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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

3.1 Forward Selection and Feature Ranking

3.2 Backward Elimination and Random Subspace Selection

3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

Kapitel 1: Intorduction i

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

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 $f : \mathbb{U} \rightarrow \mathbb{R}^n$ of objects from $\mathbb{U}$ to the feature space $\mathbb{F}$.

Similarity Model

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

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

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

Similarity versus Dissimilarity

Comments:

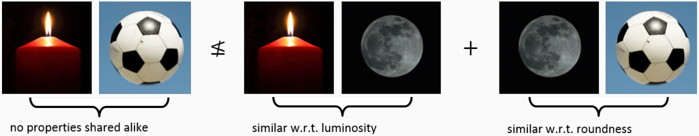
- Often, dissimilarity (distance) is measured instead of similarity
- This is a small but important difference!
  - A similarity measure (*sim*) assigns high values to similar objects
  - A dissimilarity measure (*dist*) assigns low values to similar objects
- The design of f and the definition of *sim*/*dist* 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:

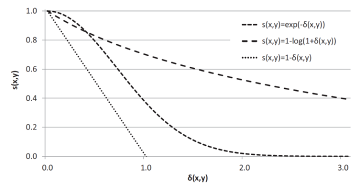


Similarity versus Dissimilarity (again)

- Transformation
    - Let $\mathbb{F}$ be a feature space and $dist : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$ be a distance function
    - Any monotonically decreasing function $f : \mathbb{R} \rightarrow \mathbb{R}$ defines a similarity function $s : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$ 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 $HammingDist(x,y) = \sum_{i=1}^{d} \begin{cases} 1 & : \quad x_i \neq y_i \\ 0 & : \quad \text{else} \end{cases}$

Kapitel 2: Challenges i

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

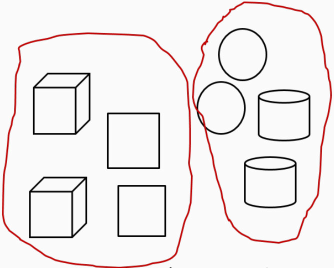
3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

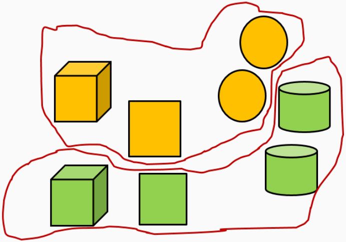
5. Clustering High Dimensional Data

A Motivating Example

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



Based on shape grouping



Based on color grouping

- 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

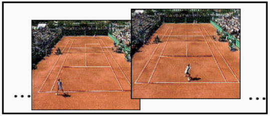
The More the Merrier or More is Less?

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

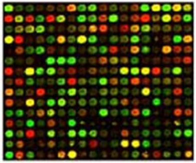
- Example: Image data
  - Low-level image descriptors (color histograms, textures, shape information ...)
  - Regional descriptors: between 16 and 1,000 features
  - ...



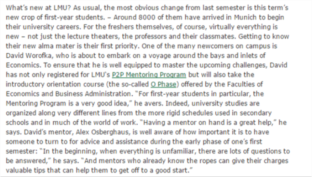
- Example: Metabolome data
  - Feature = concentration of one metabolite (intermediates/results of metabolism)
  - Bavaria newborn screening (for each baby, the blood concentrations of 43 metabolites are measured in the first 48 hours after birth)
  - between 50 and 2,000 features

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
- ...

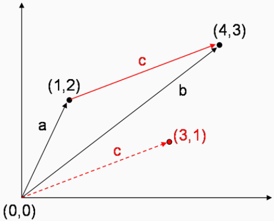
So let us have a closer look on these problems ...

