Chapter XII: Data Pre and Post Processing

Information Retrieval & Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2013/14

Chapter XII: Data Pre and Post Processing

1. Data Normalization
2. Missing Values
3. Curse of Dimensionality
4. Feature Extraction and Selection
    4.1. PCA and SVD
    4.2. Johnson–Lindenstrauss lemma
    4.3. CX and CUR decompositions
5. Visualization and Analysis of the Results
6. Tales from the Wild

Zaki & Meira, Ch. 2.4, 6 & 8

XII.1: Data Normalization

1. Centering and unit variance
2. Why and why not normalization?

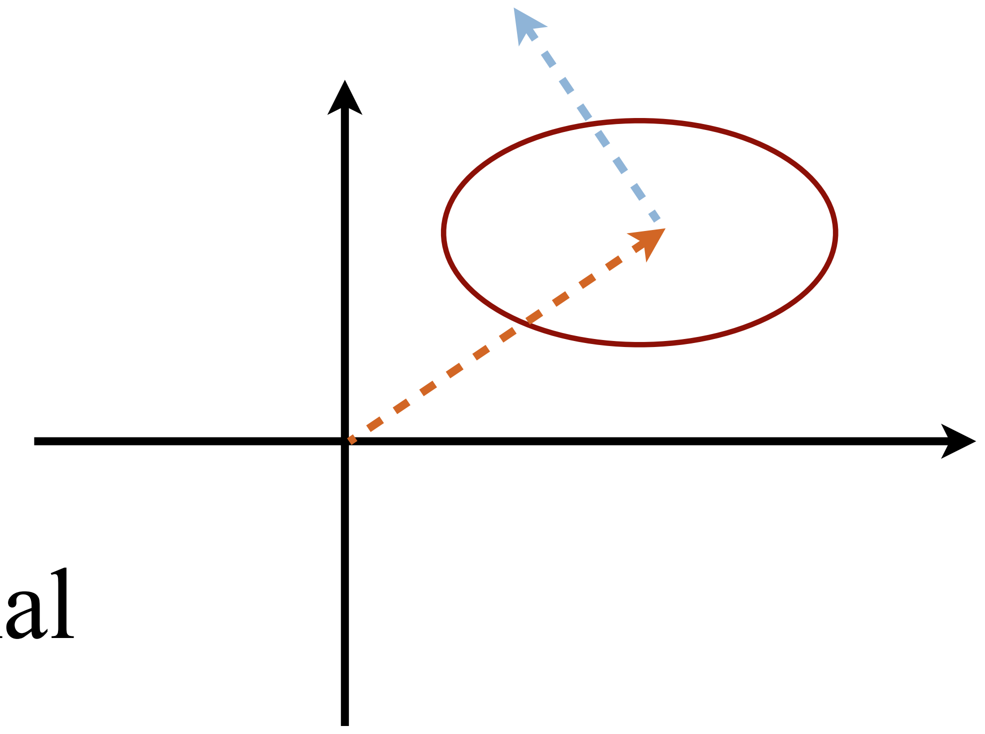
Zero centering

- Consider a data D that contains n observations over m variables
  - n-by-m matrix $\boldsymbol{D}$
- We say $\boldsymbol{D}$ is **zero centered** if $mean(\boldsymbol{d}_i) = 0$ for each column $\boldsymbol{d}_i$ of $\boldsymbol{D}$
- We can center any matrix by subtracting from its columns their means

