original problem?

Part 3: Lessons learned

- Least square problems **Ax=b** can be solved by
  - Normal equation, $\mathbf{A^TAx=A^tb \;\; --> \;\; R^TRx = A^Tb}$
  - QR decomposition, $\mathbf{Rx = Q^Tb}$
  - Singular value decomposition, $\mathbf{x_{OPT} = A^+b = V\Sigma^{-1}U^Tb}$

- If the matrix size is huge
  - Approximate with random algorithm

Conclusions

- Linear Algebraic techniques (e.g. matrix decompositions and regression) are fundamental in data mining.

- Matrix decompositions reveal structure in the data
  → **D = LMR**

- Many different decompositions with different applications exist
  → QR, SVD, k-means, NMF, SDD, ICA, etc.
- SVD is the Swiss Army knife of (numerical) linear algebra
  → ranks, kernels, norms, …
- SVD is also very useful in data analysis
  → noise removal, visualization, dimensionality reduction, …

- Non-negative matrix factorization (NMF) appears natural for non-negative data
- NMF encourages parts-based decomposition, interpretability, and sparseness

- Randomized algorithms for linear algebra computations contribute novel results and ideas, both from a theoretical as well as an applied perspective.