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

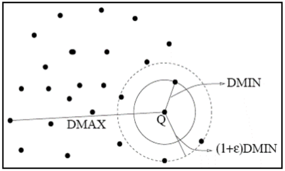
Implications from the concentration of distances:

- 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

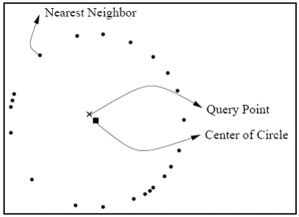
Neighborhood Concept Become Meaningless

Definition: Unstable Neighborhood

- A NN-query is unstable for a given ε 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

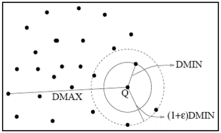


- It can be shown that with growing dimensionality, the probability that a query is unstable converges to 1



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

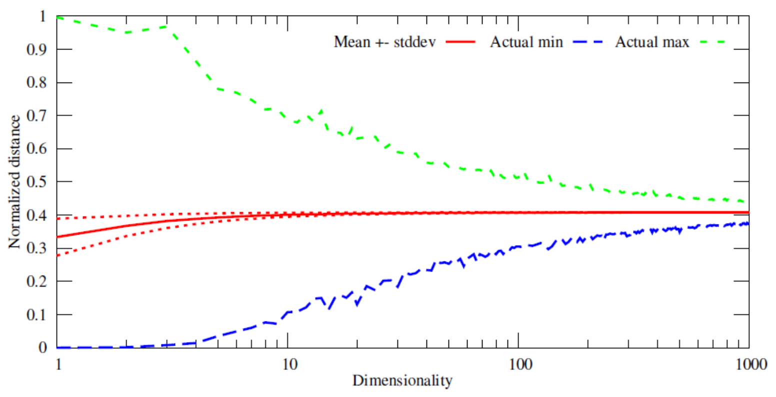
- If $\lim_{d \to \infty}(\frac{VAR_{L_2(x_i,q)}}{\mu^2_{L_2(x_i,q)}}) = 0$
- Then $\forall \varepsilon > 0 : \lim_{d \to \infty} \mathcal{P}(DMAX_d \leq (1 + \varepsilon) DMIN_d) = 1$

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.

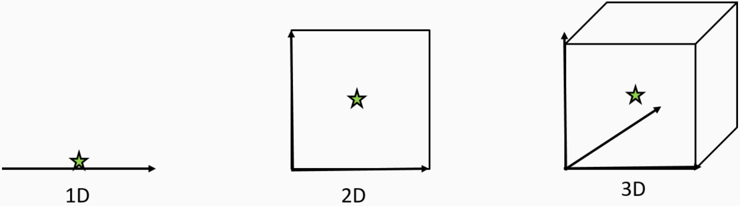
Growing Data Space

- 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. $\frac{\mu^2_{\hat{f}(0)-f(0)}}{f(0)^2} < 0.1$

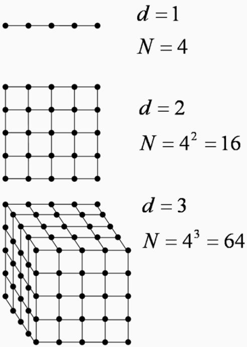
Dim.	Req. sample size to achieve 0.1 error estimate
1	4
2	19
5	768
8	43.700
10	842.000

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



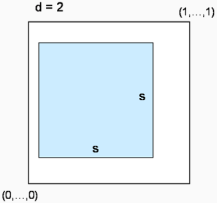
Empty Spaces and Tails

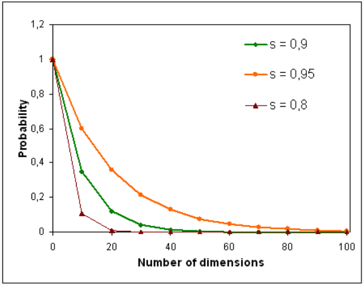
Analogously:

- Consider a simple partitioning scheme, which splits the data in each dimension in 2 halves
- For d dimensions we obtain 2^dpartitions
- 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 $\mathcal{P}(E) = s^d$





