# Knowledge Discovery in Databases II

## Lecture 2 – High Dimensional Data

Prof. Dr. Peer Kröger, Yifeng Lu

Sommer Semester 2019

Credits:

Based on material of Eirini Ntoutsi, Matthias Schubert,

Arthur Zimek, Peer Kröger, Yifeng Lu

## Table of Contents

## Kapitel 1: Intorduction i

## Feature Transformation

### Feature Transform

- Consider the following spaces:
  - $\mathbb{U}$ denotes the universe of data objects
  - $\mathbb{F} \subseteq \mathbb{R}^n$ denotes an *n*-dimensional feature space
- A feature transformation is a mapping $f : \mathbb{U} \to \mathbb{R}^n$ of objects from $\mathbb{U}$ to the feature space $\mathbb{F}$.

### Similarity Model

- A similarity model $S : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ is defined for all objects $p, q \in \mathbb{U}$ as

$$S(p,q) = sim(f(p), f(q))$$

  where $sim : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a similarity measure or a dissimilarity (distance) measure in $\mathbb{F}$.

## Similarity versus Dissimilarity

Comments:

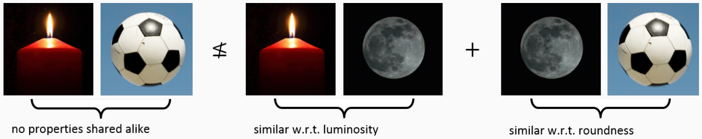
- Often, dissimilarity (distance) is measured instead of similarity
- This is a small but important difference!
    - A similarity measure (*sim*) assigns high values to similar objects
    - A dissimilarity measure (*dist*) assigns low values to similar objects
- The design of $f$ and the definition of *sim*/*dist* are important assumptions about the patterns we want to find later in the data
- As explained before, $f$ and *sim*/*dist* can be derived manually (explicit transformation and coding versus implicit Kernels) or automatically (representation learning)

## Dissimilarity

- Dissimilarity measures follow the idea of the geometric approach
    - objects are defined by their perceptual representations in a perceptual space
    - perceptual space = psychological space
    - geometric distance between the perceptual representations defines the (dis)similarity of objects
- Within the scope of Feature-based similarity
    - perceptual space = feature space $\mathbb{F}$ or feature representation space $\mathbb{R}^n$
    - geometric distance = distance function

## Distance Functions

- The distance measure *dist* is a distance function if it is reflexive, non-negative, and symmetric
- A distance function *dist* is a metric if it additionally satisfies the triangle inequality
- Comments:
  - Sound mathematical interpretation
  - Allow domain experts to model their notion of dissimilarity
  - Metric distances allow to tune efficiency of data mining approaches
  - Long-lasting discussion of whether the distance properties and in particular the metric properties reflect the perceived dissimilarity correctly, see the following contradicting example:

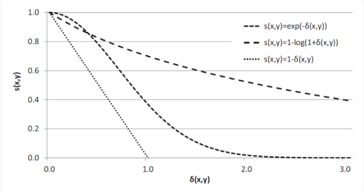


## Similarity versus Dissimilarity (again)

- Transformation
  - Let $\mathbb{F}$ be a feature space and $dist : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ be a distance function
  - Any monotonically decreasing function $f : \mathbb{R} \to \mathbb{R}$ defines a similarity function $s : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ as follows

$$\forall x, y \in \mathbb{F} : s(x, y) = f(dist(x, y))$$

- Some prominent similarity functions $(x, y \in \mathbb{F})$:
  - exponential: $s(x, y) = e^{(-dist(x,y))}$
  - logarithmic: $s(x, y) = 1 - \log(1 + dist(x, y))$
  - linear: $s(x, y) = 1 - dist(x, y)$



## Similarities: Examples (only very few)

- Dot-Product ($x, y \in \mathbb{F} \subseteq \mathbb{R}^d$)

$$x \cdot y^T = \sum_{i=1}^{d} x_i \cdot y_i = \|x\| \cdot \|y\| \cdot \cos \sphericalangle(x, y)$$

- Cosine ($x, y \in \mathbb{F} \subseteq \mathbb{R}^d$)

$$\frac{x \cdot y^T}{\|x\| \cdot \|y\|}$$

- Pearson Correlation ($x, y \in \mathbb{F} \subseteq \mathbb{R}^d$)

$$\frac{\sum_{i=1}^{d} (x_i - \bar{x}_i) \cdot (y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x}_i)^2} \cdot \sqrt{\sum_{i=1}^{d} (y_i - \bar{y}_i)^2}}$$

  where $\bar{z}_i$ denotes the mean in attribute $i$ over all data points

- Random-Walk Kernel (for graphs $x, y$)
  - Count common (random) walks in $x$ and $y$
  - Walks are sequences of nodes (connected by edges)

## Distances: Examples (only very few)

- $L_p$-norm (aka Minkowski metric) ($x, y \in \mathbb{F} \subseteq \mathbb{R}^d$)

$$L_p(x,y) = \sqrt[p]{\sum_{i=1}^{d} |x_x - y_i|^p}$$

  where

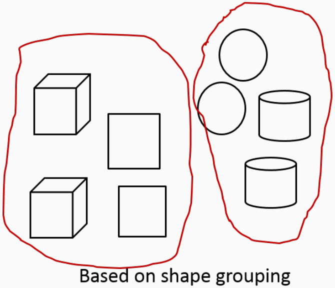
  - $p < 1$: fractional Minkowski distance
  - $p = 1$: Manhattan distance
  - $p = 2$: Euclidean distance
  - $p = \infty$: Chebyshev/Maximum distance

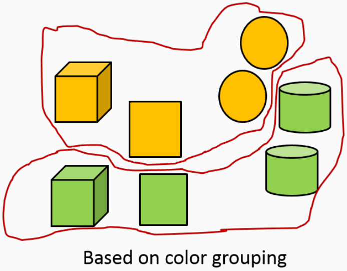
- Malahanobis distance
- Hamming distance $HammingDist(x,y) = \sum_{i=1}^{d} \begin{cases} 1 & : & x_i \neq y_i \\ 0 & : & \text{else} \end{cases}$

# Kapitel 2: Challenges i

## A Motivating Example

- Let's play the baby shapes game (truly motivating for students ...): Group the items!!!





- What about grouping based on both shape and color?
- Lesson to learn: there may be different semantic concepts (and their corresponding patterns) hidden in the data (here: shape and color)

## The More the Merrier or More is Less?

**The good old days of data mining . . .**

- Data generation and, to some extend, data storage was costly (hard to imagine but those were the days ...)
- Domain experts carefully considered which features/variables to measure before designing experiments/a feature transform/. . .
- Consequence: also data sets were well designed and potentially contained only a small number of relevant features

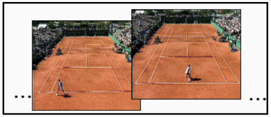
## The More the Merrier or More is Less?

**Nowadays, data science is also about integrating everything**

- Generating and storing data is easy and cheap
- People tend to measure everything they can and even more (including even more complex feature transformations)
- The Data Science mantra is often interpreted as “we can analyze data from as many sources as (technically) possible, just record anaything you can”
- Consequence: data sets are high-dimensional containing a large number of features but the relevancy of each feature for the analysis goal is not clear a priori

## High-dimensional Data is NOT a Myth

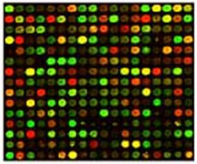
- Example: Image data
  - Low-level image descriptors (color histograms, textures, shape information ...)
  - Regional descriptors: between 16 and 1,000 features
  - ...

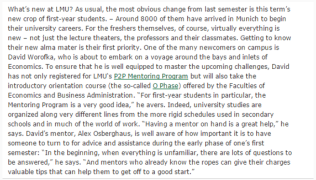


- Example: Metabolome data
  - Feature = concentration of one metabolite (intermediates/results of metabolism)
  - Bavaria newborn screening (for each baby, the blood concentrations of 43 metabolites are measured in the first 48 hours after birth)
  - between 50 and 2,000 features

## More High-dimensional Data

- Example: Microarray data (deprecated)
  - Features correspond to genes
  - Up to 20,000 features
  - Dimensionality is much higher than the sample size
- Example: Text data
  - Term frequency: features correspond to words/terms
  - Between 5,000 and 20,000 features (and even more)
  - Often, esp. in social media: abbreviations, colloquial language, special words





What’s new at LMU? As usual, the most obvious change from last semester is this term’s new crop of first-year students. – Around 8000 of them have arrived in Munich to begin their university careers. For the freshers themselves, of course, virtually everything is new – not just the lecture theaters, the professors and their classmates. Getting to know their new alma mater is their first priority. One of the many newcomers on campus is David Worofka, who is about to embark on a voyage around the bays and inlets of Economics. To ensure that he is well equipped to master the upcoming challenges, David has not only registered for LMU‘s P2P Mentoring Program but will also take the introductory orientation course (the so-called O Phase) offered by the Faculties of Economics and Business Administration. “For first-year students in particular, the Mentoring Program is a very good idea,” he avers. Indeed, university studies are organized along very different lines from the more rigid schedules used in secondary schools and in much of the world of work. “Having a mentor on hand is a great help,” he says. David’s mentor, Alex Osberghaus, is well aware of how important it is to have someone to turn to for advice and assistance during the early phase of one’s first semester: “In the beginning, when everything is unfamiliar, there are lots of questions to be answered,” he says. “And mentors who already know the ropes can give their charges valuable tips that can help them to get off to a good start.”

***Excerpt from LMU website:***
***http://tinyurl.com/qhq6byz***

## Problems with High-dimensional Data

Overview:

- Distances grow
- Contrast of distances diminish (concentration problem)
- Meaning of “neighborhood” concept
- Growing data space
- Growing hypothesis space
- Empty spaces and importance tails
- Different semantic layers
- ...

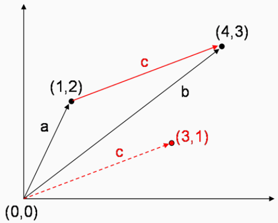
So let us have a closer look on these problems ...

## Distances Grow

The following example uses the Euclidean distance but holds for most distance measures:

- Consider 2D vectors $a = (1,2)$ and $b = (4,3)$
- The Euclidean distance between $a$ and $b$ is

$$\begin{aligned} L_2(a,b) &= L_2((1,2),(4,4)) \\ &= \sqrt{(1-4)^2 + (2-3)^2} \\ &= \sqrt{10} \end{aligned}$$



which corresponds to the norm of the difference vector $c = (3,1)$:

$$\|c\|_2 = \sqrt{3^2 + 1^2}$$

## Distances Grow

With increasing dimensionality, distances grow, too:

- Example: $L_2((1,2),(4,3)) = \sqrt{10}$
- Now double the feature vector length (double the original features): $L_2((1,2,1,2),(4,3,4,3)) = \sqrt{(3^2 + 1^2 + 3^2 + 1^2)} = \sqrt{20}$
- Effect seems not so important, values might be only in a larger scale?
- NOPE:

  **Contrast of distances is lost in high dimensional data since distances grow more and more alike!**

This is know as the Concentration of Distances problem (see next)

## Concentration of Distances

### Concentration Phenomenon

- As dimensionality grows, distance values grow, too, such that the (numerical) contrast provided by usual measures decreases or even diminishes
- In other words, the distribution of norms in a given distribution of points tends to concentrate

- Example: Euclidean norm of vectors consisting of several variables that are (assumed to be) independent and identically distributed

$$\|y\|_2 = \sqrt{y_1^2 + y_2^2 + \ldots + y_d^2}$$

- In high dimensional spaces this norm behaves unexpectedly ...

## Concentration of Distances

**Theorem: Concentration of Distances**

- Let $y$ be a $d$-dimensional vector $(y_1, ..., y_d)$ where all components $y_i (1 \leq i \leq d)$ are independent and identically distributed
- Then the mean and the variance of the Euclidean norm are:

$$\mu_{\|y\|} = \sqrt{a \cdot d - b} + \mathcal{O}(d^{-1}) \quad \text{and} \quad \sigma_{\|y\|} = b + \mathcal{O}(d^{-1/2})$$

where $a$ and $b$ are parameters depending only on the central moments of order 1, 2, 3, 4.

Interpretation:

- The norm grows proportionally to $\sqrt{d}$, but the variance remains approx. constant for large $d$ (because $\lim_{d \to \infty} d^{-const} = 0$)
- With growing dimensionality, the relative error made by taking $\mu_{\|y\|}$ instead of $\|y\|$ becomes negligible

[0] John A Lee and Michel Verleysen: ”Nonlinear Dimensionality Reduction”. Springer, 2007.

## Neighborhood Concept Become Meaningless

Implications from the concentration of distances:

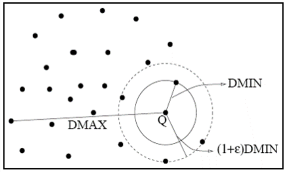
- A lot of data mining methods use distances and neighborhoods to define patterns (e.g. $k$NN classifier, density-based clustering, distance-based outlier detection, ...
- Using neighborhoods is based on a key assumption:
  - Objects that are similar to an object $o$ are in its neighborhood
  - Object that are dissimilar to $o$ are not in its neighborhood
- What if all objects are in the same neighborhood?
  - Consider the above effect on distances: $k$NN distances are almost equal to each other, i.e., the $k$ nearest neighbors are random objects

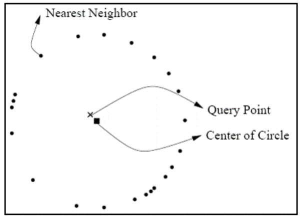
## Neighborhood Concept Become Meaningless

**Definition: Unstable Neighborhood**

- A NN-query is unstable for a given $\varepsilon$ if the distance from the query point to most data points is less than $(1 + \varepsilon)$ times the distance from the query point to its nearest neighbor

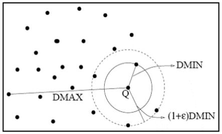
- It can be shown that with growing dimensionality, the probability that a query is unstable converges to 1





# Neighborhood Concept Become Meaningless

- Consider a $d$-dimensional query point $q$ and $n$ $d$-dimensional sample points $x_1, \dots x_n$ (independent and identically distributed)



- We define:
  $DMIN_d = \min\{L_2(x_i, q) | 1 \leq i \leq n\}$ (dist to next neighbor)
  $DMAX_d = \max\{L_2(x_i, q) | 1 \leq i \leq n\}$ (dist to farthest neighbor)

**Theorem**

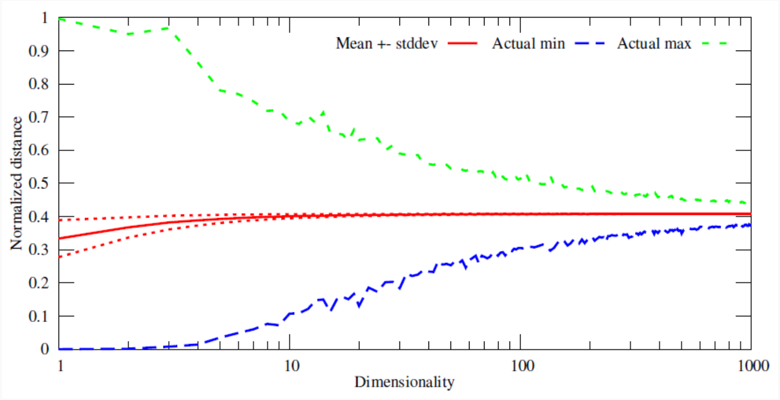
- If $\lim_{d\to\infty}(\frac{VAR_{L_2(x_i,q)}}{\mu^2_{L_2(x_i,q)}}) = 0$
- Then $\forall \varepsilon > 0 : \lim_{d\to\infty} \mathcal{P}(DMAX_d \leq (1+\varepsilon) DMIN_d) = 1$

In other words: if the precondition holds, all points converge to the same distance from the query!

[0] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft: When is "nearest neighbor" meaningful? In ICDT 1999.

## Neighborhood Concept Become Meaningless

Visually: Pairwise distances of a sample of 105 instances drawn from a uniform $[0, 1]$ distribution, normalized $(1/\sqrt{d})$.



## Neighborhood Concept Become Meaningless

- Be clear about the precondition of the Theorem!!!
- Consider the feature space of $d$ **relevant** features for a given application (i.e., truly similar objects display small distances in most features)
- Now add $d \cdot c$ additional features being independent of the initial feature space
- With increasing $c$ the distance in the independent subspace will dominate the distance in the complete feature space
- So the question is:
  How many relevant features must be similar to indicate object similarity?
  (or: how many relevant features must be dissimilar to indicate dissimilarity?)
- With increasing dimensionality the likelihood that two objects are similar in every respect gets smaller.

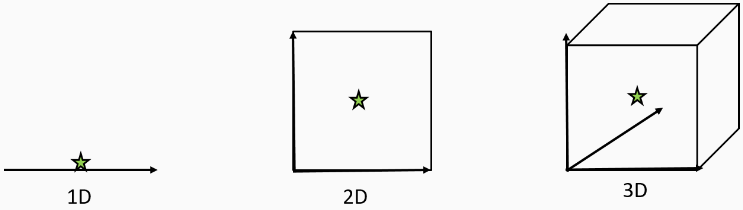
## Growing Data Space

- OK, the data space grows with increasing dimensionality
- But what are the problems?
- In low dimensional spaces we have some (intuitive) assumptions on the behavior of volumes (sphere, cube, etc.) and on the distribution of data objects
- However, basic assumptions do not hold in high dimensional spaces:
  - Spaces become sparse or even empty and the probability of one object inside a fixed range tends to become zero
  - Distribution of data has a strange behavior e.g. a normal distribution has only few objects in its center and the tails of distributions become more important

We will have a closer look on these issues ...

## Growing Hypotheses Space

- The more features, the larger the hypothesis space
- The lower the hypothesis space is,
  - the easier it is to find the correct hypothesis
  - the less examples you need to properly test hypothesis



## Growing Hypotheses Space

- Consider $f$ a unit multivariate normal distribution and normal kernel (KDE)
- The aim is to find an estimate $\hat{f}$ of $f$ at the point 0
- The relative mean square error should be fairly small, e.g. $\frac{\mu^2_{\hat{f}(0)-f(0)}}{f(0)^2} < 0.1$

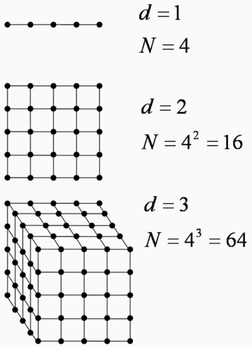
| Dim. | Req. sample size to achieve 0.1 error estimate |
|---|---|
| 1 | 4 |
| 2 | 19 |
| 5 | 768 |
| 8 | 43.700 |
| 10 | 842.000 |

Even with only 10 dimensions, we need nearly a million observations to estimate a distribution with an error less than 0.1!!!

[0] B.W. Silverman: "Density Estimation for Statistics and Data Analysis". Chapman and Hall/CRC, 1986.

## Empty Spaces and Tails

- Consider a $d$-dimensional space with partitions of constant size $1/m$
- The number of cells $N$ increases exponentially in $d$: $N = m^d$
- Suppose $x$ points are randomly placed in this space
- In low-dimensional spaces there are few empty partitions and many points per partitions
- In high-dimensional spaces there are far more partitions than points there are many empty partitions



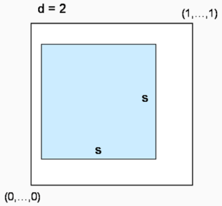
## Empty Spaces and Tails

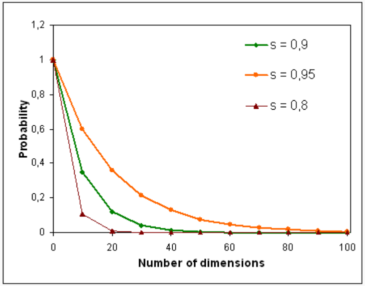
Analogously:

- Consider a simple partitioning scheme, which splits the data in each dimension in 2 halves
- For $d$ dimensions we obtain $2^d$ partitions
- Consider $n = 10^6$ samples in this space
- For $d \leq 10$ such a partition may make sense
- For $d = 100$ there are around $10^{30}$ partitions, so most partitions are empty (given the above $10^6$ points)

## Empty Spaces and Tails

- Consider a hyper-cube range query with length $s$ in all dimensions, placed arbitrarily in the data space $[0,1]^d$
- $E$ is the event that an arbitrary point lies within the query cube
- The probability for $E$ is $\mathcal{P}(E) = s^d$





$\Rightarrow$ with increasing dimensionality, even very large hyper-cube range queries are not likely to contain a point

## Empty Spaces and Tails

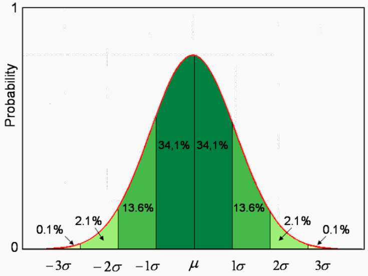
- The same holds of course for a spherical range query (instead of a cubical range query)
- Consequence: with increasing dimensionality the center of the hyper-cube (or more generally: of the data space) becomes less important and the volume of the data space concentrates in its corners (i.e. randomly distributed points tend to be on the border of the data space . . . )
- This seems to be a distortion of space compared to our 3D way of thinking — and that is actually what it is ...

