Chapter 7:

Text Processing & High Dimensional Data

Introduction

Recap Data Science Intro:

… Data contains value and knowledge …

… but to extract the knowledge data needs to be

- Stored
- Managed

up to now, we have learned about this.

Introduction

Recap Data Science Intro:

… Data contains value and knowledge …

… but to extract the knowledge data needs to be

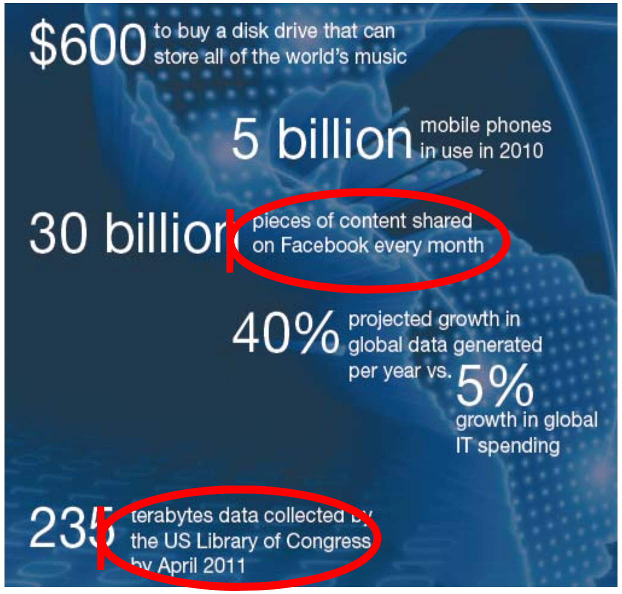
- Stored
- Managed
- And ANALYZED

up to now, we have learned about this.

Now, we will focus on this part

→ Big Data Analytics ≈ Data Mining ≈ Predictive Analytics ≈ Data Science

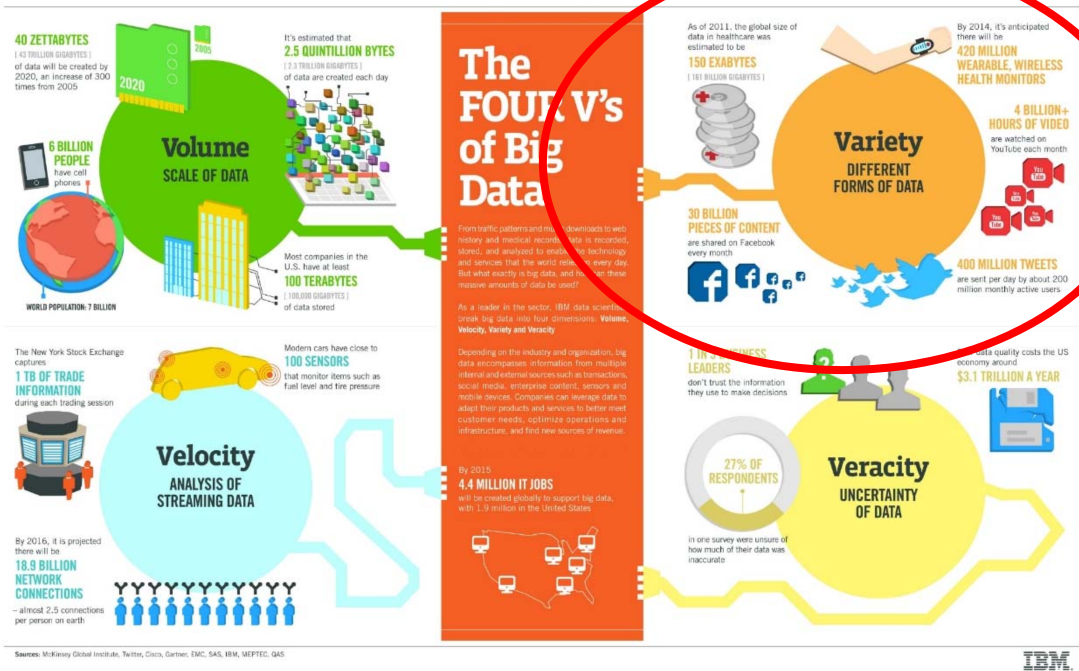
Recap Data Science Intro:



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Variety: different forms of data

- Unstructured, e.g. data in form of text
- Potentially high dimensional data



Outline

Text Processing

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

High-Dimensional Data

- Motivation
- **P**rincipal **C**omponent **A**nalysis
- **S**ingular **V**alue **D**ecomposition
- CUR

Text Processing – Motivation

Given: Set of documents

Searching for patterns in large sets of document objects

→ Analysing 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 (Common approaches - for details see KDD I)

How to handle relevancy of a term?