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

Empty Spaces and Tails

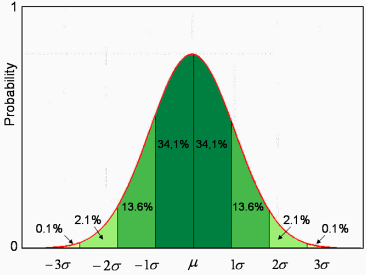
- The same holds of course for a spherical range query (instead of a cubical range query)
- Consequence: with increasing dimensionality the center of the hyper-cube (or more generally: of the data space) becomes less important and the volume of the data space concentrates in its corners (i.e. randomly 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 χ^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:

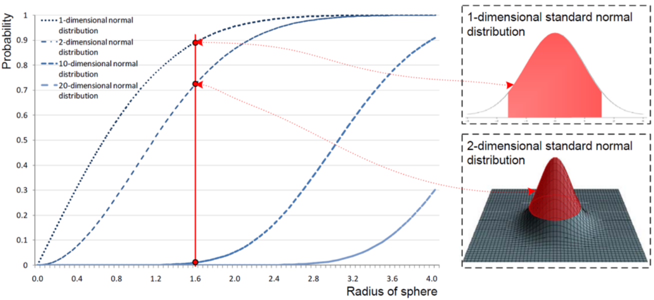
$$\frac{f(X)}{f(0)} = e^{(-\frac{1}{2}X^T X)} \approx e^{(-\frac{1}{2}{\chi^2}_{10})}$$

  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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

3.1 Forward Selection and Feature Ranking

3.2 Backward Elimination and Random Subspace Selection

3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

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.

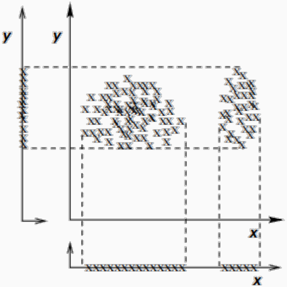
Keep in mind...

Feature selection $\neq$ Dimensionality reduction

Feature selection $\neq$ Feature extraction

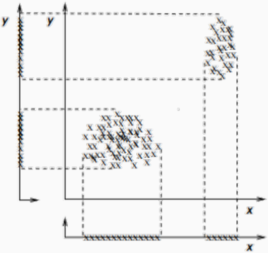
Irrelevant and Redundant Features (Unsupervised Case)

Irrelevance



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

Redundancy

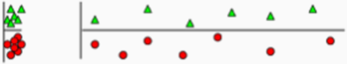


Features *x* and *y* are redundant, because *x* provides (appr.) the same information as feature *y* with regard to discriminating the two clusters

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

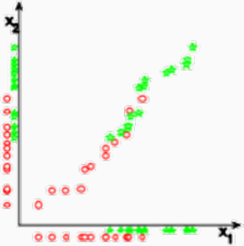
Irrelevant and Redundant Features (Supervised Case)

Irrelevance



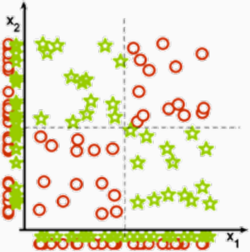
Feature *y* separates well the two classes. Feature *x* is irrelevant. Its addition “destroys” the class separation.

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, χ^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, χ^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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

3.1 Forward Selection and Feature Ranking

3.2 Backward Elimination and Random Subspace Selection

3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

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?

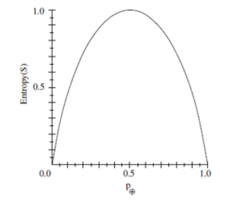
- For discrete values: determine probabilities for all value pairs.
- For real valued features:
  - Discretize the value space (reduction to the case above)
  - Use probability density functions (e.g. uniform, Gaussian,..)
- Example quality measures:
  - Information Gain
  - Chi-square χ^2-statistics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

