Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Thomas Seidl

Knowledge Discovery and Data Mining I

Winter Semester 2018/19

People

Lecturer

- Prof. Dr. Thomas Seidl

Assistants

- Max Berrendorf
- Julian Busch

Student Assistants

- Maximilian Hünemörder
- Florentin Schwarzer

Schedule

Lecture (begins: 16.10.2018)

Tu.	09:15-11:45	B U101 (Oettingenstraße. 67)

Tutorials (begin: 25.10.2018)

Th.	12:15-13:45	Lehrturm-VU107 (Prof.-Huber-Pl. 2)
Th.	14:15-15:45	Lehrturm-VU107 (Prof.-Huber-Pl. 2)
Fr.	12:15-13:45	Lehrturm-V005 (Prof.-Huber-Pl. 2)
Fr.	14:15-15:45	C 111 (Theresienstr. 41)

Exam

1. Hauptklausur:
   Mo., 25.02.19, 14:00-16:00, B 101 B 201 (Hauptgebäude)
2. Nachholklausur:
   tba

Material, Tutorials & Exam

Material (Slides, Exercises, etc.)

Available on course webpage:

http://www.dbs.ifi.lmu.de/cms/studium_lehre/lehre_master/kdd1819/index.html

Tutorial

- Python Introduction now available on website
- First exercise sheet available for download around 18.10.2018
- Prepare at home
- Presentation and discussion one week after

Exam

- Written exam at the end of semester
- All material discussed in the lecture and tutorials
- Registration via UniWorX

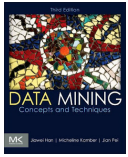
Content of this Course

1. Introduction
   - 1.1 Organisation
   - 1.2 Motivation
   - 1.3 Knowledge Discovery Process
2. Basics
   - 2.1 Data Representation
   - 2.2 Data Reduction
   - 2.3 Visualization
   - 2.4 Privacy
3. Unsupervised Methods
   - 3.1 Frequent Pattern Mining
   - 3.2 Clustering
   - 3.3 Outlier Detection
4. Supervised Methods
   - 4.1 Classification
   - 4.2 Regression
5. Advanced Topics
   - 5.1 Process Mining
   - 5.2 Outlook

Textbook / Acknowledgements

The slides used in this course are modified versions of the copyrighted original slides provided by the authors of the adopted textbooks:

- © Jiawei Han, Micheline Kamber, Jian Pei: *Data Mining – Concepts and Techniques*, 3rd ed., Morgan Kaufmann Publishers, 2011.
  http://www.cs.uiuc.edu/~hanj/bk3
- © Martin Ester and Jörg Sander: *Knowledge Discovery in Databases – Techniken und Anwendungen* Springer Verlag, 2000 (in German).



Motivation

- Data Mining = extraction of patterns from data
- Patterns
  - Regularities – examples: frequent itemsets, clusters
  - Irregularities – examples: outliers
- Not all patterns are useful
  - *"all mothers in our database are female"* $\rightsquigarrow$ trivial/known
  - *"bread, butter is frequent"* given *"bread, butter, salt is frequent"* $\rightsquigarrow$ redundant
- Aggregation of data may help: Basic statistics

What is Data Mining?

Knowledge Discovery in Databases (Data Mining)

Extraction of interesting *(non-trivial, implicit, previously unknown and potentially useful)* information or patterns from data in *large databases*

Roots of Data Mining

- Statistics
- Machine Learning
- Database Systems
- Information Visualization

Data Mining: Motivation

"Necessity is the mother of invention"

Data Explosion Problem

Tremendous amounts of data caused by

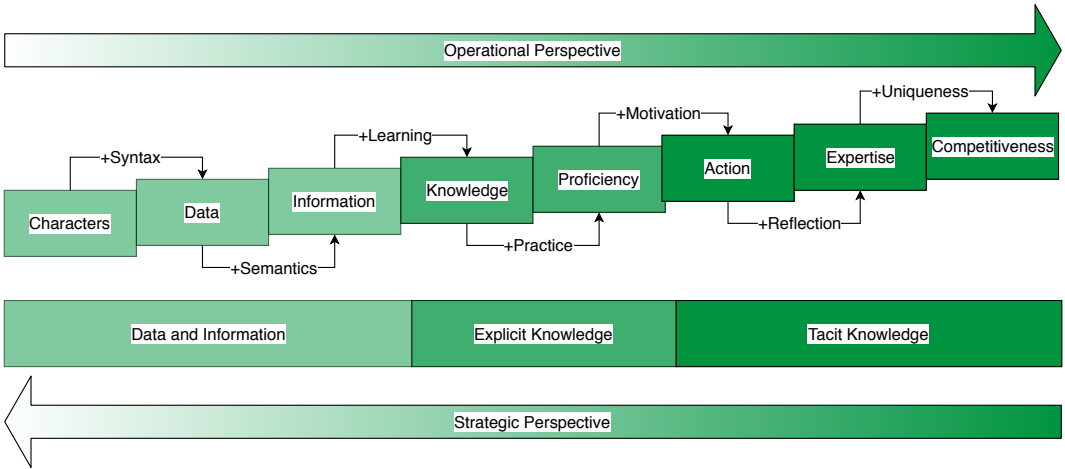
- Automated data collection
- Mature database technology

"We are drowning in data, but starving for knowledge!"

Solution

- Data Warehousing and on-line analytical processing (OLAP)
- Data Mining: Extraction of interesting knowledge (rules, regularities, patterns, constraints) from data in large databases

Data Mining: Motivation



Stairs of Knowledge (K. North) [1]

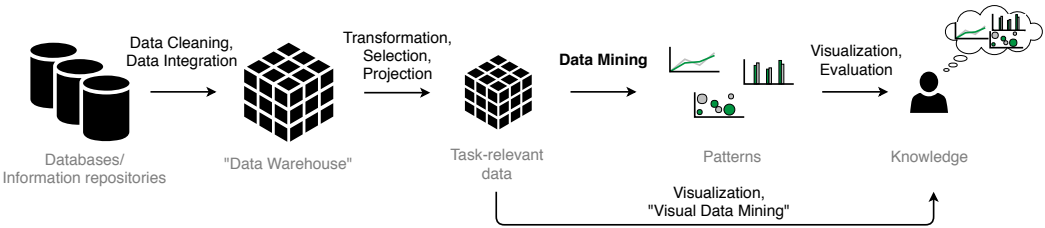
[1] Stairs of Knowledge: North, K.: Wissensorientierte Unternehmensführung - Wertschöpfung durch Wissen. Gabler, Wiesbaden 1998.

Data Mining: Potential Applications

- Database analysis and decision support
  - *Market analysis and management*:
    target marketing, customer relation management, market basket analysis, cross selling, market segmentation
  - *Risk analysis and management*:
    Forecasting, customer retention ("Kundenbindung"), improved underwriting, quality control, competitive analysis
  - *Fraud detection and management*
- Other Applications:
  - Text mining (news group, email, documents) and Web analysis.
  - Intelligent query answering

The Knowledge Discovery Process

- The KDD-Process (Knowledge Discovery in Databases)

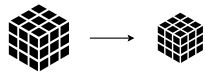


- Data Mining:
  - Frequent Pattern Mining
  - Clustering
  - Classification
  - Regression
  - Process Mining
  - ...

KDD Process: Data Cleaning & Integration

- . . . may take 60% of effort
- Integration of data from different sources
  - Mapping of attribute names, e.g. `C_Nr` $\rightarrow$ `O_Id`
  - Joining different tables, e.g. `Table1 = [C_Nr, Info1]` and `Table2 = [O_Id, Info2]`
    $\rightsquigarrow$ `JoinedTable = [O_Id, Info1, Info2]`
- Elimination of inconsistencies
- Elimination of noise
- Computation of missing values (if necessary and possible): Possible strategies e.g. default value, average value, or application specific computations

KDD Process: Focusing on Task-Relevant Data



Task

- Find useful features, dimensionality/variable reduction, invariant representation
- Creating a target data set

Selections

Select the relevant tuples/rows from the database tables, e.g., sales data for the year 2001

KDD Process: Focusing on Task-Relevant Data

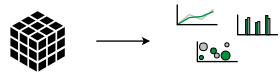
Projections

Select the relevant attributes/columns from the database tables, e.g., (`id`, `name`, `date`, `location`, `amount`) $\rightsquigarrow$ (`id`, `date`, `amount`)

Transformations, e.g.:

- Discretization of numerical attributes, e.g.,
  `amount`: [0, 100] $\rightsquigarrow$ `d_amount`: {low, medium, high}
- Computation of derived tuples/rows and derived attributes:
  - aggregation of sets of tuples, e.g., total amount per months
  - new attributes, e.g., `diff = sales current month - sales previous month`

KDD Process: Basic Data Mining Tasks



Goal

Find patterns of interest

Tasks

- Identify task: Are there labels (in the training data)?
  - *Many* $\rightsquigarrow$ Supervised learning (focus on given concepts)
  - *Some few* $\rightsquigarrow$ Semi-supervised learning (focus on few hidden concepts)
  - *None* $\rightsquigarrow$ Unsupervised learning (many hidden concepts)
- Choose fitting mining algorithm(s)

Basic Mining Tasks: Frequent Itemset Mining

Setting

Given a database of transactions, e.g.

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Motivation

Frequently co-occurring items in the set of transactions indicate correlations or causalities

Examples

- `buys(x, "diapers")`$\Rightarrow$`buys(x, "beers")` [supp: 0.5%, conf: 60%]
- `major(x, "CS")`$\wedge$`takes(x, "DB")`$\Rightarrow$`grade(x,"A")` [supp: 1.0%, conf: 75%]

Basic Mining Tasks: Frequent Itemset Mining

Applications

- Market-basket analysis
- Cross-marketing
- Catalogue design
- Also used as a basis for clustering, classification
- Association rule mining: Determine correlations between different itemsets

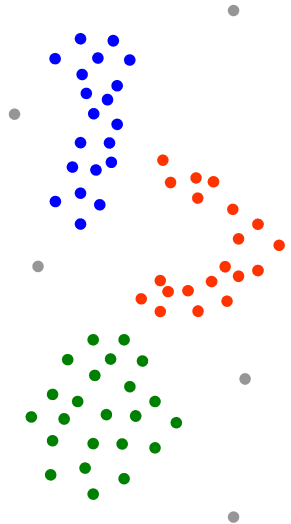
Basic Mining Tasks: Clustering

Setting

- Database of objects
- (Dis-)Similarity function between objects
- Unknown class labels

Task

Group objects into sub-groups (clusters)
"maximizing" intra-class similarity and
"minimizing" interclass similarity



Basic Mining Tasks: Clustering

Applications

- Customer profiling/segmentation
- Document or image collections
- Web access patterns
- ...

Basic Mining Tasks: Classification

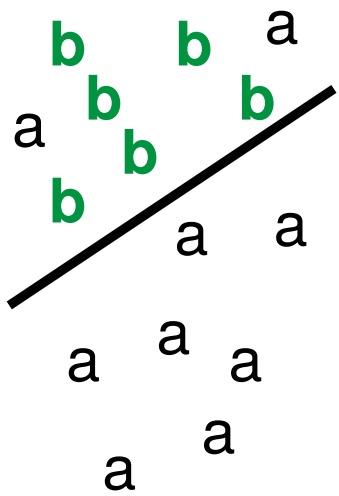
Setting

Class labels are known for a small set of "training data"

Task

Find models/functions/rules (based on attribute values of the training examples) that

- describe and distinguish classes
- predict class membership for "new" objects



Basic Mining Tasks: Classification

Applications

- Classify disease type for tissue samples from gene expression values
- Automatic assignment of categories to large sets of newly observed celestial objects
- Predict unknown or missing values (cf. KDD data cleaning & integration)
- ...