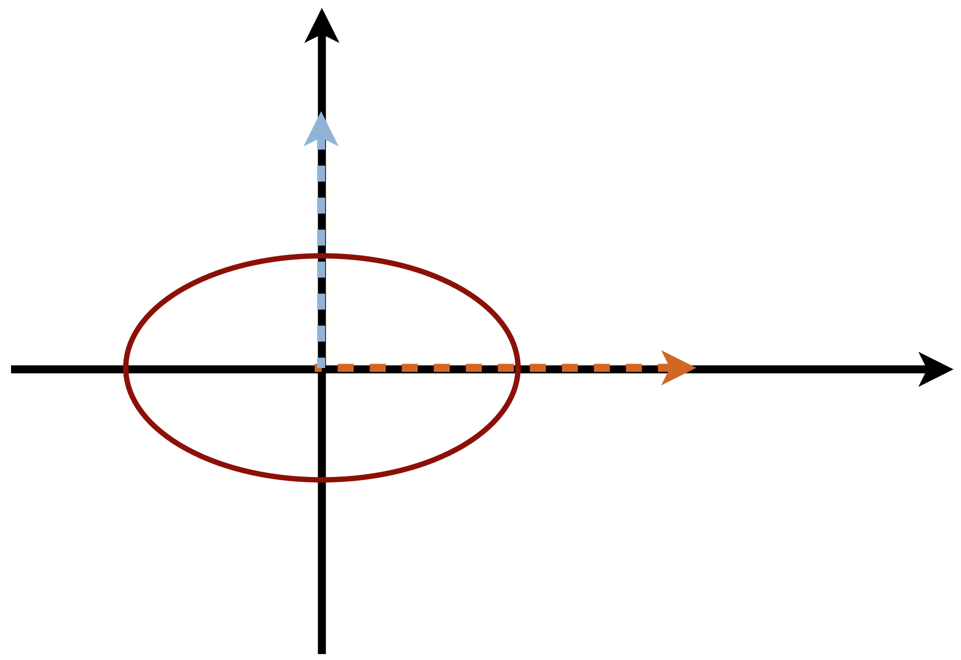
Unit variance and z-scores

- Matrix $\boldsymbol{D}$ is said to have **unit variance** if $var(\boldsymbol{d}_i) = 1$ for each column $\boldsymbol{d}_i$ of $\boldsymbol{D}$
  - The unit variance is obtained by dividing every column with its standard deviation
- Data that is zero centered and normalized to unit variance is called the **z-scores**
  - Many methods assume the input is z-scores
- We can also apply non-linear transformations before normalizing to the z-scores
  - E.g. taking logarithms (from positive data) or cubic roots (from general data) diminishes the importance of larger values

Why centering?



- Consider the red data ellipse
  - The main direction of variance is from the origin to the data
  - The second direction is orthogonal to the first
  - These don’t tell the variance of the data!
- If we center the data, the directions are correct



Why unit variance?

- Assume one observation is height in meters and other weight in grams
  - Now weight contains much higher values (for humans, at least)

    $\Rightarrow$ weight has more weight in calculations
- Division by standard deviation makes all observations equally important
  - Most values fall between –1 and 1

When not to center?

- Centering cannot be applied to all kinds of data
- It destroys non-negativity
  - E.g. NMF becomes impossible
- Centered data won't contain integers
  - E.g. counting or binary data
  - Can hurt interpretability
  - Itemset mining and BMF become impossible
- Centering destroys sparsity
  - Bad for algorithmic efficiency
  - We can retain sparsity by only chancing non-zero values

What’s wrong with unit variance?

- Dividing by standard deviation is based on the assumption that the values follow Gaussian distribution
  - Often plausible by the Law of Large Numbers
- Not all data is Gaussian
  - Integer counts
    - Especially over a small range
  - Transaction data
  - …

XII.2: Missing values

1. Handling missing values
2. Imputation

Missing values

- Missing values are common in real-world data
  - Unobserved
  - Lost in collection
  - Error in measurement device
  - …
- Data with missing values needs to be dealt with care
  - Some methods are robust to missing values
    - E.g. naïve Bayes classifiers
  - Some methods cannot (natively) handle missing values
    - E.g. support vector machines

Handling missing values

- Two common techniques to handle missing values are
  - Imputation
  - Ignoring them
- In **imputation**, the missing values are replaced with "educated guesses"
  - E.g. the mean value of the variable
    - Perhaps stratified over some class
      - The mean height vs. the mean height of the males
  - Or a model is fitted to the data and the missing values are drawn from the model
    - E.g. a low-rank matrix factorization that fits the observed values
      - This technique is used with lots of missing values in **matrix completion**

Some problems

- Imputation might impute wrong values
  - This might have significant effect on the results
  - Especially categorical data is hard
    - The effect of imputation is never “smooth”
- Ignoring records or variable with missing values might not be possible
  - There might not be any data left
- Especially binary data has the problem of distinguishing non-existent and non-observed data
  - E.g. if data says that certain species does not observed in certain area, it does not mean the species couldn’t live there