TF-IDF (Term Frequency * Inverse Document Frequency)

- *Emprical 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.

→ 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 characterisitic 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: $\mathbf{min(\infty, 3) = 3}$

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

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

Outline

Text Processing

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

High Dimensionality Data

- Motivation
- **P**rincipal **C**omponent **A**nalysis
- **S**ingular **V**alue **D**ecomposition
- CUR

Modeling data as matrices

Matrices often arise with data:

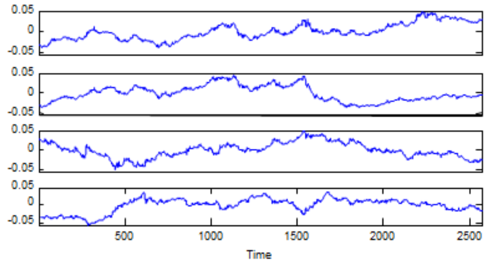
- $\boldsymbol{n}$ objects (documents, images, web pages, time series…)
- each with $\boldsymbol{m}$ features

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

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

(filter ‚for‘, ‚and‘ as stopwords)



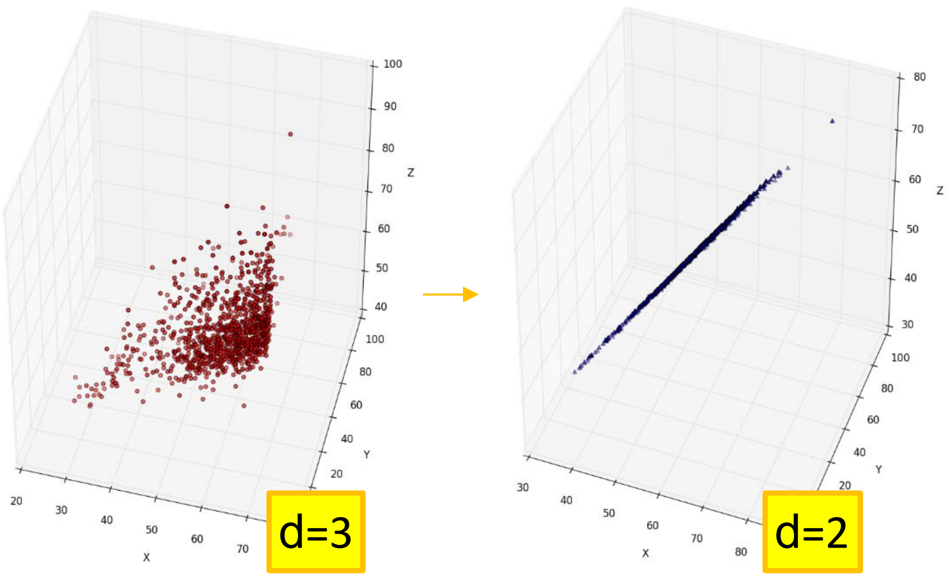
i-th series, $x^{(i)}$

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

values at time t, x_t

Why reduce Dimensions?

- Discover hidden correlations
- Remove redundant and noisy features
- Interpretation and visualization
- Easier storage and processing of the data



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

Outline

Text Processing

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

High Dimensionality Data

- Motivation
- **P**rincipal **C**omponent **A**nalysis
- **S**ingular **V**alue **D**ecomposition
- CUR

PCA Formulation

Goal of PCA: find a lower-dimensional $k < d$ representation of raw data

- **X** is $n\ x\ d$ (raw data)
- $\boldsymbol{Z = XP}$ is $n\ x\ k$ (reduces representation, P as PCA ‘scores’)
- **P is $\boldsymbol{d\ x\ k}$** (columns are k principal components)
- Variance constraints

$$\begin{bmatrix} \\ Z \\ \\ \end{bmatrix} = \begin{pmatrix} & & \\ & X & \\ & & \end{pmatrix} \begin{bmatrix} P \end{bmatrix}$$

PCA Formulation – Recall definition of Variance and Covariance

- $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n \, x \, d}}$: matrix of raw data
- $\boldsymbol{x_i}$: i-th datapoint
- $\boldsymbol{\mu}$: mean

Variance: Measure of the spread of the data:

$$\mathrm{Var(X)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

Covariance: Measure of how much two random variables vary together (zero mean assumption):