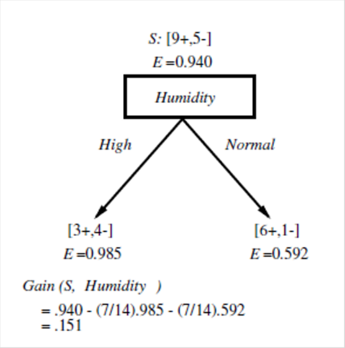
$$Gain(S, d_i) = Entropy(S) - \sum_{S_j} \frac{|S_j|}{|S|} \cdot Entropy(S_j)$$

- 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

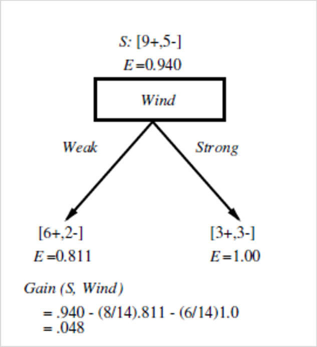
Quality of Features: Information Gain

Example

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



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

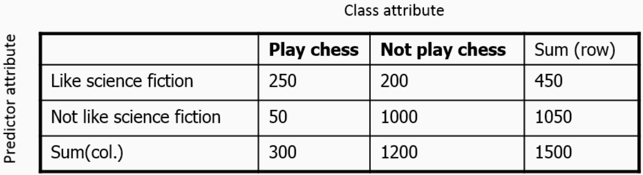


Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

- 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 χ^2 test

$$\chi^2 = \sum_{i=1}^{|C|} \sum_{j=1}^{|Values(d)|}$$

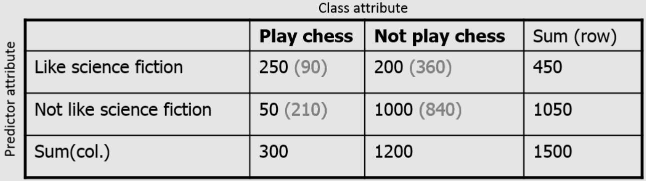
o_{ij}: observed frequency of value *j* in class *i*

o_{ij}: expected frequency of value *j* in class *i*

Quality of Features: Chi-square Statistics

Example

- Compute the χ^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

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

- Larger 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

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

- Continuous case

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

- 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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

3.1 Forward Selection and Feature Ranking

3.2 Backward Elimination and Random Subspace Selection

3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

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 the closest object having the same class $NN^R_{c_i = C(o)}(o)$ (within-class nearest neighbor) in subspace R, and the closest object belonging to another class $NN^R_{c_j \neq C(o)}(o)$ (between-classes nearest neighbor), 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

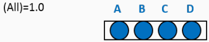
General Approach

- 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 function J (to be maximized), if $X \subset Y$ then $J(X) \leq J(Y)$.
- Branch and Bound starts from the full set F and removes features using a depth-first strategy
- Nodes whose objective function are smaller 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.