XII.3: Curse of Dimensionality

1. The Curse
2. Some oddities of high-dimensional spaces

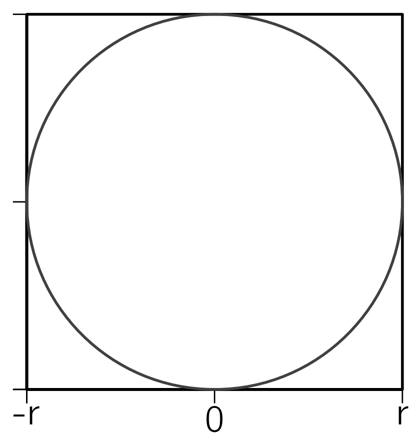
Curse of dimensionality

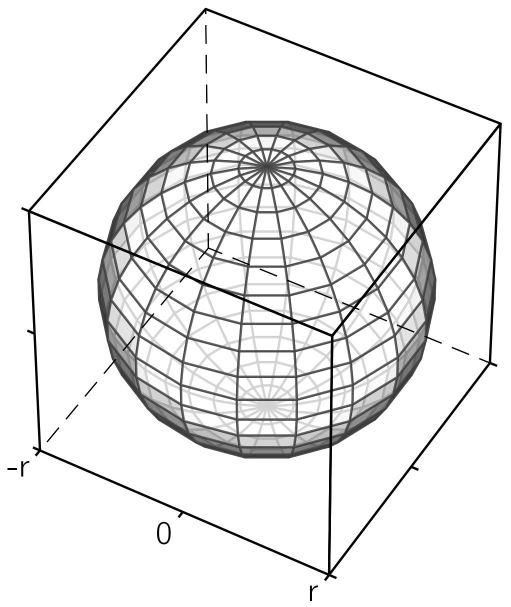
- Many data mining algorithms need to work in high-dimensional data
- But life gets harder as dimensionality increases
  - The volume grows too fast
    - 100 points evenly-spaced points in unit interval have max distance between adjacent points of 0.01
    - To get that distance for adjacent points in 10-dimensional unit hypercube requires 10^{20} points
    - Factor of 10^{18} increase
- High-dimensional data also makes algorithms slower

Hypersphere and hypercube

- Hypercube is *d*-dimensional cube with edge length *2r*
  - Volume: $\mathrm{vol}(H_d(2r)) = (2r)^d$
- Hypersphere is the *d*-dimensional ball of radius *r*
  - $\mathrm{vol}(S_1(r)) = 2r$
  - $\mathrm{vol}(S_2(r)) = \pi r^2$
  - $\mathrm{vol}(S_3(r)) = 4/3\ \pi r^3$
  - $\mathrm{vol}(S_d(r)) = K_d r^d$, where $K_d = \dfrac{\pi^{d/2}}{\Gamma(d/2+1)}$
    - $\Gamma(d/2 + 1) = (d/2)!$ for even *d*