Basic Mining Tasks: Regression

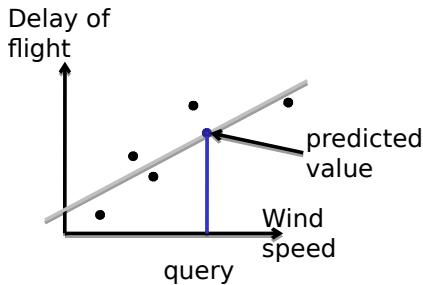
Setting

Numerical output values are known for a small set of "training data"

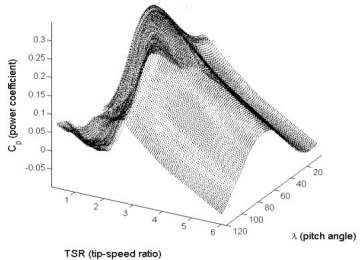
Task

Find models/functions/rules (based on attribute values of the training examples) that

- describe the numerical output values of the training data
- predict the numerical value for "new" objects



Basic Mining Tasks: Regression

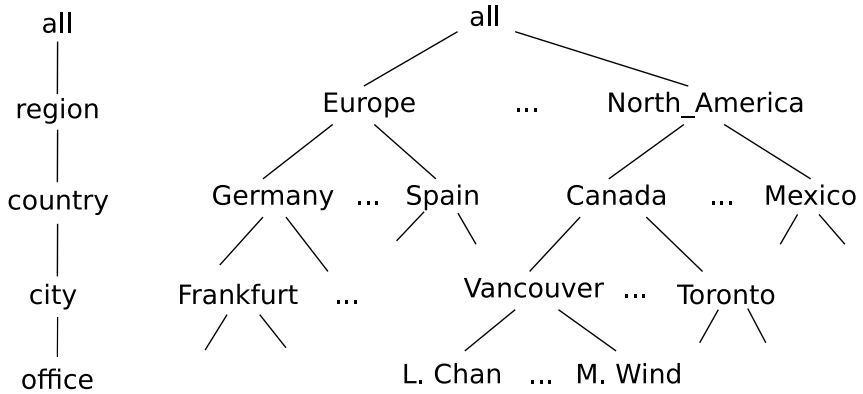


Applications

- Build a model of the housing values, which can be used to predict the price for a house in a certain area
- Build a model of an engineering process as a basis to control a technical system
- ...

Basic Mining Tasks: Generalization Levels

- Generalize, summarize, and contrast data characteristics
- Based on attribute aggregation along concept hierarchies
  - Data cube approach (OLAP)
  - Attribute-oriented induction approach



Basic Mining Tasks: Other Methods

Outlier Detection

Find objects that do not comply with the general behaviour of the data (fraud detection, rare events analysis)

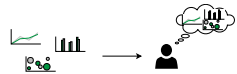
Trends and Evolution Analysis

Sequential patterns (find re-occurring sequences of events)

Methods for special data types, and applications

- Process Mining
- Spatial Data Mining
- Graphs
- ...

KDD Process: Evaluation and Visualization



- Pattern evaluation and knowledge presentation: Visualization, transformation, removing redundant patterns, etc.
- Integration of visualization and data mining:
  - *data* visualization
  - data mining *result* visualization
  - data mining *process* visualization
  - *interactive* visual data mining
- Different types of 2D/3D plots, charts and diagrams are used, e.g. box-plots, trees, scatterplots, parallel coordinates
- Use of discovered knowledge

Summary

- Data mining = Discovering interesting patterns from large amounts of data
- A natural evolution of database technology, machine learning, statistics, visualization, in great demand, with wide applications
- A KDD process includes data cleaning, data integration, data selection, transformation, data mining, pattern evaluation, and knowledge presentation
- Data mining functionalities: characterization, discrimination, association, classification, clustering, outlier and trend analysis, etc.

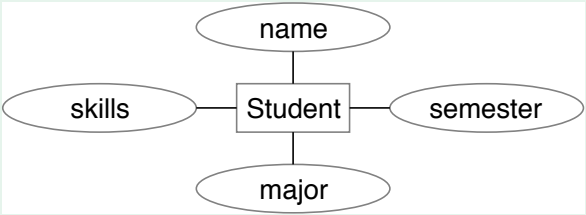
References

	Conference	Journal
Data Mining and KDD	KDD, PKDD, SDM, PAKDD, ICDM, ...	Data Mining and Knowledge Discovery, ...
Database Field	ACM-SIGMOD, ACM-PODS, VLDB, ICDE, EDBT, CIKM, ...	ACM-TODS, J. ACM, IEEE-TKDE, JIIS, VLDBJ, ...
AI and Machine Learning	Machine learning, AAAI, IJCAI, ICLR, ...	Machine Learning, Artificial Intelligence, ...
Statistics	Joint Stat. Meeting, ...	Annals of Statistics, ...
Visualization	CHI (Comp. Human Interaction), ...	IEEE Trans. Visualization and Computer Graphics, ...

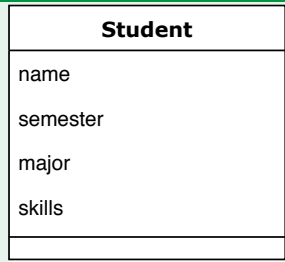
Agenda

Objects and Attributes

Entity-Relationship Diagram (ER)



UML Class Diagram



Data Tables (Relational Model)

name	sem	major	skills
Ann	3	CS	Java, C, R
Bob	1	CS	Java, PHP
Charly	4	History	Piano
Debra	2	Arts	Painting

Overview of (Attribute) Data Types

Simple Data Types

Numeric/metric, Categorical/nominal, ordinal

Composed Data Types

Sets, sequences, vectors

Complex Data Types

- Multimedia: Images, videos, audio, text, documents, web pages, etc.
- Spatial, geometric: Shapes, molecules, geography, etc.
- Structures: Graphs, networks, trees, etc.

Simple Data Types: Numeric Data

Numeric Data

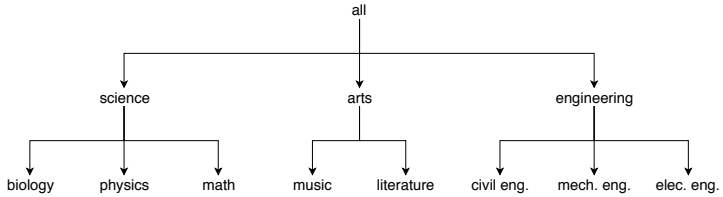
- Numbers: natural, integer, rational, real numbers
- Examples: age, income, shoe size, height, weight
- Comparison: difference
- Example: 3 is more similar to 30 than to 3,000

Simple Data Types: Categorical Data

- "Just identities"
- Examples:
  - occupation = { butcher, hairdresser, physicist, physician, ... }
  - subjects = { physics, biology, math, music, literature, ... }
- Comparison: How to compare values?
  - Trivial metric:

$$d(p, q) = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{else} \end{cases}$$

  - Generalization hierarchy: Use path length



Generalization: Metric Data

Metric Space

Metric space (O, d) consists of object set O and *metric distance* function $d : O \times O \to \mathbb{R}^{\geq 0}$ which fulfills:

Symmetry:	$\forall p, q \in O : d(p, q) = d(q, p)$
Identity of Indiscernibles:	$\forall p, q \in O : d(p, q) = 0 \iff p = q$
Triangle Inequality:	$\forall p, q, o \in O : d(p, q) \leq d(p, o) + d(o, q)$

Example: Points in 2D space with Euclidean distance

Simple Data Types: Ordinal

Characteristic

There is a (total) order $\leq$ on the set of possible data values O:

$$\begin{aligned}
\text{Transitivity:} &\quad \forall p, q, o \in O : p \leq q \wedge q \leq o \implies p \leq o \\
\text{Antisymmetry:} &\quad \forall p, q \in O : p \leq q \wedge q \leq p \implies p = q \\
\text{Totality:} &\quad \forall p, q \in O : p \leq q \vee q \leq p
\end{aligned}$$

Examples

- Words & lexicographic ordering: *high* $\leq$ *highschool* $\leq$ *highscore*
- (Vague) sizes: *tiny* $\leq$ *small* $\leq$ *medium* $\leq$ *big* $\leq$ *huge*
- Frequencies: *never* $\leq$ *seldom* $\leq$ *rarely* $\leq$ *occasionally* $\leq$ *sometimes* $\leq$ *often* $\leq$ *frequently* $\leq$ *regularly* $\leq$ *usually* $\leq$ *always*

Composed Data Types: Sets

Characteristic

Unordered collection of individual values

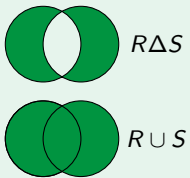
Example

- skills = { Java, C, Python }

Comparison

- Symmetric Set Difference:

$$\begin{aligned} R \Delta S &= (R - S) \cup (S - R) \\ &= (R \cup S) - (R \cap S) \end{aligned}$$



- Jaccard Distance: $d(R, S) = \frac{|R \Delta S|}{|R \cup S|}$

Composed Data Types: Sets

Bitvector Representation

- Given a set S, an ordered base set $B = (b_1, \ldots, b_n)$, create binary vector $r \in \{0,1\}^n$ with $r_i = 1 \iff b_i \in S$.
- Hamming distance: Sum of different entries (equals cardinality of symmetric set difference)

Example

- Base: B = (Math, Physics, Chemistry, Biology, Music, Arts, English)
- S = { Math, Music, English } = (1,0,0,0,1,0,1)
- R = { Math, Physics, Arts, English } = (1,1,0,0,0,1,1)
- Hamming(R, S) = 3

Composed Data Types: Sequences, Vectors

Characteristic

- Put n values of a domain D together
- Order does matter: $I_n \to D$ for an index set $I_n = \{1, \ldots, n\}$

Examples

(Simple) sum	$d_1(o,q) = \sum_{i=1}^{n} \lvert o_i - q_i \rvert$	(Manhattan)
Root of sum of squares	$d_2(o,q) = \sqrt{\sum_{i=1}^{n} (o_i - q_i)^2}$	(Euclidean)
Maximum	$d_3(o,q) = \max_{i=1}^{n} \lvert o_i - q_i \rvert$	(Maximum)
General formula	$d_4(o,q) = \sqrt[p]{\sum_{i=1}^{n} \lvert o_i - q_i \rvert^p}$	(Minkowski)
Weighting of dimensions	$d_5(o,q) = \sqrt[p]{\sum_{i=1}^{n} w_i \cdot \lvert o_i - q_i \rvert^p}$	(Weighted Minkowski)

Complex Data Types

Components

- Structure: graphs, networks, trees
- Geometry: shapes/contours, routes/trajectories
- Multimedia: images, audio, text, etc.

Similarity models: Approaches

- Direct measures – highly data type dependent
- Feature engineering – explicit vector space embedding with hand-crafted features
- Feature learning – explicit vector space embedding learned by machine learning model, e.g. neural network
- Kernel trick – implicit vector space embedding

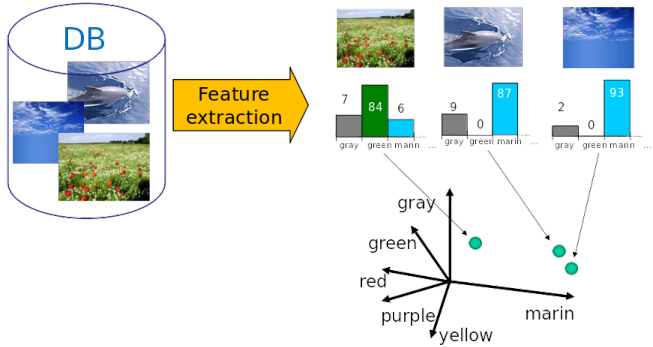
Complex Data Types

Examples for similarity models

	Direct	Feature engineering	Feature learning	Kernel-based
Graphs	Structural Alignment	Degree Histograms	Node embeddings	Label Sequence Kernel
Geometry	Hausdorff Distance	Shape Histograms	Spectral Neural Network	Spatial Pyramid Kernel
Sequences	Edit Distance	Symbol Histograms	Recurrent neural network (RNN)	Cosine Distance

Feature Extraction

- Objects from database DB are mapped to feature vectors



- Feature vector space
  - Points represent objects
  - Distance corresponds to (dis-)similarity

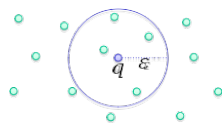
Similarity Queries

- Similarity queries are basic operations in (multimedia) databases
- Given: Universe O, database DB, distance function d and query object q

Range query

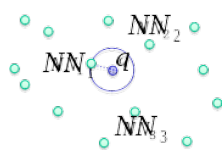
Range query for range parameter $\epsilon \in \mathbb{R}_0^+$:

$$range(DB, q, d, \epsilon) = \{o \in DB \mid d(o, q) \leq \epsilon\}$$



Nearest neighbor query

$$NN(DB, q, d) = \{o \in DB \mid \forall o' \in DB : d(o, q) \leq d(o', q)\}$$



Similarity Queries

k-nearest neighbor query

k-nearest neighbor query for parameter $k \in \mathbb{N}$:

$$NN(DB, q, d, k) \subset DB \text{ with } |NN(DB, q, d, k)| = k \text{ and}$$

$$\forall o \in NN(DB, q, d, k), o' \in DB - NN(DB, q, d, k) : d(o, q) \leq d(o', q)$$