## Empty Spaces and Tails

And that also means, that the tails of a distribution become extremely important

- Consider standard density function $f$
- Consider $\hat{f}$ with

$$\hat{f}(x) = \begin{cases} 0 & f(x) < 0.01 \\ f(x) & \text{else} \end{cases}$$



- Rescaling $\hat{f}$ to a density function will make very little difference in 1D, since very few data points occur in regions where $f$ is very small

## Empty Spaces and Tails

But for high dimensional data:

- More than half of the data has less then $1/100$ of the maximum density $f(0)$ (for $\mu = 0$)
- Example: 10-dimensional Gaussian distribution $X$:

$$\frac{f(X)}{f(0)} = e^{(-\frac{1}{2}X^TX)} \approx e^{(-\frac{1}{2}\chi^2_{10})}$$

  since the median of the $\chi^2_{10}$ distribution is 9.34, the median of $\frac{f(X)}{f(0)}$ is $e^{\frac{-9.34}{2}} = 0.0094$
- Thus, most objects occur at the tails of the distribution
- In other words, in contrast to the low dimensional case, regions of relatively very low density can be extremely important parts

## Empty Spaces and Tails

But for high dimensional data:

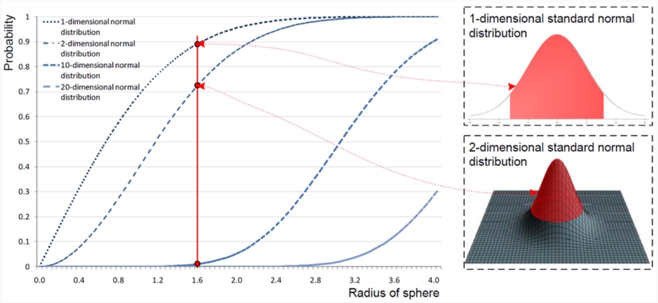
- More than half of the data has less then $1/100$ of the maximum density $f(0)$ (for $\mu = 0$)
- Example: 10-dimensional Gaussian distribution $X$:

$$\frac{f(X)}{f(0)} = e^{(-\frac{1}{2}X^T X)} \approx e^{(-\frac{1}{2}\chi^2{}_{10})}$$

  since the median of the $\chi^2{}_{10})$ distribution is 9.34, the median of $\frac{f(X)}{f(0)}$ is $e^{\frac{-9.34}{2}} = 0.0094$
- Thus, most objects occur at the tails of the distribution
- In other words, in contrast to the low dimensional case, regions of relatively very low density can be extremely important parts

## Empty Spaces and Tails

Example: $(\mu = 0, \sigma = 1)$



- 1D: 90% of the mass of the distribution lies between $-1.6$ and 1.6
- 10D: 99% of the mass of the distribution is at points whose distance from the origin is greater than 1.6
- Thus, it is difficult to estimate the density, except for enormous samples becausein very high dimensions virtually the entire sample will be in the tails

## Further Problems

- Patterns and models on high-dimensional data are often hard to interpret, e.g. long decision rules
- Efficiency in high-dimensional spaces is often limited because e.g. index structures degenerate and distance computations are much more expensive
- There may be different semantic layers so pattern might only be observable in subspaces or projected spaces (cf. the baby shape game)
- Cliques of correlated features dominate the object description

## The Case Kröger versus Tresp

- Summarizing: the higher the dimensionality, the worse is the expected outcome of the mining algorithm (i.e., dimensionality is a curse, says Kröger)
- Well, not in general, the Kernel trick shows the opposite: through the extension of the data space with new attributes, the mining algorithm (e.g. a SVM classifier) gets more accurate (i.e., dimensionality is a blessing, says Tresp in his ML course)
- So: Who is right???????? – Both – What????

## The Case Kröger versus Tresp

- Look at what we assumed for the curse: attributes are independent (and often even uniformly distributed)
- These attributes are likely to be irrelevant for the mining task
- And the blessing: a Kernel (if it works) adds relevant attributes (even more relevant than the original ones)
- Message: high-dimensional data is tricky and the curse can come by as several problems
    - Some are due to irrelevant attributes, so try to get rid of irrelevant attributes and keep the relevant ones
    - Some are instead of relevant attributes, so among the relevant attributes, try to get rid of redundant ones

# Kapitel 3: Feature Selection i

## Feature Selection

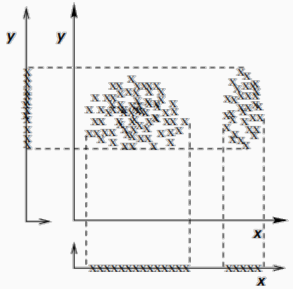
- A task to remove irrelevant and/or redundant features
  - Irrelevant features:
    - Not useful for a given task
    - Probably decrease accuracy
  - Redundant features:
    - Strongly correlated with another relevant feature
    - Does not drop the accuracy, but may drop efficiency, explainability, etc.
- Deleting irrelevant and redundant features can improve the quality as well as the efficiency of the methods and the found patterns.
- New feature space: Delete all useless features from the original feature space.

**Keep in mind...**

Feature selection $\neq$ Dimensionality reduction
Feature selection $\neq$ Feature extraction

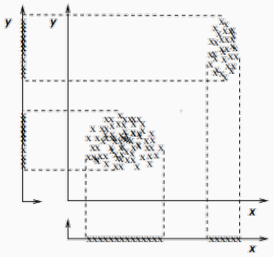
# Irrelevant and Redundant Features (Unsupervised Case)

**Irrelevance**



Feature *y* is irrelevant, because if we omit *x*, we have only one cluster, which is uninteresting.

**Redundancy**

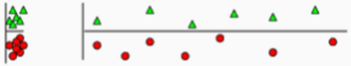


Features *x* and *y* are redundant, because *x* provides (appr.) the same information as feature *y* with regard to discriminating the two clusters

0 Source: Feature Selection for Unsupervised Learning, Dy and Brodley, Journal of Machine Learning Research 5 (2004)

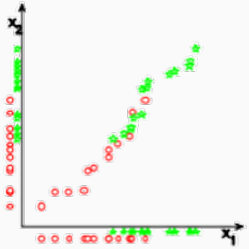
## Irrelevant and Redundant Features (Supervised Case)

**Irrelevance**



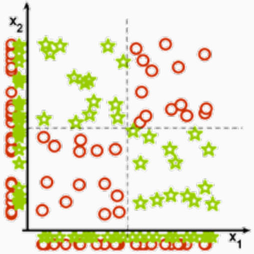
Feature $y$ separates well the two classes. Feature $x$ is irrelevant. Its addition “destroys” the class separation.

**Redundancy**



Features $x_1$ and $x_2$ are redundant.

**Individually irrelevant together relevant**



[0] Source: http://www.kdnuggets.com/2014/03/machine-learning-7-pictures.html

## Problem Definition

- **Input:** Vector space $F = d_1 \times \cdots \times d_n$, dimensions $D = \{d_1, \ldots, d_n\}$.
- **Output:** a minimal subspace $M$ over dimensions $D' \subseteq D$ which is optimal for a given data mining task.
    - Minimality increases the efficiency, reduces the effects of the curse of dimensionality and increases interpretability.

**Challenges:**

- Optimality depends on the given task.
- There are $2^d$ possible solution spaces (exponential complexity)
- This search space is similar to the frequent itemset mining problem, but:
    - There is often no monotonicity in the quality of subspace (which is important for efficient searching)
    - Features might only be useful in combination with other certain features.

$\Rightarrow$ For many popular criteria, feature selection is an exponential problem.

$\Rightarrow$ Most algorithms employ search heuristics.

## Two Main Components (Steps)

1. Feature subset generation
   - Single dimensions
   - Combinations of dimensions (subspaces)
2. Feature subset evaluation
   - Importance scores like information gain, $\chi^2$
   - Performance of a learning algorithm

$\Rightarrow$ How to select/evaluate features? How to traverse the search space?

## Feature Selection/Evaluation Methods

1. Filter methods
   – Explores the general characteristics of the data, independent of the learning algorithm.
2. Wrapper methods
   – The learning algorithm is used for the evaluation of the subspace.
3. Embedded methods
   – The feature selection is part of the learning algorithm.

## Feature Selection/Evaluation Methods

- Filter methods
  - Basic idea: assign an “importance” score to each feature to filter out useless ones
  - Examples: information gain, $\chi^2$-statistic, TF-IDF for text...
  - Disconnected from the learning algorithm.
  - Pros:
    - Fast and generic
    - Simple to apply
  - Cons:
    - Doesn’t take into account interactions between features
    - Individually irrelevant features, might be relevant together
    - Too generic?

## Feature Selection/Evaluation Methods

- Wrapper methods
  - A learning algorithm is employed and its performance is used to determine the quality of selected features.
  - Pros:
    - take feature dependencies into account
    - interaction between feature subset search and model selection
  - Cons:
    - higher risk of overfitting than filter techniques
    - very computationally intensive, especially if building the classifier has a high computational cost.

## Feature Selection/Evaluation Methods

- Embedded methods
  - Such methods integrate the feature selection in model building
  - Example: decision tree induction algorithm: at each decision node, a feature has to be selected.
  - Pros:
    - less computationally intensive than wrapper methods.
  - Cons:
    - specific to a learning method

## Search Strategies in the Feature Space

- Forward selection
  - Start with an empty feature space and add relevant features
- Backward selection
  - Start with all features and remove irrelevant features
- Branch-and-bound
  - Find the optimal subspace under the monotonicity assumption
- Randomized
  - Randomized search for a $k$ dimensional subspace
- ...

# Kapitel 3: Feature Selection

## General Idea

**Input**

- Target dimensionality $k \leq d$
- Training set of $n$-dimensional feature vectors with features $d_1, d_2, \ldots, d_n$ and target variable $C$

**General Approach**

- Compute the quality $q(d_i, C)$ for each dimension $d_i \in \{d_1, ..., d_n\}$ to predict the correlation to $C$
- Sort the dimensions $d_1, ..., d_n$ w.r.t. $q(d_i, C)$
- Select the best$k$ dimensions

**Basic Assumption**

- Attribute independence (no correlations between features)

## General Idea

**Key Concept**

- Quality of feature $d_i$: How suitable is the feature for predicting the value of class attribute $C$?
- Statistical measures
  - Rely on distributions over feature values and target values
  - How strong is the correlation between both value distributions?
  - How good does splitting the values in the feature space separate values in the target dimension?

## Quality of Features

How to measure the distribution?

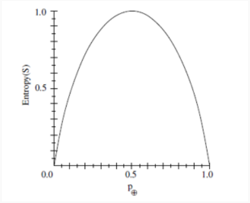
- For discrete values: determine probabilities for all value pairs.
- For real valued features:
  - Discretize the value space (reduction to the case above)
  - Use probability density functions (e.g. uniform, Gaussian,..)
- Example quality measures:
  - Information Gain
  - Chi-square $\chi^2$-statistics
  - Mutual Information

## Quality of Features: Entropy

- Idea: Evaluate class discrimination in each dimension (Used in ID3 algorithm for decision trees)
- It uses entropy, a measure of pureness of the data set $S$ w.r.t. the class labels $c_i \in C$

$$Entropy(S) = \sum_{c_i \in C} -p_{c_i} \cdot \log_2(p_{c_i})$$

where $p_{c_i}$ is the relative frequency of class $c_i$ in $S$



## Quality of Features: Entropy

### Example

- Let $S$ be a collection of positive and negative examples for a binary classification problem, i.e., $C = \{+, -\}$
- Then $\mathit{Entropy}(S) = -p_{+}\log_2(p_{+}) - p_{-}\log_2(p_{-})$
  - $p_{+}$ is the percentage of positive examples in $S$
  - $p-$ is the percentage of negative examples in $S$
- Example splits:
  - Let $S : [9+, 5-]$: $\mathit{Entropy}(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log(\frac{5}{14}) = 0.940$
  - Let $S : [7+, 7-]$: $\mathit{Entropy}(S) = -\frac{7}{14}\log_2(\frac{7}{14}) - \frac{7}{14}\log(\frac{7}{14}) = 1$
  - Let $S : [14+, 0-]$: $\mathit{Entropy}(S) = -\frac{14}{14}\log_2(\frac{14}{14}) - \frac{0}{14}\log(\frac{0}{14}) = 0$
- Obviously: Entropy is 0, when all samples belong to the same class while Entropy is 1, when there is an equal number of samples in all splits

## Quality of Features: Information Gain

- The information gain $Gain(S, d_i)$ of a feature $d_i$ relative to a training set $S$ measures the gain reduction in $S$ due to splitting on $d_i$, i.e., the entropy of the data set $S$ before splitting minus the weighted sum of the entropies of all splits $S_j$ in a given feature $d_i$:

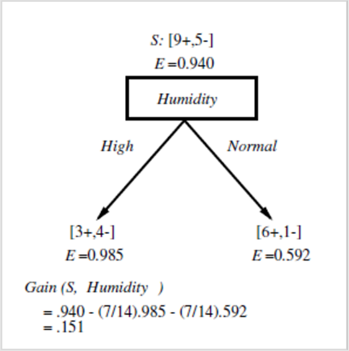
$$Gain(S, d_i) = Entropy(S) - \sum_{S_j} \frac{|S_j|}{|S|} \cdot Entropy(S_j)$$

- For nominal attributes: use attribute values for splitting, i.e. each possible value $v_j$ in $d_i$ defines one split and $S_j$ contains all objects having $v_j$ in $d_i$
- For real valued attributes: Determine a splitting position $v$ in the value set and split e.g. into $S_1$ containing all objects with values $\leq v$ and $S_2$ containing all objects with values $> v$ in $d_i$

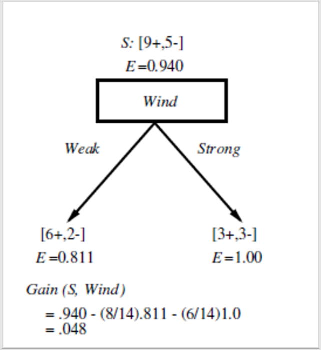
## Quality of Features: Information Gain

### Example

- Which dimension, “Humidity” or “Wind”, is better?



*Gain (S, Humidity)*
$= .940 - (7/14).985 - (7/14).592$
$= .151$

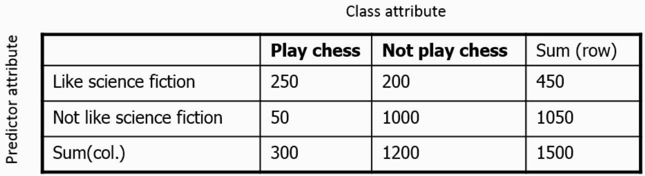


*Gain (S, Wind)*
$= .940 - (8/14).811 - (6/14)1.0$
$= .048$

- Larger values are better!

## Quality of Features: Chi-square Statistics

- Idea: Measures the independence of a feature *d* from the class variable *C*
- Contingency table: divide data based on a split value *s* or based on discrete values
- Example: Does “liking science fiction movies” imply “playing chess”?



Class attribute

Predictor attribute

| | **Play chess** | **Not play chess** | Sum (row) |
|---|---|---|---|
| Like science fiction | 250 | 200 | 450 |
| Not like science fiction | 50 | 1000 | 1050 |
| Sum(col.) | 300 | 1200 | 1500 |

- Chi-square $\chi^2$ test

$$\chi^2 = \sum_{i=1}^{|C|} \sum_{j=1}^{|Values(d)|} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

$o_{ij}$: observed freq. of value *j* in class *i*
$e_{ij}$: expected freq. of value *j* in class *i*

## Quality of Features: Chi-square Statistics

### Example

- Compute the $\chi^2$ values for the following table (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

Class attribute

Predictor attribute

| | **Play chess** | **Not play chess** | Sum (row) |
|---|---|---|---|
| Like science fiction | 250 (90) | 200 (360) | 450 |
| Not like science fiction | 50 (210) | 1000 (840) | 1050 |
| Sum(col.) | 300 | 1200 | 1500 |

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{1000-840)^2}{840} = 507.93$$

- Smaller values are better!

## Quality of Features: Mutual Information

- In general, the Mutual Information (MI) between two variables $x$ and $y$ measures how much knowing one of these variables reduces uncertainty about the other
- In our case, it measures how much information a feature contributes to making the correct classification decision, i.e., $x$ is the dimension $d_i$ we want to evaluate and $y$ is the class variable $C$.
- MI is based on probability distributions:
  - $p(x)$ and $p(y)$ are the marginal probability distributions of $x$ and $y$, respectively
  - $p(x, y)$ is the joint probability distribution function

## Quality of Features: Mutual Information

- Discrete case

$$MI(x,y) = \sum_{x_i \in x} \sum_{y_i \in y} p(x_i, y_i) \cdot \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}$$

- Continuous case

$$MI(x,y) = \int_x \int_y p(x,y) \cdot \log \frac{p(x,y)}{p(x)p(y)} dxdy$$

- Interpretation: if *x* and *y* are statistically independent, then
    - $p(x,y) = p(x) \cdot p(y)$ and, thus, $\log(1) = 0$
    - Or in other words: knowing *x* does not reveal anything about *y*

## Forward Selection and Feature Ranking: Discussion

**Advantages**

- Efficiency: it compares each feature $\{d_1, d_2, \ldots, d_n\}$ separately to the class attribute $C$ (and takes the best $k$) instead of testing $\binom{n}{k}$ subspaces
- Works already for rather small sample sizes

**Limitations**

- Independency assumption: Classes and features must display a direct correlation
- In case of correlated features: Always selects the features having the strongest direct correlation to the class variable, even if the features are strongly correlated with each other

# Kapitel 3: Feature Selection

## Backward Elimination: General Idea

### General Approach

- Start with the complete feature space and delete redundant features
- Greedy Backward Elimination
  1. Generate the subspaces $R$ of the feature space $F$
  2. Evaluate subspaces $R$ with the quality measure $q(R)$
  3. Select the best subspace $R*$ w.r.t. $q(R)$
  4. If $R*$ has the target dimensionality, terminate else start backward elimination on $R*$.

### Remarks

- Useful in supervised and unsupervised setting (in the latter scenario, $q(R)$ measures structural characteristics)
- Greedy search if there is no monotonicity on $q(R)$; for monotonous measures, branch and bound can be employed

## Supervised Quality Measure: Distance-based

- Idea: Subspace quality can be evaluated by the distance between the within-class nearest neighbor and the between-classes nearest neighbor
- Quality criterion: For each object $o$ from the data set $S$, compute distance to the closest object having the same class $NN^R_{c_i=C(o)}(o)$ (within-class nearest neighbor distance) in subspace $R$, and to the closest object belonging to another class $NN^R_{c_j \neq C(o)}(o)$ (between-classes nearest neighbor distance), where $C(o)$ denotes the class label of object $o$ in subspace $R$:

$$q(R) = \frac{1}{S} \cdot \sum_{o \in S} \frac{NN^R_{c_j \neq C(o)}(o)}{NN^R_{c_i=C(o)}(o)}$$

- Remark: $q(R)$ is not monotonous: by deleting a dimension, the quality can increase or decrease

## Supervised Quality Measure: Model-based

- Idea: Directly employ the data mining algorithm to evaluate the subspace, e.g. by training a Naive Bayes classifier
- Practical aspects:
  - Success of the data mining algorithm must be measurable (e.g. class accuracy)
  - Runtime for training and applying the classifier should be low
  - The classifier parameterization should not be of great importance
  - Test set should have a moderate number of instances

## Backward Elimination: Discussion

**Advantages**

- Considers complete subspaces (multiple dependencies are used)
- Can recognize and eliminate redundant features

**Limitations**

- Tests w.r.t. subspace quality usually requires much more effort
- All solutions employ heuristic greedy search which do not necessarily find the optimal feature space

## Branch and Bound: General Idea

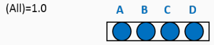
### General Approach

- Given: A classification task over the feature space $F$
- Aim: Select the $k$ best dimensions to learn the classifier
- Backward elimination approach “Branch and Bound” is guaranteed to find the optimal feature subset under the monotonicity assumption
- The monotonicity assumption states that for two feature subsets $X, Y \in F$ and a feature selection criterion $J$, if $X \subset Y$ then
  - $J(X) \leq J(Y)$ if $J$ is maximized
  - $J(X) \geq J(Y)$ if $J$ is minimized
- Branch and Bound starts from the full set $F$ and removes features using a depth-first strategy
- Nodes whose objective function are smaller (greater) than the current best are not explored since the monotonicity assumption ensures that their children will not contain a better solution

## Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality *d* = 1.