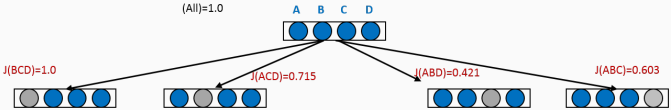
● selected feature ● removed feature



Branch and Bound: Example

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

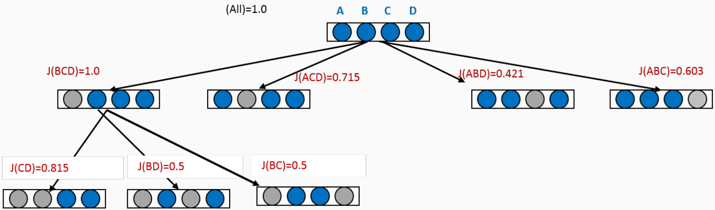
🔵 selected feature ⚪ removed feature



Branch and Bound: Example

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

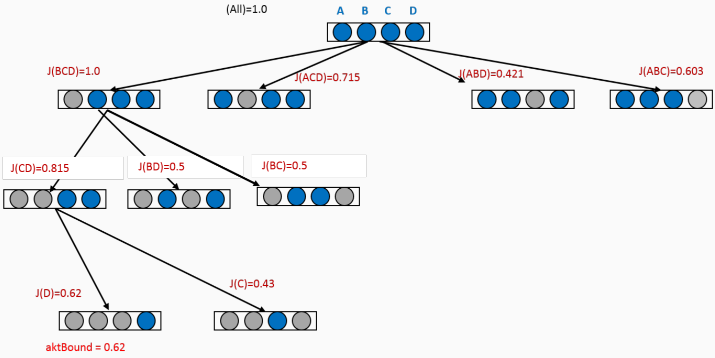
● selected feature ○ removed feature



Branch and Bound: Example

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

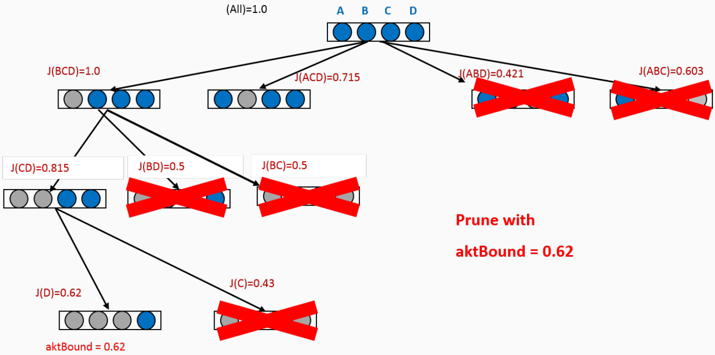
● selected feature ○ removed feature



Branch and Bound: Example

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

● selected feature ○ removed feature



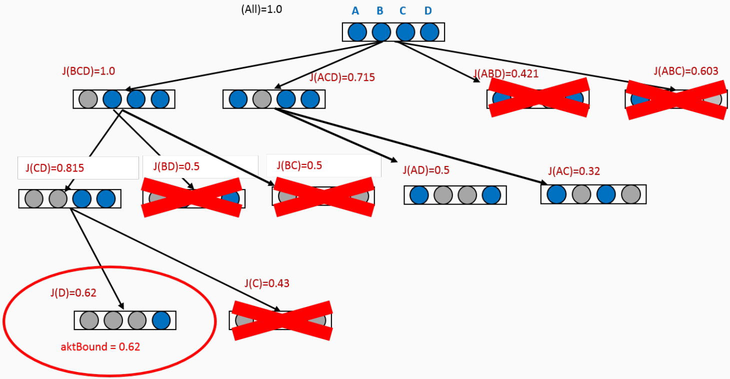
Prune with

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 A in R
  - $X_R^c(u)$: Amount of all identical vectors A in R having class label $c \in C$
  - Inconsistency of u in R: $IC_R(u) = X_R(u) - \max_{c \in C} X_R^c(u)$

  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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

3.1 Forward Selection and Feature Ranking

3.2 Backward Elimination and Random Subspace Selection

3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

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

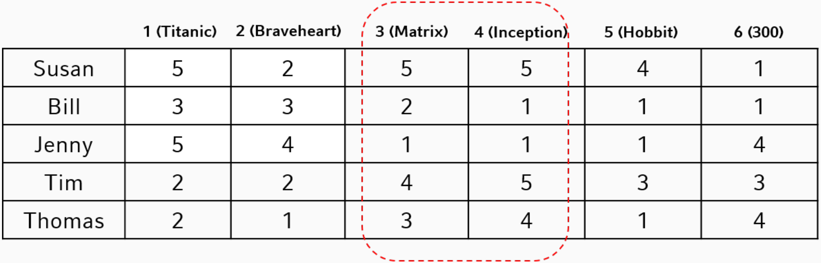
Feature Clustering: General Idea

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, χ^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

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

Vorlesungsteam

Vorlesungsteam

Kapitel 5: Clustering High-dim Data i

1. Intorduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

Vorlesungsteam

Vorlesungsteam