Ranking query

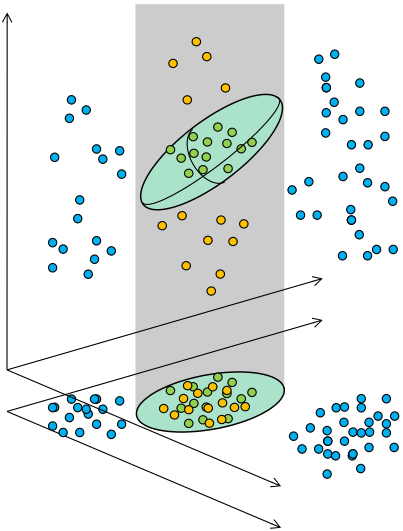
Ranking query (partial sorting query): "get next" functionality for picking database objects in an increasing order w.r.t. their distance to q:

$$\forall i \leq j : d(q, rank_{DB,q,d}(i)) \leq d(q, rank_{DB,q,d}(j))$$

Similarity Search

- Example: Range query $range(DB, q, d, \epsilon) = \{o \in DB \mid d(o, q) \leq \epsilon\}$
- Naive search by sequential scan
  - Fetch database objects from secondary storage (e.g. disk): $O(n)$
  - Check distances individually: $O(n)$
- Fast search by applying database techniques
  - Filter-refine architecture
    - Filter: Boil database DB down to (small) candidate set $C \subseteq DB$
    - Refine: Apply exact distance calculation to candidates from C only
  - Indexing structures
    - Avoid sequential scans by (hierarchical or other) indexing techniques
    - Data access in (fast) $O(n)$, $O(\log n)$ or even $O(1)$

Filter-Refine Architecture

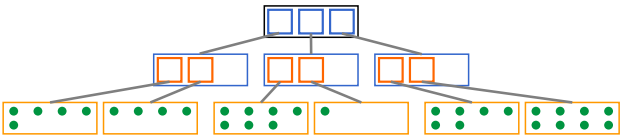


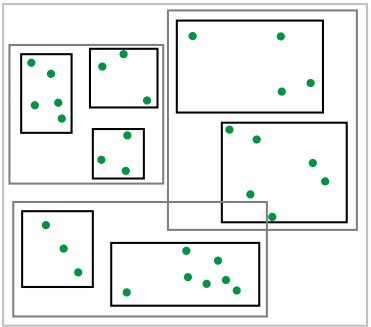
- Principle of multi-step search:
  1. Fast filter step produces candidate set $C \subset DB$ (by approximate distance function d')
  2. Exact distance function d is calculated on candidate set C only.
- Example: Dimensionality reduction[a]
- ICES[b] criteria for filter quality
  - I ndexable – Index enabled
  - C omplete – No false dismissals
  - E fficient – Fast individual calculation
  - S elective – Small candidate set

[a] GEMINI: Faloutsos 1996; KNOP: Seidl & Kriegel 1998
[b] Assent, Wenning, Seidl: ICDE 2006

Indexing

- Organize data in a way that allows for fast access to relevant objects, e.g. by heavy pruning.





- R-Tree as an example for spatial index structure:
  - Hierarchy of minimum bounding rectangles
  - Disregard subtrees which are not relevant for the current query region

Indexing

- Example: Phone book
- Indexed using alphabetical order of participants
- Instead of sequential search:
  - Estimate region of query object (interlocutor)
  - Check for correct branch
  - Use next identifier of query object
  - Repeat until query is finished

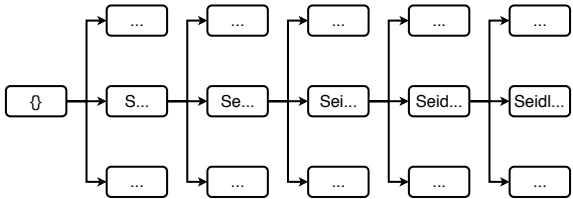


Image source: hierher/flickr, Licence: CC BY 2.0

Agenda

Data Reduction

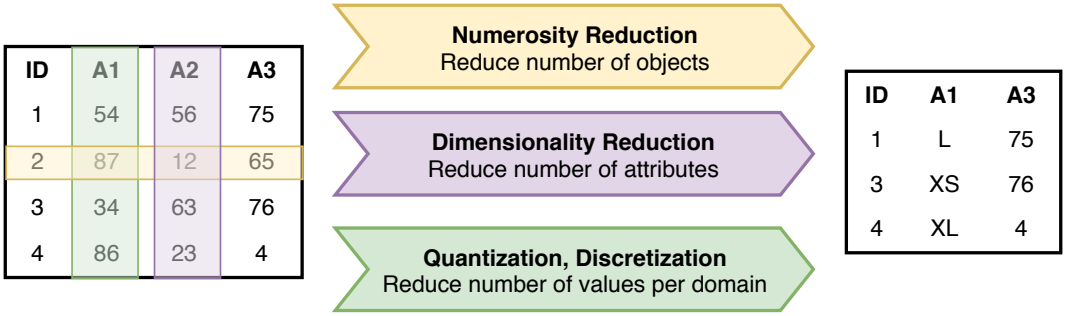
Why data reduction?

- Better perception of patterns
  - Raw (tabular) data is hard to understand
  - Visualization is limited to (hundreds of) thousands of objects
  - Reduction of data may help to identify patterns
- Computational complexity
  - Big data sets cause prohibitively long runtime for data mining algorithms
  - Reduced data sets are useful the more the algorithms produce (almost) the same analytical results

How to approach data reduction?

- Data aggregation (basic statistics)
- Data generalization (abstraction to higher levels)

Data Reduction Strategies



Numerosity reduction

Reduce number of objects

- Sampling (loss of data)
- Aggregation (model parameters, e.g., center / spread)

Data Reduction Strategies

Dimensionality reduction

Reduce number of attributes

- Linear methods: feature sub-selection, Principal Components Analysis, Random projections, Fourier transform, Wavelet transform
- Non-linear methods: Multidimensional scaling (force model)

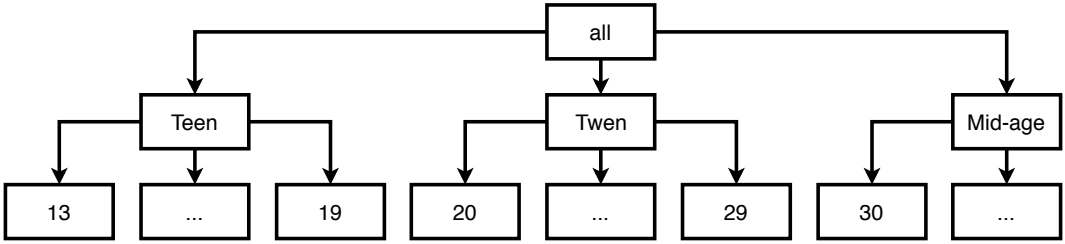
Quantization, discretization

Reduce number of values per domain

- Binning (various types of histograms)
- Generalization along hierarchies (OLAP, attribute-oriented induction)

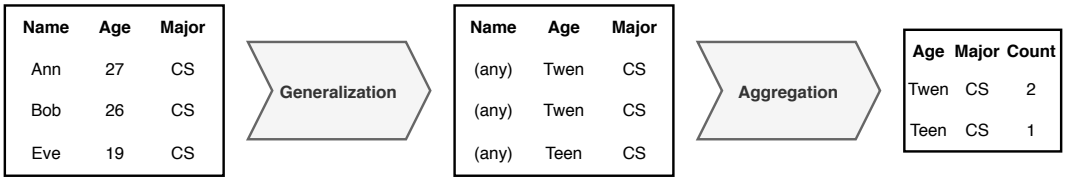
Data Generalization

- Quantization is a special case of generalization
  - E.g., group age (7 bits) to age_range (4 bits)
- Dimensionality reduction is degenerate quantization
  - Dropping age reduces 7 bits to zero bits
  - Corresponds to generalization of age to "all" = "any age" = no information



Data Aggregation

- Aggregation is numerosity reduction (= less tuples)
- Generalization yields duplicates: Merge duplicate tuples and introduce (additional) counter attribute



Basic Aggregates

- Central tendency: Where is the data located? Where is it centered?
  - Examples: mean, median, mode, etc. (see below)
- Variation, spread: How much do the data deviate from the center?
  - Examples: variance / standard deviation, min-max-range, ...

Examples

- Age of students is around 20
- Shoe size is centered around 40
- Recent dates are around 2020
- Average income is in the thousands

Distributive Aggregate Measures

Distributive Measures

The result derived by applying the function to n aggregate values is the same as that derived by applying the function on all the data without partitioning.

Examples

- $count(D_1 \cup D_2) = count(D_1) + count(D_2)$
- $sum(D_1 \cup D_2) = sum(D_1) + sum(D_2)$
- $min(D_1 \cup D_2) = min(min(D_1), min(D_2))$
- $max(D_1 \cup D_2) = max(max(D_1), max(D_2))$

Algebraic Aggregate Measures

Algebraic Measures

Can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function.

Examples

- $avg(D_1 \cup D_2) = \dfrac{sum(D_1 \cup D_2)}{count(D_1 \cup D_2)} = \dfrac{sum(D_1) + sum(D_2)}{count(D_1) + count(D_2)}$
  $\neq avg(avg(D_1), avg(D_2))$
- $standard\_deviation(D_1 \cup D_2)$

Holistic Aggregate Measures

Holistic Measures

There is no constant bound on the storage size which is needed to determine/describe a sub-aggregate.

Examples

- *median*: value in the middle of a sorted series of values (=50% quantile)

$$median(D_1 \cup D_2) \neq simple\_function(median(D_1), median(D_2))$$

- *mode*: value that appears most often in a set of values
- *rank*: k-smallest / k-largest value (cf. quantiles, percentiles)

Measuring the Central Tendency

Mean – (weighted) arithmetic mean

Well-known measure for central tendency ("average").

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{x}_w = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Mid-range

Average of the largest and the smallest values in a data set:

$$(max + min)/2$$

- Algebraic measures
- Applicable to numerical data only (sum, scalar multiplication)

What about categorical data?

Measuring the Central Tendency

Median

- Middle value if odd number of values
- For even number of values: average of the middle two values (numeric case), or one of the two middle values (non-numeric case)
- Applicable to ordinal data only (an ordering is required)
- Holistic measure

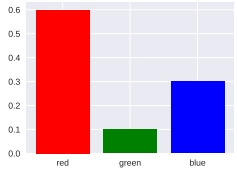
Examples

- never, never, never, rarely, *rarely*, often, usually, usually, always
- tiny, small, big, big, *big*, *big*, big, big, huge, huge
- tiny, tiny, small, *medium, big*, big, large, huge

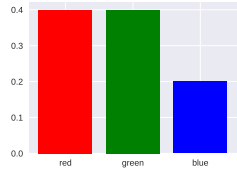
What if there is no ordering?

Measuring the Central Tendency

Unimodal



Bimodal



Mode

- Value that occurs most frequently in the data
- Example: *blue*, red, *blue*, yellow, green, *blue*, red
- Unimodal, bimodal, trimodal, ...: There are 1, 2, 3, ... modes in the data (multi-modal in general), cf. mixture models
- There is no mode if each data value occurs only once
- Well suited for categorical (i.e., non-numerical) data

Measuring the Dispersion of Data

Variance

- Applicable to numerical data, scalable computation:

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right]$$

  - Calculation by two passes: numerically much more stable
  - Single pass: calculate sum of squares and square of sum in parallel
- Measures the spread around the mean
- It is zero if and only if all the values are equal
- Standard deviation: Square root of the variance
- Both the standard deviation and the variance are algebraic

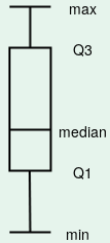
Boxplot Analysis

Five-number summary of a distribution

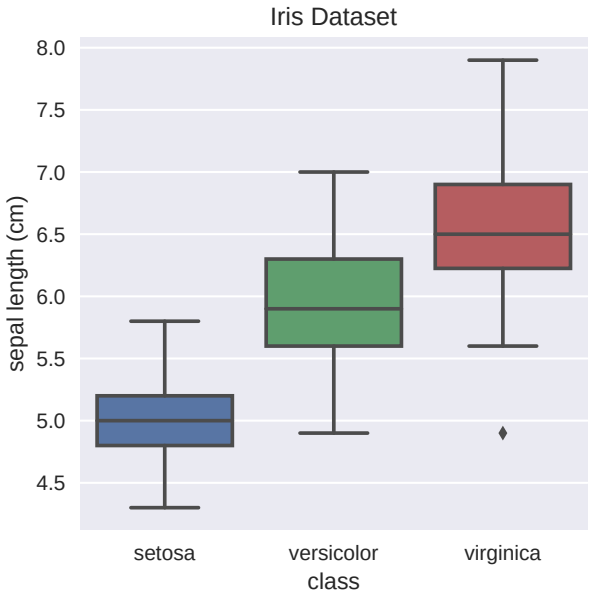
- Minimum, Q1, Median, Q3, Maximum
- Represents 0%, 25%, 50%, 75%, 100%-quantile of the data
- Also called "25-percentile", etc.