Hypersphere within hypercube

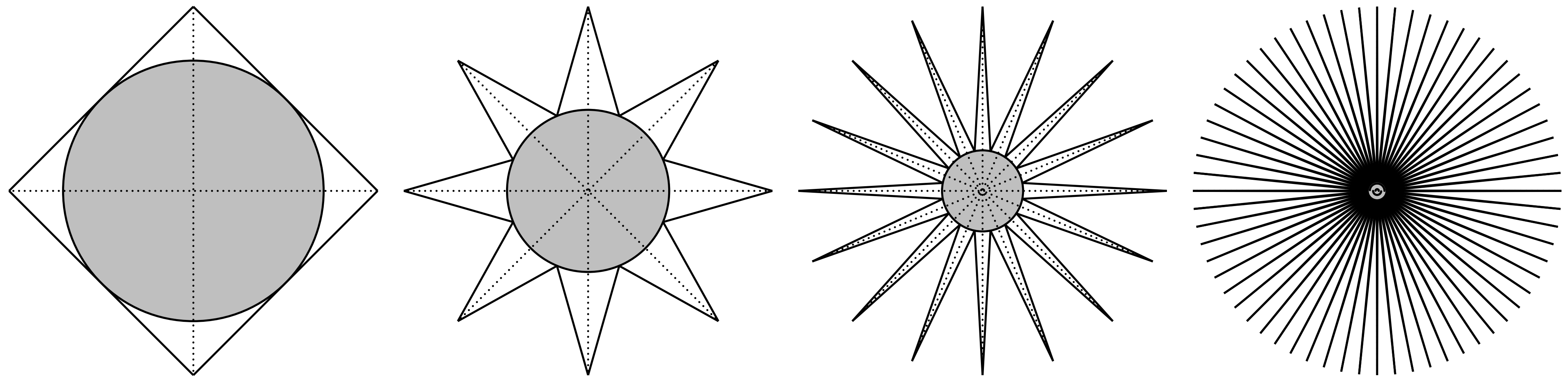




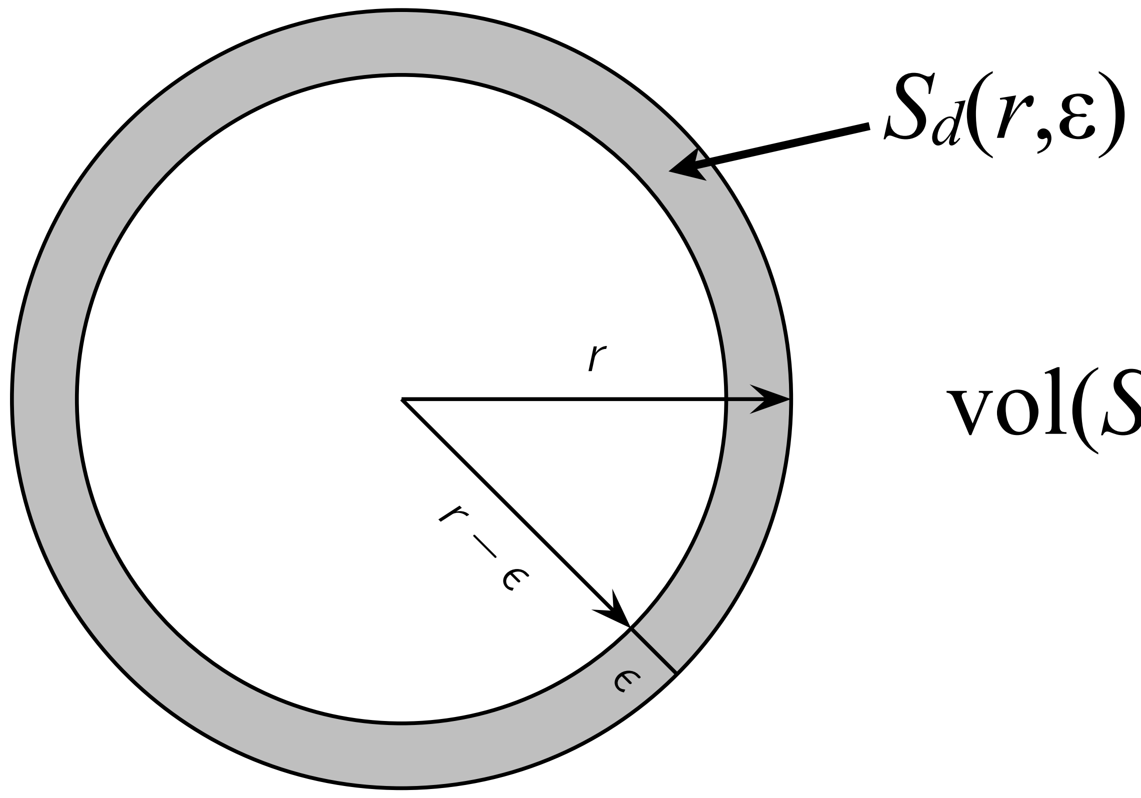
Mass is in the corners!

