LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK

LEHRSTUHL FÜR DATENBANKSYSTEME
UND DATA MINING

Lecture Notes to
Big Data Management and Analytics
Winter Term 2017/2018

Text Processing and High-Dimensional Data

© Matthias Schubert, Matthias Renz, Felix Borutta, Evgeniy Faerman, Christian Frey, Klaus Arthur Schmid, Daniyal Kazempour, Julian Busch

DBS

Outline

- Text Processing
  - Motivation
  - Shingling of Documents
  - Similarity-Preserving Summaries of Sets

- High-Dimensional Data
  - Motivation
  - Principal Component Analysis
  - Singular Value Decomposition
  - CUR

Text Processing – Motivation

Given: Set of documents

Searching for patterns in large sets of document objects

→ Analyzing the similarity of objects

In many applications the documents are not identical, yet they share large portions of their text:

- Plagiarism
- Mirror Pages
- Articles from the same source

Problems in the field of Text Mining:

- Stop words (e.g. for, the, is, which ,…)
- Identify word stem
- High dimensional features (d > 10‘000)
- Terms are not equally relevant within a document
- The frequency of terms are often $h_i = 0$ → very sparse feature space

→ We will focus on character-level similarity, not ‚*similar meaning*‘

Text Processing – Motivation

How to handle relevancy of a term?

TF-IDF (Term Frequency * Inverse Document Frequency)

- *Empirical probability of term t in document d:* $\boldsymbol{TF(t,d)} = \frac{n(t,d)}{max_{w \in d} n(w,d)}$

  frequency n(t,d) := number of occurrences of term (word) t in document d

- *Inverse probability of t regarding all documents:* $\mathbf{IDF(t)} = \frac{|DB|}{|\{d|d \in DB \wedge t \in d\}|}$

- *Feature vector is given by:* $r(d) = (TF(t_1,d) * IDF(t_1), \dots, TF(t_n,d) * IDF(t_n))$

How to handle sparsity?

Term frequency often 0 => diversity of mutual Euclidean distances quite low → other distance measures required:

- ***Jaccard Coefficient:*** $d_{Jaccard}(D_1, D_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}$ (Documents → set of terms)
- ***Cosinus Coefficient:*** $d_{Cosinus}(D_1, D_2) = \frac{\langle D_1, D_2 \rangle}{\|D_1\| * \|D_2\|}$ (useful for high-dim. data)

Shingling of Documents

General Idea: construct a set of short strings that appear within a document

***K*- shingles**

Definition: A *k*-shingle is any substring of length *k* found within the document. (aka k-grams)

→ Associate with each document the set of *k*-shingles that appear n times within that document

Hashing Shingles:

Idea: pick hash function that maps strings of length *k* to some number of buckets and treat the resulting bucket number as the shingle

→ set representing document is then set of integers

Similarity-Preserving Summaries of Sets

Problem: Sets of shingles are large
→ replace large sets by much smaller representations called ‚*signatures*‘

Matrix representation of Sets
Characteristic matrix:

- columns correspond to the sets (documents)
- rows correspond to elements of the universal set from which elements (shingles) of the columns are drawn

Example:

- universal set: {A,B,C,D,E},
- S1 = {A,D}, S2 = {C}, S3={B,D,E}, S4={A,C,D}

documents

shingles

Element	S1	S2	S3	S4
A	1	0	0	1
B	0	0	1	0
C	0	1	0	1
D	1	0	1	1
E	0	0	1	0

Similarity-Preserving Summaries of Sets

Minhashing

Idea: To minhash a set represented by a column c_i of the characteristic matrix, pick a permutation of the rows. The value of the minhash is the number of the first row, in the permutated order, with $h(c_i) = 1$

Example:
Suppose the order of rows ‚BEADC'

- h(S1) = A
- h(S2) = C
- h(S3) = B
- h(S4) = A

Element	S1	S2	S3	S4
B	0	0	1	0
E	0	0	1	0
A	1	0	0	1
D	1	0	1	1
C	0	1	0	1

Similarity-Preserving Summaries of Sets

Minhashing and Jaccard Similarity

The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets.

Three different classes of similarity between sets (documents)

- Type X rows have 1 in both cols
- Type Y rows have 1 in one of the columns
- Type Z rows have 0 in both rows

Example

Considering the cols of S1 and S3:
The probability that h(S1) = h(S3) is given by:

$$SIM(S1, S3) = \frac{x}{(x+y)} = \frac{1}{4}$$

(Note that x is the size of $S1 \cap S2$ and (x+y) is the size of $S1 \cup S2$)

Element	S1	S2	S3	S4
B	0	0	1	0
E	0	0	1	0
A	1	0	0	1
D	1	0	1	1
C	0	1	0	1

Similarity-Preserving Summaries of Sets

Minhash Signatures

- Pick a random number n of permutations of the rows
- Vector $[h_1(S), h_2(S), \ldots, h_n(S)]$ represents the minhash signature for S
- Put the specific vectors together in a matrix, forms the *signature matrix*
- Note that the *signature matrix* has the same number of columns as input matrix M but only n rows

How to compute minhash signatures:

1. Compute $h_1(S), h_2(S), \ldots, h_n(S)$
2. For each row r: For each column c do the following:
   (a) if c has 0 in row r, do nothing
   (b) if c has 1 in row r then for each $i = 1, 2, \ldots, n$ set
   $SIG(i,c) = \min(SIG(i,c), h_i(r))$

→ Signature matrix allows to estimate the Jaccard similarities of the underlying sets!

Similarity-Preserving Summaries of Sets

Minhash Signatures - Example

- Suppose two hash functions : $h_1(x) = (x+1)\ mod\ 5$ and $h_2(x)\ = (3x+1)\ mod\ 5$

Element	S1	S2	S3	S4	h1(x)	h2(x)
0	**1**	**0**	**0**	**1**	**1**	**1**
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

initialization

1.	*s1*	*s2*	*s3*	*s4*
h1	∞	∞	∞	∞
h2	∞	∞	∞	∞

2.	*s1*	*s2*	*s3*	*s4*
h1	1	∞	∞	1
h2	1	∞	∞	1

1st row

Check Sig for S1 and S4:

$$SIG(i,c)\ =\ \min(SIG(i,c)\ ,h_i(r))$$

S1: $\mathbf{\min(\infty, 1) = 1}$

$\mathbf{\min(\infty, 1) = 1}$

S4: $\boldsymbol{min(\infty, 1) = 1}$

$\boldsymbol{min(\infty, 1) = 1}$

Similarity-Preserving Summaries of Sets

Minhash Signatures - Example

- Suppose two hash functions : $h_1(x) = x + 1 \, mod \, 5$ and $h_2(x) = (3x + 1) \bmod 5$