Boxplot

- Boundaries: first and third quartiles
- Height: inter-quartile range (IQR)
- The median is marked by a line within the box
- Whiskers: minimum and maximum
- Outliers: usually values more than $1.5 \cdot$ IQR below Q1 or above Q3



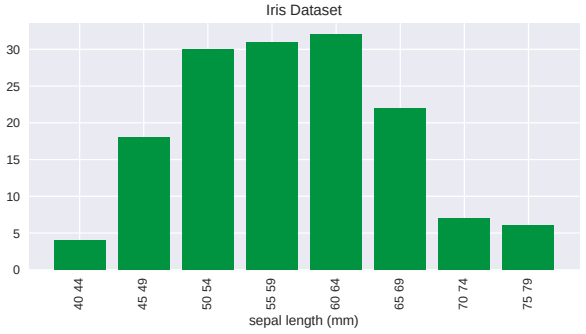
Boxplot Example



Data Generalization

- Which partitions of the data to aggregate?
- All data
  - Overall mean, overall variance: too coarse (overgeneralized)
- Different techniques to form groups for aggregation
  - Binning – histograms, based on value ranges
  - Generalization – abstraction based on generalization hierarchies
  - Clustering (see later) – based on object similarity

Binning Techniques: Histograms



- Histograms use binning to approximate data distributions
- Divide data into bins and store a representative (sum, average, median) for each bin
- Popular data reduction and analysis method
- Related to quantization problems

Equi-width Histograms

- Divide the range into N intervals of equal size: uniform grid
- If A and B are the lowest and highest values of the attribute, the width of intervals will be $(B - A)/N$

Positive

- Most straightforward

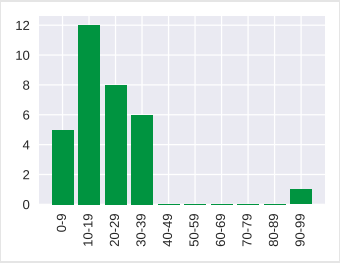
Negative

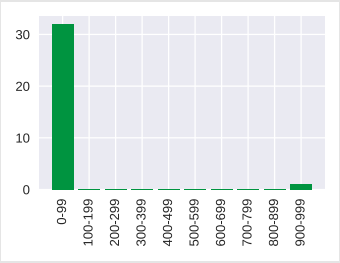
- Outliers may dominate presentation
- Skewed data is not handled well

Equi-width Histograms

Example

- Sorted data, 10 bins: 5, 7, 8, 8, 9, 11, 13, 13, 14, 14, 14, 15, 17, 17, 17, 18, 19, 23, 24, 25, 26, 26, 26, 27, 28, 32, 34, 36, 37, 38, 39, 97
- Insert 999





Equi-height Histograms

Divide the range into N intervals, each containing approx. the same number of samples (*quantile-based approach*)

Positive

- Good data scaling

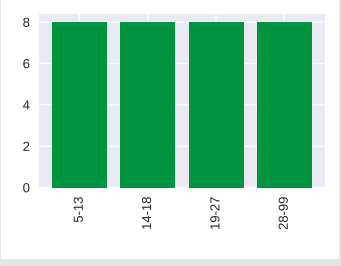
Negative

- If any value occurs often, the equal frequency criterion might not be met (intervals have to be disjoint!)

Equi-height Histograms

Example

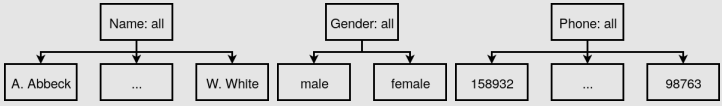
- Same data, 4 bins: 5, 7, 8, 8, 9, 11, 13, 13, 14, 14, 14, 15, 17, 17, 17, 18, 19, 23, 24, 25, 26, 26, 26, 27, 28, 32, 34, 36, 37, 38, 39, 97



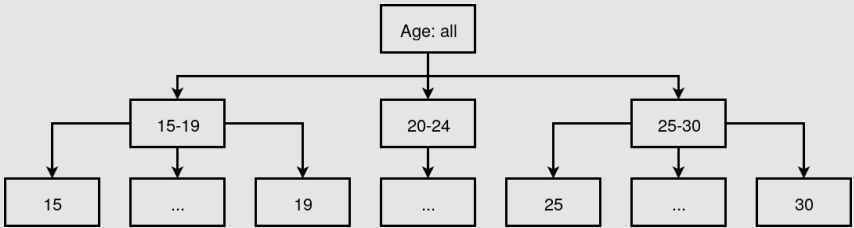
- Median = 50%-quantile
  - More robust against outliers (cf. value 999 from above)
  - Four bin example is strongly related to boxplot

Concept Hierarchies: Examples

No (real) hierarchies

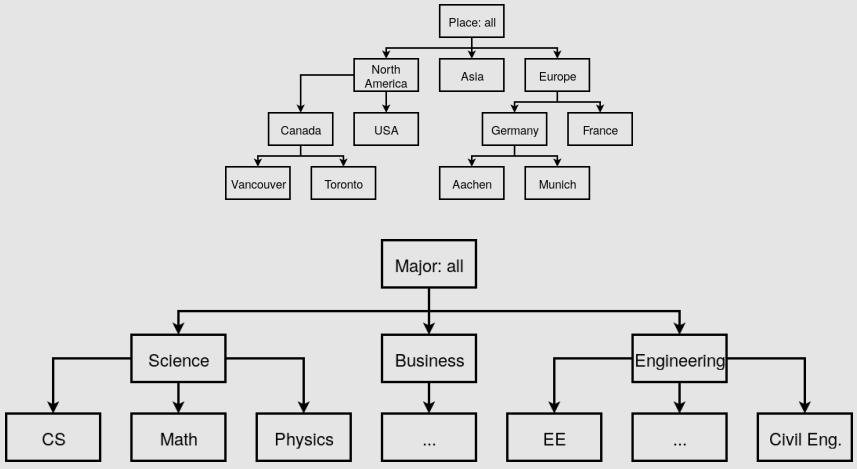


Set grouping hierarchies



Concept Hierarchies: Examples

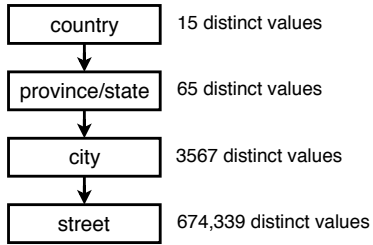
Schema hierarchies



Concept Hierarchy for Categorical Data

- Concept hierarchies can be specified by experts or just by users

- Heuristically generate a hierarchy for a set of (related) attributes
  - based on the number of distinct values per attribute in the attribute set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy

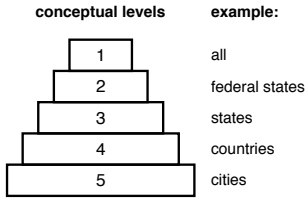


- Fails for counter examples: 20 distinct years, 12 months, 7 days_of_week, but not "year $<$ month $<$ days_of_week" with the latter on top

Summarization-based Aggregation

Data Generalization

A process which abstracts a large set of task-relevant data in a database from low conceptual levels to higher ones.



- Approaches:
  - Data-cube approach (OLAP / Roll-up) – manual
  - Attribute-oriented induction (AOI) – automated

Basic OLAP Operations

Roll up

Summarize data by climbing up hierarchy or by dimension reduction.

Drill down

Reverse of roll-up. From higher level summary to lower level summary or detailed data, or introducing new dimensions.

Slice and dice

Selection on one (slice) or more (dice) dimensions.

Pivot (rotate)

Reorient the cube, visualization, 3D to series of 2D planes.

Example: Roll up / Drill down

Query

```
SELECT *
FROM business
GROUP BY country, quarter
```

Roll-Up

```
SELECT *
FROM business
GROUP BY continent, quarter
```

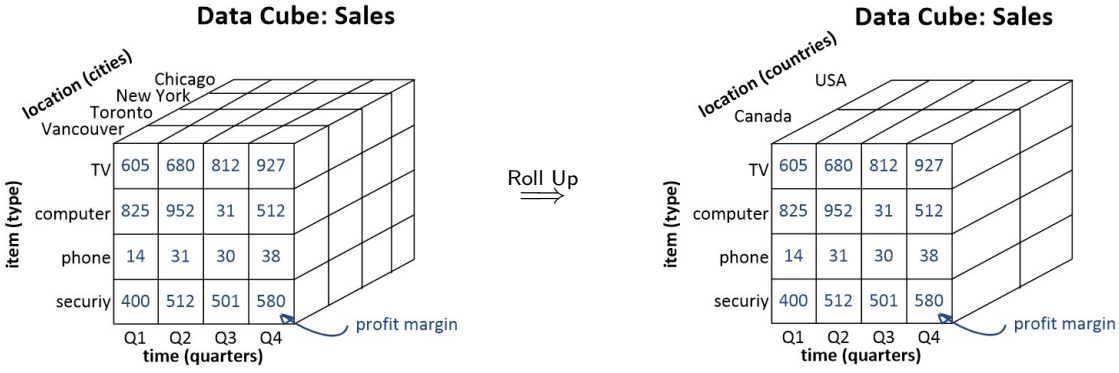
```
SELECT *
FROM business
GROUP BY country
```

Drill-Down

```
SELECT *
FROM business
GROUP BY city, quarter
```

```
SELECT *
FROM business
GROUP BY country, quarter, product
```

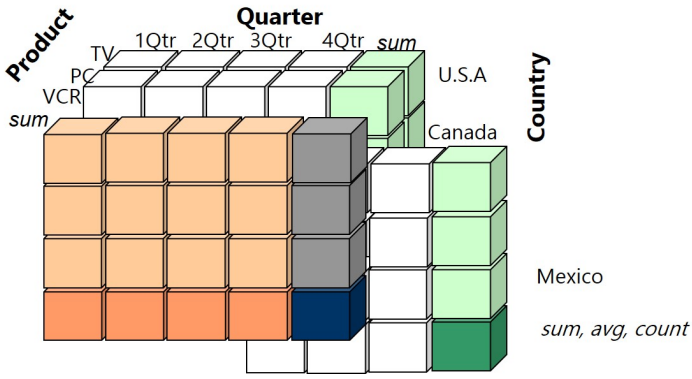
Example: Roll up in a Data Cube



Example: Slice Operation

```
SELECT income
FROM   time t, product p, country c
WHERE  p.name = 'VCR'
```

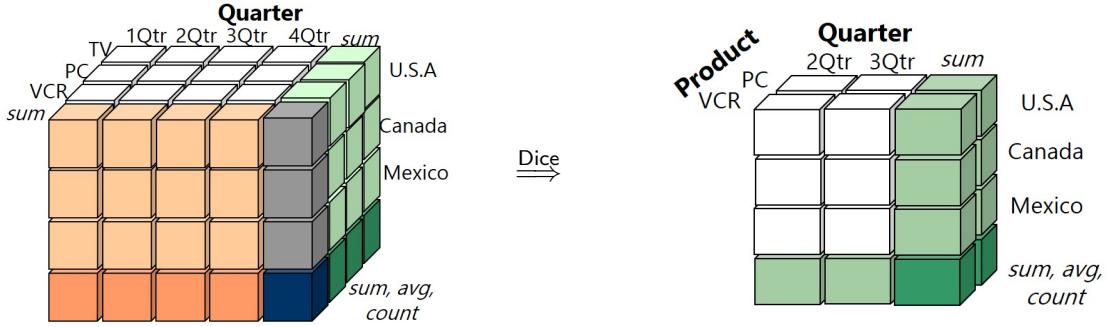
VCR dimension is chosen



Example: Dice Operation

```
SELECT income
FROM   time t, product p, country c
WHERE  p.name = 'VCR' OR p.name = 'PC' AND t.quarter BETWEEN 2 AND 3
```

sub-data cube over PC, VCR and quarters 2 and 3 is extracted



Example: Pivot (rotate)

year	17			18			19		
product	TV	PC	VCR	TV	PC	VCR	TV	PC	VCR
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

↓ Pivot (rotate) ↓

product	TV			PC			VCR		
year	17	18	19	17	18	19	17	18	19
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

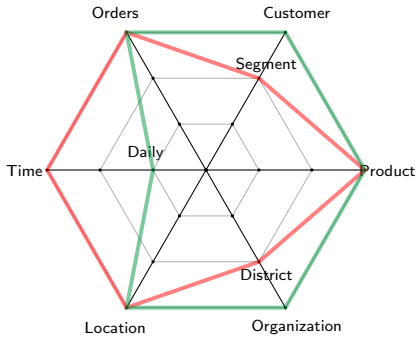
Basic OLAP Operations

Other operations

- *Drill across:* involving (across) more than one fact table
- *Drill through:* through the bottom level of the cube to its back-end relational tables (using SQL)

Specifying Generalization by a Star-Net

- Each circle is called a *footprint*
- Footprints represent the granularities available for OLAP operations



Discussion of OLAP-based Generalization

- Strength
  - Efficient implementation of data generalization
  - Computation of various kinds of measures, e.g., count, sum, average, max
  - Generalization (and specialization) can be performed on a data cube by roll-up (and drill-down)
- Limitations
  - Handles only dimensions of simple non-numeric data and measures of simple aggregated numeric values
  - Lack of intelligent analysis, can't tell which dimensions should be used and what levels the generalization should reach

Attribute-Oriented Induction (AOI)

- Apply aggregation by merging identical, generalized tuples and accumulating their respective counts.
- *Data focusing:* task-relevant data, including dimensions, and the result is the *initial relation*
- *Generalization Plan:* Perform generalization by either *attribute removal* or *attribute generalization*

Attribute-Oriented Induction (AOI)

Attribute Removal

Remove attribute A if:

- there is a large set of distinct values for A but there is no generalization operator (concept hierarchy) on A, or
- A's higher level concepts are expressed in terms of other attributes (e.g. *street* is covered by *city*, *state*, *country*).