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i y_i)$$

Covariance Matrix: Variance of all features and the pairwise correlations between them (zero mean assumption):

$$\Sigma_X = \begin{pmatrix} Var(X_1) & \cdots & Cov(X_1, X_d) \\ \vdots & \ddots & \vdots \\ Cov(X_d, X_1) & \cdots & Var(X_d) \end{pmatrix} = \frac{1}{n} X^T X$$

PCA Formulation

Goal of PCA: find a lower-dimensional $k < d$ representation of raw data

- **X** is $n\ x\ d$ (raw data)
- $\boldsymbol{Z} = \boldsymbol{XP}$ is $n\ x\ k$ (reduces representation, PCA 'scores')
- **P is $\boldsymbol{d}\ \boldsymbol{x}\ \boldsymbol{k}$** (columns are k principal components)
- Variance constraints

- **Q**: Which constraints in reduced representation?
  - No feature correlation, i.e. all off-diagonals in C_Z are zero → avoids redundancy
  - Rank-ordered features by variance

PCA Solution

All matrices have an eigendecomposition:

- $\boldsymbol{C_x} = \boldsymbol{U\,\Lambda\,U^T}$ (eigendecomposition)
- $\boldsymbol{\Lambda}$ is $d\ x\ d$ (diagonals are **sorted** *eigenvalues*, off-diagonals are zero)
- $\boldsymbol{U}$ is $d\ x\ d$ (columns are *eigenvectors*, **sorted** by their associated eigenvalues)

The d eigenvectors are orthonormal directions of max variance

- Associated eigenvalues equal variance in these directions
- 1st eigenvector is direction of max variance (variance is λ_1)

PCA - Which k<d to choose for dimensional reduction?

Visualization: Pick top 2 or 3 dimensions for plotting purposes

Analysis: Capture ‚*most*' variance in the data

- As eigenvalues are sorted variances in the directions specified by eigenvectors, we can choose a fraction of retained variance:

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

E.g. choose k such that we retain 95% of the variance

Excursus: Eigenvectors and eigenvalues

Definition of the *algebraic eigenvalue problem*:

Let A be a square $d \; x \; d$ matrix. If there exists a real scalar λ and a $d \; x \; 1$ vector $v \; \neq 0$, such that:

$$Av = \lambda v,$$

then λ is called an **eigenvalue** of A and v is the associated **eigenvector**.

How to find eigenvalues / eigenvectors of A?

- Solving the equation: $\det(A - \lambda \, \mathrm{I}_{dxd}) = 0$ yields the eigenvalues
- For each eigenvalue λ_i, we find its eigenvector by solving the system of equations $(A - \lambda_i \, I_{dxd}) \, v_i = 0$

Excursus: Eigenvectors and eigenvalues

Example:

$$A = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$$

$$A - \lambda * I_{2x2} = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2-\lambda & 3 \\ 2 & 1-\lambda \end{pmatrix}$$

$$\det(A - \lambda * \mathrm{I}_{2x2}) = (2-\lambda)(1-\lambda) - 6 = \lambda^2 - 3\lambda - 4 = (\lambda + 1) * (\lambda - 4)$$

→ Largest eigenvalue (in magnitude) is $\lambda_1 = 4$, smallest eigenvalue $\lambda_2 = -1$

$$(A - \lambda_1 * \mathrm{I}_{2x2})v_1 = \begin{pmatrix} -2 & 3 \\ 2 & -3 \end{pmatrix} v_1 = \vec{0} \Rightarrow v_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

$$(A - \lambda_2 * \mathrm{I}_{2x2})v_2 = \begin{pmatrix} 3 & 3 \\ 2 & 2 \end{pmatrix} v_2 = \vec{0} \Rightarrow v_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

PCA Solution

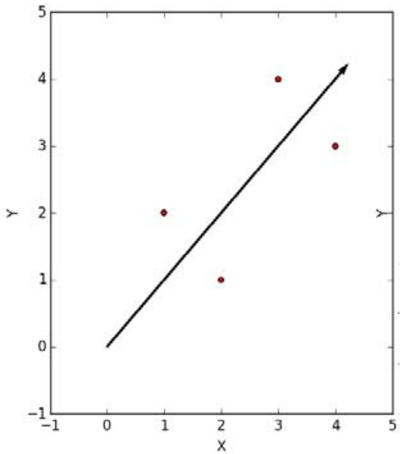
Computation

- Treat a set of tuples as a matrix M
- Find eingevectors for $M^T M$ or MM^T
  - Eigenvectors can be thought of a rotation in high-dimensional space
  - Principal eigenvector yields the axis along which the variance of the data is maximizied