Element	S1	S2	S3	S4	h1(x)	h2(x)
0	1	0	0	1	1	1
1	**0**	**0**	**1**	**0**	**2**	**4**
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

2nd row
Check Sig for S3:
$SIG(i, c) = \min(SIG(i, c), h_i(r))$

S3: $\mathbf{\min(\infty, 2) = 2}$
$\mathbf{\min(\infty, 4) = 4}$

initialization

1.	***s1***	***s2***	***s3***	***s4***
h1	∞	∞	∞	∞
h2	∞	∞	∞	∞

2.	***s1***	***s2***	***s3***	***s4***
h1	***1***	∞	∞	***1***
h2	***1***	∞	∞	***1***

3.	***s1***	***s2***	***s3***	***s4***
h1	1	∞	2	1
h2	1	∞	4	1

Similarity-Preserving Summaries of Sets

Minhash Signatures - Example

- Suppose two hash functions : $h_1(x) = x + 1 mod\ 5$ and $h_2(x) = (3x + 1) mod 5$

Element	S1	S2	S3	S4	h1(x)	h2(x)
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	**0**	**1**	**0**	**1**	**3**	**2**
3	1	0	1	1	4	0
4	0	0	1	0	0	3

3rd row

Check Sig for S2 and S4:

$SIG(i,c) = \min(SIG(i,c), h_i(r))$

S2: $\min(\infty, 3) = 3$

$\min(\infty, 2) = 2$

S4: $\min(1,3) = 1$

$\min(1,2) = 1$

initialization

1.	*s1*	*s2*	*s3*	*s4*
h1	∞	∞	∞	∞
h2	∞	∞	∞	∞

2.	*s1*	*s2*	*s3*	*s4*
h1	***1***	∞	∞	***1***
h2	***1***	∞	∞	***1***

3.	*s1*	*s2*	*s3*	*s4*
h1	1	∞	2	1
h2	1	∞	4	1

4.	*s1*	*s2*	*s3*	*s4*
h1	1	3	2	1
h2	1	2	4	1

Similarity-Preserving Summaries of Sets

Minhash Signatures - Example

- Suppose two hash functions : $h_1(x) = x + 1 mod\ 5$ and $h_2(x) = (3x + 1) mod 5$

Element	S1	S2	S3	S4	h1(x)	h2(x)
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	**1**	**0**	**1**	**1**	**4**	**0**
4	0	0	1	0	0	3

4th row
Check Sig for S1,S3,S4:
$SIG(i,c) = \min(SIG(i,c), h_i(r))$

S1: $\min(1,4) = 1$
$\boldsymbol{min(1,0) = 0}$

S3: $\min(2,4) = 2$
$\boldsymbol{min(4,0) = 0}$

S4: $\min(1,4) = 1$
$\boldsymbol{min(1,0) = 0}$

initialization

1.	s1	s2	s3	s4
h1	∞	∞	∞	∞
h2	∞	∞	∞	∞

2.	s1	s2	s3	s4
h1	*1*	∞	∞	*1*
h2	*1*	∞	∞	*1*

3.	s1	s2	s3	s4
h1	1	∞	2	1
h2	1	∞	4	1

4.	s1	s2	s3	s4
h1	1	3	2	1
h2	**1**	2	**4**	1

5.	s1	s2	s3	s4
h1	1	3	2	1
h2	**0**	2	**0**	**0**

Similarity-Preserving Summaries of Sets

Minhash Signatures - Example

- Suppose two hash functions : $h_1(x) = x + 1 mod\ 5$ and $h_2(x) = (3x + 1) mod 5$

Element	S1	S2	S3	S4	h1(x)	h2(x)
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	**0**	**0**	**1**	**0**	**0**	**3**

5th row
Check Sig for S3:
$SIG(i, c) = \min(SIG(i, c), h_i(r))$

S3: $\mathbf{\min(2, 0) = 0}$
$\min(0,3) = 0$

initialization

1.	*s1*	*s2*	*s3*	*s4*
h1	∞	∞	∞	∞
h2	∞	∞	∞	∞

2.	*s1*	*s2*	*s3*	*s4*
h1	***1***	∞	∞	***1***
h2	***1***	∞	∞	***1***

3.	*s1*	*s2*	*s3*	*s4*
h1	1	∞	2	1
h2	1	∞	4	1

4.	*s1*	*s2*	*s3*	*s4*
h1	1	3	2	1
h2	1	2	4	1

5.	*s1*	*s2*	*s3*	*s4*
h1	1	3	2	1
h2	0	2	0	1

6.	*s1*	*s2*	*s3*	*s4*
h1	1	3	**0**	1
h2	0	2	0	0

Modeling data as matrices

Matrices often arise with data:

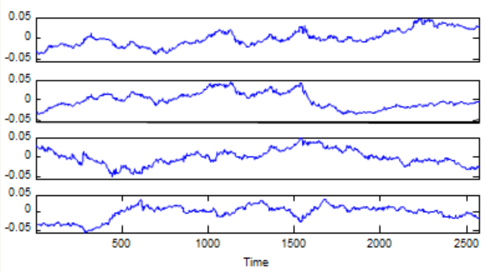
- $\boldsymbol{n}$ objects (documents, images, web pages, time series…)
- each with $\boldsymbol{m}$ features

→ Can be represented by an $\boldsymbol{n\ x\ m}$ matrix

doc1	Two for wine and wine for two
doc2	Wine for me and wine for you
doc3	You for me and me for you

$$TDM := \begin{array}{cc} \begin{matrix}\text{Two} & \text{wine} & \text{Me} & \text{you}\end{matrix} & \\ \begin{pmatrix} 2 & 2 & 0 & 0 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 2 \end{pmatrix} & \begin{matrix}\text{Doc1} \\ \text{Doc2} \\ \text{Doc3}\end{matrix} \end{array}$$

(filter ‚for‘, ‚and‘ as stopwords)