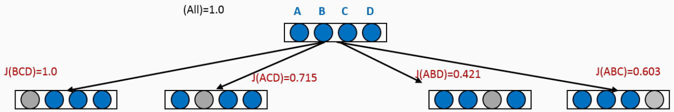
● selected feature ● removed feature



## Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality *d* = 1.

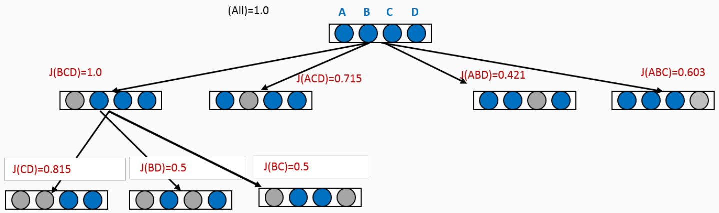
● selected feature ● removed feature



## Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality $d$ = 1.

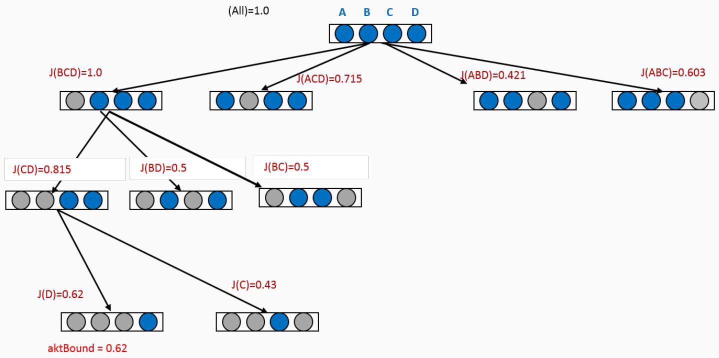
● selected feature ● removed feature



## Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality *d* = 1.

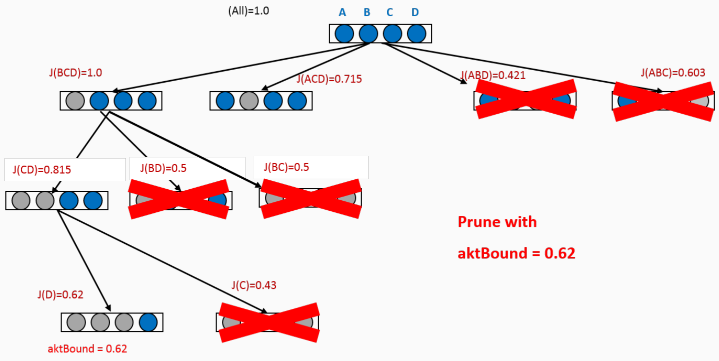
● selected feature ● removed feature



# Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality *d* = 1.

● selected feature ● removed feature



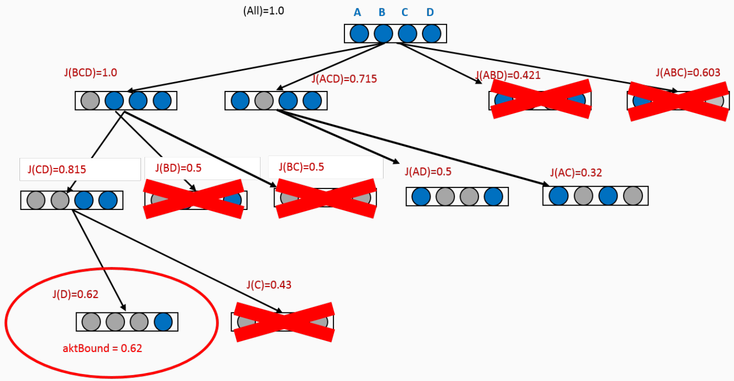
**Prune with**

**aktBound = 0.62**

## Branch and Bound: Example

Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality *d* = 1.

● selected feature ● removed feature



## Branch and Bound: Quality Measures

### Subspace Inconsistency (IC)

- Given a data set $S$ (works best for categorical data)
- Idea: Having identical vectors $u, v$ ($u_i = v_i, 1 \leq i \leq d$) in subspace $R$ but the class labels are different $(C(u) \neq C(v))$, this subspace displays an inconsistent labeling
- Measuring the inconsistency of a subspace $R$
  - $X_R(u)$: Amount of all identical vectors $u$ in $R$
  - $X_R^c(u)$: Amount of all identical vectors $u$ in $R$ having class label $c \in C$
  - Inconsistency of $u$ in $R$: $\quad IC_R(u) = X_R(u) - \max_{c \in C} X_R^c(u)$

  Then, inconsistency of subspace $R$ is

$$IC(R) = \frac{\sum_{u \in S} IC_R(u)}{|S|}$$

- Monotonicity: $R_1 \subset R_2 \Rightarrow IC(R_1) \geq IC(R_2)$

## Backward Elimination: Discussion

**Advantages**

- Monotonicity allows efficient search for optimal solutions
- Well-suited for binary or discrete data (identical vectors are very likely with decreasing dimensionality)

**Limitations**

- Useless without groups of identical features (real-valued vectors)
- Worse-case runtime complexity remains exponential in the number of features $d$

## Random Subspace Selection: General Idea

### General Approach

- Idea: Select $n$ random subspaces having the target dimensionality $k$ out of the $\binom{d}{k}$ many possible subspaces and evaluate each of them
- Needs quality measures for complete subspaces
- Trade-off between quality and effort depends on $n$
- Good alternative to forward selection if quality measure is not monotonic

- Different randomization approaches exist (see next subsection):
  - Genetic algorithms
  - $k$-medoids feature clustering
  - ...

# Kapitel 3: Feature Selection

## Genetic Algorithms: General Idea

### General Approach

- Idea: Randomized search through genetic algorithms
- Genetic Algorithms encode individual states in the search space as bit-strings
- Population (of current solutions) is a subset of all possible $k$-dimensional subspaces
- Fitness function: quality measure for a subspace
- Algorithmic schema to find the best solution in the search space by mixing/changing the population in each iteration (stops e.g. if the best solution of the current population is less fit than the best solution in the previous population)
- Each iteration manages a specific population from which the next population is obtained

## Genetic Algorithms: Population Generation

- Operators on the population ($k$-dim subspaces) to create candidates for the next population:
  - Mutation: dimension $d_i$ in subspace $R$ is replaced by dimension $d_j$ with a likelihood of $x\%$
  - Crossover: combine two subspaces $R_1$ and $R_2$, i.e., unite the features sets of $R_1$ and $R_2$ and delete random dimensions until dimensionality is $k$ again
- Selection for next population: All subspaces having at least a quality of $y\%$ of the best fitness in the current generation are copied to the next generation
- Free tickets: Additionally each subspace is copied into the next generation with a probability of $u\%$
- Remark: Many variants on the basic algorithmic schema, e.g. different operations, efficient convergence by “Simulated Annealing” (likelihood of free tickets decreases with the iterations), ...

## Genetic Algorithms: Discussion

**Advantages**

- Can escape from local optima during the search
- Often good approximations of the optimal solutions

**Limitations**

- Runtime ( is not bounded (in the original schema)
- Configuration depends on many parameters which have to be tuned to achieve good quality results in efficient time

## Feature Clustering: General Idea

### General Approach

- Given: A feature space $F$ and an unsupervised data mining task
- Target: Reduce $F$ to a subspace of $k$ (original) dimensions while reducing redundancy
- Idea: Cluster the features in the space of objects and select one representative feature for each of the clusters (this is equivalent to clustering in a transposed data matrix)
- Problem: often many more samples than features so transposed data matrix has many more features than samples

## Feature Clustering: Example

- Typical example: item-based collaborative filtering
- E.g. features 3 and 4 are similar over all persons so they could be "merged" to one feature

| | 1 (Titanic) | 2 (Braveheart) | 3 (Matrix) | 4 (Inception) | 5 (Hobbit) | 6 (300) |
|---|---|---|---|---|---|---|
| Susan | 5 | 2 | 5 | 5 | 4 | 1 |
| Bill | 3 | 3 | 2 | 1 | 1 | 1 |
| Jenny | 5 | 4 | 1 | 1 | 1 | 4 |
| Tim | 2 | 2 | 4 | 5 | 3 | 3 |
| Thomas | 2 | 1 | 3 | 4 | 1 | 4 |

## Feature Clustering: Example

- Work around for the “many features” problem: specialized feature similarity measures, e.g.
  - Cosine similarity
  - Pearson correlation
- Algorithmic schema
  - Cluster features with a *k*-medoid clustering method based on correlation
  - Select the medoids to span the target data space
- Remark
  - For group/cluster of dependent features there is one representative feature
  - Other clustering algorithms could be used as well, e.g. approximate clustering methods for performance reasons

## Feature Clustering: Discussion

**Advantages**

- Depending on the clustering algorithm quite efficient
- Unsupervised method

**Limitations**

- Results are usually not deterministic (partitioning clustering results depend on initialization)
- Representatives are usually unstable for different clustering methods and parameters
- Method captures pairwise correlations and dependencies among features but multiple dependencies are not considered

## Summary: Feature Selection

- Forward-Selection examines each dimension separately and selects the $k$-best to span the target space
    - Greedy Selection based on Information Gain, $\chi^2$ statistics or Mutual Information
- Backward-Elimination start with the complete feature space and successively remove the worst dimensions
    - Greedy Elimination with model-based and nearest-neighbor based approaches
    - Branch and Bound Search (monotonicity required!) based on inconsistency
- $k$-dimensional Projections directly search in the set of $k$-dimensional subspaces for the best suited
    - Genetic algorithms (any quality measures possible, e.g. those from backward elimination)
    - Feature clustering based on correlation

## Discussion: Feature Selection

- Many algorithms based on different heuristics
- There are two reason to delete features:
  - Redundancy: Features can be expressed by other features
  - Missing correlation to the target variable
- Often even approximate results are capable of increasing efficiency and quality in a data mining tasks
- Caution: Selected features need not to have a causal connection to the target variable, but both might depend on the same mechanisms in the data space (hidden variables)
- Different indicators to consider in the comparison of before and after selection performance, e.g. model performance, time, dimensionality, ...

## Feature Selection — Further Readings

- I. Guyon, A. Elisseeff: An Introduction to Variable and Feature Selection, Journal of Machine Learning Research 3, 2003.
- H. Liu and H. Motoda, Computations methods of feature selection, Chapman & Hall/ CRC, 2008.
- A.Blum and P. Langley: Selection of Relevant Features and Examples in Machine Learning, Artificial Intelligence (97),1997.
- H. Liu and L. Yu: Feature Selection for Data Mining (WWW), 2002.
- L.C. Molina, L. Belanche, Â. Nebot: Feature Selection Algorithms: A Survey and Experimental Evaluations, ICDM 2002, Maebashi City, Japan.
- P. Mitra, C.A. Murthy and S.K. Pal: Unsupervised Feature Selection using Feature Similarity, IEEE Transacitons on pattern analysis and Machicne intelligence, Vol. 24. No. 3, 2004.
- J. Dy, C. Brodley: Feature Selection for Unsupervised Learning, Journal of Machine Learning Research 5, 2004.
- M. Dash, H. Liu, H. Motoda: Consistency Based Feature Selection, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, 2000.

# Kapitel 4: Feature Reduction and Metric Learning i

## Kapitel 4: Feature Reduction and Metric Learning ii

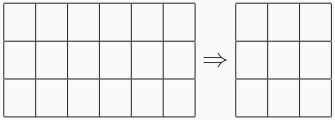
5. Clustering High Dimensional Data

## Overview

- Idea: Instead of removing features, try to find a low dimensional feature space generating the original space as accurate as possible:
  - Redundant features are summarized
  - Irrelevant features are weighted by small values or are “erased” (in the best case of course, the new feature space should contain no irrelevant features anymore)
- Some sample methods (among lots of others):
  - Reference point embedding
  - Principal component analysis (PCA)
  - Singular value decomposition (SVD)
  - Fischer-Faces (FF) and Relevant Component Analysis(RCA)
  - Large Margin Nearest Neighbor (LMNN)

## Feature Reduction Task

- **Goal**: Describe data with fewer features (reduce number of columns)
- Be clear: (like in feature selection) there will always be an information loss



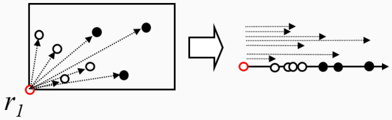
- There are supervised and unsupervised methods

# Kapitel 4: Feature Reduction and Metric Learning

## General Approach

- Idea: Describe the position of each object by their distances to a set of reference points
- Given: Vector space $F = D_1 \times ... \times D_n$ where $D = \{D_1, ..., D_n\}$
- Target: A $k$-dimensional space $R$ which yields optimal solutions for a given data mining task
- Method: For each reference point $R = \{r_1, ..., r_k\}$ and a distance measure *dist*, transform vector $x \in F$ as follows:

$$r_R(x) = \begin{pmatrix} dist(r_1, x) \\ \vdots \\ dist(r_k, x) \end{pmatrix}$$



## Diskussion

- Distance measure is usually determined by the application
- Selection of reference points can be important (use centroids of the classes or cluster-centroids, points on the margin of the data space, use random samples, ...)

**Advantages**

- Simple approach which is easy to implement
- The transformed vectors yields lower and upper bounds of the exact distances (What the hell is that good for???)

**Disadvantages**

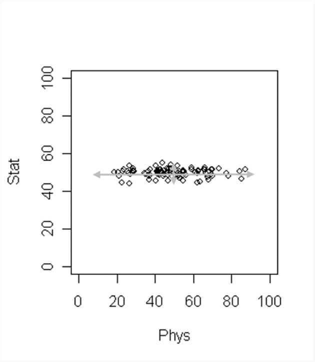
- Even using *d* reference points does not reproduce a *d*-dimensional feature space
- Selecting good reference points is important but very difficult

# Kapitel 4: Feature Reduction and Metric Learning

## Introduction

### Motivation

- 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?



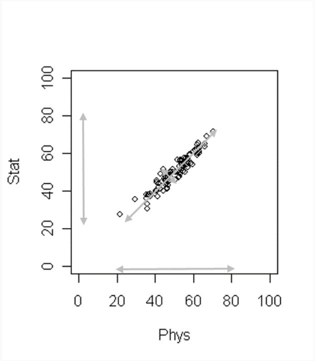
Answer:
Physics because the variation along that axis is larger

Source: `http://astrostatistics.psu.edu/su09/lecturenotes/pca.html`

## Introduction

### Motivation

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



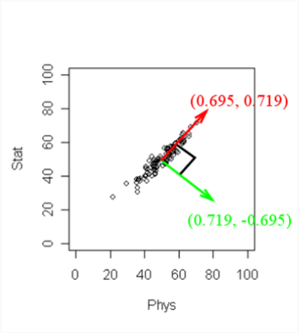
Answer:
We should take a linear combination of the two grades (that represents the direction of highest variance) to get the best results

Source: `http://astrostatistics.psu.edu/su09/lecturenotes/pca.html`

## Introduction

### Motivation

- 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 the first



Source: `http://astrostatistics.psu.edu/su09/lecturenotes/pca.html`

## Normalizing Data

A feature $X$ can be normalized by substracting its values with the mean $\bar{X}$ and dividing by the standard deviation $s_X$, e.g. $\tilde{X} = \frac{X - \bar{X}}{s_X}$.

**Example**:

Consider the following body heights measured in different units:

| | Person A | Person B | Person C | mean | sd |
|---|---|---|---|---|---|
| body height (cm) | 180.00 | 172.00 | 175.00 | 175.67 | 4.04 |
| body height (m) | 1.80 | 1.72 | 1.75 | 1.76 | 0.04 |
| body height (feet) | 5.91 | 5.64 | 5.74 | 5.76 | 0.13 |

After normalizing, we always obtain the normalized body height (no matter which unit we used):

| | Person A | Person B | Person C | mean | sd |
|---|---|---|---|---|---|
| normalized body height | 1.07 | -0.91 | -0.16 | 0.00 | 1.00 |

## Normalizing Data

Normalizing all features in a data set, can have several advantages:

- It puts all features into *comparable* units, i.e., we make sure that all normalized features have mean 0 and standard deviation of 1
- It can avoid numerical instabilites in several algorithms, e.g. if a feature has very low / high values
- It helps in computing meaningful *distances* between observations
- Finally, if we want to find directions of highest variances, it might be better to do this on normalized data

## Normalizing: Covariance vs. Correlation

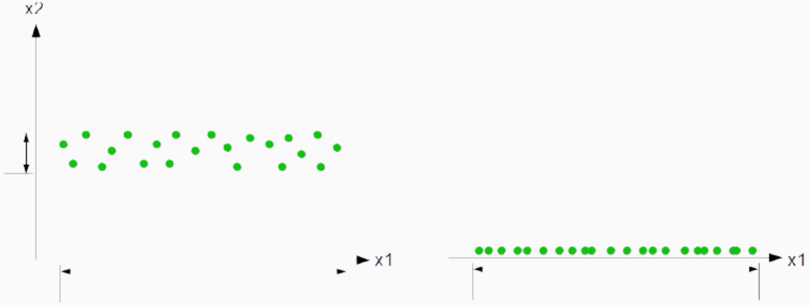
- Sure, there are many ways to do normalization
- here, we will use the notion common in Statistics, where the **variance** of a normalized feature is always 1, its mean is always 0
- The **covariance** of two normalized features $\tilde{X} = \frac{X-\bar{X}}{s_X}$ and $\tilde{Y} = \frac{Y-\bar{Y}}{s_Y}$ is the same as the **correlation** of the non-normalized features $X$ and $Y$.
- One can proof this with the help of

$$s_{\tilde{X}\tilde{Y}} = \frac{1}{n-1} \sum_{i=1}^{n} (\tilde{x}_i - \bar{\tilde{x}})(\tilde{y}_i - \bar{\tilde{y}}) = \ldots = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{s_X} \frac{(y_i - \bar{y})}{s_Y} = r_{XY}.$$

## PCA Intuition

*Example I*:

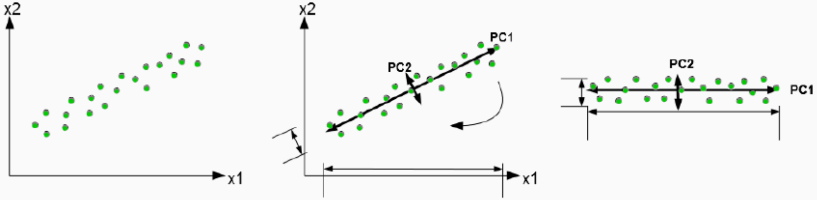
- Feature $x_1$ explains most of the variation
- Feature $x_2$ has a lower variance than $x_1$
- If we disregard $x_2$ and project the points into the 1-dimensional space of $x_1$, we do not lose much information w.r.t. variability



## PCA Intuition

*Example II*:

- $x_1$ and $x_2$ are correlated and have similar variances.
- Find a new orthogonal axes (e.g. PC1 and PC2), where PC1 explains most of the variation
- Rotate the points and consider PC1 and PC2 as new coordinate system (situation as in the previous example)
- We can now project points onto PC1 and disregard PC2 (hopefully without losing much information)