Fraction of volume hypersphere has of surrounding hypercube:

$$\lim_{d\to\infty}\frac{\mathrm{vol}(S_d(r))}{\mathrm{vol}(H_d(2r))} = \lim_{d\to\infty}\frac{\pi^{d/2}}{2^d\Gamma(d/2+1)} \to 0$$



Volume of thin shell of hypersphere



$$\mathrm{vol}(S_d(r,\varepsilon)) = \mathrm{vol}(S_d(r)) - \mathrm{vol}(S_d(r-\varepsilon))$$
$$= K_d r^d - K_d(r-\varepsilon)^d$$

Fraction of volume in the shell: $\dfrac{\mathrm{vol}(S_d(r,\epsilon))}{\mathrm{vol}(S_d(r))} = 1 - \left(1 - \dfrac{\epsilon}{r}\right)^d$

$$\lim_{d\to\infty} \frac{\mathrm{vol}(S_d(r,\epsilon))}{\mathrm{vol}(S_d(r))} = \lim_{d\to\infty} 1 - \left(1 - \frac{\epsilon}{r}\right)^d \to 1$$

Mass is in the shell!

XII.4: Feature Extraction and Selection

1. Dimensionality reduction and PCA
   1.1. PCA
   1.2. SVD
2. Johnson–Lindenstrauss lemma
3. CX and CUR decompositions

Dimensionality reduction

- Aim: reduce the number of features/dimensions by replacing them with new ones
  - The new features should capture the "essential part" of the data
  - What is considered essential defines what method to use
  - Vice versa, using wrong dimensionality reduction can lead to non-sensical results
- Usually dimensionality reduction methods work on numerical data
  - For categorical or binary data, feature selection can be more appropriate

Principal component analysis

- The goal of the **principal component analysis** (PCA) is to project the data onto linearly uncorrelated variables in (possibly) lower-dimensional subspace that preserves as much of the variance of the original data as possible
  - Also known as Karhunen–Lòeve transform or Hotelling transform
    - And with many other names, too
- In matrix terms, we want to find a column-orthogonal n-by-r matrix $\boldsymbol{U}$ that projects n-dimensional data vector $\boldsymbol{x}$ into r-dimensional vector $\boldsymbol{a} = \boldsymbol{U}^T\boldsymbol{x}$

Deriving the PCA: 1-D case (1)

- We assume our data is normalized to z-scores
- We want to find a unit vector $\boldsymbol{u}$ that maximizes the variance of the projections $\boldsymbol{u}^T\boldsymbol{x}_i\boldsymbol{u}$
  - Scalar $\boldsymbol{u}^T\boldsymbol{x}_i$ gives the coordinate of $\boldsymbol{x}_i$ along $\boldsymbol{u}$
  - As data is normalized, its mean is 0, which has coordinate 0 when projected to $\boldsymbol{u}$
- The variance of the projection is

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{u}^T\boldsymbol{x}_i - \mu_{\boldsymbol{u}})^2$$

$$= \boldsymbol{u}^T\boldsymbol{\Sigma}\boldsymbol{u}$$

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^T$$

The covariance matrix for centered data

Deriving the PCA: 1-D case (2)

- To maximize variance σ^2, we maximize $J(\boldsymbol{u}) = \boldsymbol{u}^T \boldsymbol{\Sigma u} - \lambda(\boldsymbol{u}^T \boldsymbol{u} - 1)$
  - The second term is to ensure $\boldsymbol{u}$ is a unit vector
- Solving the derivative gives $\boldsymbol{\Sigma u} = \lambda \boldsymbol{u}$
  - $\boldsymbol{u}$ is an eigenvector and λ is an eigenvalue
  - Further $\boldsymbol{u}^T \boldsymbol{\Sigma u} = \boldsymbol{u}^T \lambda \boldsymbol{u}$ implying that $\sigma^2 = \lambda$
    - To maximize variance, we need to take the largest eigenvalue
- Thus, the **first principal component** $\boldsymbol{u}$ is the dominant eigenvector of the covariance matrix $\boldsymbol{\Sigma}$