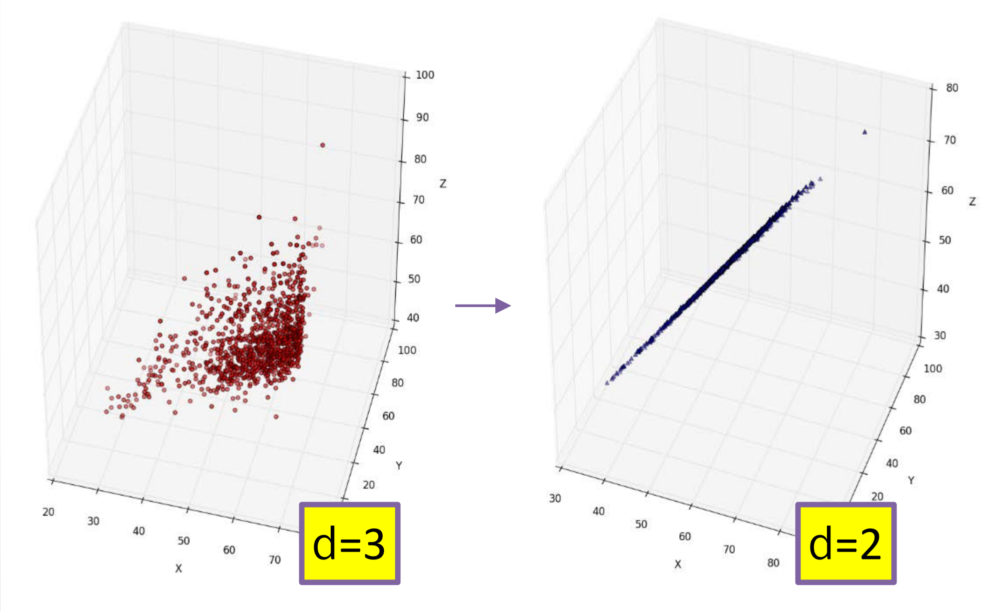
i-th series, $x^{(i)}$

$$X_{NxM} := \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(M)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(M)} \\ \vdots & \vdots & \ddots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \dots & x_N^{(M)} \end{pmatrix}$$

values at time t, x_t

Why reducing the dimesionality makes sense?

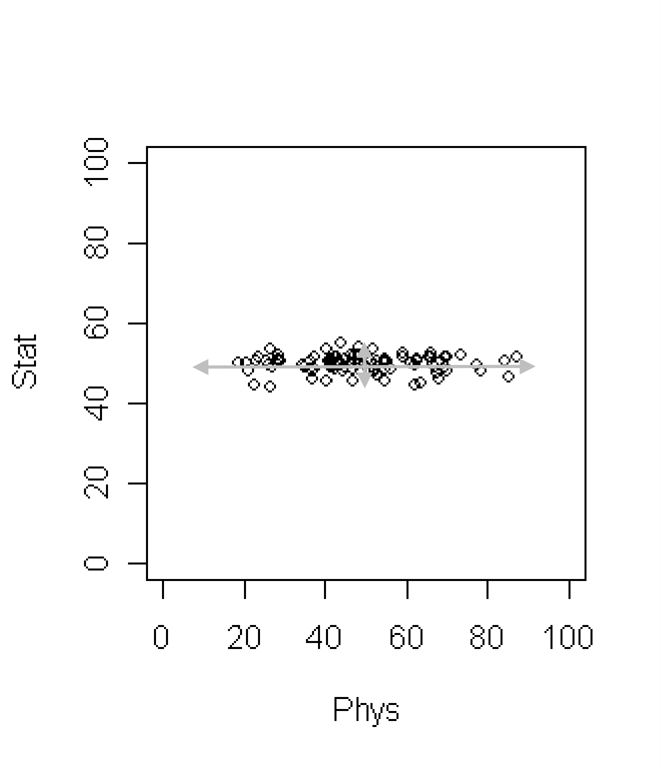
- discover hidden correlations
- remove redundant and noisy features
- interpretation and visualization
- easier storage and processing of the data
- transform a high-dimensional sparse matrix into a low-dimensional dense matrix



Axes of k-dimensional subspace are effective representation of the data

Principal Component Analysis (PCA): A simple example 1/3

- Consider the grades of students in Physics and Statistics.
- If we want to compare among the students, which grade should be more discriminative? Statistics or Physics?

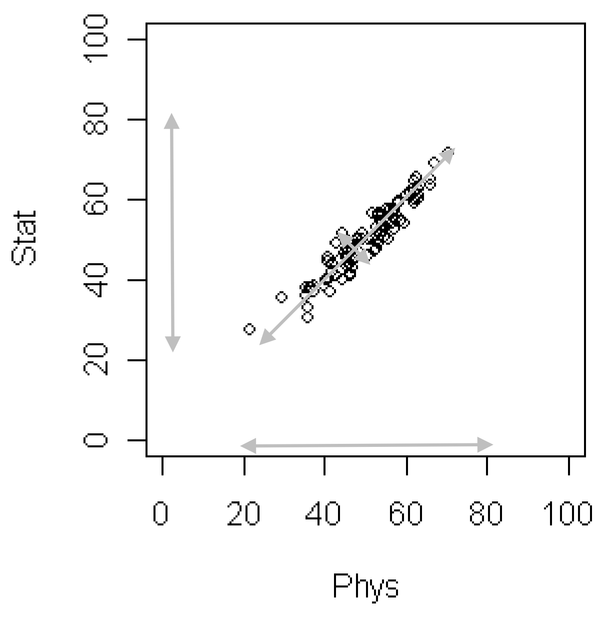


Physics since the variation along that axis is larger.

Based on: http://astrostatistics.psu.edu/su09/lecturenotes/pca.html

Principal Component Analysis (PCA): A simple example 2/3

- Suppose now the plot looks as below.
- What is the best way to compare students now?



We should take linear combination of the two grades to get the best results.

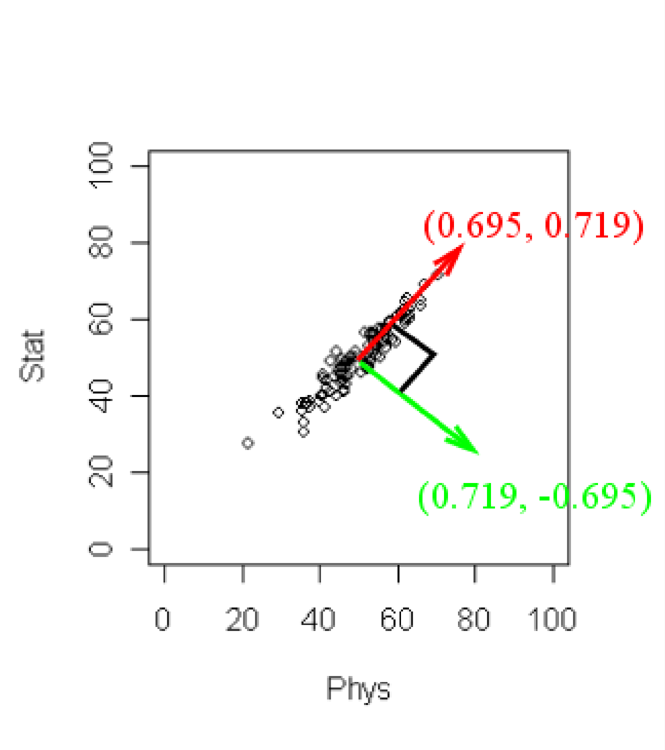
Here the direction of maximum variance is clear.