Attribute Generalization

If there is a large set of distinct values for A, and there exists a set of generalization operators (i.e., a concept hierarchy) on A, then select an operator and generalize A.

Attribute Oriented Induction: Example

Name	Gender	Major	Birth place	Birth data	Residence	Phone	GPA
Jim Woodman	M	CS	Vancouver, BC, Canada	8-12-81	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-80	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-75	125 Austin Ave., Burnaby	420-5232	3.83
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Name: large number of distinct values, no hierarchy – removed
- Gender: only two distinct values – retained
- Major: many values, hierarchy exists – generalized to Sci., Eng., Biz.
- Birth_place: many values, hierarchy – generalized, e.g., to country
- Birth_date: many values – generalized to age (or age_range)
- Residence: many streets and numbers – generalized to city
- Phone number: many values, no hierarchy – removed
- Grade_point_avg (GPA): hierarchy exists – generalized to good, ...
- Count: additional attribute to aggregate base tuples

Attribute Oriented Induction: Example

- Initial Relation:

Name	Gender	Major	Birth place	Birth data	Residence	Phone	GPA
Jim Woodman	M	CS	Vancouver, BC, Canada	8-12-81	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-80	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-75	125 Austin Ave., Burnaby	420-5232	3.83
...	...	...	...	...	...	...	...

- Prime Generalized Relation:

Gender	Major	Birth region	Age Range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
...	...	...	...	...	...	...

- Crosstab for generalized relation:

	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

Attribute Generalization Control

- Problem: How many distinct values for an attribute?
  - *Overgeneralization:* values are too high-level
  - *Undergeneralization:* level not sufficiently high
  - Both yield tuples of poor usefulness
- Two common approaches
  - *Attribute-threshold control:* default or user-specified, typically 2-8 values
  - *Generalized relation threshold control:* control the size of the final relation/rule, e.g., 10-30

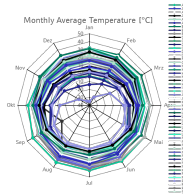
Next Attribute Selection Strategies for Generalization

- Aiming at *minimal degree of generalization*
  - Choose attribute that reduces the number of tuples the most
  - Useful heuristic: choose attribute with highest number of distinct values.
- Aiming at *similar degree of generalization* for all attributes
  - Choose the attribute currently having the least degree of generalization
- *User-controlled*
  - Domain experts may specify appropriate priorities for the selection of attributes

Agenda

Data Visualization

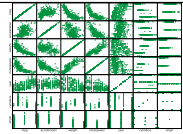
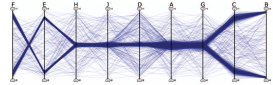
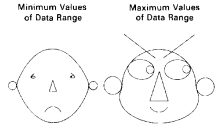
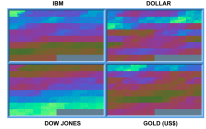
- Patterns in large data sets are hardly perceived from tabular numerical representations
- Data visualization transforms data in visually perceivable representations ("a picture is worth a thousand words")
- Combine capabilities:
  - Computers are good in number crunching (and data visualization by means of computer graphics)
  - Humans are good in visual pattern recognition



Monthly average temperature [°C]

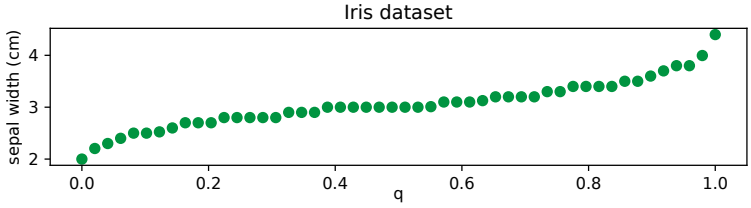
Städte Ø	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
Abu Dhabi	25	27	31	36	40	41	42	43	42	37	31	27
Acapulco	32	31	32	32	33	33	33	33	33	33	32	32
Anchorage	-4	-2	0	6	13	17	18	17	13	5	-3	-5
Antalya	15	16	19	22	27	32	35	36	32	27	21	17
Athen	13	14	17	20	26	30	34	34	29	24	18	14
Atlanta	11	13	18	23	26	30	31	31	28	23	17	12
Bangkok	32	33	35	36	35	34	33	33	33	32	32	32
Bogota	20	19	19	19	19	18	18	18	19	19	19	20
Buenos Aires	30	28	26	23	19	16	15	17	19	21	26	29
Caracas	30	28	30	30	31	32	32	32	33	32	31	30
Casablanca	18	18	20	21	22	25	26	27	26	24	21	19
Chicago	0	1	9	16	21	26	29	28	24	17	9	2
Colombo (Sri Lanka)	31	31	32	32	32	31	31	31	31	31	31	31
Dallas	13	16	21	25	29	33	36	36	32	26	19	14
Denver	7	8	14	14	21	28	32	30	25	18	12	6
Faro (Algarve)	16	16	19	21	23	27	29	29	26	23	19	17
Grand Canyon (Arizona)	6	8	13	15	21	27	29	27	25	18	12	6
Harare	27	26	27	26	24	21	22	24	28	29	28	27
Helsinki	-3	-3	2	9	15	20	23	21	17	9	3	0
Heraklion (Kreta)	15	16	18	20	24	27	30	30	27	24	20	17
Hongkong	19	20	23	26	30	32	33	33	32	30	25	21
Honolulu	26	26	27	27	28	30	30	31	30	30	28	27
Houston	16	19	23	27	30	33	34	35	32	28	21	17
Irkutsk	-14	-9	1	9	16	23	24	21	16	7	-4	-13
Istanbul	9	9	13	17	23	27	30	30	26	20	15	11
Jakutsk (Nordostsibirien)	-35	-28	-10	3	14	23	26	21	11	-3	-25	-34
Johannesburg	25	25	24	22	20	17	17	20	24	25	25	25
Kairo	19	20	24	27	32	35	35	35	34	30	25	20
Kapstadt	27	27	26	24	21	18	18	18	19	22	24	26
Kathmandu	18	21	25	28	28	29	28	28	28	26	23	20
Larnaka (Zypern)	17	18	20	23	26	31	33	34	31	28	23	19
Las Palmas	21	20	22	23	24	25	27	28	28	27	24	22
Las Vegas	15	16	23	26	31	38	40	39	35	27	20	14
Lhasa	9	10	13	17	21	24	23	22	21	17	13	10
Lima	26	26	27	24	21	20	19	18	19	20	22	24
Lissabon	14	15	18	20	23	27	28	29	27	22	17	15

Data Visualization Techniques

Type	Idea	Examples
Geometric	Visualization of geometric transformations and projections of the data	 Scatterplots  Parallel Coordinates
Icon-Based	Visualization of data as icons	 Chernoff Faces  Stick Figures
Pixel-oriented	Visualize each attribute value of each data object by one coloured pixel	 Recursive Patterns
Other		Hierarchical Techniques, Graph-based Techniques, Hybrid-Techniques, ...

Slide credit: Keim, Visual Techniques for Exploring Databases, Tutorial Slides, KDD 1997.

Quantile Plot



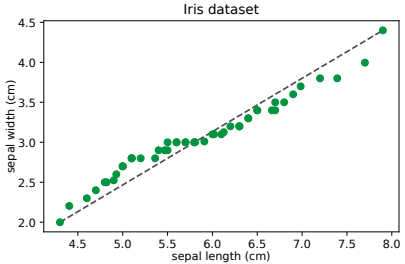
Characteristic

The p-quantile x_p is the value for which the fraction p of all data is less than or equal to x_p.

Benefit

Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)

Quantile-Quantile (Q-Q) Plot



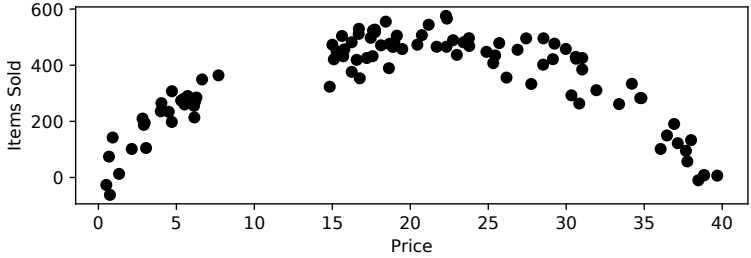
Characteristic

Graphs the quantiles of one univariate distribution against the corresponding quantiles of another.

Benefit

Allows the user to compare to distributions against each other.

Scatter Plot



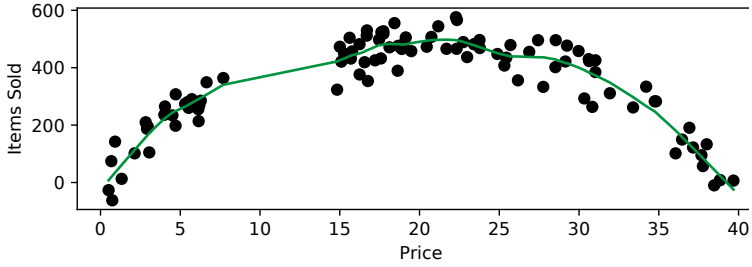
Characteristic

Each pair of values is treated as a pair of coordinates and plotted as points in the plane.

Benefit

Provides a first look at bivariate data to see clusters of points, outliers, etc.

Loess Curve



Characteristic

Loess curve is fitted by setting two parameters: a smoothing parameter, and the degree of the polynomials that are fitted by the regression.

Benefit

Adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence.

Scatterplot Matrix

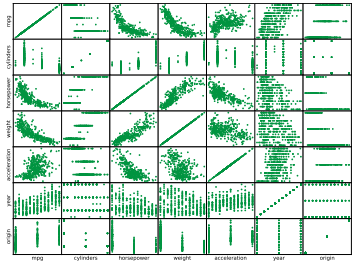
Characteristic

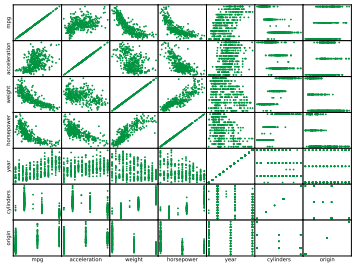
Matrix of scatterplots for pairs of dimensions

Ordering

Ordering of dimensions is important:

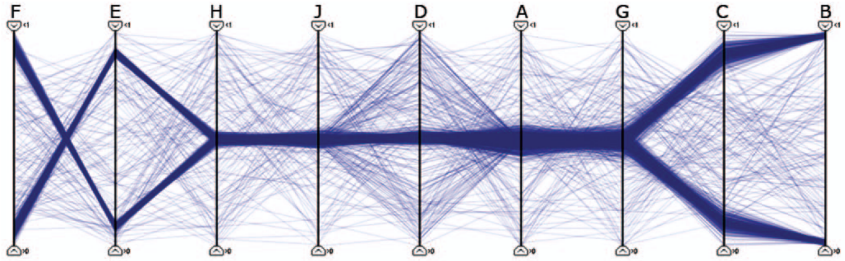
- Reordering improves understanding of structures and reduces clutter
- Interestingness of orderings can be evaluated with quality metrics (e.g. Peng et al.)





Clutter Reduction in Multi-Dimensional Data Visualizazion Using Dimension Reordering, IEEE Symp. on Inf. Vis., 2004.