## PCA Intuition

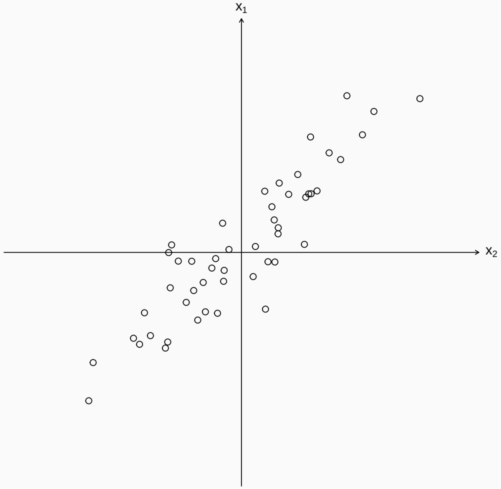
- PCA finds the optimal rotation such that the transformed data explains the variability of the data best
- The new axis are the principal components (also called "eigenvectors" because PCA is technically an Eigen-Decomposition); for a $d$-dimensional data set we always get $d$ principal components
- The variance along each eigenvector (called "eigenvalue") is decreasing, i.e. the first eigenvector has the highest eigenvalue, while the $d$-th eigenvector has the smallest eigenvalue
- This can be used for dimensionality reduction: if we pick the $k$-th first eigenvectors as new axes and transform the $d$-dimensional data into the new $k$-dimensional space, this transformation is optimal w.r.t. loss of total variance

## PCA Procedure

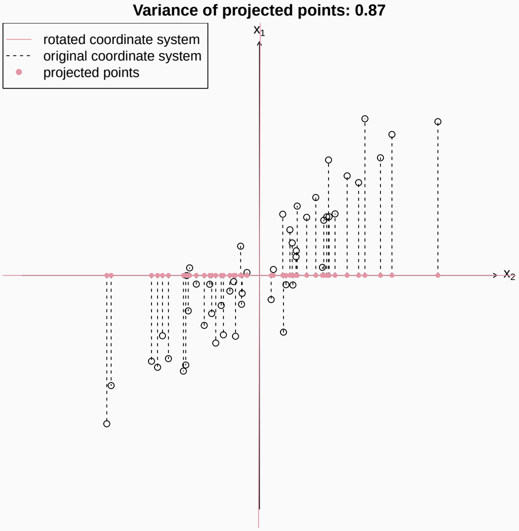
### General procedure

1. Rotate the original *p*-dimensional coordinate system until the first PC that explains most of the variation is found
2. Fix the first PC and proceed with rotating the remaining $p - 1$ coordinates until the second PC (which is orthogonal to the first PC) is found that explains most of the *remaining* variation, etc.
3. We can reduce the dimensions by projecting the points onto the first, say $k < p$, PC

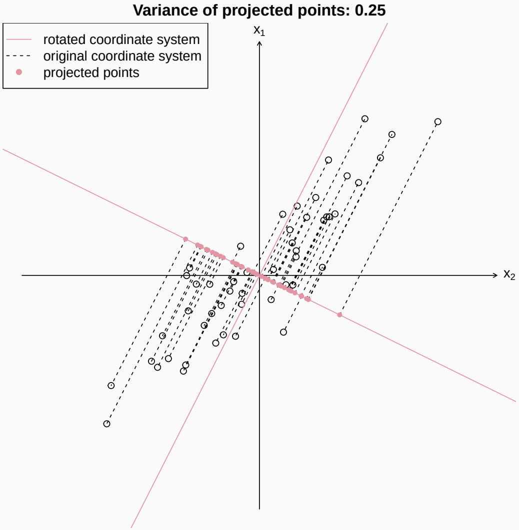
## PCA Intuition: Find first PC



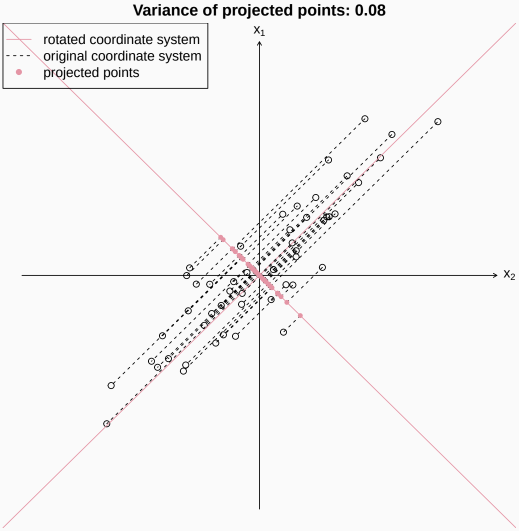
## PCA Intuition: Animation



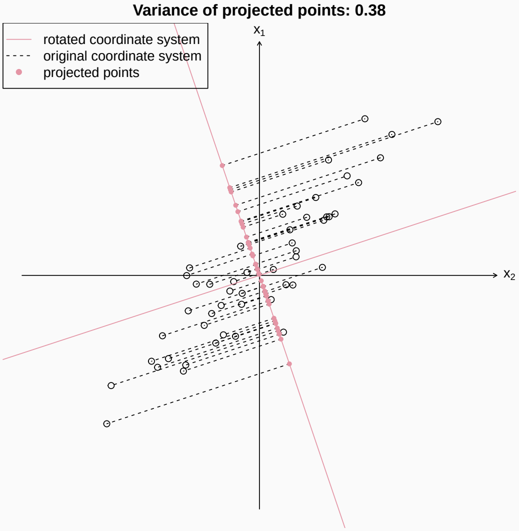
# PCA Intuition: Animation



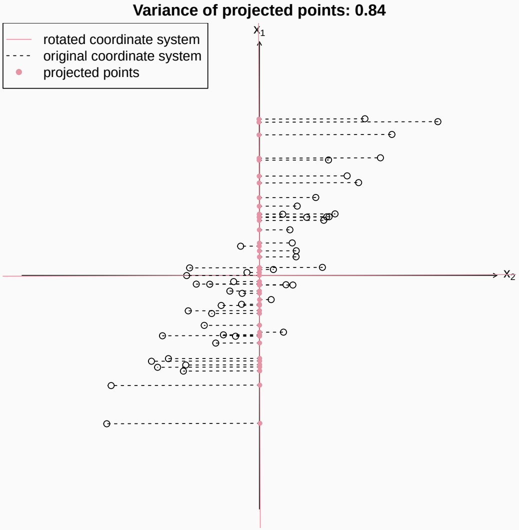
## PCA Intuition: Animation



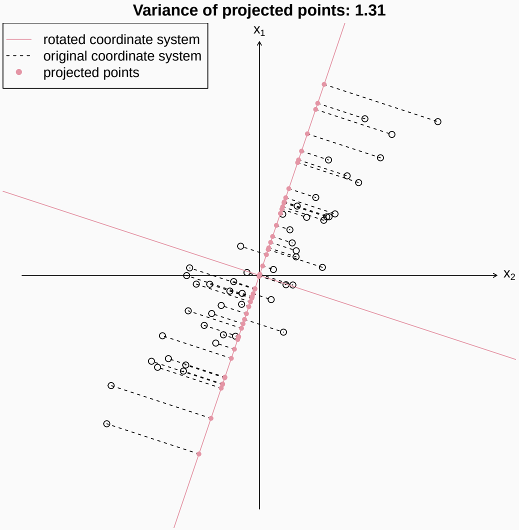
# PCA Intuition: Animation



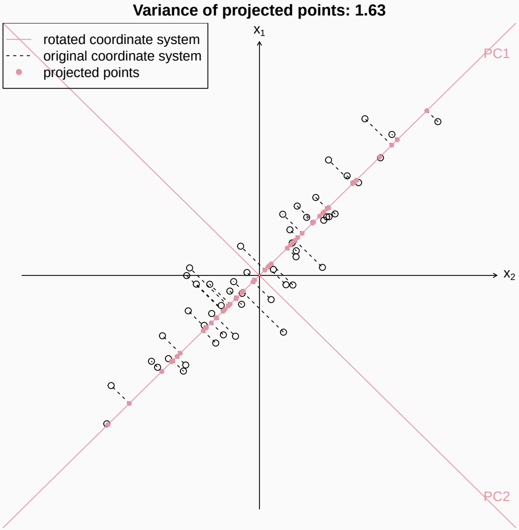
## PCA Intuition: Animation



# PCA Intuition: Animation



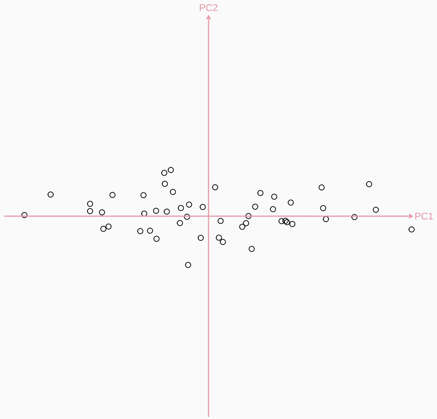
## PCA Intuition: Animation



## PCA Intuition: Reduce dimensionality

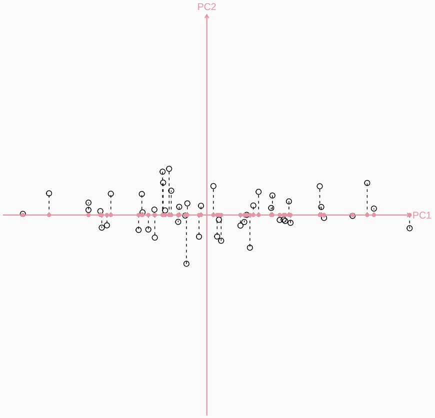
Rotate the points and use PC1 and PC2 as new coordinate system.

Here, the PC1 axis explains most of the variance:



# PCA Intuition: Reduce dimensionality

Dimensionality can be reduced by projecting the points onto the PC1 (and by disregarding PC2). The hope is that we won't lose much information this way.



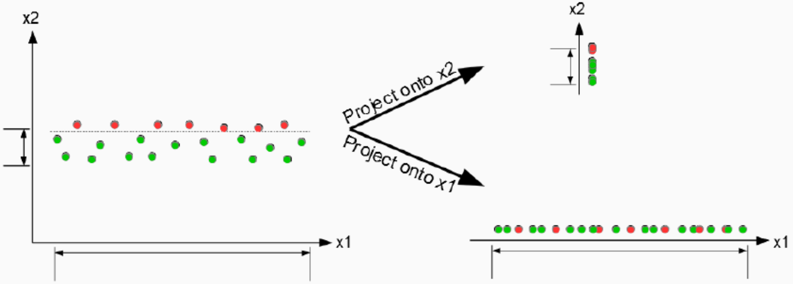
## PCA Intuition: Summary

**Idea:** Transform an original set of correlated metric features to a new set of uncorrelated (orthogonal) metric features, called principal components (PC), that explain the variability in the data.

- The objective is to investigate if only a few PC account for most of the variability in the original data.
- If the objective is fulfilled, we can use fewer PCs to reduce the dimensionality.
- The PCs remove collinearity of the input variables as they are orthogonal to each other.

## PCA Intuition: Final Remarks

- PCA is used for dimensionality reduction by disregaring dimensions with lower variability.
- There is always an information loss, especially for other criteria.
- **Attention:** dimensionality reduciton can worsen the classification accuracy when the task is to classify two groups:



## Deriving the First PC Mathematically

Aim: Find a new set of features (PC scores, eigenvectors) $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ based on the original data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ so that

- each PC score $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ is a linear combination of the original metric features with coefficient weights (so-called **loading vectors**) $\mathbf{a}_1, \ldots, \mathbf{a}_p$, i.e.

$$\mathbf{pc}_j = a_{j1}\mathbf{x}_1 + a_{j2}\mathbf{x}_2 + \ldots + a_{jp}\mathbf{x}_p = \mathbf{X}\mathbf{a}_j.$$

- the set is mutually uncorrelated: $Cov(\mathbf{pc}_j, \mathbf{pc}_k) = 0, \forall j \neq k$.
- the variances (eigenvalues) of the PC scores decrease:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_p, \quad \text{where } \lambda_k := Var(\mathbf{pc}_k).$$

## Deriving the First PC Mathematically

We look for the loading vector $\mathbf{a}_1 = (a_{11}, a_{21}, \ldots, a_{p1})^\top$ that maximizes the variance of $\mathbf{pc}_1$:

$$\max_{\mathbf{a}_1} \; Var(\mathbf{pc}_1) = Var(\mathbf{X}\mathbf{a}_1) = \mathbf{a}_1^\top \Sigma \mathbf{a}_1$$

subject to the normalization constraint $\mathbf{a}_1^\top \mathbf{a}_1 = \sum_{k=1}^{p} a_{k1}^2 = 1$.

The constraint is required for identifiability reasons, otherwise we could maximize the variance by just increasing the values in $\mathbf{a}_1$.

Repeat this maximization step for the other PCs and additionally use the orthogonality constraint, i.e. for the second PC:

$$\mathbf{a}_2^\top \mathbf{a}_1 = 0.$$

## Example: The Olympic Heptathlon Data

The `heptathlon` data set (e.g. available in the R package `HSAUR3`) contains the competition results of 25 athletes in 7 disciplines for the Olympics held in Seoul in 1988.

- **Aim**: Rank the athletes according to their overall performance in all 7 disciplines.
- **Idea**: Use PCA to reduce the dimensionality (i.e., reduce the results of the 7 disciplines to one dimension) and compare the scores of the first PC with the official scores.

## Example: The Olympic Heptathlon Data

Features of the `heptathlon` data:

- `hurdles`: results 100m hurdles (in seconds).
- `highjump`: results high jump (in m).
- `shot`: results shot putt (in m).
- `run200m`: results 200m race (in seconds).
- `longjump`: results long jump (in m).
- `javelin`: results javelin (in m).
- `run800m`: results 800m race (in seconds).
- `score`: total score of the official scoring system.

## Example: The Olympic Heptathlon Data

The features `hurdles`, `run200m` and `run800m` are time measurements, i.e. low values are better. For all other features high values are better.

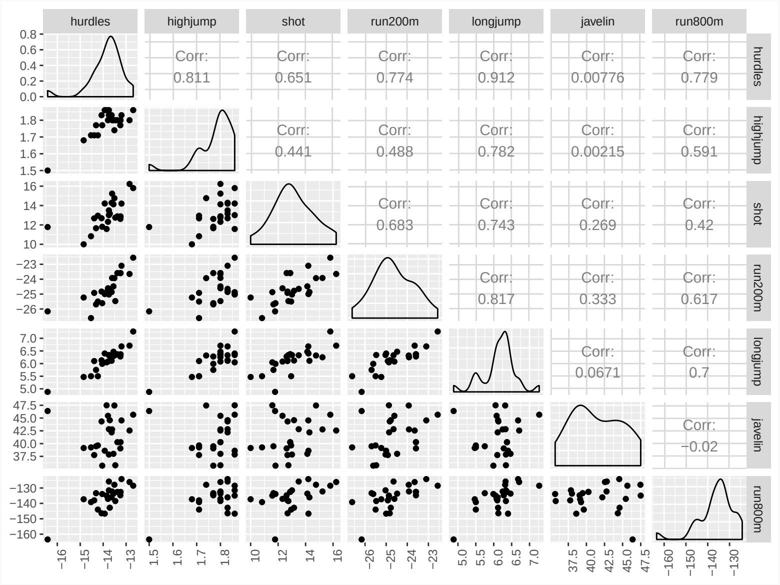
Results of the best and worst participant:

| | hurdles | highjump | shot | run200m | longjump | javelin | run800m | score |
|---|---|---|---|---|---|---|---|---|
| Joyner-Kersee (USA) | 12.7 | 1.86 | 15.8 | 22.6 | 7.27 | 45.7 | 129 | 7291 |
| Launa (PNG) | 16.4 | 1.50 | 11.8 | 26.2 | 4.88 | 46.4 | 163 | 4566 |

We use negative time measurements so that higher values are better and therefore all features have the same direction:

| | hurdles | highjump | shot | run200m | longjump | javelin | run800m | score |
|---|---|---|---|---|---|---|---|---|
| Joyner-Kersee (USA) | -12.7 | 1.86 | 15.8 | -22.6 | 7.27 | 45.7 | -129 | 7291 |
| Launa (PNG) | -16.4 | 1.50 | 11.8 | -26.2 | 4.88 | 46.4 | -163 | 4566 |

## Scatter Plot Matrix



## Perform PCA

- If features are on very different scales, PCA should be carried out on the correlation matrix (which is equivalent to the correlation matrix if normalized features are used).
- As the features of the `heptathlon` data are on different scales, we perform the PCA based on the correlation matrix.
- Alternatively, we could also perform the PCA based on the covariance matrix but on the normalized `heptathlon` data.
- The result contains:
  - The loadings $\mathbf{a}_1, \dots, \mathbf{a}_p$,
  - The PC scores $\mathbf{pc}_1, \dots, \mathbf{pc}_p$ and
  - The variance $\lambda_1, \dots, \lambda_p$ (or standard deviation) of the PC scores.

## Proportion of Explained Variance

- The total variance of the $p$ PC scores is equal the total variance of the original features, i.e.,

$$\sum_{j=1}^{p} \lambda_j = s_1^2 + s_2^2 + \cdots + s_p^2,$$

where $\lambda_j$ is the variance of the $j$th PC and $s_j^2$ is the sample variance of variable $\mathbf{x}_j$.

- The proportion of explained variance of the $j$-th PC is

$$\frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

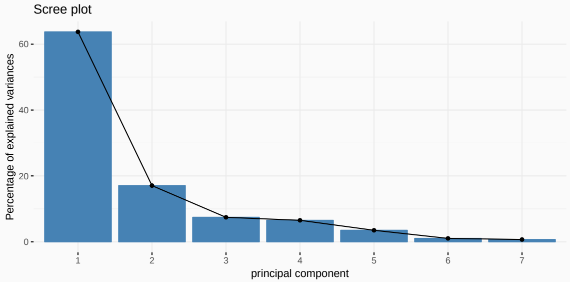
- The first $k$ PCs account for a proportion

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

## Choosing the Number of PCs

Two simple rules of thumb for choosing the number of PCs:

1. Retain the first *k* components, which explain a large proportion of the total variation, e.g., 80-90%.
2. Use a scree plot: Plot the component variances vs. the component number and look for an *elbow*. For components after the *elbow*, the variance decreases more slowly.



## PC Scores vs. Official Scores

The first PC explains $63,72\%$ of the variation of the `heptathlon`, the loadings of the first PC are:

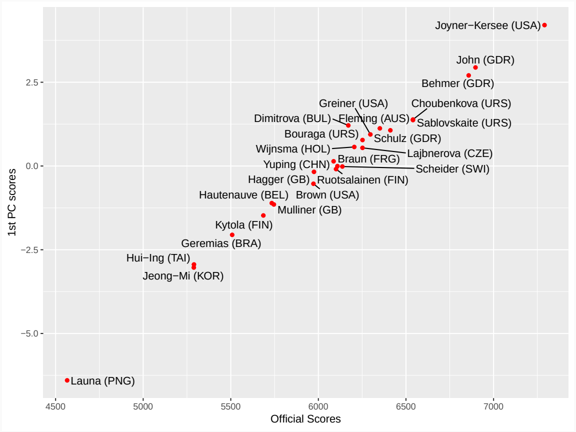
| hurdles | highjump | shot | run200m | longjump | javelin | run800m |
|---|---|---|---|---|---|---|
| 0.4529 | 0.3772 | 0.3631 | 0.4079 | 0.4562 | 0.0754 | 0.3750 |

Dimensionality reduction:

- Project all 8 features onto the first PC.
- Compare the scores of the first PC with the official scores used to rank the athletes.