In general → PCA

Based on:
http://astrostatistics.psu.edu/su09/lecturenotes/pca.html

Principal Component Analysis (PCA): A simple example 3/3

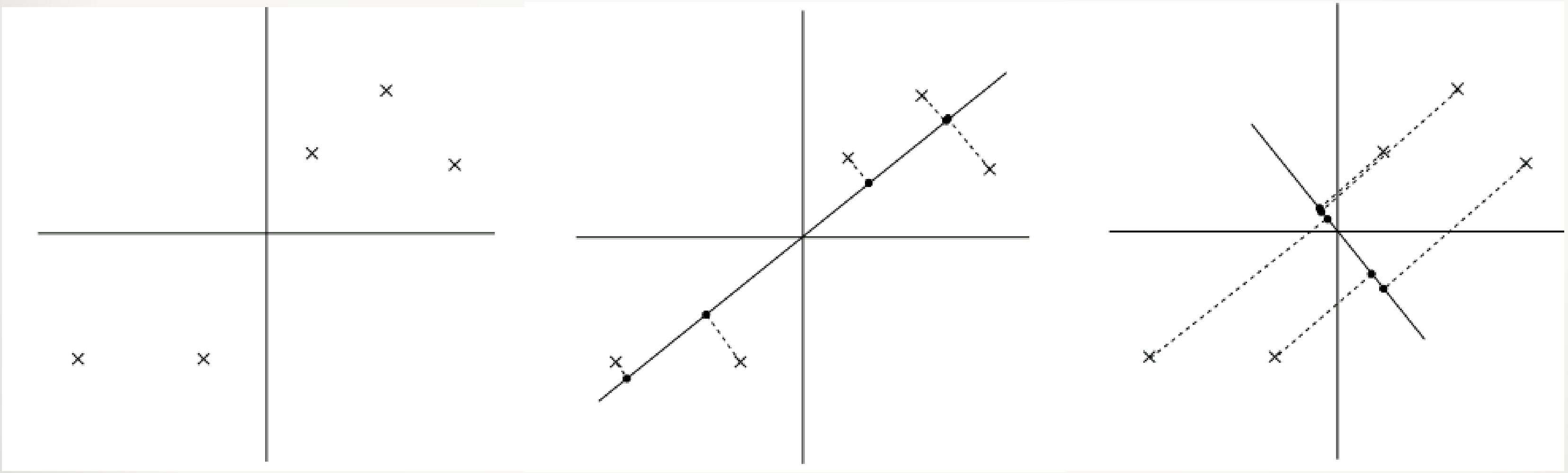
- PCA returns two principal components
  - The first gives the direction of the maximum spread of the data.
  - The second gives the direction of maximum spread perpendicular to



Based on:
http://astrostatistics.psu.edu/su09/lecturenotes/pca.html

Intuition

- to represent information, data objects has to be distinguishable
- if all objects have the same attribute value (+ noise), objects are not different from each other
- maximize the diversity between the objects
- the variance in a direction describes this diversity



Initial data | Direction 1 | Direction 2

Idea: Always choose the direction that maximizes the variance of the projected data

Principal Component Analysis (PCA)

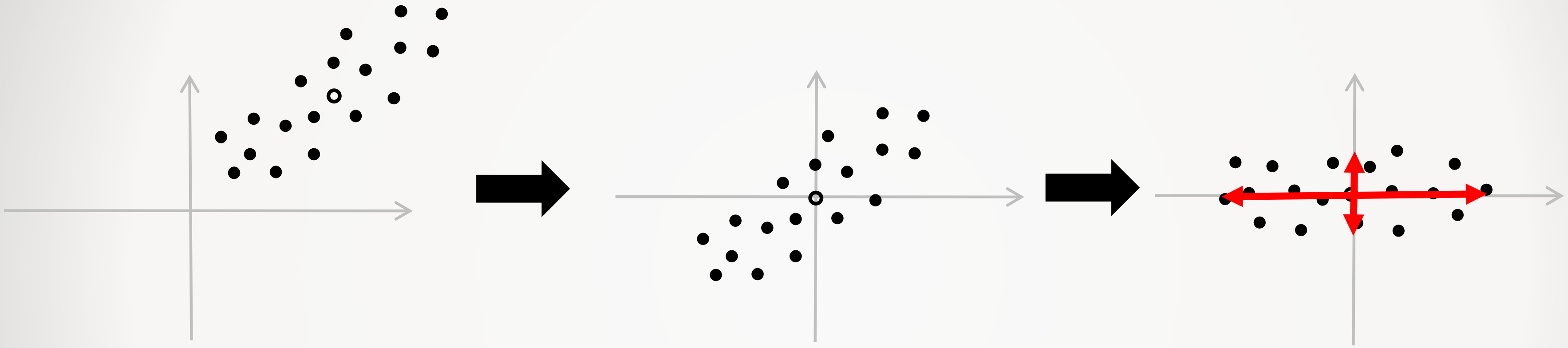
- PCA computes the most meaningful basis to re-express noisy data
- Think of PCA as choosing a new coordinate system for the data, the principal components being the unit vectors along the axes
- PCA asks: *Is there another basis, which is a linear combination of the original basis, that best expresses our dataset?*
- General form: $PX=Y$

  where P is a linear transformation, X is the original dataset and Y the re-representation of this dataset.

  - P is a matrix that transforms X into Y
  - Geometrically, P is a *rotation* and a *stretch* which again transforms X into Y
  - The eigenvectors are the rotations to the new axes
  - The eigenvalues are the amount of stretching that needs to be done
- The p's are the principal components
  - Directions with the largest variance … those are the most important, most *principal.*

Principal Component Analysis (PCA)

Idea: Rotate the data space in a way that the principal components are placed along the main axis of the data space
=> Variance analysis based on principal components



- Rotate the data space in a way that the direction with the largest variance is placed on an axis of the data space
- Rotation is equivalent to a basis transformation by an orthonormal basis
  - Mapping is equal of angle and preserves distances:

$$x \cdot B = x\left(b_{*,1},\dots,b_{*,d}\right) = \left(\langle x, b_{*,1}\rangle,\dots,\langle x, b_{*,d}\rangle\right) \quad mit \quad \underset{i \neq j}{\forall} \langle b_i, b_j \rangle = 0 \wedge \underset{1 \le i \le d}{\forall} \|b_i\| = 1$$

- B is built from the largest variant direction which is orthogonal to all previously selected vectors in B.