Parallel Coordinates



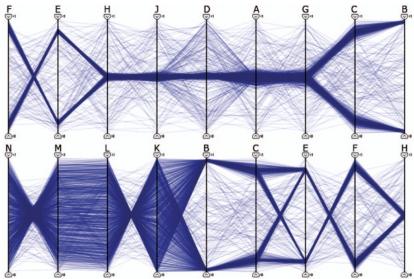
Characteristics

- d-dimensional data space is visualised by d parallel axes
- Each axis is scaled to min-max range
- Object = polygonal line intersecting axis at value in this dimension

Parallel Coordinates

Ordering

- Again, the ordering of the dimensions is important
- Quality metric for interestingness of ordering
- Quality or interestingness of orderings depends on what you want to visualize



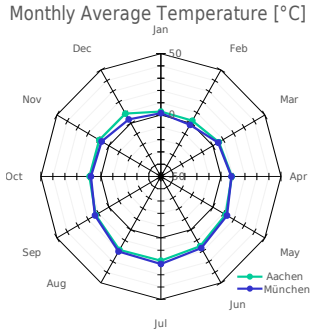
- Visualize clusters
- Visualize correlations between dimensions

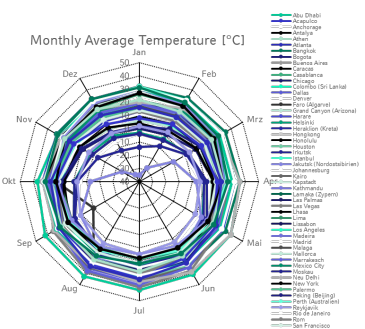
Bertini et al., Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization, Trans. on Vis. and Comp. Graph., 2011.

Spiderweb Model

Characteristics

- Illustrate any single object by a polygonal line
- Contract origins of all axes to a global origin point
- Works well for few objects only





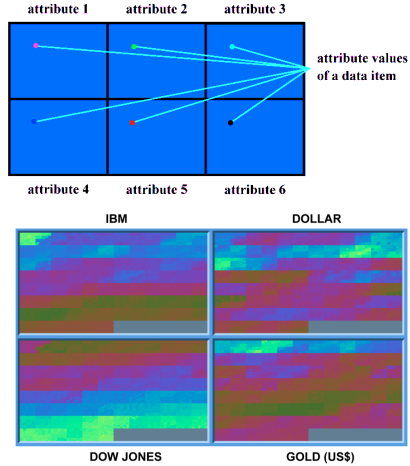
Pixel-Oriented Techniques

Characteristics

- Each data value is mapped onto a colored pixel
- Each dimension is shown in a separate window

How to arrange the pixel ordering?

One strategy: Recursive Patterns iterated line and column-based arrangements



Figures from Keim, Visual Techniques for Exploring Databases, Tutorial Slides, KDD 1997.

Chernoff Faces

Characteristics

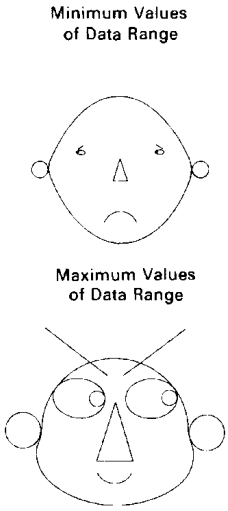
Map d-dimensional space to facial expression, e.g. length of nose = dim 6; curvature of mouth = dim 8

Advantage

Humans can evaluate similarity between faces much more intuitively than between high-dimensional vectors

Disadvantages

- Without dimensionality reduction only applicable to data spaces with up to 18 dimensions
- Which dimension represents what part?



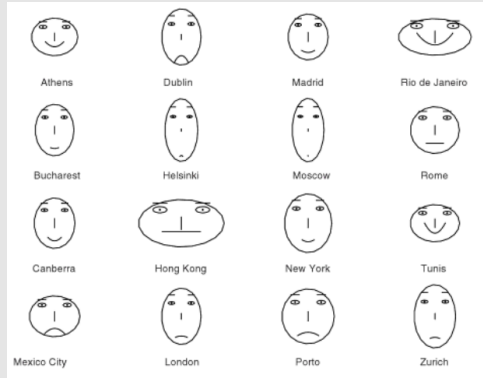
Figures taken from Mazza, Introduction to Information Visualization, Springer, 2009.

Chernoff Faces

Example: Weather Data

City	Precip. average	Temp. average	Temp. max average	Temp. min average	Record max	Record min
Athens	37	17	21	13	42	−3
Bucharest	58	11	16	5	49	−23
Canberra	62	12	19	6	42	−10
Dublin	74	10	12	6	28	−7
Helsinki	63	5	8	1	31	−36
Hong Kong	218	23	25	21	37	2
London	75	10	13	5	35	−13
Madrid	45	13	20	7	40	−10
Mexico City	63	17	23	11	32	−3
Moscow	59	4	8	1	35	−42
New York	118	12	17	8	40	−18
Porto	126	14	18	10	34	−2
Rio de Janeiro	109	25	30	20	43	7
Rome	80	15	20	11	37	−7
Tunis	44	18	23	13	46	−1
Zurich	107	9	12	6	35	−20

Table 4.1 Annual climatic values in Celsius of some world cities. Values from http://www.weatherbase.com.



Figures from Riccardo Mazza, Introduction to Information Visualization, Springer, 2009.

Chernoff Faces

Example: Finance Data

FIGURE 3
Facial Representation of Financial Performance (1 to 5 Years Prior to Failure)

	FEDERAL				
Date / Dimensions	Year to Failure 5	4	3	2	1
1. Return on Assets	0.10	0.11	0.06	0.03	−0.16
2. Debt Service	3.66	3.79	1.55	0.78	−14.11
3. Cash Flows	1.53	1.48	1.39	1.35	0.94
4. Capitalization	0.22	0.20	0.18	0.16	−0.02
5. Current Ratio	71.40	89.10	97.85	96.80	58.21
6. Cash Turnover	24.03	25.92	25.62	27.40	71.26
7. Receivables Turnover	5.25	4.46	4.26	4.36	9.56
8. Inventory Turnover	5.38	4.77	4.57	4.44	5.34
9. Sales per Dollar Working Capital	6.74	6.33	7.02	7.61	−45.77
10. Retained Earning/ Total Assets	0.32	0.30	0.01	−0.01	−0.26
11. Total Assets	0.94	.76	0.39	0.45	0.43

Figure from Huff et al., Facial Representation of Multivariate Data, Journal of Marketing, Vol. 45, 1981, pp. 53-59.

Agenda

Data Privacy

Situation

- Huge volume of data is collected
- From a variety of devices and platforms (e.g. Smartphones, Wearables, Social Networks, Medical systems)
- Capturing human behaviors, locations, routines, activities and affiliations
- Providing an opportunity to perform data analytics

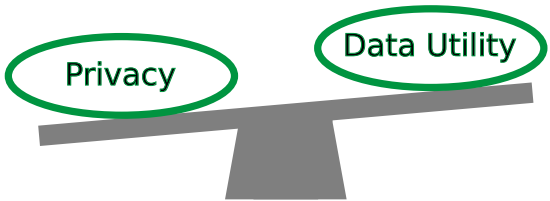
Data Abuse is inevitable

- It compromises individual's privacy
- Or breaches the security of an institution

Data Privacy

- These privacy concerns need to be mitigated
- They have prompted huge research interest to *protect data*
- But,

$$\text{Strong Privacy Protection} \implies \text{Poor Data Utility}$$
$$\text{Good Data Utility} \implies \text{Weak Privacy Protection}$$



Challenge

Find a good trade-off between Data Utility and Privacy

Data Privacy

Objectives of Privacy Preserving Data Mining

- Ensure data privacy
- Maintain a good trade-off between data utility and privacy

Paradigms

- k-Anonymity
- l-Diversity
- Differential Privacy

Linkage Attack

Method

Different public records can be linked to it to breach privacy

Hospital Records

Private	*Public*			
Name	Sex	Age	Zip	Disease
Alice	F	29	52062	Breast Cancer
Janes	F	27	52064	Breast Cancer
Jones	M	21	52066	Lung Cancer
Frank	M	35	52072	Heart Disease
Ben	M	33	52078	Fever
Betty	F	37	52080	Nose Pains

Public Records from Sport Club

Public				
Name	Sex	Age	Zip	Sport
Alice	F	29	52062	Tennis
Theo	M	41	52066	Golf
John	M	24	52062	Soccer
Betty	F	37	52080	Tennis
James	M	34	82066	Soccer

k-Anonymity

k-Anonymity

Privacy paradigm for protecting data records before *publication*

Three kinds of attributes:

1. *Key Attributes*: Uniquely identifiable attributes (e.g., Name, Social Security Number, Telephone Number)
2. *Quasi-identifier*: Groups of attributes that can be combined with external data to uniquely re-identify an individual (e.g. (Date of Birth, Zip Code, Gender))
3. *Sensitive Attributes*: An attacker should not be able to combine these with the key attributes. (e.g. Disease, Salary, Habit, Location etc.)

k-Anonymity

Attention

Hiding key attributes alone does not guarantee privacy.

An attacker may be able to break the privacy by combining the quasi-identifiers from the data with those from publicly available information.

Definition: k-Anonymity

Given a set of quasi-identifiers in a database table, the database table is said to be *k-Anonymous*, if the sequence of records in each quasi-identifier exists at least k times.

Ensure privacy by *Suppression* or *Generalization* of quasi-identifiers.

k-Anonymity: Suppression

Suppression

Accomplished by replacing a part or the entire attribute value by placeholder, e.g. "?" (= generalization)

Example

- Suppress Postal Code: 52062 $\mapsto$ 52???
- Suppress Gender: Male $\mapsto$?; Female $\mapsto$?

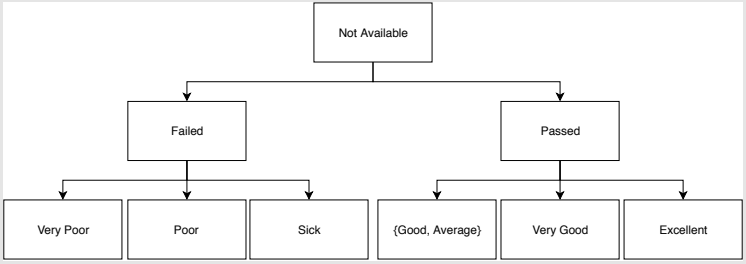
k-Anonymity: Generalization

Generalization

Accomplished by aggregating values from fine levels to coarser resolution using generalisation hierarchy.

Example

Generalize exam grades:



Shortcomings: Background Knowledge Attack

Background Knowledge Attack

Lack of diversity of the sensitive attribute values (homogeneity)

Example

- *Background Knowledge*: Alice is (i) 29 years old and (ii) female
- *Homogeneity*: All 2*-aged females have Breast Cancer.
  $\Longrightarrow$ Alice has BC!

Release

Quasi Identifier			*Sensitive*
Sex	Age	Zip	Disease
F	2?	520??	Breast Cancer
F	2?	520??	Breast Cancer
M	2?	520??	Lung Cancer
M	3?	520??	Heart Disease
M	3?	520??	Fever
F	3?	520??	Nose Pains

This led to the creation of a new privacy model called *l*-diversity

l-Diversity

Distinct l-Diversity

An quasi-identifier is l-diverse, if there are at least l different values. A dataset is l-diverse, if all QIs are l-diverse.

Example

Not "diverse"

Quasi Identifier	Sensitive
QI 1	Headache
QI 1	Headache
QI 1	Headache
QI 2	Cancer
QI 2	Cancer

2-diverse

Quasi Identifier	Sensitive
QI 1	Headache
QI 1	Cancer
QI 1	Headache
QI 2	Headache
QI 2	Cancer

l-Diversity

Other Variants

- *Entropy l-Diversity*: For each equivalent class, the entropy of the distribution of its sensitive values must be at least l
- *Probabilistic l-Diversity*: The most frequent sensitive value of an equivalent class must be at most $1/l$

Limitations

- Not necessary at times
- Difficult to achieve: For large record size, many equivalent classes will be needed to satisfy l-Diversity
- Does not consider the distribution of sensitive attributes

Background Attack Assumption

- k-Anonymity and l-Diversity make assumptions about the adversary
- They at times fall short of their goal to prevent data disclosure
- There is another privacy paradigm which does not rely on background knowledge, called *Differential Privacy*

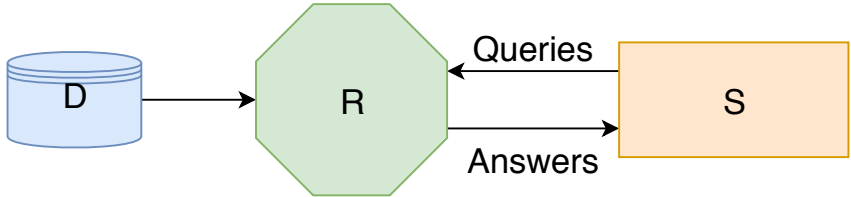
Differential Privacy

Core Idea

Privacy through data perturbation.

- The addition or removal of one record from a database should not reveal any information to an adversary, i.e. your *presence* or *absence* does not reveal or leak any information.
- Use a randomization mechanism to perturb query results of `count`, `sum`, `mean` functions, as well as other statistical query functions.