Example of 1-D PCA

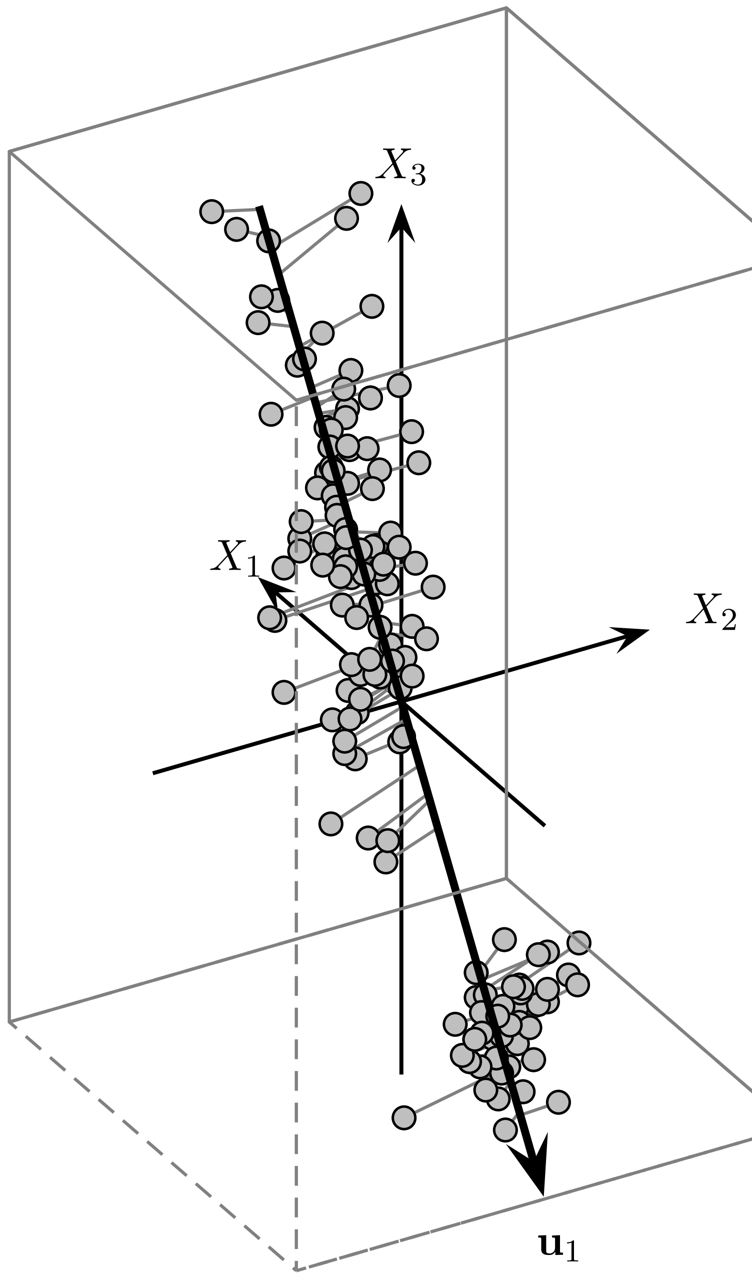


Figure 7.2: Best One-dimensional or Line Approximation

Deriving the PCA: *r* dimensions

- The second principal component should be orthogonal to the first one and maximize the variance
  - Adding this constraint and deriving shows that the second principal component is the eigenvector associated with the second-highest eigenvalue
  - Further, to find *r* principal components, we take the eigenvectors of $\boldsymbol{\Sigma}$ associated to the *r* largest eigenvalues
  - The total variance is the sum of the eigenvalues
- It also turns out that maximizing the variance minimizes the mean squared error

$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{U}^T\boldsymbol{x}\boldsymbol{U}\|^2$$

Computing the PCA

- We can compute the covariance matrix and its top-k eigenvectors
- Or we can use SVD
  - Because covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{X}\boldsymbol{X}^T$ and if $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$
  - This approach is preferred due to numerical stability
    - Computing the covariance matrix can cause numerical stability issues with the eigendecomposition

Kernel PCA

- PCA separates linear correlations
  - But what if the correlations are not linear?
- We can use the kernel trick as with SVMs, say
  - Map the input space into higher-dimensional feature space and find linear correlations there
- Basic idea: replace $\boldsymbol{\Sigma}$ with (centered) kernel matrix $\boldsymbol{K}$
  - n-by-n matrix with $k_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_\mathrm{i})^T\phi(\boldsymbol{x}_\mathrm{j})$
- We cannot compute the principal vectors directly
  - They're expressed using $\phi(\boldsymbol{x})$
  - But we can project $\phi(\boldsymbol{x})$ onto the principal direction using kernels

Problems with PCA and SVD

- Many characteristics of the original data are lost
  - Non-negativity
  - Integrality
  - Sparsity
  - …
- Also, the computation can be costly for big matrices
  - Although there exists approximate methods to do SVD in a single sweep of the matrix

Johnson–Lindenstrauss lemma

- Finding the decomposition can be expensive
- Decompositions give only *global* guarantees
  - Any pair of points can have very different distances
- Can we guarantee *local* similarity?