PCA steps

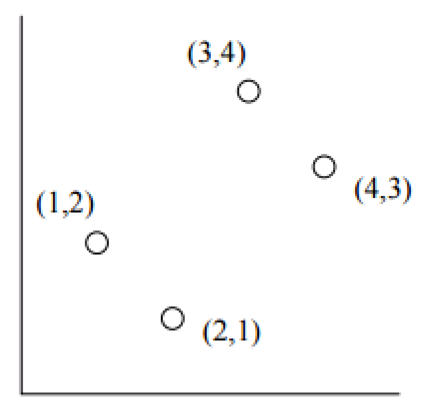
Feature reduction using PCA

1. Compute the covariance matrix Σ
2. Compute the eigenvalues and the corresponding eigenvectors of Σ
3. Select the *k* biggest eigenvalues and their eigenvectors (V‘)
4. The *k* selected eigenvectors represent an orthogonal basis
5. Transform the original $n \times d$ data matrix D with the $d \times k$ basis V‘:

$$D \cdot \mathrm{V}' = \begin{pmatrix} \mathrm{x}_1 \\ \vdots \\ \mathrm{x}_\mathrm{n} \end{pmatrix} \left(v_1', \cdots, v_k' \right) = \begin{pmatrix} \langle \mathrm{x}_1, v_1' \rangle & \cdots & \langle \mathrm{x}_1, v_k' \rangle \\ \vdots & \ddots & \vdots \\ \langle \mathrm{x}_\mathrm{n}, v_1' \rangle & \cdots & \langle \mathrm{x}_\mathrm{n}, v_k' \rangle \end{pmatrix}$$

Example of transformation

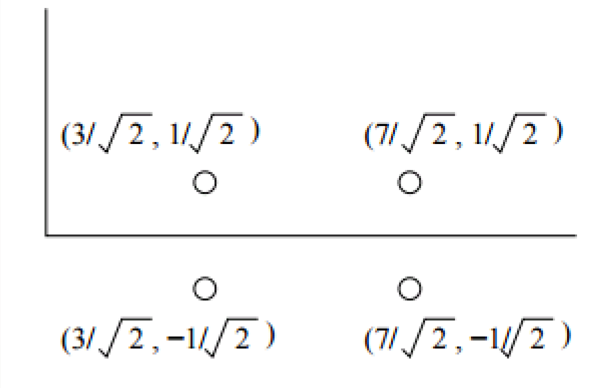
- Original



Eigenvectors

$$\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \qquad \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

In the rotated coordinate system



- Transformed data

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 1/\sqrt{2} \\ 3/\sqrt{2} & -1/\sqrt{2} \\ 7/\sqrt{2} & 1/\sqrt{2} \\ 7/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

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

Percentage of variance explained by PCA

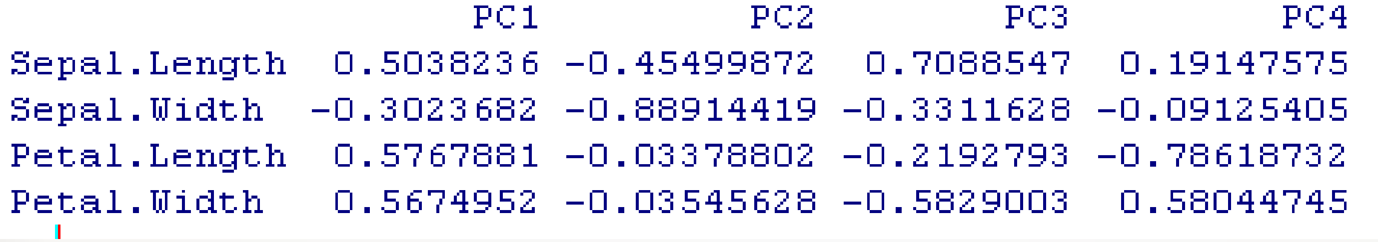
- Let *k* be the number of top eigenvalues out of *d* (*d* is the number of dimensions in our dataset)
- The percentage of variance in the dataset explained by the *k* selected eigenvalues is:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

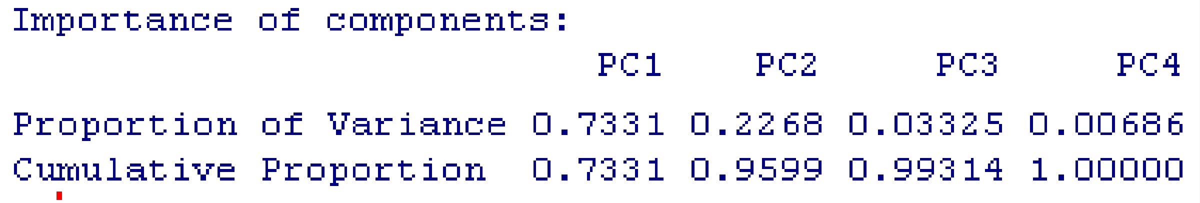
- Similarly, you can find the variance explained by each principal component
- Rule of thumb: keep enough to explain **85%** of the variation

PCA results interpretation

- Example: iris dataset (d=4), results from R
- 4 principal components



```
                    PC1         PC2        PC3         PC4
Sepal.Length  0.5038236 -0.45499872  0.7088547  0.19147575
Sepal.Width  -0.3023682 -0.88914419 -0.3311628 -0.09125405
Petal.Length  0.5767881 -0.03378802 -0.2192793 -0.78618732
Petal.Width   0.5674952 -0.03545628 -0.5829003  0.58044745
```