# PC Scores vs. Official Scores

The scores of the first PC $\mathbf{pc}_1$ have a similar ranking as the scores of the official scoring system:



## Discussion

**Advantage**

- Considers arbitrary correlations between features
- Selected subspace is optimal w.r.t. loss of variance

**Disadvantage**

- Assumption: components with high variance are useful to discover the desired patterns
- Considers only linear correlations (work-around: Kernel-PCA, see later)

# Kapitel 4: Feature Reduction and Metric Learning

## Idea

- PCA is an eigenvalue decomposition of the $d \times d$ covariance matrix $\Sigma = D^T D$ of the (normalized) data matrix $D$:

$$\Sigma = VEV^T$$

  such that

  - $V = (pc_1, ..., pc_d)$, is a $d \times d$ matrix whose columns are the pairwise independent unit vectors, the eigenvectors
  - $E = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix}$ is a $d \times d$ diagonal matrix, the diagonal elements are the eigenvalues of the corresponding eigenvectors
- The decomposition can be found e.g. based on numerical algorithms

## Compute the SVD

- SVD is a generalization of the eigenvalue decomposition
- Let $D$ be the $n \times d$ data matrix ($n$ objects, $d$ dimensions) and let $k$ be its rank (max number of independent rows/ columns)
- We can decompose $D$ into matrices $O, S, A$ with $D = OSA^T$ or

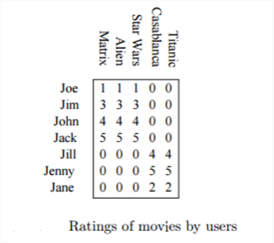
$$\underbrace{\begin{pmatrix} x_{1,1} & \dots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,d} \end{pmatrix}}_{D} = \underbrace{\begin{pmatrix} o_{1,1} & \dots & o_{1,k} \\ \vdots & \ddots & \vdots \\ o_{n,1} & \dots & o_{n,k} \end{pmatrix}}_{O} \cdot \underbrace{\begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_k \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} a_{1,1} & \dots & a_{1,d} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \dots & a_{k,d} \end{pmatrix}}_{A^T}$$

such that

- $O$ is a $n \times k$ column-orthonormal matrix (each of its columns is a unit vector and the dot product of any two columns is 0)
- $S$ is a diagonal $k \times k$ matrix
- $A$ is a $k \times d$ column-orthonormal matrix. Note that we always use $A$ in its transposed form, so it is the rows of $A^T$ that are orthonormal

## SVD: Example I

- *D* contains movie ratings by users
  - The corresponding SVD shows two concepts “science fiction” and “romance”
  - *S* shows the strength of these concepts
  - *A* relates movies to concepts



| | Matrix | Alien | Star Wars | Casablanca | 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 |

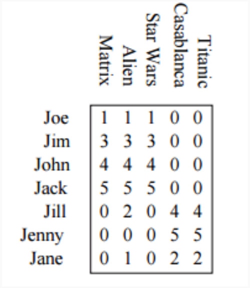
Ratings of movies by users

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{pmatrix}}_{D} = \underbrace{\begin{pmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{pmatrix}}_{O} \cdot \underbrace{\begin{pmatrix} 12.4 & 0 \\ 0 & 9.5 \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{pmatrix}}_{A^T}$$

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

## SVD: Example II

- Now a slightly different $D$
  - The corresponding SVD shows three concepts “science fiction” and “romance” and ???



| | Matrix | Alien | Star Wars | Casablanca | 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 | 2 | 0 | 4 | 4 |
| Jenny | 0 | 0 | 0 | 5 | 5 |
| Jane | 0 | 1 | 0 | 2 | 2 |

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{pmatrix}}_{D} = \underbrace{\begin{pmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & -09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & .-73 & -.67 \\ .07 & -.29 & .32 \end{pmatrix}}_{O} \cdot \underbrace{\begin{pmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{pmatrix}}_{A^T}$$

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

## Dimensionality Reduction with SVD

- To reduce dimensionality, we can set the smallest singular values to 0 in $S$ and eliminate the corresponding columns in $O$ and rows in $A^T$ (check previous examples)
- How Many Singular Values Should We Retain?
  - Rule of thumb: retain enough singular values to make up 90% of the energy in $S$
  - Energy is defined in terms of the singular values (matrix $S$)
  - In the previous example, the total energy is: $(12.4)^2 + (9.5)^2 + (1.3)^2 = 245.70$
  - The retained energy is: $(12.4)^2 + (9.5)^2 = 244.01 > 99\%$

## Connection between SVD and PCA

- PCA is applying SVD on the covariance matrix $\Sigma = D^T D$
- SVD means: $D = OSA^T$
- Thus:

$$\Sigma = D^T D = (OSA^T)^T OSA^T = AS^T(O^T O)SA^T$$

- Since $O$ is an orthonormal matrix, $O^T O$ is the identity:

$$AS^T(O^T O)SA^T = A(S^T S)A^T$$

- $S$ is a diagonal matrix, so transposing has no effect:

$$A(S^T S)A^T = AS^2 A^T = A \begin{pmatrix} \lambda_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_k^2 \end{pmatrix} A^T$$

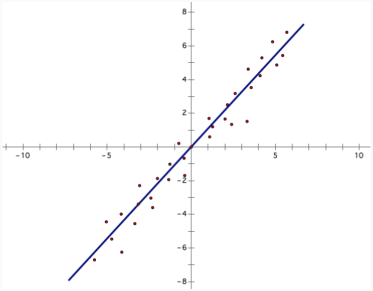
## Connection between SVD and PCA

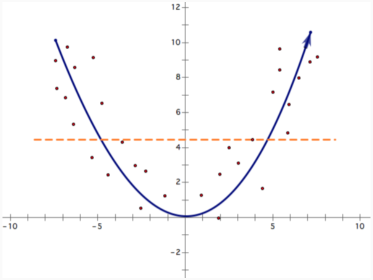
- Here: $A$ is a matrix of eigenvectors
- Eigenvalues of the covariance matrix = squared singular values of $D$
- Conclusion: Eigenvalues and eigenvectors of the covariance matrix $S$ can be determined by the SVD of the data matrix $D$ (or in other words: SVD is a method to perform PCA)
- SVD is sometimes a better way to perform PCA (Large dimensionalities e.g., text data)
- SVD can cope with dependent dimensions ($k < d$ is an ordinary case in SVD)

# Kapitel 4: Feature Reduction and Metric Learning

## Motivation

Consider the following scenarios:



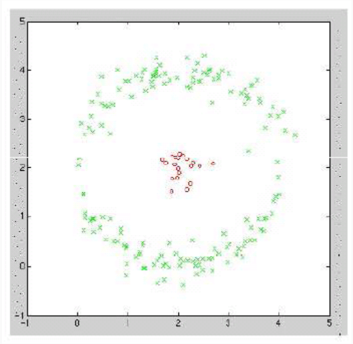


- PCA will be effective since data is linearly correlated

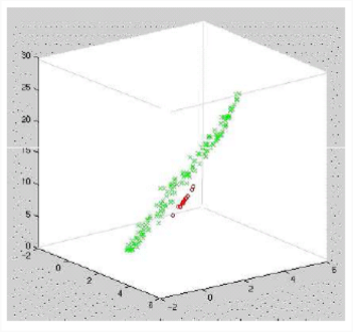
- PCA may find the orange line as the first component

## Basic Idea

Recall: the solution of linear classifiers (e.g. SVMs) for non-linear problems is “make them linear!” using a suitable feature mapping



- No linear separation of classes possible

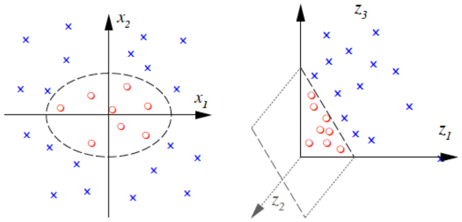


- Mapping $\mathbb{R}^2 \to \mathbb{R}^3$ with $(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$

## Kernel Trick

- Since a high-dimensional mapping can still have negative impact, the Kernel trick is used whenever possible (see KDD I lecture)
- Given the intended mapping $\Phi$, the Kernel is usually defined as $K(x,y) = \Phi(x)^T\Phi(y)$
- Example: Degree-$d$ polynomials: $K(x,y) = (x^Ty + c)^d$ with an arbitrary constant $c$, e.g. for $d = 2$:

$$\Phi : R^2 \rightarrow R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt(2)x_1x_2, x_2^2)$$



(Image source:

`http://i.stack.imgur.com/qZV3s.png`)

## Kernel PCA Using SVD

- Recall the SVD $D = OSA^T$
- $A$ is a $k$-dimensional basis of the eigenvectors of $DD^T$ (originally $d \times d$)
- Analogously, $O$ is a $k$-dimensional basis of eigenvectors of $DD^T$
- $DD^T$ is a Kernel matrix for the linear Kernel (i.e., no mapping made - cf. KDD I) or any other Kernel
- $A$ and $O$ are related as follows:

$$D = OSA^T \Rightarrow O^T D = O^T OSA^T = SA^T \Rightarrow S^{-1} O^T D = A^T$$

i.e. each $d$-dimensional eigenvector in $A$ is a linear combination of vectors in $D$ (original or mapped!) and the $n$ $k$-dimensional eigenvectors in $O^T$ ($O$ is $n \times k$)

# Kernel PCA Using SVD

- Let $K(x,y) = \Phi(x)^T\Phi(y)$ be a kernel for the non-linear transformation $\Phi$
- Assume: $K(x,y)$ is known, but $\Phi(x)$ is not explicitly given
- Let $K$ be the Kernel matrix of $D$ w.r.t. $K(x,y)$, i.e.

$$K = \begin{pmatrix} K(x_1,x_1) & \dots & K(x_1,x_n) \\ \vdots & \ddots & \vdots \\ K(x_n,x_1) & \dots & K(x_n,x_n) \end{pmatrix}$$

- The eigenvalue decomposition of $K$ is $K = VSV^T$ where $V$ is a $n$-dimensional basis from eigenvectors of $K$
- Dimensionality Reduction through mapping of $y \in D$ w.r.t $V$ to

$$\hat{y} = \begin{pmatrix} \Phi(y)^T(\sum_{i=1}^n v_{i,1}\Phi(x_i)) \\ \vdots \\ \Phi(y)^T(\sum_{i=1}^n v_{i,k}\Phi(x_i)) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n v_{i,1}(\Phi(y)^T\Phi(x_i)) \\ \vdots \\ \sum_{i=1}^n v_{i,k}(\Phi(y)^T\Phi(x_i)) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n v_{i,1}K(y,x_i) \\ \vdots \\ \sum_{i=1}^n v_{i,k}K(y,x_i) \end{pmatrix}$$

## Matrix Factorization as Optimization Task

- BTW, SVD (and, thus PCA) is a matrix decomposition that can be formalized as optimization task

$$D = OSA^T = \underbrace{\left( O \begin{pmatrix} \sqrt{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sqrt{\lambda_k} \end{pmatrix} \right)}_{U} \underbrace{\left( \begin{pmatrix} \sqrt{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sqrt{\lambda_k} \end{pmatrix} A^T \right)}_{V^T} = UV^T$$

- As an optimization problem: $L(U,V) = \|D - UV^T\|_f^2$

  subject to $\forall_{i \neq j} : \langle v_i, v_j \rangle = 0 \wedge \langle u_i, u_j \rangle$

  using the squared Frobenius Norm of an $n \times m$ matrix $M$: $\|M\|_f^2 = \sum_{i=1}^{n} \sum_{i=1}^{m} |m_{i,j}|^2$

# Kapitel 4: Feature Reduction and Metric Learning

## Fisher Faces

### Fisher Faces

- Idea: Use examples from a training set (supervised!) to increase the discriminative power of the target space
- Minimize the similarity between objects from different classes (between class scatter matrix: $\sigma_b$)
  Use covariance matrix of the class centroids for $\Sigma_b$
- Maximize similarity between objects belonging to the same class (within class scatter matrix $\Sigma_w$)
  Use average covariance matrix of all classes for $\Sigma_w$
- Determine new basis vectors $b_i$ by maximizing

$$\frac{b_i^T \Sigma_b b_i}{b_i^T \Sigma_w b_i}$$

subject to $\forall_{i \neq j} : \langle b_i, b_j \rangle = 0$

## Fisher Faces

Remarks on Fisher Faces

- The vector having the largest eigenvalue corresponds to the normal vector of the separating hyper plane in linear discriminant analysis or Fisher‘s discriminant analysis. (cf. KDD I)
- Fischer Faces are limited due to the assumption of mono-modal classes: each class is assumed to follow one multivariate Gaussian
- Multi-modal or non-Gaussian distributions are not modeled well
- Many variants (e.g. Relevant Component Analysis (RCA), Large Margin Nearest Neighbor (LMNN)

## Summary

- Linear basis transformation yield a rich framework to optimize feature spaces
- Unsupervised methods delete low variant dimensions (PCA und SVD)
- Kernel PCA allows to compute PCA in non-linear kernel spaces
- Basic assumption: direction of highest variance bear the most relevant information
- Supervised methods try to minimize the within class distances while maximizing between class distances (Fischer Faces and variants)

## Further Readings

- S. Deerwester, S. Dumais, R. Harshman: Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science, Vol. 41, 1990
- L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.
- K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classication. Journal of Machine Learning Research, 10:207,244, 2009.
- P. Comon. Independent component analysis, a new concept? Signal Processing, 36(3):287-314, 1994.
- J. Davis, B. Kulis, S. Sra, and I. Dhillon. Information theoretic metric learning. In in NIPS 2006 Workshop on Learning to Compare Examples, 2007.
- A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, USA, pages 11-18, 2003.

# Kapitel 5: Clustering High-dim Data i

# Kapitel 5: Clustering High-dim Data ii

# Kapitel 5: Clustering High-dim Data

## Motivating Example

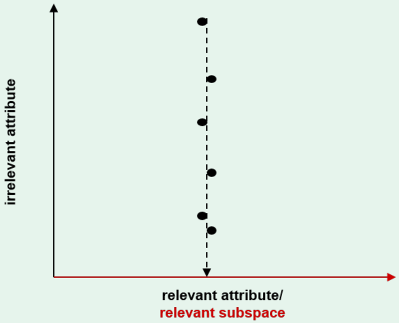
### Customer Recommendation / Target Marketing

- Data: customer ratings for given products
  - Rows (objects): customers (millions?)
  - Columns (features): products (hundreds to thousands)
  - Value $x_{ij}$ in the data matrix is the rating of product $i$ by user $j$
- Task: Cluster customers to find groups of persons that share similar preferences or disfavor (e.g. to do personalized target marketing)
- Challenge: customers may be grouped differently according to different preferences/disfavors, i.e. different subsets of products (See: baby shapes game)

## Curse of Dimensionality (Revisited)

### Relevant and irrelevant attributes

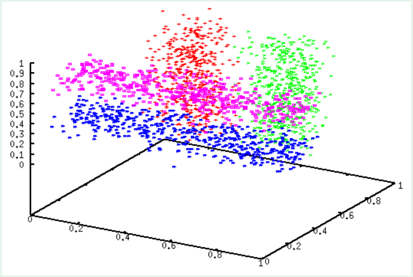
- Not all features, but a subset of the features may be relevant for clustering
- Groups of similar (e.g. “dense”) points may be identified when considering only these features
- Different subsets of attributes may be relevant for different clusters



## Curse of Dimensionality (Revisited)

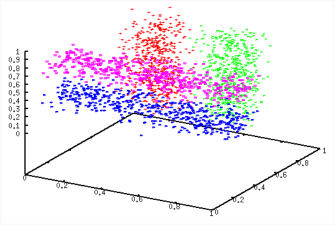
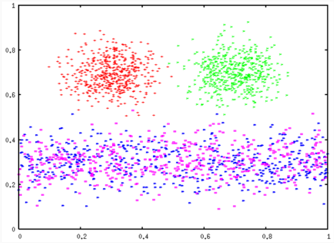
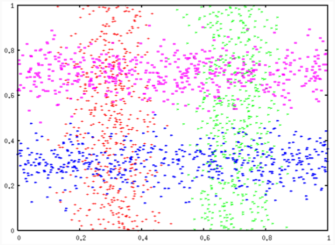
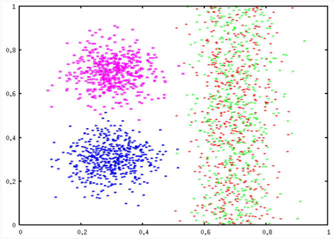
### Effect on clustering

- Traditional distance functions give equal weight to all dimensions
- However, not all dimensions are of equal importance
- Adding irrelevant dimensions ruins any clustering based on a distance function that equally weights all dimensions



## Curse of Dimensionality (Revisited)

- Example: different attributes are relevant for different clusters

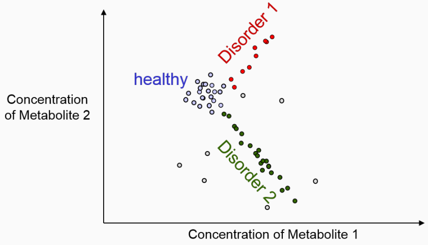





## Curse of Dimensionality (Revisited)

It can even be a little bit more complex ...

- Task: Cluster test persons to find groups of individuals with similar correlation among the concentrations of metabolites indicating homogeneous metabolic behavior (e.g. disorder)

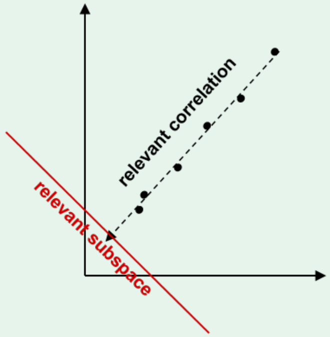
- Challenge: different metabolic disorders appear through different correlations of (subsets of) metabolites



## Curse of Dimensionality (Revisited)

### Correlation among attributes

- A subset of features may be correlated
- Groups of similar (e.g. “dense”) points may be identified when considering this correlation of features only
- Different correlations of attributes may be relevant for different clusters

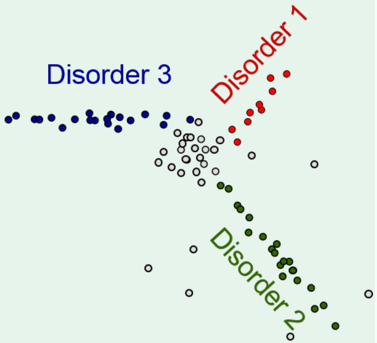


## Curse of Dimensionality (Revisited)

### Why not feature selection/reduction

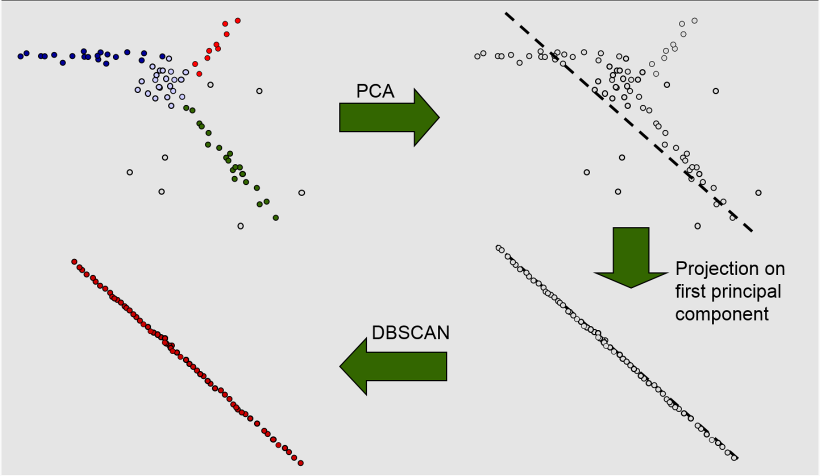
- (Unsupervised) feature selection or feature reduction (e.g. PCA) is global, i.e., it transforms the original feature space into one new representation

- We face a local feature relevance/correlation: some features (or combinations of them) may be relevant for one cluster, but may be irrelevant for a second one, i.e., we need multiple representations



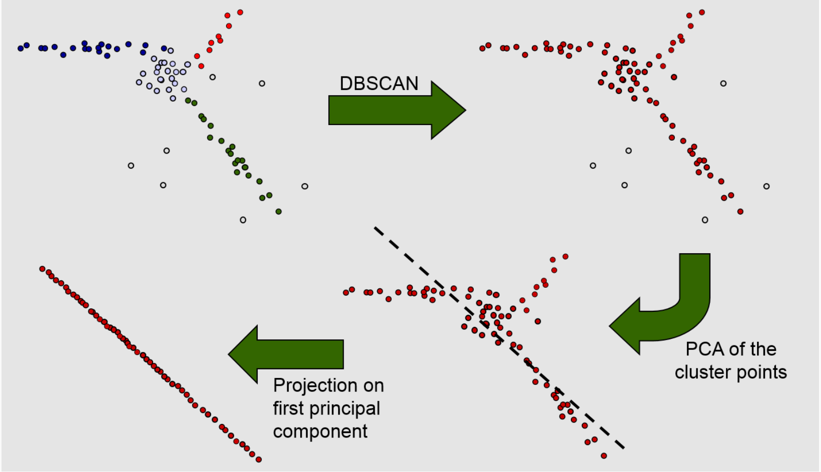
## Curse of Dimensionality (Revisited)

### Example: use PCA (target dim = 1) before clustering



## Curse of Dimensionality (Revisited)

### Example: cluster first, then find correlations (with PCA)


DBSCAN
PCA of the cluster points
Projection on first principal component

## Problem Summary

- Feature relevance and correlation
  - Usually, no clusters in the full dimensional space
  - Often, clusters are hidden in subspaces of the data, i.e. only a subset of features is relevant for the clustering
  - E.g. a group of genes play a common role in a subset of experimental conditions but may behave completely different in other conditions
- Local feature relevance/correlation
  - For each cluster, a different subset of features or a different correlation of features may be relevant
  - E.g. different genes are responsible for different phenotypes other conditions
- Overlapping clusters (different semantic concepts)
  - Clusters may overlap, i.e. an object may be clustered differently in varying subspaces
  - E.g. a gene plays different functional roles depending on the environment other conditions

## General Problem Setting

**Search for clusters in (in general arbitrarily oriented) subspaces of the original feature space**

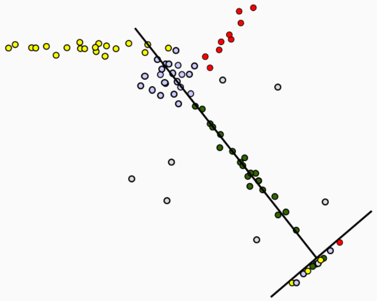
**Challenges**

- Find the correct subspace of each cluster (Search space virtually infinite: all possible arbitrarily oriented subspaces of a feature space)
- Find the correct cluster in each relevant subspace (Search space depends on clustering algorithm)

- Even worse: both challenges depend on each other:
  - In order to determine the correct subspace of a cluster, we need to know (at least some) cluster members
  - In order to determine the correct cluster memberships, we need to know the subspaces of all clusters

## General Problem Setting

- Hmm, it is really not so easy, especially in practice ...
- Even if we found the relevant subspace, we still might have a hard time to find the correct cluster(s)
- Here, clusters (yellow, red, green, blue) and noise (gray) are not separable in the “correct” subspace . . .