Differential Privacy



Definition

A randomized mechanism $R(x)$ provides ϵ-*differential privacy* if for any two databases D_1 and D_2 that differ on at most one element, and all outputs $S \subseteq Range(R)$

$$\frac{Pr[R(D_1) \in S]}{Pr[R(D_2) \in S]} \leq \exp(\epsilon)$$

ϵ is a parameter called *privacy budget/level*.

Data Perturbation

Data perturbation is achieved by noise addition.

Some Kinds of Noise

- Laplace noise
- Gaussian noise
- Exponential Mechanism

Agenda

What is Frequent Pattern Mining?

Setting: Transaction Databases

A database of transactions, where each transaction comprises a set of items, e.g. one transaction is the basket of one customer in a grocery store.

Frequent Pattern Mining

Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

Applications

Basket data analysis, cross-marketing, catalogue design, loss-leader analysis, clustering, classification, recommendation systems, etc.

What is Frequent Pattern Mining?

Task 1: Frequent Itemset Mining

Find all subsets of items that occur together in many transactions.

Example

Which items are bought together frequently?

$$
\begin{aligned}
D = \{ \quad & \{ \quad butter, bread, milk, sugar\}, \\
& \{ \quad butter, flour, milk, sugar\}, \\
& \{ \quad butter, eggs, milk, salt\}, \\
& \{ \quad eggs\}, \\
& \{ \quad butter, flour, milk, salt, sugar\}\}
\end{aligned}
$$

$\rightsquigarrow$ 80% of transactions contain the itemset {milk, butter}

What is Frequent Pattern Mining?

Task 2: Association Rule Mining

Find all rules that correlate the presence of one set of items with that of another set of items in the transaction database.

Example

98% of people buying tires and auto accessories also get automotive service done

Agenda

Mining Frequent Itemsets: Basic Notions

- **Items** $I = \{i_1, \ldots, i_m\}$: a set of literals (denoting items)
- **Itemset** X: Set of items $X \subseteq I$
- **Database** D: Set of *transactions* T, each transaction is a set of items $T \subseteq I$
- Transaction T contains an itemset X: $X \subseteq T$
- **Length** of an itemset X equals its cardinality $|X|$
- k-**itemset**: itemset of length k
- (Relative) **Support** of an itemset: $supp(X) = |\{T \in D \mid X \subseteq T\}|/|D|$
- X is **frequent** if $supp(X) \geq minSup$ for threshold $minSup$.

Goal

Given a database D and a threshold $minSup$, find all frequent itemsets $X \in Pot(I)$.

Mining Frequent Itemsets: Basic Idea

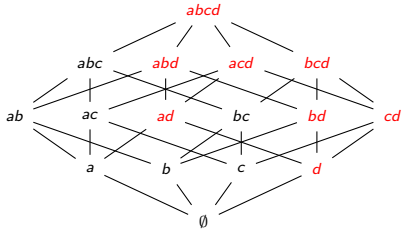
Naïve Algorithm

Count the frequency of all possible subsets of I in the database D.

Problem

Too expensive since there are 2^m such itemsets for m items (for $|I| = m$, $2^m =$ cardinality of the powerset of I).

Mining Frequent Patterns: Apriori Principle



- frequent
- non-frequent

Apriori Principle (anti-monotonicity)

- Any non-empty subset of a frequent itemset is frequent, too!

$$A \subseteq I : supp(A) \geq minSup \implies \forall \emptyset \neq A' \subset A : supp(A') \geq minSup$$

- Any superset of a non-frequent itemset is non-frequent, too!

$$A \subseteq I : supp(A) < minSup \implies \forall A' \supset A : supp(A') < minSup$$

Apriori Algorithm

Idea

- First count the 1-itemsets, then the 2-itemsets, then the 3-itemsets, and so on
- When counting $(k+1)$-itemsets, only consider those $(k+1)$-itemsets where all subsets of length k have been determined as frequent in the previous step

Apriori Algorithm

variable C_k: candidate itemsets of size k
variable L_k: frequent itemsets of size k
$L_1 = \{\text{frequent items}\}$
for $(k = 1;\ L_k \neq \emptyset;\ k{+}{+})$ **do**

Produce candidates.

join L_k with itself to produce C_{k+1} ▷ JOIN STEP
discard $(k+1)$-itemsets from C_{k+1} that ... ▷ PRUNE STEP
... contain non-frequent k-itemsets as subsets

C_{k+1} = candidates generated from L_k

Prove candidates.

for each transaction $T \in D$ **do**
Increment the count of all candidates in C_{k+1} ...
... that are contained in T

L_{k+1} = candidates in C_{k+1} with *minSupp*
return $\bigcup_k L_k$

Apriori Algorithm: Generating Candidates – Join Step

Requirements for Candidate $(k+1)$-itemsets

- *Completeness*: Must contain all frequent $(k+1)$-itemsets (superset property $C_{k+1} \supseteq L_{k+1}$)
- *Selectiveness*: Significantly smaller than the set of all $(k+1)$-subsets

Suppose the itemsets are sorted by any order (e.g. lexicographic)

Step 1: Joining ($C_{k+1} = L_k \bowtie L_k$)

- Consider frequent k-itemsets p and q
- p and q are joined if they share the same first $(k-1)$ items.

Apriori Algorithm: Generating Candidates – Join Step

Example

- $k = 3$ ($\implies k + 1 = 4$)
- $p = (a, c, f) \in L_k$
- $q = (a, c, g) \in L_k$
- $r = (a, c, f, g) \in C_{k+1}$

SQL example

insert into C_{k+1}

select $p.i_1, p.i_2, \ldots, p.i_k, q.i_k$

from $L_k : p, L_k : q$

where $p.i_1 = q.i_1, \ldots, p.i_{k-1} = q.i_{k-1}, p.i_k < q.i_k$

Apriori Algorithm: Generating Candidates – Prune Step

Step 2: Pruning ($L_{k+1} = \{X \in C_{k+1} \mid supp(X) \geq minSup\}$)

- *Naïve*: Check support of every itemset in C_{k+1} $\rightsquigarrow$ inefficient for huge C_{k+1}
- *Better*: Apply Apriori principle first: Remove candidate $(k+1)$-itemsets which contain a non-frequent k-subset s, i.e., $s \notin L_k$

Pseudocode

for all $c \in C_{k+1}$ **do**
  for all k-subsets s of c **do**
    if $s \notin L_k$ **then**
      Delete c from C_{k+1}

Apriori Algorithm: Generating Candidates – Prune Step

Example

- $L_3 = \{acf, acg, afg, afh, cfg\}$
- Candidates after join step: $\{acfg, afgh\}$
- In the pruning step: delete $afgh$ because $fgh \notin L_3$, i.e. fgh is not a frequent 3-itemset (also $agh \notin L_3$)
- $C_4 = \{acfg\} \rightsquigarrow$ check the support to generate L_4

Apriori Algorithm: Full example

Database

TID	items
0	acdf
1	bce
2	abce
3	aef

minSup = 0.5

Alphabetic Ordering

k	candidate	prune	count	threshold
1	a		3	a
	b		2	b
	c		3	c
	d		1	
	e		3	e
	f		2	f
2	ab		1	
	ac		2	ac
	ae		2	ae
	af		2	af
	bc		2	bc
	be		2	be
	bf		0	
	ce		2	ce
	cf		1	
	ef		1	
3	ace		1	
	acf	with cf		
	aef	with ef		
	bce		2	bce

Frequency-Ascending Ordering

k	candidate	prune	count	threshold
1	d		1	
	b		2	b
	f		2	f
	a		3	a
	c		3	c
	e		3	e
2	bf		0	
	ba		1	
	bc		2	bc
	be		2	be
	fa		2	fa
	fc		1	
	fe		1	
	ac		2	ac
	ae		2	ae
	ce		2	ce
3	bce		2	bce
	ace		1	

Counting Candidate Support

Motivation

Why is counting supports of candidates a problem?

- Huge number of candidates
- One transaction may contain many candidates

Solution

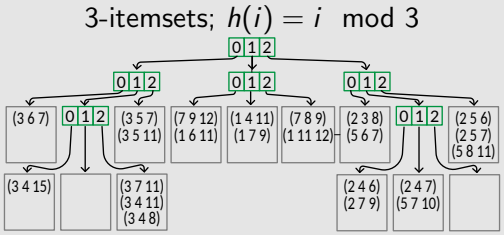
Store candidate itemsets in hash-tree

Counting Candidate Support: Hash Tree

Hash-Tree

- Leaves contain itemset lists with their support (e.g. counts)
- Interior nodes comprise hash tables
- *subset* function to find all candidates contained transaction

Example

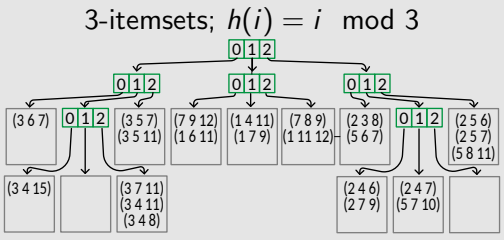


Hash-Tree: Construction

Search

- Start at the root (level 1)
- At level d: Apply hash function h to d-th item in the itemset

Example

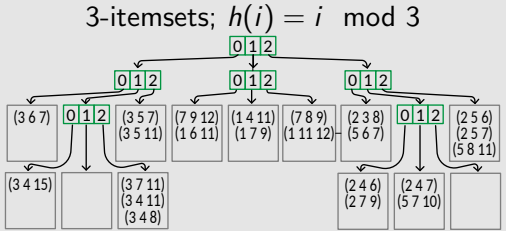


Hash-Tree: Construction

Insertion

- Search for the corresponding leaf node
- Insert the itemset into leaf; if an overflow occurs:
  - Transform the leaf node into an internal node
  - Distribute the entries to the new leaf nodes according to the hash function h

Example



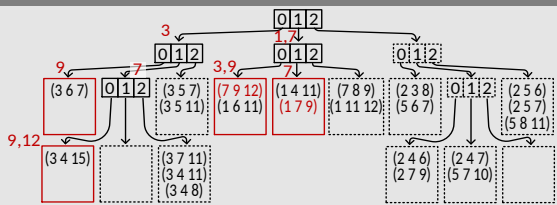
Hash-Tree: Counting

Search all candidates of length k in transaction $T = (t_1, \ldots, t_n)$

- At root:
  - Compute hash values for all items $t_1, \ldots, t_{n-k+1}$
  - Continue search in all resulting child nodes
- At internal node at level d (reached after hashing of item t_i):
  - Determine the hash values and continue the search for each item t_j with $i < j \leq n - k + d$
- At leaf node:
  - Check whether the itemsets in the leaf node are contained in transaction T

Example

3-itemsets;
$h(i) = i \mod 3$
Transaction:
$\{1, 3, 7, 9, 12\}$



Apriori – Performance Bottlenecks

Huge Candidate Sets

- 10^4 frequent 1-itemsets will generate 10^7 candidate 2-itemsets
- To discover a frequent pattern of size 100, one needs to generate $2^{100} \approx 10^{30}$ candidates.

Multiple Database Scans

- Needs n or $n + 1$ scans, where n is the length of the longest pattern

Is it possible to mine the complete set of frequent itemsets without candidate generation?

Mining Frequent Patterns *Without Candidate Generation*

Idea

- Compress large database into compact tree structure; complete for frequent pattern mining, but avoiding several costly database scans (called *FP-tree*)
- Divide compressed database into *conditional databases* associated with one frequent item

FP-Tree Construction

minSup=2/12

Database

TID	Items
1	c
2	cd
3	cef
4	cef
5	bcd
6	bcd
7	bcdg
8	bde
9	bd
10	bh
11	bi
12	b

2.1

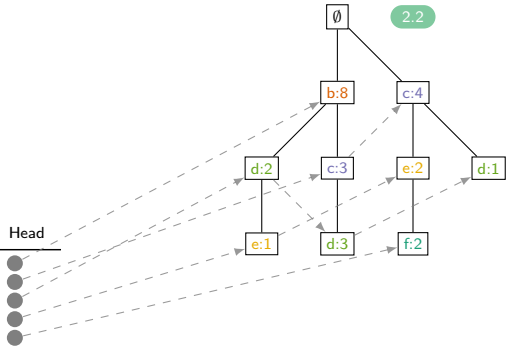
Freq. Item
c
cd
cef
cef
bcd
bcd
bcd
bde
bd
b
b
b

1

Header Table

Item	Frequency
b	8
c	7
d	6
e	3
f	2

Head



1. Scan DB once, find frequent 1-itemsets (single items); Order frequent items in frequency descending order
2. Scan DB again:
   - 2.1 Keep only freq. items; sort by descending freq.
   - 2.2 Does path with common prefix exist?
     Yes: Increment counter; append suffix;
     No: Create new branch

Benefits of the FP-Tree Structure

Completeness

- never breaks a long pattern of any transaction
- preserves complete information for frequent pattern mining

Compactness

- reduce irrelevant information – infrequent items are gone
- frequency descending ordering: more frequent items are more likely to be shared
- never be larger than the original database (if not count node-links and counts)
- Experiments demonstrate compression ratios over 100

Mining Frequent Patterns Using FP-Tree

General Idea: (Divide-and-Conquer)

Recursively grow frequent pattern path using the FP-tree

Method

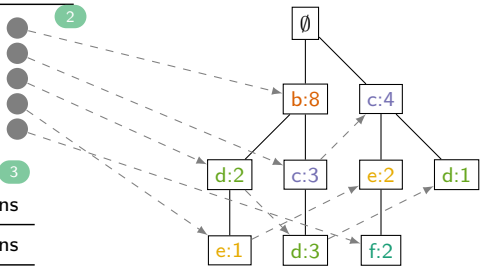
1. Construct conditional pattern base for each node in the FP-tree
2. Construct conditional FP-tree from each conditional pattern-base
3. Recursively mine conditional FP-trees and grow frequent patterns obtained so far;
   If the conditional FP-tree contains a single path, simply enumerate all the patterns

Major Steps to Mine FP-Tree: Conditional Pattern Base

Header Table 1

Item	Frequency	Head
b	8	
c	7	
d	6	
e	3	
f	2	

2



Conditional Pattern 3

Item	Cond. Patterns
b	∅
c	b:3, ∅
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

1. Start from header table
2. Visit all nodes for this item (following links)
3. Accumulate all transformed prefix paths to form conditional pattern base (the frequency can be read from the node).