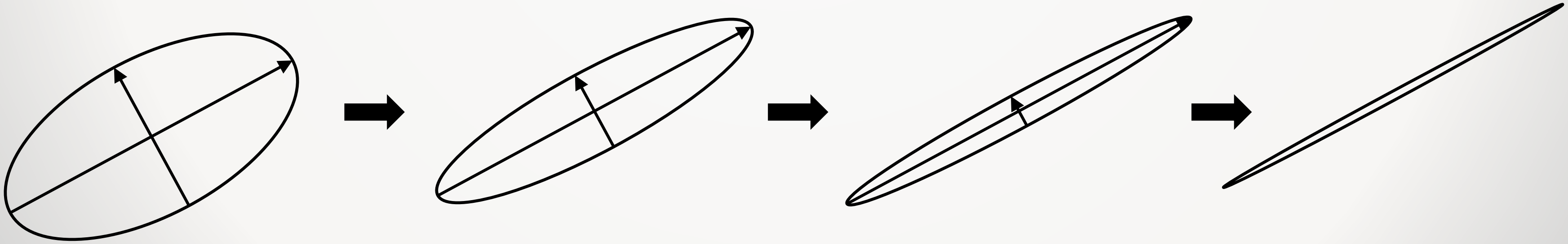
```
Importance of components:
                          PC1    PC2     PC3     PC4

Proportion of Variance 0.7331 0.2268 0.03325 0.00686
Cumulative Proportion  0.7331 0.9599 0.99314 1.00000
```

Computing PCA via Power Iteration

Problem:

- Computing the eigenvalues with standard algorithms is often expensive (many algorithm are well-known)
- Standard methods often involve matrix inversions ($O(n^3)$)
- For large matrixes more efficient methods are required:
- Most prominent is the power iterations method ($O(n^2)$)

Intuition: Multiplying a matrix with itself increases the strongest direction relative to the other direction.



Power Iterations general idea

- given: data n×d matrix X and the corresponding covariance matrix $\Sigma=(X-\mu(X)^T(X-\mu(X))$ where $\mu(X)$ is the mean vector of X.
- consider the eigenvalue decomposition of $\Sigma = V^T \Lambda V$ where $V = (v_1, \ldots, v_d)$: is the column wise orthonormal eigenvector basis

$\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}$: is the diagonal eigenvalue matrix

Note: $\Sigma^t = (V^T \Lambda V)^t = V^T \Lambda V \cdot V^T \Lambda V \cdot \ldots \cdot V^T \Lambda V = V^T \Lambda^t V$

$$= V^T \begin{bmatrix} \lambda_1^t & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d^t \end{bmatrix} V$$

What is the *i*th power of a diagonal matrix ?

- if PCA is well-defined all $\lambda >= 0$
- taking the i^{th} power: All values $\lambda > 1$ increase with the power and all λ values < 0 decrease exponentially fast.
- When normalizing the λ by $\sum_{i=1}^{d} \lambda_i$, we observe the following: for $\lambda_i \neq \lambda_j$ and $t \rightarrow \infty$: $\exists \lambda_{i*}: \frac{\lambda_{i*}^t}{\sum_{i=1}^{d} \lambda_i^t} \rightarrow 1$ and $\forall j \neq i*: \frac{\lambda_j^t}{\sum_{i=1}^{d} \lambda_i^t} \rightarrow 0$
- under normalization over all diagonal entries, only one component remains.
- Thus: the rank of Σ^t converges to 1 and the only component remaining is the strongest eigenvector.

Determining the Strongest Eigenvalue

The following algorithm computes the strongest eigenvalue of matrix M:

```
Input: d×d data matrix M
$x_0$ = random unit vector
$M_0$ = M
while || $x_i/||x_i|| - x_{i-1}/||x_{i-1}||$ ||> ε do
    $M_{i+1} = M^T_i M_i$
    $x_i = M_{i+1} x_0$
    i=i+1
return $x_i/||x_i||$
```

Why does this work?

$$M^t x = [v_1 , \ldots, v_d] \begin{bmatrix} 0 & .. & 0 \\ .. & \lambda_j^t & .. \\ 0 & .. & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} x = [v_1 , \ldots, v_d] \begin{bmatrix} 0 & .. & 0 \\ .. & \lambda_j^t & .. \\ 0 & .. & 0 \end{bmatrix} \begin{bmatrix} \langle v_1, x \rangle \\ \vdots \\ \langle v_d, x \rangle \end{bmatrix}$$

$$= [v_1, .., v_d] \begin{bmatrix} 0 \\ \lambda_j^t \langle v_j, x \rangle \\ 0 \end{bmatrix} = \begin{bmatrix} v_{1,1} \cdot 0 + .. + v_{1,j} \cdot \lambda_j^t \langle v_j, x \rangle + v_{1,d} \cdot 0 \\ \vdots \\ v_{d,1} \cdot 0 + .. + v_{d,j} \cdot \lambda_j^t \langle v_j, x \rangle + v_{d,d} \cdot 0 \end{bmatrix} = v_j \cdot \lambda_j^t \langle v_j, x \rangle$$

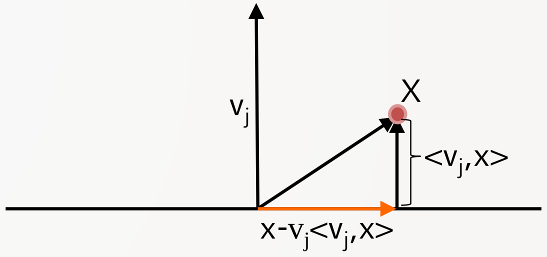
in other words the $M^T x$ has the same direction as the strongest eigenvector v_j.

Power Iterations: the complete method

- we now have a method to determine the strongest eigenvector
- to compute the k-strongest eigenvectors we proceed as follows:

```
For i=1 to k:
  determine the strongest eigenvector v_i
  reproject data X to the space being orthogonal to v_i:
  x’ = x-v_i<v_i,x>
output the v_i
```

- explanation for the reprojection:



- if there are two equally strong eigenvalues $\lambda_i = \lambda_j$ then the algorithm returns an arbitrary vector from $span(v_i, v_j)$
- for $\lambda_i \approx \lambda_j$: the algorithm converges slower

Conclusion

- PCA is an important method for feature reduction
- general and complete for eigenvalue decomposition are often inefficient(compute the characteristic polynomial, using matrix inversion etc.)
- Power iterations are linear in the size of the matrix, i.e. quadratic in the dimension d.
- Power iterations compute only the k strongest eigenvalues but not all (stop when k strongest v are found)
- rely only on matrix multiplications

Singular Value Decomposition (SVD) – Generalization of the eigenvalue decomposition

Let $X_{n \times d}$ be a data matrix and let k be its rank. We can decompose X into matrices U, Σ, V as follows:

$$\begin{array}{ccccccc} \boldsymbol{X} & & \boldsymbol{U} & & \boldsymbol{\Sigma} & & \boldsymbol{V^T} \\ \begin{pmatrix} x_{1,1} & \dots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,d} \end{pmatrix} & = & \begin{pmatrix} u_{1,1} & \dots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,n} \end{pmatrix} & * & \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{pmatrix} & * & \begin{pmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{d,1} & \cdots & v_{d,d} \end{pmatrix} \end{array}$$