Johnson–Lindenstrauss lemma. Given $\varepsilon > 0$ and an integer n, let k be a positive integer such that $k \geq k_0 = O(\varepsilon^{-2} \log n)$. For every set X of n points in $\mathbb{R}^d$ there exists $F\colon \mathbb{R}^d \to \mathbb{R}^k$ such that for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$

$$(1-\varepsilon)\,\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leqslant \|F(\mathbf{x}_i) - F(\mathbf{x}_j)\|^2 \leqslant (1+\varepsilon)\,\|\mathbf{x}_i - \mathbf{x}_j\|^2$$

How to find the projections?

- We need to find an k-by-d matrix $\boldsymbol{R} = (r_{ij})$ such that function $\boldsymbol{x} \mapsto \boldsymbol{Rx}$ satisfies JL
- Remarkably, if we select $r_{ij} \sim N(0,1)$, $\boldsymbol{R}$ satisfies JL with high probability
  - That is, JL holds for *all* points of X with high probability
- Achlioptas has show that we can also select $\Pr[r_{ij} = 1] = 1/2$ and $\Pr[r_{ij} = -1] = 1/2$ or $\Pr[r_{ij} = 1] = 1/6$, $\Pr[r_{ij} = 0] = 2/3$, $\Pr[r_{ij} = -1] = 1/6$
  - Sparse matrix

CX and CUR decompositions

- Sometimes we want to retain the original features
  - Interpretability
  - Sparsity
  - …
- We can select the most important features and work only on them
- There are many ways to do feature selection
  - CX and CUR decompositions are one option

The CX factorization

- Given a data matrix ***D***, find a subset of columns of ***D*** in matrix ***C*** and a matrix ***X*** s.t. $||\boldsymbol{D} - \boldsymbol{CX}||_F$ is minimized
  - Interpretability: if columns of ***D*** are easy to interpret, so are columns of ***C***
  - Sparsity: if all columns of ***D*** are sparse, so are columns of ***C***
  - Feature selection: selects actual columns
  - Approximation accuracy: if $\boldsymbol{D}_k$ is the rank-*k* truncated SVD of ***D*** and ***C*** has *k* columns, then with high probability

$$\|D - CX\|_F \leqslant O(k\sqrt{\log k})\, \|D - D_k\|_F$$

[Boutsidis, Mahoney & Drineas, KDD ’08, SODA ’09]

The CUR factorization

- Given data matrix $\boldsymbol{D}$, its **CUR factorization** is $\boldsymbol{D} \approx \boldsymbol{CUR}$, where matrix $\boldsymbol{C}$ has r columns of $\boldsymbol{D}$ and matrix $\boldsymbol{R}$ has r rows of $\boldsymbol{D}$ and $\boldsymbol{U}$ is arbitrary mixing matrix
  - The aim is to minimize $\|\boldsymbol{D} - \boldsymbol{CUR}\|_F$
  - We also have approximation results for CUR, but they require many more rows and columns
- The CUR decomposition selects "stereotypical" rows *and* columns

Computing CX and CUR — the idea

- The columns (and rows in CUR) are selected randomly
  - The probability of sampling each row or column is proportional to its L_2-norm
    - Heavy rows and columns are more probable
- After $\boldsymbol{C}$ is obtained, the $\boldsymbol{X}$ in CX is computed using the pseudo-inverse
- To compute the $\boldsymbol{U}$ in the CUR, we first take the submatrix of $\boldsymbol{D}$ defined by the Cartesian product of row indices in $\boldsymbol{R}$ and column indices in $\boldsymbol{C}$
  - The final $\boldsymbol{U}$ is the pseudo-inverse of this matrix

Summary

- Normalizing the data can be crucial
- Missing values need to be dealt with
- High-dimensional data is a problem for many data mining methods
  - Computational complexity
  - Everything is evenly far from everything
- Many ways to address the problem
  - PCA gives dimensionality reduction with global guarantees
  - JL lemma tells us we can also achieve local guarantees
  - Feature selections retains important features of the data