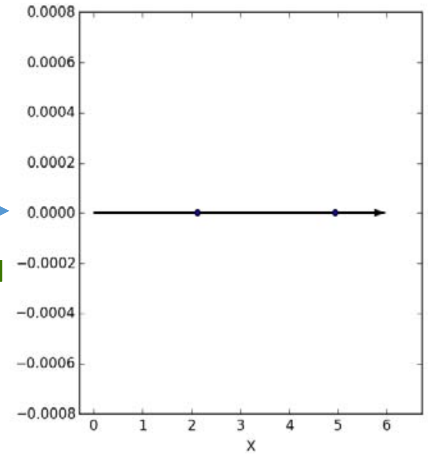
→ High-dimensional data can be replaced by its projection onto the most important axes

PCA Example

$$X = \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{pmatrix} \Rightarrow X^T X = \begin{pmatrix} 30 & 28 \\ 28 & 30 \end{pmatrix}$$



Projection of 2d points onto a one-dimensional space



→ **Eigenvalues:** solving $\det(X^T X - \lambda I) = 0$ yields $\lambda_1 = 58, \lambda_2 = 2$

→ **Eigenvectors:** solving $(X^T X - \lambda_i I) v_i$ yields $E = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$

→ **Projection** of data on principal component by using first k columns of E:

$$\boldsymbol{XE_1} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{2} & \mathbf{1} \\ \mathbf{3} & \mathbf{4} \\ \mathbf{4} & \mathbf{3} \end{pmatrix} \begin{pmatrix} \mathbf{1/\sqrt{2}} \\ \mathbf{1/\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \mathbf{3/\sqrt{2}} \\ \mathbf{3/\sqrt{2}} \\ \mathbf{7/\sqrt{2}} \\ \mathbf{7/\sqrt{2}} \end{pmatrix}$$

Outline

Text Processing

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

High Dimensionality Data

- Motivation
- **P**rincipal **C**omponent **A**nalysis
- **S**ingular **V**alue **D**ecomposition
- CUR

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

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

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

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

Singular Value Decomposition (SVD)

Computing SVD of a Matrix

Connected to eingevalues 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 Eigenwert-Problem:
$Av = \lambda v$

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

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

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

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 $s_i = 0$

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

→ 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 mxn 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

$$
\begin{array}{l}
\text{Joe} \rightarrow \\ \\ \\ \\ \\ \\ \\
\end{array}
\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{pmatrix}
=
\underbrace{\begin{array}{cc} \text{Sci-Fi concept} & \text{romance concept} \\ \left(\begin{array}{cc} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .6 \\ 0 & .75 \\ 0 & .30 \end{array}\right) \end{array}}
* \underbrace{\begin{pmatrix} 12.4 & 0 \\ 0 & 9.5 \end{pmatrix}}
* \underbrace{\begin{pmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{pmatrix}}
$$

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**

Outline

Text Processing

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

High Dimensionality Data

- Motivation
- **P**rincipal **C**omponent **A**nalysis
- **S**ingular **V**alue **D**ecomposition
- CUR

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 *columns and rows of A*

$$\begin{pmatrix} \quad & A & \quad \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} * \begin{pmatrix} U \end{pmatrix} * \begin{pmatrix} \quad R \quad \end{pmatrix}$$

CUR Decomposition

- *In large-data applications 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* $\boldsymbol{m\ x\ r}$ ***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* $\boldsymbol{U} = \boldsymbol{Y}(\boldsymbol{\Sigma}^+)^2\ \boldsymbol{X^t}$

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: $C \in \mathbb{R}^{m\,x\,r}$
1. **For** x = 1 : n **do**
2. P(x) = $\sum_i (m_{i,x})^2$ / $\|M\|_F^2$
3. **For** y = 1 : r **do**
4. Pick $z \in 1{:}n$ based on Prob(x)
5. 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

	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:

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

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

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

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

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

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 $\boldsymbol{W} = \boldsymbol{X}\ \boldsymbol{\Sigma}\ \boldsymbol{Y^T}$*
  - *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

$$U = Y\,(\Sigma^+)^2 X^T = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1/5 & 0 \\ 0 & 1/5 \end{pmatrix}^2\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1/25 \\ 1/25 & 0 \end{pmatrix}$$

Sources

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* .