- **X (Input data matrix)** is a $n \times d$ matrix (e.g. n customers, d products)
- **U (Left singular vectors)** is a $n \times n$ column-orthonormal matrix
- ***Σ* (Singular values)** is a diagonal $n \times d$ with the elements being the singular values of X
- **V (Right singular vecors)** is a $d \times d$ column-orthonormal matrix

Singular Value Decomposition (SVD)

Computing SVD of a Matrix

Connected to eingenvalues of matrices $X^T X$ and $X X^T$

$$X^T X = (U \Sigma V^T)^T U \Sigma V^T = (V^T)^T \Sigma^T U^T U \Sigma V^T = V \Sigma^2 V^T$$

→ Multiplying each side with V:

$$(X^T X) V = V \Sigma^2$$

remember the eigenvalue problem:
$Av = \lambda v$

→ Same algorithm that computes the *eigenpairs* for $X^T X$ gives us matrix V for SVD

→ Square root of the eigenvalues of $X^T X$ gives us the singular values of X

→ U can be found by the same procedure as V, just with XX^T

Singular Value Decomposition (SVD)

How to reduce the dimensions?

Let X= $U\,\Sigma\,V^T$ (with rank(A) = r) and Y = $U\,S\,V^T$, with $S\ \in\ \mathbb{R}^{r\,x\,r}$ where $s_i = \lambda_i\ (i = 1\,,\dots,k)$ else $s_i = 0$

$$\begin{pmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{pmatrix} = \begin{pmatrix} u_{1,1} & \cdots & u_{1,r} & u_{1,n} \\ \vdots & \ddots & \vdots & \vdots \\ u_{n,1} & \cdots & u_{n,r} & u_{n,n} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & \cdots \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \cdots & \lambda_r & \vdots \\ \vdots & \dots & \dots & \lambda_d \end{pmatrix} \begin{pmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{r,1} & \cdots & v_{r,d} \\ v_{d,1} & \cdots & v_{d,d} \end{pmatrix}$$

→ New matrix Y is a **best rank-k approximation to X**

Singular Value Decomposition (SVD) – Example

Ratings of movies by users

	Matrix	Alien	Star Wars	Cassablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Let A be a m × n matrix, and let r be the rank of A

Here:

- a rank-2 matrix representing ratings of movies by users
- 2 underlying concepts: science-fiction + romance

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

Singular Value Decomposition (SVD) – Example

Ratings of movies by users - SVD

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{pmatrix} = \begin{pmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .6 \\ 0 & .75 \\ 0 & .30 \end{pmatrix} * \begin{pmatrix} 12.4 & 0 \\ 0 & 9.5 \end{pmatrix} * \begin{pmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{pmatrix}$$

$$X = U * \Sigma * V^T$$

X: Raw data of user -movie-ratings

U: Connects people to ‚concepts‘

Σ: ‚strength‘ of each concept

V^T: Relates movies to concepts

Singular Value Decomposition (SVD) – Example

Ratings of movies by users - SVD Interpretation

$$\begin{array}{r}\text{Joe}\longrightarrow\\ \\ \\ \\ \\ \\ \\ \end{array}\begin{pmatrix}1&1&1&0&0\\3&3&3&0&0\\4&4&4&0&0\\5&5&5&0&0\\0&0&0&4&4\\0&0&0&5&5\\0&0&0&2&2\end{pmatrix}=\underbrace{\overset{\text{Sci-Fi concept}\quad\text{romance concept}}{\begin{pmatrix}.14&0\\.42&0\\.56&0\\.70&0\\0&.6\\0&.75\\0&.30\end{pmatrix}}}*\underbrace{\begin{pmatrix}12.4&0\\0&9.5\end{pmatrix}}*\underbrace{\begin{pmatrix}.58&.58&.58&0&0\\0&0&0&.71&.71\end{pmatrix}}$$

People's preferences to specific concepts (e.g. Joe exclusively likes sci-fi movies but rates them low)

Data provides more Information about the sci-fi genre and the people who like it

First three movies (Matrix, Alien, Star Wars) are assigned exclusively to sci-fi genre, whereas the other two belong to the romance ‚concept'

SVD and low-rank approximations

Summary

Basic SVD Theorem: Let A be an m x n matrix with rank p

- Matrix A can be expressed as $A = U \Sigma V^T$
- Truncate SVD of A yields 'best' rank-k approximation given by $A_k = U_k \Sigma_k V_k^T$,with $k < d$

Properties of truncated SVD:

- Often used in data analysis via PCA
- Problematic w.r.t sparsity, interpretability, etc.

Problems with SVD / Eigen-analysis

Problems: arise since structure in the data is not respected by mathematical operations on the data

Question: Is there a 'better' low-rank matrix approximations in the sense of …

- … **structural properties** for certain application
- … respecting **relevant structure**
- … **interpretability** and **informing intuition**

→ **Alternative: CX and CUR matrix decompositions**

CX and CUR matrix decompositions

Definition CX : A CX decomposition is a low-rank approximation explicitly expressed in terms of a small number of *columns of A.*

Definition CUR : A CUR matrix decomposition is a low-rank approximation explicitly expressed in terms of a small number of _**c**olumns and **r**ows of A._

$$A \approx C * U * R$$

CUR Decomposition

- *In large-data aplications the raw data matrix M tend to be very sparse (e.g. matrix of customers/products , movie recommendation systems…)*

- *Problem with SVD :*
  - *Even if M is sparse, the SVD yields two dense matrices U and V*

- *Idea of CUR Decomposition:*
  - *By sampling a sparse Matrix M, we create two sparse matrices C (‘columns’) and R (‘rows’)*

CUR Definition

*Input: let $\boldsymbol{M}$ be a $\boldsymbol{m\ x\ n}$ **matrix***

1.Step:

- *Choose a number $\boldsymbol{r}$ of ‘concepts’ (c.f. rank of matrix)*
    - *Perform biased Sampling of $\boldsymbol{r}$ cols from $\boldsymbol{M}$ and create a $\boldsymbol{m}\times\boldsymbol{r}$ **matrix** $\boldsymbol{C}$*
    - *Perform biased Sampling of $\boldsymbol{r}$ rows from $\boldsymbol{M}$ and create a $\boldsymbol{r}\times\boldsymbol{n}$ **matrix** $\boldsymbol{R}$*

2.Step:

- *Construct $\boldsymbol{U}$ from $\boldsymbol{C}$ and $\boldsymbol{R}$:*
    - *Create a $\boldsymbol{r\ x\ r}$ **matrix** $\boldsymbol{W}$ by the intersection of the chosen cols from C and rows from R*
    - *Apply SVD on $\boldsymbol{W} = \boldsymbol{X\ \Sigma\ Y^t}$*
    - *Compute $\boldsymbol{\Sigma^+}$, the moore-penrose pseudoinverse of Σ*
    - *Compute $\boldsymbol{U} = \boldsymbol{Y}(\boldsymbol{\Sigma^+})^2\ \boldsymbol{X^t}$*

CUR – how to sample rows and cols from M?