- Rather, we would need a new cluster model (recall: cluster = group of similar objects)
- What is the concept of similarity that all members of a cluster share in this example (and does not include members from other clusters/noise)???

## A Taxonomy of Approaches

**Subspace Clustering (restricted to axis-parallel subspace)**

- Find all clusters in all subspaces (allow overlaps)
- Usually bottom-up subspace search

**Projected Clustering (restricted to axis-parallel subspace)**

- Each point is assigned to one subspace cluster or noise
- Usually top-down subspace search

**Correlation Clustering (explores arbitrarily oriented subspaces**

- Each point is assigned to one subspace cluster or noise
- Bottom-up and top-down subspace search approaches

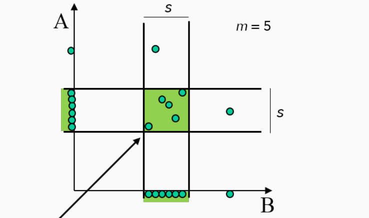
## Kapitel 5: Clustering High-dim Data

## Bottom-up Subspace Cluster Search

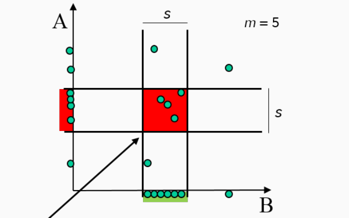
- Similar to Branch-and-Bound feature selection: Start with 1-D subspaces or subspace clusters and merge them to compute higher dimensional ones
- Most approaches transfer this problem into a frequent item set mining problem
- In this case, the cluster criterion must implement the downward closure (monotonicity) property:
    - If the criterion holds for a $k$-dimensional subspace $S$, then it also holds for any ($k$–1)-dimensional projection of $S$
    - Use the reverse implication for pruning: If the criterion does not hold for a ($k$–1)-dimensional projection of $S$, then the criterion also does not hold for $S$
- Some approaches use other search heuristics (especially if monotonicity does not hold) like best-first-search, greedy-search, ...

## Downward-closure Property: Example

- Consider a simple cluster criterion (density of grid cells): If a cell *C* of side length *s* contains more than *m* points, it represents a cluster
- Monotonicity: if *C* contains more than *m* points in subspace *S* then *C* also contains more than *m* points in any subspace $T \subset S$



Cell *C* contains more than *m*=5 points in subspace „AB“
=> Also in subspaces „A“⊂ „AB“ and „B“⊂ „AB“

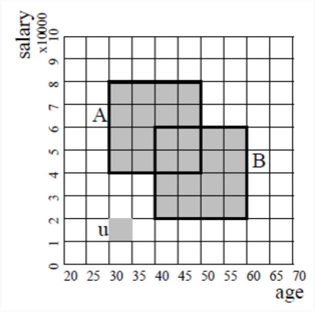


Cell *C* contains less than *m*=5 points in subspace „A“
=> Also in subspace „AB“

## CLIQUE: Idea

- Probably the first bottom-up algorithm
- It uses a density-grid-based cluster model (similar to previous slide)

- Clusters are "dense regions" in the feature space
- Partition the feature space into $\xi$ equal sized parts in each dimension
- A unit is the intersection of one interval from each dimension
- Unit $u$ is dense if it contains more than $\tau$ objects

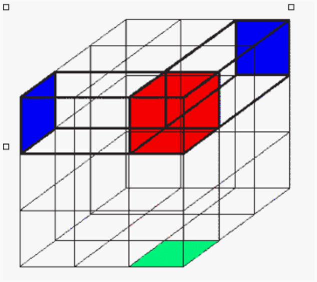


- Clusters are maximal sets of connected dense units (e.g., $A \cup B$)
- Two-step approach (1. subspace search, 2. clustering)

## CLIQUE: Subspace Search

Step 1: Find subspaces with dense units

- Explore Downward Closure property of dense cells (APRIORI-style search)
- Candidate generation
  - Based on $D_{k-1}$, the set of $(k-1)$-dimensional dense units, generate candidate set $C_k$ by self joining $D_{k-1}$
  - Join condition: units share first $k-2$ dimensions
  - Discard those candidates which have a $k-1$ projection not included in $D_{k-1}$
  - For the remaining candidates: check density



| | |
|---|---|
| 🟦 | 2-dim. dense unit (∈D2) |
| 🟥 | 3-dim. candidate unit |
| 🟩 | 2-dim. unit which has to be checked |

## CLIQUE: Cluster Search

Step 2: Find clusters as maximal sets of connected dense units

- Given: a set of dense units $D$ in the same $k$-dimensional subspace $S$
- Output: A partition of $D$ into clusters $D_1, \dots, D_k$ of connected dense units
- The problem is equivalent to finding connected components in a graph
    - Nodes: dense units
    - Edge between two nodes if the corresponding dense units have a common face (neighboring units)
    - Depth-first search algorithm: Start with a unit $u$ in $D$, assign it to a new cluster ID and find all the units it is connected to
    - Repeat if there are nodes not yet visited

## CLIQUE: Discussion

- Input parameters: $\xi$ and $\tau$ specifying the density threshold
- Output: all clusters in all subspaces, clusters may overlap
- Simple but efficient cluster model
- Uses a fixed density threshold for all subspaces (in order to ensure the downward closure property)
    - To represent a cluster, a unit in 10D must contain as many points (or more) as in 2D . . .
    - For a cluster $C$ in subspace $S$, all clusters in all projections of $S$ are also reported as clusters (extremely high redundancy)
    - Which of the redundant information is more interesting? ($D$ in $S$ or $D'$ in $T \subset S$; $D'$ may contain more units/points)
- Worst case runtime?

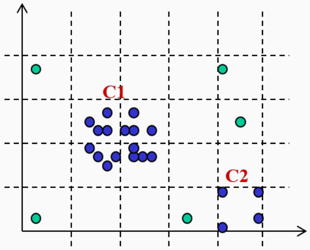
## CLIQUE: Variants

There are different variations of CLIQUE varying mainly in terms of ...

- ... different density definition, e.g. using Entropy for subspaces rather than simple counts of units
- ... different grid construction methods, e.g. adaptive intervals (but then, no downward closure property is given anymore)

Drawbacks of a grid-based clustering model:

- Positioning of the grid influences the clustering
- Selection of $\xi$ and $\tau$ is very sensitive
- Example: Either $C_2$ and $C_1$ are found as clusters or none of them

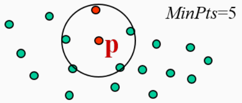


## SUBCLU: Idea

Use DBSCAN model (maximal density-connected sets): density is defined w.r.t. the location of points not w.r.t. the data space:

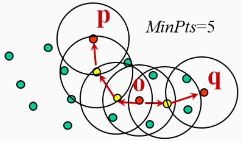
**Core Points**

- Points finding more than *MinPts* other points in its $\varepsilon$-neighborhood



**Density connectivity**

- Core points may have core points in their $\varepsilon$-neighborhood
- Build transitive chains of such core points to find connected sets

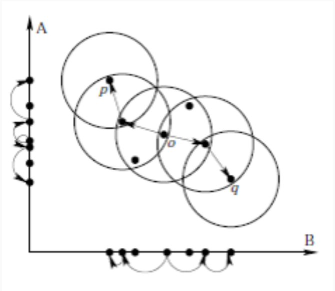


## SUBCLU: Basics

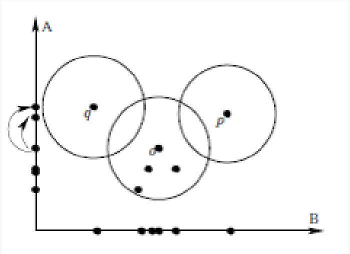
- Richer cluster model: detects clusters of arbitrary shapes and locations (in the corresponding subspaces)
- Naive approach: Apply DBSCAN in all possible subspaces (exponential runtime!)
- Idea: Exploit clustering information from previous step (subspaces)
- Density-connected clusters are not monotonic, but density connected sets are (see next slide)
    - If $C$ is a density connected set in subspace $S$ then $C$ is a density connected set in any subspace $T \subset S$
    - But, if $C$ is a cluster in $S$, it need not to be a cluster in $T \subset S$ because maximality might be violated, i.e., in $T$ there may be additional points density connected to the points in $C$
    - A cluster in a higher-dimensional subspace $S$ will be a subset of a cluster in the a projection $T$
- Thus, APRIORI-style subspace search is possible

## SUBCLU: Monotonicoty

Example (circles indicate $\varepsilon$-neighborhood):



(a) $p$ and $q$ are density-connected via $o$



(b) $p$ and $q$ are not density-connected

$p$ and $q$ density connected in $\{A, B\}$, thus, they are also density connected in $\{A\}$ and $\{B\}$

$p$ and $q$ not density connected in $\{B\}$, thus, they are not density connected in $\{A, B\}$, although they are density connected in $\{A\}$

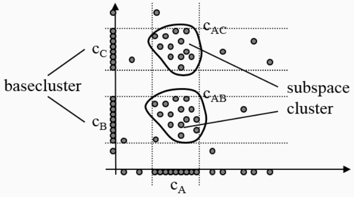
## SUBCLU: Algorithm/Discussion

- Algorithm
  - All subspaces that contain any density-connected set are computed using the bottom-up approach (similar to CLIQUE/APRIORI)
  - Density-connected clusters are computed using a specialized DBSCAN run in the resulting subspace to generate the subspace clusters
- Discussion
  - Input: $\varepsilon$ and *MinPts* specifying the density threshold
  - Output: all clusters in all subspaces, clusters may overlap
  - Uses a fixed density threshold for all subspaces
  - Advanced but costly cluster model

## Bottom-up Variants: FIRES

Uses a different bottom-up heuristic for subspace search:

- Starts with 1-dimensional clusters called base clusters (generated by applying any traditional clustering algorithm to each 1-dimensional subspace)
- Merges these clusters to generate subspace cluster approximations by applying a clustering of the base clusters using a variant of DBSCAN (similarity between two clusters $C_1$ and $C_2$ is defined by $|C_1 \cap C_2|$) in order to “jump” to maximal dimensional subspaces
- Refines the resulting cluster approximations using any traditional clustering algorithm on the points within the approximations



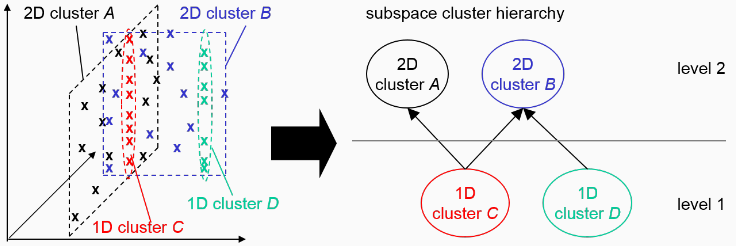
## Bottom-up Variants: P3C

Similar idea to FIRES:

- Cluster cores are hyper-rectangular approximations of subspace clusters
- Subspace search is APRIORI-style: cluster cores are computed bottom-up from “significant” 1D intervals
- Significant 1D intervals are determined using a hypothesis test:
  - Hypothesis: no clusters, i.e. points are randomly distributed
  - If an interval contains significantly more points than expected, the hypothesis is rejected (significant 1D interval)
- Significant 1D intervals follow the downward closure property
- Postprocessing: Cluster cores initialize an EM fuzzy clustering of all data points

## Bottom-up Variants: DISH

Find subspace cluster hierarchies (lower dimensional clusters embedded in higher dimensional ones):



- Integrate a proper distance function into hierarchical clustering
- Learns distance function instance-based bottom-up

## Summary: Subspace Clustering

Several different variants exist; most of them suffer from a key limitation: global density thresholds

- In order to ensure the downward closure property, the density threshold must be fixed globally
- Consequence: the points in an e.g. 20-dimensional subspace cluster must be as dense as in an e.g. 2-dimensional cluster
- This is a rather optimistic assumption since the data space grows exponentially with increasing dimensionality (see “curse” discussion)
- Consequences:
    - A strict threshold will most likely produce only lower dimensional clusters
    - A loose threshold will most likely produce higher dimensional clusters but also a huge amount of (potentially meaningless) low dimensional clusters

# Kapitel 5: Clustering High-dim Data

## Top-down Subspace Search: Approaches

Several different variants for top-down subspace search:

- Cluster-based search
  - Learn the subspace of a cluster in the entire *d*-dimensional feature space
  - Start with full-dimensional clusters
  - Iteratively refine the cluster memberships of points and the subspaces of the cluster
- Instance-based search
  - Learn for each point its subspace preference in the entire *d*-dimensional feature space
  - The subspace preference specifies the subspace in which each point “clusters best”
  - Merge points having similar subspace preferences to generate the clusters
- Random search: learn the subspace preference of a cluster or a point from randomly sampled points

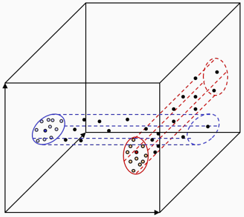
## Top-down Subspace Search: Key Limitation

How should we learn the subspace preference of a cluster or a point?

- Most approaches rely on the so-called “locality assumption”: the subspace preference can be learned from the local neighborhood in the *d*-dimensional space
- The subspace is usually learned from the local neighborhood of cluster representatives/cluster members in the entire feature space:
  - Cluster-based approach: the local neighborhood of each cluster representative is evaluated in the *d*-dimensional space to learn the “correct” subspace of the cluster
  - Instance-based approach: the local neighborhood of each point is evaluated in the *d*-dimensional space to learn the “correct” subspace preference of this point (i.e. the subspace in which the cluster exists that accommodates this point)
- Random search approaches do not suffer from the locality assumption but usually need many sampling rounds

## PROCLUS

- Cluster-based top-down approach: we learn the subspace for each cluster
- $K$-medoid cluster model
  - Cluster is represented by its medoid
  - To each cluster a subspace (of relevant attributes) is assigned
  - Each point is assigned to the nearest medoid (where the distance to each medoid is based on the corresponding subspace of the medoid)
  - Points that have a large distance to their nearest medoid(s) are classified as noise



## PROCLUS

3-phase algorithm (input: number of clusters $k$, average dimensionality of subspaces $L$):

**Phase 1**: Initialization of cluster medoids

- Ideally we want a set of centroids, where each centroid comes from a different cluster
- We do not know which are these $k$ points though, so we choose a superset $M$ of $b \cdot k$ medoids such that they are well separated
    - Chose a random sample ($S$) of $a \cdot k$ data points
    - Out of $S$, select $b \cdot k$ points ($M$) by greedy selection: medoids are picked iteratively so that the current medoid is well separated from the medoids that have been chosen so far
    - Additional input parameters $a$ and $b$ are introduced for performance reasons
- This procedure has nothing to do with subspace/projected clustering but could be applied fro any $k$-partitioning algorithm

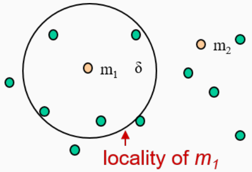
## PROCLUS

**Phase 2**: Iterative phase (works similar to any *k*-medoid clustering)

- *k* randomly chosen medoids from *M* (Phase 1) are the initial cluster medoids
- Replace the “bad” medoids with other points in *M* if the quality of the clustering increases
- Procedure:
    - Find dimensions related to the medoids
    - Assign data points to the medoids
    - Evaluate the clusters formed
    - Find the bad medoids, and try to improve the result by replacing these bad medoids

## PROCLUS: Find Cluster Dimensions

- For each medoid $m_i$, let $2 \cdot \delta$ be the distance to its closest medoid
- All the data points within $\delta$ will be surely assigned to the medoid $m_i$ (this set $L_i$ is the locality of $m_i$)



- Intuition: to each medoid we want to associate those dimensions where the points are closed to the medoid in that dimension
- Compute the average distance $X_{i,j}$ along each dimension $j$ from the points in $L_i$ to $m_i$
- Calculate for $m_i$ the mean $Y_{i,j}$ and standard deviation $\sigma_{i,j}$ of $X_{i,j}$
- Calculate $Z_{i,j} = (X_{i,j} - Y_{i,j})/\sigma_{i,j}$
- Choose $k \cdot l$ smallest values $Z_{i,j}$ with at least 2 chosen for each medoids (to ensure that cluster preferences are lower dimensional)
- Output: A set of $k$ medoids and their associated dimensions

## PROCLUS: Point Assignment and Cluster Evaluation

- Assign each data point to its closest medoid using Manhattan segmental distance (only relevant dimensions count)
  Manhattan segmental distance: For any two points $x_1$, $x_2$ and any set of dimensions $D$:

$$MSdist_D(x_1, x_2) = \frac{\sum_{i \in D} |x_{1i} - x_{2i}|}{|D|}$$

- Evaluate a cluster $C_i$ using average *MSdist* from the points in $C_i$ to the centroid of $C_i$ along dimension $j$
- Replace bad medoids with random points from $M$
- Terminate if the clustering quality does not increase after a given number of current medoids have been exchanged with medoids from $M$ (it is not clear, if there is another hidden parameter in that criterion)

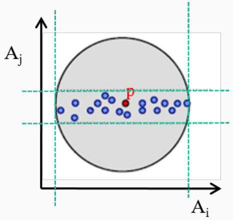
## PROCLUS

**Phase 3**: Refinement

- Reassign subspaces to medoids as above (but use only the points assigned to each cluster rather than the locality of each cluster, i.e., $C_i$ not $L_i$)
- Reassign points to medoids
- Points that are not in the locality of any medoid are classified as noise

## PreDeCon

- Use DBSCAN cluster model instead of *k*-medoid clustering
- Use DBSCAN with a distance that adopts to the subspace preferences of points



- Given the input parameters $\delta$ and $\kappa >> 1$, for each point *p*, the *d*-dimensional subspace preference vector $w^p$ of *p* is defined using the following components:

$$w_i^p = \begin{cases} 1 & \text{if} \quad VAR_i > \delta \\ \kappa & \text{if} \quad VAR_i \leq \delta \end{cases}$$

where

$$VAR_i = \frac{\sum_{q \in N_\varepsilon(p)} dist(p_i, q_i)^2}{|N_\varepsilon(p)|}$$

## PreDeCon

- Preference weighted distance function

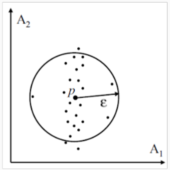
$$dist_p(p,q) = \sqrt{\sum_{i=1}^{d} w_i^p \cdot (p_i - q_i)^2}$$

can be made symmetric:

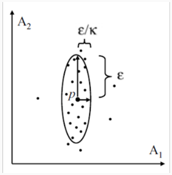
$$dist_{pref}(p,q) = \max\{dist_p(p,q), dist_q(q,p)\}$$

- Preference $\varepsilon$-neighborhood of a point uses $dist_{pref}$

*simple*
ε-neighborhood



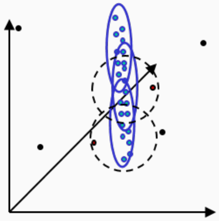
$\leftrightarrow$

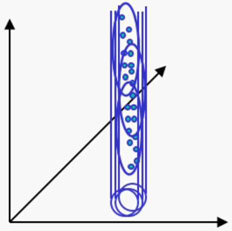


*preference weighted*
ε-neighborhood

## PreDeCon

- Preference weighted core points, direct density reachability, reachability and connectivity are defined based on preference neighborhood
- A subspace preference cluster is a maximal density connected set of points associated with a certain subspace preference vector
- Clusters may have arbitrary shape in the subspace projection





## Variants

- DOC
    - Clusters are “dense” hyper-rectangles
    - Computes at most one cluster in each run
    - Random sampling for subspace learning
- COSA
    - Learns a distance matrix that can be used for clustering
    - Instance-based locality assumption

# Kapitel 5: Clustering High-dim Data

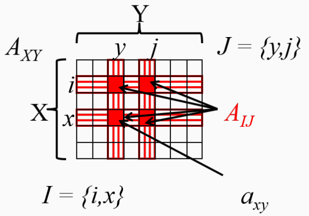
## Overview

- Challenges and Approaches, Basic Models
    - Constant Biclusters
    - Biclusters with Constant Values in Rows or Columns
    - Pattern-based Clustering: Biclusters with Coherent Values
    - Biclusters with Coherent Evolutions
- Algorithms for
    - Constant Biclusters Pattern-based Clustering: Biclusters with Coherent Values

## General Idea

- Pattern-based clustering relies on patterns in the data matrix
- Simultaneous clustering of rows and columns of the data matrix (hence “bi-clustering”).
- Data matrix $A = (X, Y)$ with set of rows (objects) $X$ and set of columns (features) $Y$

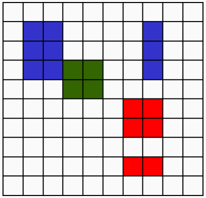
- $a_{xy}$ is the element in row $x$ and column $y$
- Submatrix $A_{IJ} = (I, J)$ with subset of rows $I \subseteq X$ and subset of columns $J \subseteq Y$ contains those elements $a_{ij}$ with $i \in I$ and $j \in J$



## General Idea

General aim of biclustering approaches:

- Find a set of submatrices $\{(I_1, J_1), (I_2, J_2), ..., (I_k, J_k)\}$ of the data matrix $A = (X, Y)$ such that each submatrix (= bicluster) meets a given homogeneity criterion



Some values often used by bicluster models:

- Mean of row $i$: $a_{i,J} = 1/|J| \sum_{j \in J} a_{ij}$
- Mean of column $j$: $a_{I,j} = 1/|I| \sum_{i \in I} a_{ij}$
- Mean of all elements:

$$a_{I,J} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} = 1/|J| \sum_{j \in J} a_{Ij} = 1/|I| \sum_{i \in I} a_{iJ}$$

## General Idea

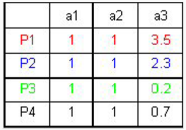
Different types (models) of biclusters

- constant biclusters
- biclusters with constant values on columns and/or constant values on rows
- biclusters with coherent values (aka. pattern-based clustering)
- biclusters with coherent evolutions

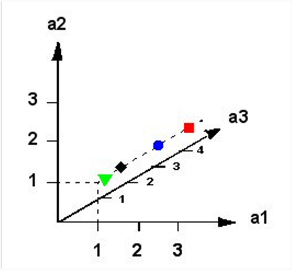
## Constant Biclusters

Cluster Model

- All points share identical value in selected attributes
- The constant value $\mu$ is a typical value for the cluster
- Thus, the cluster model is $a_{ij} = \mu$
- Obviously a special case of an axis-parallel subspace cluster
- Example:



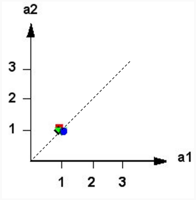
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 1 | 3.5 |
| P2 | 1 | 1 | 2.3 |
| P3 | 1 | 1 | 0.2 |
| P4 | 1 | 1 | 0.7 |



## Constant Biclusters

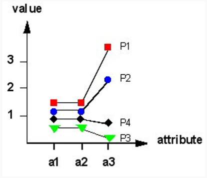
- Interpretation: points are located on the bisecting line of participating attributes

|  | a1 | a2 |
|---|---|---|
| P1 | 1 | 1 |
| P2 | 1 | 1 |
| P3 | 1 | 1 |
| P4 | 1 | 1 |



- Visualization using Parallel Coordinates: identical constant lines

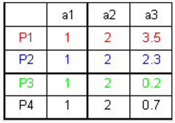
|  | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 1 | 3.5 |
| P2 | 1 | 1 | 2.3 |
| P3 | 1 | 1 | 0.2 |
| P4 | 1 | 1 | 0.7 |



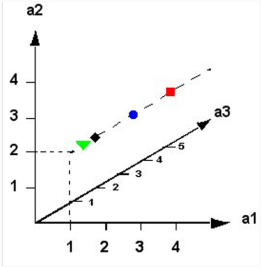
## Biclusters with Constant Column Values

Cluster Model

- Cluster model for $A_{IJ} = (I, J)$ with $a_{ij} = \mu + c_j$
- $c_j$ is an adjustment value for column $j$
- Results in axis-parallel subspace clusters
- Example:



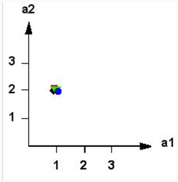
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 2 | 3.5 |
| P2 | 1 | 2 | 2.3 |
| P3 | 1 | 2 | 0.2 |
| P4 | 1 | 2 | 0.7 |



# Biclusters with Constant Column Values

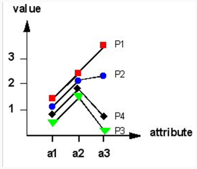
- Interpretation: points are located on the same values (not necessary bisecting line)

| | a1 | a2 |
|---|---|---|
| P1 | 1 | 2 |
| P2 | 1 | 2 |
| P3 | 1 | 2 |
| P4 | 1 | 2 |



- Visualization using Parallel Coordinates: identical lines

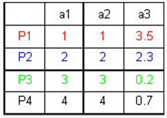
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 2 | 3.5 |
| P2 | 1 | 2 | 2.3 |
| P3 | 1 | 2 | 0.2 |
| P4 | 1 | 2 | 0.7 |



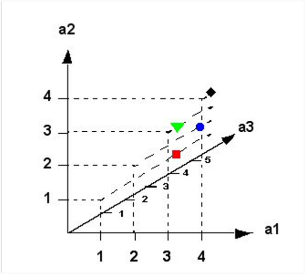
# Biclusters with Constant Row Values

Cluster Model

- Cluster model for $A_{IJ} = (I, J)$ with $a_{ij} = \mu + r_i$
- $r_i$ is an adjustment value for column $i$
- Results in points building a sparse hyperplane parallel to irrelevant axes (axis-parallel subspace cluster)
- Example:



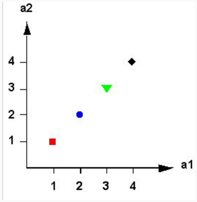
|  | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 1 | 3.5 |
| P2 | 2 | 2 | 2.3 |
| P3 | 3 | 3 | 0.2 |
| P4 | 4 | 4 | 0.7 |



## Biclusters with Constant Row Values

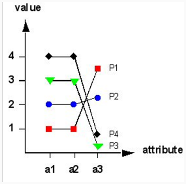
- Interpretation: points are accommodated on the bisecting line of participating attributes

| | a1 | a2 |
|---|---|---|
| P1 | 1 | 1 |
| P2 | 2 | 2 |
| P3 | 3 | 3 |
| P4 | 4 | 4 |



- Visualization using Parallel Coordinates: parallel constant lines

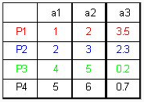
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 1 | 3.5 |
| P2 | 2 | 2 | 2.3 |
| P3 | 3 | 3 | 0.2 |
| P4 | 4 | 4 | 0.7 |



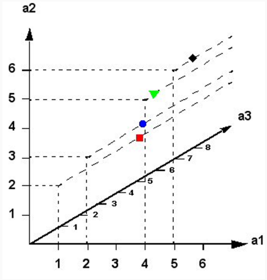
## Biclusters with Coherent Values

Cluster Model

- Based on a particular form of covariance between rows and columns: $a_{ij} = \mu + r_i + c_j$
- Special cases: constant rows ($c_j = 0$) and constant columns ($r_i = 0$)
- Results in points building a hyperplane parallel to axes of irrelevant attributes (axis-parallel subspace cluster)
- Example:



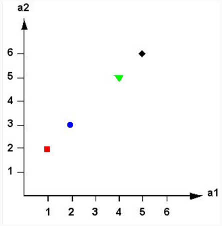
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 2 | 3.5 |
| P2 | 2 | 3 | 2.3 |
| P3 | 4 | 5 | 0.2 |
| P4 | 5 | 6 | 0.7 |



## Biclusters with Coherent Values

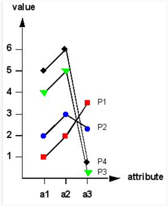
- Interpretation: increasing one-dimensional line in relevant subspace

|  | a1 | a2 |
|---|---|---|
| P1 | 1 | 2 |
| P2 | 2 | 3 |
| P3 | 4 | 5 |
| P4 | 5 | 6 |



- Visualization using Parallel Coordinates: parallel lines

|  | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 1 | 2 | 3.5 |
| P2 | 2 | 3 | 2.3 |
| P3 | 4 | 5 | 0.2 |
| P4 | 5 | 6 | 0.7 |

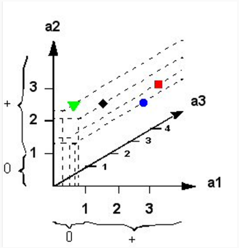


## Biclusters with Coherent Evolution

Cluster Model

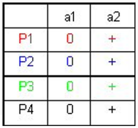
- For all rows, all pairs of attributes change simultaneously, e.g.
  - Discretized attribute space: coherent state-transitions: similar to grid-based axis parallel approaches (see above)
  - Change in same direction irrespective of the quantity
- Example with coherent state-transitions:

| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 0.5 | 1.5 | 3.5 |
| P2 | 0.7 | 1.3 | 2.3 |
| P3 | 0.3 | 2.3 | 0.2 |
| P4 | 0.8 | 2.1 | 0.7 |

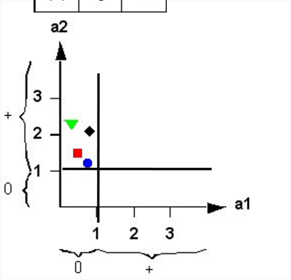


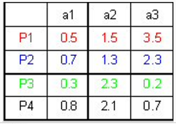
# Biclusters with Coherent Evolution

- Interpretation: points are in certain (grid-cell-like) half spaces
- Visualization using Parallel Coordinates: all lines cross border between states (in the same direction)

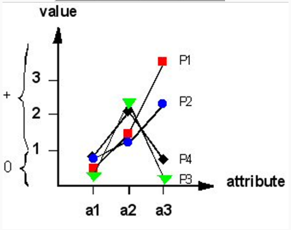


| | a1 | a2 |
|---|---|---|
| P1 | 0 | + |
| P2 | 0 | + |
| P3 | 0 | + |
| P4 | 0 | + |



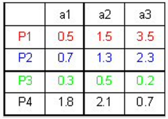


| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 0.5 | 1.5 | 3.5 |
| P2 | 0.7 | 1.3 | 2.3 |
| P3 | 0.3 | 2.3 | 0.2 |
| P4 | 0.8 | 2.1 | 0.7 |

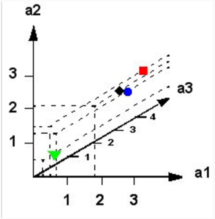


## Biclusters with Coherent Evolution

- Change in same direction — general idea: find a subset of rows and columns, where a permutation of the set of columns exists such that the values in every row are increasing
- Clusters do not form a subspace but rather half-spaces
- Example with change in same direction:



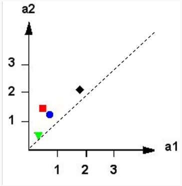
| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 0.5 | 1.5 | 3.5 |
| P2 | 0.7 | 1.3 | 2.3 |
| P3 | 0.3 | 0.5 | 0.2 |
| P4 | 1.8 | 2.1 | 0.7 |



## Biclusters with Coherent Evolution

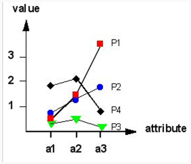
- Interpretation: points are in certain (non axis-parallel) half spaces

|  | a1 | a2 |
|---|---|---|
| P1 | 0.5 | 1.5 |
| P2 | 0.7 | 1.3 |
| P3 | 0.3 | 0.5 |
| P4 | 1.8 | 2.1 |

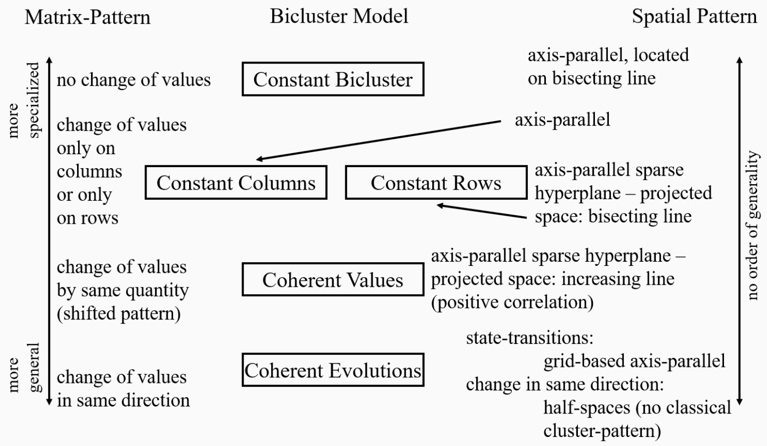


- Visualization using Parallel Coordinates: all lines increasing

|  | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 0.5 | 1.5 | 3.5 |
| P2 | 0.7 | 1.3 | 2.3 |
| P3 | 0.3 | 0.5 | 0.2 |
| P4 | 1.8 | 2.1 | 0.7 |



## Bicluster Models: Big Picture


Matrix-Pattern
Bicluster Model
Spatial Pattern
more specialized
more general
no change of values
Constant Bicluster
axis-parallel, located on bisecting line
change of values only on columns or only on rows
axis-parallel
Constant Columns
Constant Rows
axis-parallel sparse hyperplane – projected space: bisecting line
change of values by same quantity (shifted pattern)
Coherent Values
axis-parallel sparse hyperplane – projected space: increasing line (positive correlation)
change of values in same direction
Coherent Evolutions
state-transitions: grid-based axis-parallel
change in same direction: half-spaces (no classical cluster-pattern)
no order of generality

## Biclustering Algorithms

Algorithms for Constant Biclusters

- Classical problem statement by Hartiganin 1972
- Quality measure for a bicluster: variance of the submatrix $A_{IJ}$:

$$VAR(A_{IJ}) = \sum_{i \in I, j \in J} (a_{ij} - a_{I,J})^2$$

- Recursive split of data matrix into two partitions
- Each split chooses the maximal reduction in the overall sum of squares for all biclusters
- Avoids partitioning into $|X||Y|$ singularity-biclusters (optimizing the sum of squares) by comparing the reduction with the reduction expected by chance

## Biclustering Algorithms

Algorithms for Biclusters with Constant Values in Rows or Columns

- Simple approach: normalization to transform the biclusters into constant biclusters and follow the first approach
- Some application-driven approaches with special assumptions in the bioinformatics community
- Constant values on columns: general axis-parallel subspace/projected clustering
- Constant values on rows: special case of general correlation clustering
- Both cases special case of approaches to biclusters with coherent values

## Biclustering Algorithms

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

- Classical approach by Cheng and Church 2000
- Introduced the term biclustering to analysis of gene expression data
- Quality of a bicluster: mean squared residue value $H$ defined as follows

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{i,J} - a_{I,j} + a_{I,J})^2$$

- Submatrix $(I,J)$ is considered a bicluster, if $H(I,J) < \delta$
- If $\delta = 0$, the biclusters are perfect

## Biclustering Algorithms

- The model for a perfect bicluster predicts value $a_{ij}$ by a row-constant, a column-constant, and an overall cluster-constant:

$$a_{ij} = a_{i,J} + a_{I,j} - a_{I,J} = \mu + r_i + c_j$$

  with $\mu = a_{I,J}$, $r_i = a_{i,J} - a_{I,J}$, and $c_j = a_{I,j} - a_{I,J}$, i.e., each row and column exhibits absolutely consistent bias
- For a non-perfect bicluster, the prediction of the model deviates from the true value by a residue

$$a_{ij} = res(a_{ij}) + a_{i,J} + a_{I,j} - a_{I,J}$$

  with $res(a_{ij}) = a_{ij} - a_{i,J} - a_{I,j} + a_{I,J}$ which is the optimization criterion $H(I,J)$ above

## Biclustering Algorithms

Related Algorithms

- *p*-cluster algorithm specializes the $\delta$-bicluster-property to a pairwise property of two objects in two attributes
- FLOC uses a randomized procedure
- MaPle introduces improved pruning
- CoClus follows a *k*-means-like approach

## Biclustering: Summary

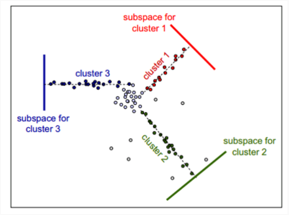
Related Algorithms

- Biclustering models do not fit exactly into the spatial intuition behind subspace, projected, or correlation clustering
- Models rather make sense in view of a data matrix
- Strong point: the models generally do not rely on the locality assumption
- Models differ substantially, so a careful selection is a non-trivial task

# Kapitel 5: Clustering High-dim Data

## Motivation

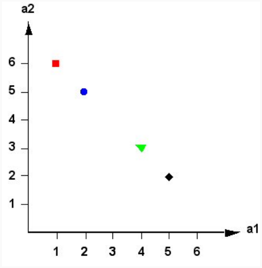
- As we have seen before, there may be more than simple (independent) feature relevancy: feature correlation
- Example:
  - Cluster 3 exists in an axis-parallel subspace (so far so good)
  - However, clusters 1 and 2 exist in (different) arbitrarily oriented subspaces that represent different correlations among attributes

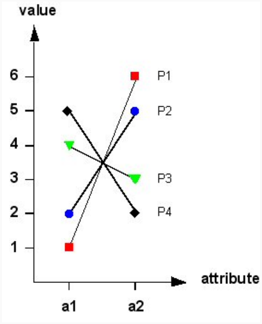


- Can the methods introduced so far capture those patterns?

## Motivation

- Subspace/projected clustering methods
  - Obviously, these methods are restricted to axis-parallel clusters, i.e., assume feature independence
- Pattern-based methods
  - Can find simple pairwise positive correlations (coherent evolution)
  - However, negative correlations have no additive pattern:

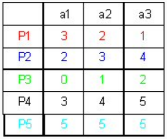




## Motivation

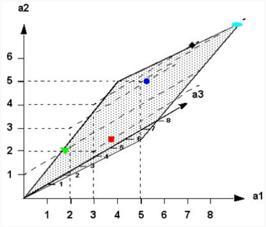
- Pattern-based methods (cont.)
    - Also, more complex relationships between several features are out of scope of pattern-based approaches
    - Example:

      Complex correlation among three attributes: $a1 - 2 \cdot a2 + a3 = 0$

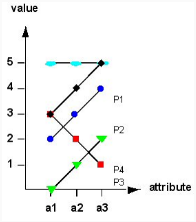


| | a1 | a2 | a3 |
|---|---|---|---|
| P1 | 3 | 2 | 1 |
| P2 | 2 | 3 | 4 |
| P3 | 0 | 1 | 2 |
| P4 | 3 | 4 | 5 |
| P5 | 5 | 5 | 6 |

Data matrix



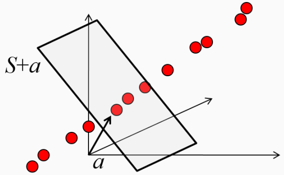
Data space

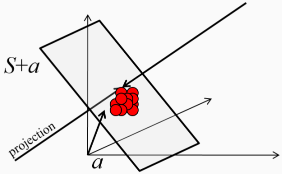


Parallel coordinates

# Correlation Clustering: General Idea

- To find clusters in subspaces of correlated attributes, a more general approach is necessary
- This more general approach is called oriented clustering aka. generalized subspace/projected clustering aka. correlation clustering in the literature[1]
- Assumption: any cluster is located in an arbitrarily oriented affine subspace $S + a$ of $\mathbb{R}^d$



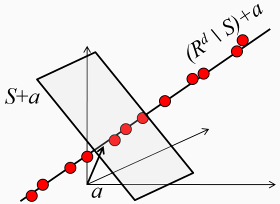


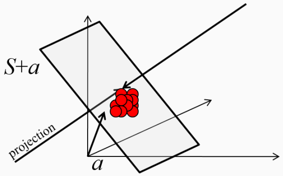
[1]Note: different notion of “Correlation Clustering” in machine learning community, e.g. cf. Bansal, Blum, S. Chawla: Correlation clustering. Machine Learning, 56:89–113, 2004.