Sample columns for C:

```
**Input:** matrix $M \in \mathbb{R}^{m\,x\,n}$ , sample size $r$
**Output:** $C \in \mathbb{R}^{m\,x\,r}$
For x = 1 : n do
        P(x) = $\sum_i (m_{i,x})^2$ / $\|M\|_F^2$
  For y = 1 : r do
        Pick $z \in 1{:}n$ based on Prob(x)
        C( : , y) = M( : , z) / $\sqrt{r * P(z)}$
```

Frobenius-Norm:

$$\|M\|_F = \sqrt{\sum_i \sum_j (m_{i,j})^2}$$

(sampling of R for rows analogous)

CUR Definition

Example - Sampling

$$\begin{array}{l|ccccc|}
 & \text{Matrix} & \text{Alien} & \text{Star Wars} & \text{Cassablanca} & \text{Titanic} \\
\hline
\text{Joe} & 1 & 1 & 1 & 0 & 0 \\
\text{Jim} & 3 & 3 & 3 & 0 & 0 \\
\text{John} & 4 & 4 & 4 & 0 & 0 \\
\text{Jack} & 5 & 5 & 5 & 0 & 0 \\
\text{Jill} & 0 & 0 & 0 & 4 & 4 \\
\text{Jenny} & 0 & 0 & 0 & 5 & 5 \\
\text{Jane} & 0 & 0 & 0 & 2 & 2 \\
\hline
\end{array}$$

Sample columns:

$$\sum_i m_{i,1} = \sum_i m_{i,2} = \sum_i m_{i,3} = 1^2 + 3^2 + 4^2 + 5^2 = 51$$

$$\sum_i m_{i,4} = \sum_i m_{i,5} = 4^2 + 5^2 + 2^2 = 45$$

$$FrobeniusNorm: \|M\|_F^2 = 243$$

$\Rightarrow P(x_1) = P(x_2) = P(x_3) = \frac{51}{243} = 0.210$

$\Rightarrow P(x_4) = P(x_5) = \frac{45}{243} = 0.185$

CUR Definition

Example - Sampling

	Matrix	Alien	Star Wars	Cassablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Sample columns:

- Let r = 2
- Randomly choosen columns, e.g. Star Wars + Cassablanca

$$[1,3,4,5,0,0,0]^T \frac{1}{\sqrt{r * P(x_3)}} = [1,3,4,5,0,0,0]^T \frac{1}{\sqrt{2 * 0.210}}$$
$$= [1.54, 4.63, 6.17, 7.72, 0, 0,0]^T$$

$$[0,0,0,0,4,5,2]^T \frac{1}{\sqrt{r * P(x_4)}} = [0,0,0,0,4,5,2]^T \frac{1}{\sqrt{2 * 0.185}}$$
$$= [0, 0, 0, 0, 6.58, 8.22, 3.29]^T$$

$$=> C = \begin{pmatrix} 1.54 & 0 \\ 4.63 & 0 \\ 6.17 & 0 \\ 7.72 & 0 \\ 0 & 6.58 \\ 0 & 8.22 \\ 0 & 3.29 \end{pmatrix}$$

R is constructed analogous

CUR Definition

*Input: let $\boldsymbol{M}$ be a $\boldsymbol{m\ x\ n}$ **matrix***

1.Step:

- *Choose a number $\boldsymbol{r}$ of 'concepts' (c.f. rank of matrix)*
  - *Perform biased Sampling of $\boldsymbol{r}$ cols from $\boldsymbol{M}$ and create a $\boldsymbol{m\ x\ r}$ **matrix** $\boldsymbol{C}$*
  - *Perform biased Sampling of $\boldsymbol{r}$ rows from $\boldsymbol{M}$ and create a $\boldsymbol{r\ x\ n}$ **matrix** $\boldsymbol{R}$*

2.Step:

- *Construct $\boldsymbol{U}$ from $\boldsymbol{C}$ and $\boldsymbol{R}$:*
  - *Create a $\boldsymbol{r\ x\ r}$ **matrix** $\boldsymbol{W}$ by the intersection of the chosen cols from C and rows from R*
  - *Apply SVD on $\boldsymbol{W} = \boldsymbol{X\ \Sigma\ Y^T}$*
  - *Compute $\boldsymbol{\Sigma^+}$, the moore-penrose pseudoinverse of Σ*
  - *Compute $\boldsymbol{U} = \boldsymbol{Y}(\boldsymbol{\Sigma^+})^{\boldsymbol{2}}\ \boldsymbol{X^T}$*

CUR Definition

Example – Calculating U

	Matrix	Alien	Star Wars	Cassablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Suppose C (Star Wars, Cassablance) and R (Jenny, Jack)

→ W as intersection of cols from C and rows from R:

$$W = \begin{pmatrix} 0 & 5 \\ 5 & 0 \end{pmatrix}$$

Ensure the correct order!

→ SVD applied on W:

$$W = \begin{pmatrix} 0 & 5 \\ 5 & 0 \end{pmatrix} = X\,\Sigma\,Y^T = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

→ Pseudo-Inverse of Σ

(replace diagonal entries with their numerical inverse)

$$\Sigma^+ = \begin{pmatrix} 1/5 & 0 \\ 0 & 1/5 \end{pmatrix}$$

→ Compute U

$$U = Y\,(\Sigma^+)^2 X^T = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1/5 & 0 \\ 0 & 1/5 \end{pmatrix}^2 \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1/25 \\ 1/25 & 0 \end{pmatrix}$$

High Dimensionality Data

[1] Less is More: Compact Matrix Decomposition for Large Sparse Graphs, Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos, Proceedings of the 2007 SIAM International Conference on Data Mining. 2007, 366-377

[2] Rajaraman, A.; Leskovec, J. & Ullman, J. D. (2014), *Mining Massive Datasets* .