## Correlation Clustering: General Idea

- Affine subspace $S + a$ of $\mathbb{R}^d$ is interesting if a set of points clusters within this subspace
- Points may exhibit high variance in perpendicular subspace $(\mathbb{R}^d\ S) + a$

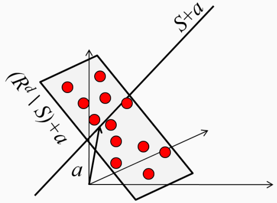
  $\Rightarrow$ points form a hyperplane located in this subspace $(\mathbb{R}^d\ S) + a$

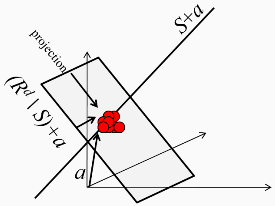




## Correlation Clustering: General Idea

- So in general: points on a hyperplane appear to follow linear dependencies among the attributes participating in the description of the hyperplane
- Another example with a 1D subspace / 2D hyperplane





- How can we capture that???

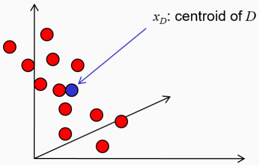
## PCA Revisited

- Assume, we know the cluster members, how can we capture the relevant subspace?
- In order to find the directions of high/low variance of a set of points *D*, we could use PCA

- So we could compute the covariance matrix $\Sigma_D$ of *D*, i.e.,

$$\Sigma_D = \frac{1}{|D|} \sum_{x \in D} (x - \mu_D)(x - \mu_D)^T,$$

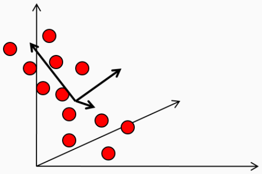
where $\mu_D$ is the mean (or centroid) of the points in *D*, and compute the eigen decomposition.



## PCA Revisited

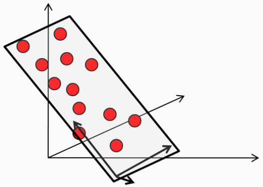
- The decomposition $\Sigma_D = V_D E_D V_D^T$ gives us
  - The $d$ eigen values in the diagonal matrix $E_D$ representing the variance along the direction of highest variance
  - The $d$ $d$-dimensional eigenvectors in the matrix $V_D$ indicating the directions of highest variance sorted by decreasing eigen values

- $E_D$: $d \times d$ diagonal matrix with eigen values
- $V_D$: $d \times d$ orthonormal matrix with eigen vectors



## PCA Computes the Correlation

- If points in $D$ forming $\lambda$-dimensional hyperplane, this hyperplane is spanned by the first $\lambda$ eigenvectors of $V_D$ (called "strong" eigenvectors), denoted by $\tilde{V}_D$
- The subspace where the points cluster is spanned by the remaining $d - \lambda$ eigenvectors (called "weak" eigenvectors), denoted by $\hat{V}_D$
- For the eigensystem, the sum of the smallest $d - \lambda$ eigenvalues is minimal under all possible transformations
- Thus, points in $D$ cluster optimally dense in this subspace



- Note again: the subspace where the points cluster is spanned by the weak eigenvectors $\hat{V}_D$

## General Model for Correlation Clusters

- As a general model for correlation clusters [2], we consider a $\lambda$-dimensional hyperplane accommodating the points of a correlation cluster $C$ defined by an equation system of $d - \lambda$ equations for $d$ variables and the affinity (e.g. the mean point $\mu_C$ of all cluster members):

$$\hat{V}_C^T x = \hat{V}_C^T \mu_C$$

- The equation system approximately fulfilled for all points $x \in C$
- This is quantitative model for the cluster allowing for probabilistic prediction (classification)
- Note: correlations are observable, linear dependencies are merely an assumption to explain the observations — predictive model allows for evaluation of assumptions and experimental refinements

[2] Achtert, Böhm, Kriegel, Kröger, Zimek: Deriving quantitative models for correlation clusters. In Proc. SIGKDD, Philadelphia, PA, 2006

## PCA-based Correlation Cluster Approaches

- Many algorithms include PCA for feature evaluation into the clustering process similar to (axis-parallel) variance analysis
- But how do we know *C*, the set of points we should apply PCA on ion order to get the correct subspace?
- Similar to the axis-parallel versions, we need some assumptions (very often a locality assumption, see above)
- Examples:
    - ORCLUS[3]: integrates distance function into *k*-means, relies on a cluster-based locality assumption
    - 4C[4]: integrates distance function into DBSCAN, relies on instance-based locality assumption
    - Extensions to 4C: COPAC and ERiC

[3] Aggarwal, Yu: Finding generalized projected clusters in high dimensional space. In Proc. SIGMOD, Dallas, TX, 2000

[4] Böhm, Kailing, Kröger, and Zimek: Computing clusters of correlation connected objects. In Proc. SIGMOD, Paris, France, 2004

## ORCLUS: Basics

- A generalized projected cluster is a set of orthonormal vectors $E$ and a set of points $C$ such that the points in $C$ are closely clustered in the subspace defined by the vectors $E$ (where $|E| \leq d$)
- Input:
  - The number of clusters $k$
  - The average dimensionality of the subspace of the clusters, $l$
- Output:
  - A set of $k$ clusters and their associated subspaces of dimensionality $l$
- Main idea
  - To find the subspace of a cluster $C_i$, compute the covariance matrix $M_i$ for $C_i$ and determine the eigenvectors
  - Pick “approx.” $l$ eigenvectors with the smallest eigenvalues

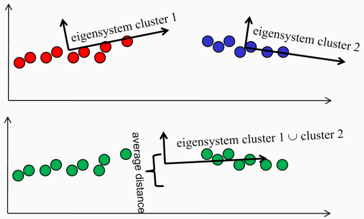
## ORCLUS: Overview of the Algorithm

- Very similar to PROCLUS but use PCA on locality of cluster representatives instead of variance analysis
- $k$-means like approach (uses centroids rather than medoids)
- Start with $k_c > k$ seeds
- Assign points to clusters according to distance function based on the eigensystem of the current cluster (starting with axes of data space, i.e. Euclidean distance)
- The eigensystem is iteratively adapted based on the updated cluster members (i.e., not only optimize the centroids but also the eigensystems of each cluster)
- Reduce the number of clusters $k_c$ in each iteration by merging best-fitting cluster pairs

## ORCLUS: Merging Clusters

- Find best fitting pair of clusters: least average distance in the projected space spanned by weak eigenvectors of the merged clusters

- Two clusters $C_i$ and $C_j$ exist in possibly different subspaces
- Compute the subspace of their union $C_i \cup C_j$ (take the smallest $l$ eigenvalues)



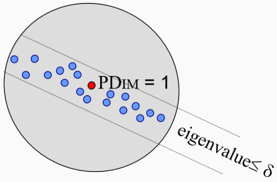
- Check mean square error (MSE) of the points from the new centroid in this new subspace
- Assess average distance in all merged pairs of clusters and finally merge the best fitting pair (smallest MSE)

## 4C: Computing Correlation Connected Clusters

- Density-based cluster-paradigm, see PREDECON
- Extend a cluster from a seed as long as a density-criterion is fulfilled — otherwise pick another seed unless all data objects are assigned to a cluster or noise
- Density criterion: minimal required number of points in the neighborhood of a point
- Neighborhood: distance between two points ascertained based on the eigensystems of both compared points

## 4C: Distance Adaption

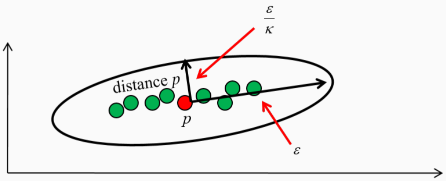
- Perform PCA on the local neighborhood S of p to find subspace correlations
- A parameter $\delta$ discerns large from small eigenvalues
- $CorDim(S) = \#\text{eigenvalues} > \delta$
- In the eigenvalue matrix $E_p$ of $p$, large eigenvalues are replaced by 1, small eigenvalues by a value $\kappa >> 1$



- The adopted eigenvalue matrix of $p$ is denoted by $E'_p$

# 4C: Distance Adaption

- Effect on distance measure:



- Distance is not symmetric:
  - distance of $p$ and $q$ w.r.t. $p$

$$\sqrt{(p-q) \cdot V_p \cdot E'_p \cdot V_p^T \cdot (p-q)^T}$$

    relies on $E'_p$ and $V_p$
  - distance of $p$ and $q$ w.r.t. $q$

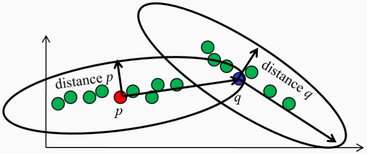
$$\sqrt{(q-p) \cdot V_q \cdot E'_q \cdot V_q^T \cdot (q-p)^T}$$

    relies on $E'_q$ and $V_q$

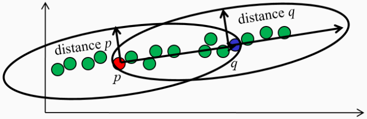
## 4C: Distance Adaption

- Since DBSCAN needs a symmetric distance measure, use the maximum
- Then, *p* and *q* are “correlation-$\varepsilon$-neighbors” if

$$\max \left\{ \begin{array}{c} \sqrt{(p-q) \cdot V_p \cdot E'_p \cdot V_p^T \cdot (p-q)^T}, \\ \sqrt{(q-p) \cdot V_q \cdot E'_q \cdot V_q^T \cdot (q-p)^T} \end{array} \right\} \leq \varepsilon$$



*p* and *q* are no $\varepsilon$-neighbors



*p* and *q* are $\varepsilon$-neighbors

(Circles indicate the $\varepsilon$ threshold)

## 4C: Summary

- Finds arbitrary number of clusters
- Requires specification of density-thresholds
    - $\mu$ (minimum number of points): rather intuitive
    - $\varepsilon$ (radius of neighborhood): hard to guess
- Additional paramters:
    - Maximal dimensionality $\lambda$ of correlation clusters (required, otherwise, 4C can also find full-dimensional clusters like DBSCAN)
    - $\delta$ for distinction between strong and weak eigen vectors
- These two parameters are erased by the extensions such as COPAC: take the minimum number of strong eigen vectors that explain a given portion of the overall variance (see PCA)

## Correlation Clustering with PCA

- PCA is a mature technique, allows construction of a broad range of similarity measures for local correlation of attributes
- Can be replaced by similar techniques like relevant component analysis (RCA), independent component analysis (ICA), etc.
- Key limitation still: all approaches suffer from the locality assumption because PCA only works, if the set of points on which PCA is performed is a fairly good approximation of a subspace/correlation cluster

We need a new principle to employ correlation clustering (coming up)

## Correlation Clustering with Hough Transform

This new principle is the Hough transform (originally developed image analysis)

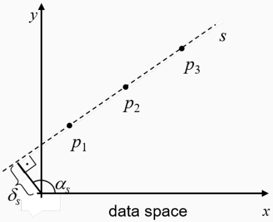
- Transform each object into a so-called parameter space representing all possible subspaces accommodating this object (i.e. all hyperplanes through this object)
- This parameter space is a continuum of all these subspaces
- However, the subspaces are represented by a considerably small number of parameters
- This transform is a generalization of the Hough Transform (which is designed to detect linear structures in 2D images) for arbitrary dimensions

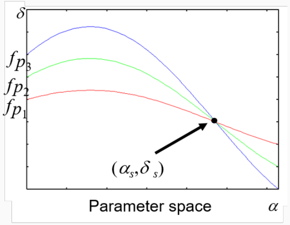
## The Hough Transform in 2D

Check out the board ...

## The General Hough Transform

- For each $d$-dimensional point $p$ there is an infinite number of ($d$-1)-dimensional hyper-planes through $p$
- Each of these hyper-planes $s$ is defined by $(p, \alpha_1, \dots, \alpha_{d-1})$, where $\alpha_1, \dots, \alpha_{d-1}$ is the normal vector $n_s$ of the hyper-plane $s$
- The function $f_p(\alpha_1, \dots, \alpha_{d-1}) = \delta_s = \langle p, n_s \rangle$ maps $p$ and $\alpha_1, \dots, \alpha_{d-1}$ onto the distance $\delta_s$ of the hyper-plane $s$ to the origin
- The parameter space plots the graph of this function

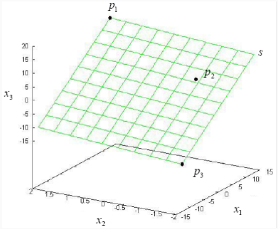


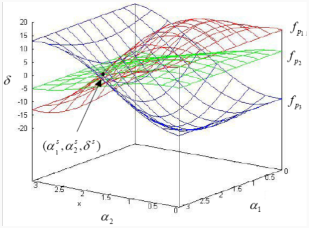


## The General Hough Transform

Properties:

- Point in the data space = sinusoide curve in the parameter space
- Point in the parameter space = hyper-plane in the data space
- Points on a common hyper-plane in the data space (cluster) = sinusoide curves intersecting at one point in the parameter space
- Intersection of sinusoide curves in the parameter space = hyper-plane accommodating the corresponding points in data space

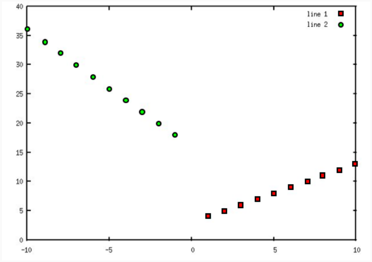




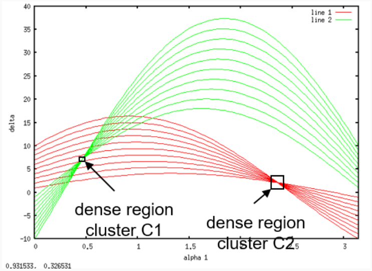
## Using the Hough Transform to Detect Clusters

So we can use the parameter space to detect correlation clusters:

- Determine all intersection points of at least *m* curves in the parameter space (where *m* is a minimum number of points in a cluster threshold) to detect (*d*-1)-dimensional clusters
- Exact solution (check all pair-wise intersections) is usually too costly



data space



parameter (hough) space

## CASH: Clustering in Arbitrary Subspaces Using the Hogh Transform

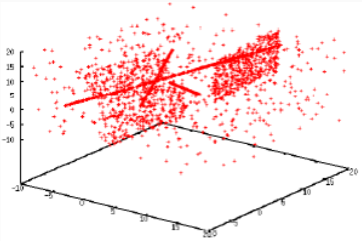
The CASH algorithm employs a grid-based bisecting search (input: minimum number of points *m*, and minimum number of splits *s*):

- Find dense regions in parameter space
- Define dense grid cells as cells still intersected by *m* parametrization functions after *s* splits
- Construct the candidate grid cells by recursively splitting the parameter space along one dimension (in fixed order)
- Always split the candidate cell with the highest number of intersecting functions next (best first)
- Prune candidate cells where less than *m* functions intersect
- If a cell has been split *s* times and still intersects *m* functions, a cluster is found; delete corresponding functions (from other candidate cells) and continue splits until no candidate is remaining

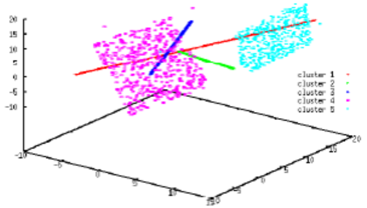
## CASH: Clustering in Arbitrary Subspaces Using the Hogh Transform

- Dense grid cell represents ($d$-1)-dimensional linear structure
- Transform corresponding data objects in corresponding ($d$-1)-dimensional space and repeat the search recursively to find lower dimensional clusters
- This is comparable to a bottom-up search — WHY?

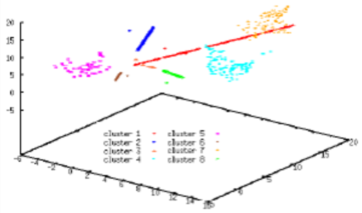
## An Empirical Comparison



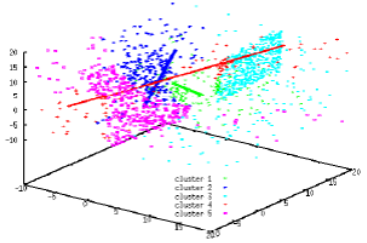
(a) Data set DS2.



(b) CASH – Cluster 1 - 5.



(c) 4C – Cluster 1 - 8.



(d) ORCLUS – Cluster 1 - 5.

## Summary

- Traditional clustering in high dimensional spaces is most likely meaningless with increasing dimensionality (curse of dimensionality)
- Clusters may be found in (generally arbitrarily oriented) subspaces of the data space
- The partitioning need not be unique (clusters may overlap)
- The subspaces may be axis-parallel or arbitrarily oriented
- Analysis of this general problem:
  - Sub-problem 1: search for clusters
  - Sub-problem 2: search for subspaces

## Summary

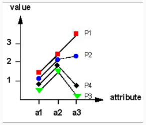
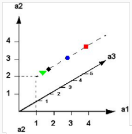
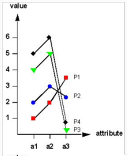
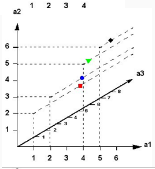
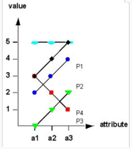
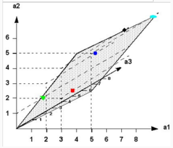
- Analysis of the 2nd sub-problem (subspace search)
  - A naive solution would examine all possible subspaces to look for clusters
  - The search space of all possible arbitrarily oriented subspaces is infinite
  - We need assumptions and heuristics to develop a feasible solution
- What assumptions did we get to know here to solve the subspace search problem?
  - The search space is restricted to certain subspaces
  - A clustering criterion that implements the downward closure property enables efficient search heuristics
  - The locality assumption enables efficient search heuristics
  - Assuming simple additive models (“patterns”) enables efficient search heuristics
  - . . .

## Summary

- Remember: also for the clustering problem (1st sub-problem) we need assumptions and heuristics that have an impact on the algorithms' properties
  - Cluster model (what patterns to look for
  - Model specific assumptions/parameters, e.g. number of clusters need to be specified, results are not deterministic e.g. due to randomized procedures, ...

## Summary

Overview (classification according to sub-problem 2):

| Matrix-Pattern | | Problem | Spatial Pattern | |
|---|---|---|---|---|
|  | Constant values in columns, change of values only on rows | Subspace / Projected Clustering | Axis-parallel hyperplanes |  |
|  | From constant values in rows and columns (no change of values) to arbitrary change of values in common direction | Pattern-based / Bi-Clustering | Special cases of axis-parallel to special cases of arbitrarily oriented hyperplanes |  |
|  | No particular pattern | Correlation Clustering | Arbitrarily oriented hyperplanes |  |

## Outlook

- Recall, (unsupervised) outlier detection is very similar (but often considered to be orthogonal) to clustering
- Outlier detection in high dimensional spaces is also often very hard and similar problems can occur, e.g. points are only outliers when considering specific subspaces
- There are several approaches to extend full-dimensional outlier detection methods analogously so that subspace outliers can be detected
- If you want, ask Prof. Kröger to sketch one ...