Properties of FP-Tree for Conditional Pattern Bases

Node-Link Property

For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.

Prefix Path Property

To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P needs to be accumulated, and its frequency count should carry the same count as node a_i.

Major Steps to Mine FP-Tree: Conditional FP-Tree

Conditional Pattern

Item	Cond. Patterns
b	$\emptyset$
c	b:3, $\emptyset$
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2

Example: *e*-conditional FP-Tree

Item	Frequency
c	2
b	**1**
d	**1**

$\emptyset \mid e$

c:2

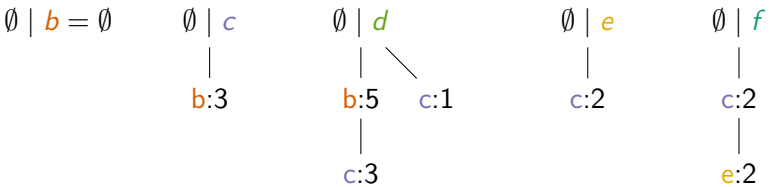
Construct conditional FP-tree from each conditional pattern-base

- The prefix paths of a suffix represent the conditional basis $\rightsquigarrow$ can be regarded as transactions of a database.
- For each pattern-base:
  - Accumulate the count for each item in the base
  - Re-sort items within sets by frequency
  - Construct the FP-tree for the frequent items of the pattern base

Major Steps to Mine FP-Tree: Conditional FP-Tree

- Build conditional FP-Trees for each item

Item	Cond. Patterns
b	$\emptyset$
c	b:3, $\emptyset$
d	bc:3, b:2, c:1
e	c:2, bd:1
f	ce:2



Major Steps to Mine FP-Tree: Recursion

Base Case: Single Path

If the conditional FP-tree contains a single path, simply enumerate all the patterns (enumerate all combinations of sub-paths)

Example

$\emptyset \mid f$

c:2

e:2

$\rightsquigarrow$

All frequent patterns concerning f:

f,

fc, fe

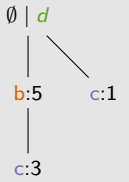
fce

Major Steps to Mine FP-Tree: Recursion

Recursive Case: Non-degenerated Tree

If the conditional FP-tree is not just a single path, create conditional pattern base for this smaller tree, and recurse.

Example



Conditional Pattern Base

Item	Cond. Patterns
b	∅
c	b:3, ∅

$\emptyset \mid db = \emptyset$

Principles of Frequent Pattern Growth

Pattern Growth Property

Let X be a frequent itemset in D, B be X's conditional pattern base, and Y be an itemset in B. Then $X \cup Y$ is a frequent itemset in D if and only if Y is frequent in B.

Example

"abcdef" is a frequent pattern, if and only if

- "abcde" is a frequent pattern, and
- "f" is frequent in the set of transactions containing "abcde"

Why Is Frequent Pattern Growth Fast?

Performance study[1] shows: FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection

Reasoning:

- No candidate generation, no candidate test (Apriori algorithm has to proceed breadth-first)
- Use compact data structure
- Eliminate repeated database scan
- Basic operation is counting and FP-tree building

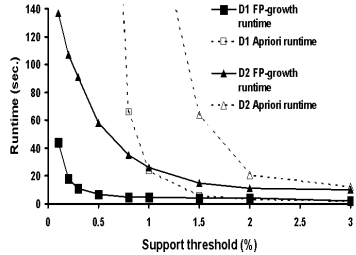


Image Source: [1]

[1] Han, Pei & Yin, *Mining frequent patterns without candidate generation*, SIGMOD'00

Maximal or Closed Frequent Itemsets

Challenge

Often, there is a huge number of frequent itemsets (especially if minSup is set too low), e.g. a frequent itemset of length 100 contains $2^{100} - 1$ many frequent subsets

Closed Frequent Itemset

Itemset X is *closed* in dataset D if for all $Y \supset X : supp(Y) < supp(X)$.

$\Rightarrow$ The set of closed frequent itemsets contains complete information regarding its corresponding frequent itemsets.

Maximal Frequent Itemset

Itemset X is *maximal* in dataset D if for all $Y \supset X : supp(Y) < minSup$.

$\Rightarrow$ The set of maximal itemsets does not contain the complete support information

$\Rightarrow$ More compact representation

Agenda

Simple Association Rules: Introduction

Example

Transaction database:

$$D = \{ \{ butter, bread, milk, sugar\}, \{ butter, flour, milk, sugar\}, \{ butter, eggs, milk, salt\}, \{ eggs\}, \{ butter, flour, milk, salt, sugar\}\}$$

Frequent itemsets:

items	support
{butter}	4
{milk}	4
{butter, milk}	4
{sugar}	3
{butter, sugar}	3
{milk, sugar}	3
{butter, milk, sugar}	3

Question of interest

- If milk and sugar are bought, will the customer always buy butter as well? milk, sugar $\Rightarrow$ butter?
- In this case, what would be the probability of buying butter?

Simple Association Rules: Basic Notions

Let *Items, Itemset, Database, Transaction, Transaction Length, k-itemset, (relative) Support, Frequent Itemset* be defined as before. Additionally:

- The items in transactions and itemsets are **sorted** lexicographically: itemset $X = (x_1, \ldots, x_k)$, where $x_1 \leq, \ldots, \leq x_k$
- **Association rule**: An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ are two itemsets with $X \cap Y = \emptyset$
- Note: simply enumerating all possible association rules is not reasonable!

What are the interesting association rules w.r.t. D?

Interestingness of Association Rules

Goal

Quantify the interestingness of an association rule with respect to a transaction database D.

Support

- Frequency (probability) of the entire rule with respect to D:

$$supp(X \Rightarrow Y) = P(X \cup Y) = \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|D|} = supp(X \cup Y)$$

- "Probability that a transaction in D contains the itemset."

Interestingness of Association Rules

Confidence

- Indicates the strength of implication in the rule:

$$\begin{aligned} conf(X \Rightarrow Y) = P(Y \mid X) &= \frac{|\{T \in D \mid X \subseteq T\} \cap \{T \in D \mid Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} \\ &= \frac{|\{T \in D \mid X \subseteq T \wedge Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} \\ &= \frac{|\{T \in D \mid X \cup Y \subseteq T\}|}{|\{T \in D \mid X \subseteq T\}|} = \frac{supp(X \cup Y)}{supp(X)} \end{aligned}$$

- "Conditional probability that a transaction in D containing the itemset X also contains itemset Y."

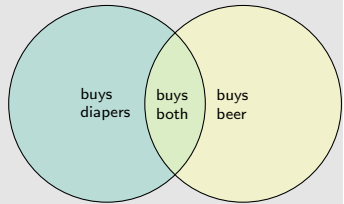
Interestingness of Association Rules

Rule form

"Body $\Rightarrow$ Head [support, confidence]"

Association rule examples

- buys diapers $\Rightarrow$ buys beer [0.5 %, 60%]
- major in CS $\wedge$ takes DB $\Rightarrow$ avg. grade A [1%, 75%]



Mining of Association Rules

Task of mining association rules

Given a database D, determine all association rules having a $supp \geq minSup$ and a $conf \geq minConf$ (so-called *strong association rules*).

Key steps of mining association rules

1. Find frequent itemsets, i.e., itemsets that have $supp \geq minSup$ (e.g. Apriori, FP-growth)
2. Use the frequent itemsets to generate association rules
   - For each itemset X and every nonempty subset $Y \subset X$ generate rule $Y \Rightarrow (X \setminus Y)$ if *minSup* and *minConf* are fulfilled
   - We have $2^{|X|} - 2$ many association rule candidates for each itemset X

Mining of Association Rules

Example

- Frequent itemsets:

1-itemset	count	2-itemset	count	3-itemset	count
{ a }	3	{ a,b }	3	{ a,b,c }	2
{ b }	4	{ a,c }	2		
{ c }	5	{ b,c }	4		

- Rule candidates
  - From 1-itemsets: None
  - From 2-itemsets: $a \Rightarrow b$; $b \Rightarrow a$; $a \Rightarrow c$; $c \Rightarrow a$; $b \Rightarrow c$; $c \Rightarrow b$
  - From 3-itemsets: $a, b \Rightarrow c$; $a, c \Rightarrow b$; $c, b \Rightarrow a$; $a \Rightarrow b, c$; $b \Rightarrow a, c$; $c \Rightarrow a, b$

Generating Rules from Frequent Itemsets

Rule generation

- For each frequent itemset X:
  - For each nonempty subset Y of X, form a rule $Y \Rightarrow (X \setminus Y)$
  - Delete those rules that do not have minimum confidence
- Note:
  - Support always exceeds *minSup*
  - The support values of the frequent itemsets suffice to calculate the confidence
- Exploit anti-monotonicity for generating candidates for strong association rules!
  - $Y \Rightarrow Z$ not strong $\implies$ for all $A \subseteq D : Y \Rightarrow Z \cup A$ not strong
  - $Y \Rightarrow Z$ not strong $\implies$ for all $Y' \subseteq Y$: $(Y \setminus Y') \Rightarrow (Z \cup Y')$ not strong

Generating Rules from Frequent Itemsets

Example: $minConf = 60\%$

$conf(a \Rightarrow b) = 3/3$ ✓
$conf(b \Rightarrow a) = 3/4$ ✓
$conf(a \Rightarrow c) = 2/3$ ✓
$conf(c \Rightarrow a) = 2/5$ ✗
$conf(b \Rightarrow c) = 4/4$ ✓
$conf(c \Rightarrow b) = 4/5$ ✓
$conf(b, c \Rightarrow a) = 1/2$ ✗
$conf(a, c \Rightarrow b) = 1$ ✓
$conf(a, b \Rightarrow c) = 2/3$ ✓
$conf(a \Rightarrow b, c) = 2/3$ ✓
$conf(b \Rightarrow a, c) = 2/4$ ✗ (pruned with $b, c \Rightarrow a$)
$conf(c \Rightarrow a, b) = 2/5$ ✗ (pruned with $b, c \Rightarrow a$)

itemset	count
{ a }	3
{ b }	4
{ c }	5
{ a,b }	3
{ a,c }	2
{ b,c }	4
{ a,b,c }	2

Interestingness Measurements

Objective measures

Two popular measures:

- Support
- Confidence

Subjective measures [Silberschatz & Tuzhilin, KDD95]

A rule (pattern) is interesting if it is

- *unexpected* (surprising to the user) and/or
- *actionable* (the user can do something with it)

Criticism to Support and Confidence

Example 1 [Aggarwal & Yu, PODS98]

- Among 5000 students
  - 3000 play basketball (=60%)
  - 3750 eat cereal (=75%)
  - 2000 both play basket ball and eat cereal (=40%)
- Rule "play basketball $\Rightarrow$ eat cereal [40%, 66.7%]" is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%
- Rule "play basketball $\Rightarrow$ not eat cereal [20%, 33.3%]" is far more accurate, although with lower support and confidence
- Observation: "play basketball" and "eat cereal" are negatively correlated

Not all strong association rules are interesting and some can be misleading.

- Augment the support and confidence values with interestingness measures such as the correlation: "A $\Rightarrow$ B [*supp*, *conf*, *corr*]"

Other Interestingness Measures: Correlation

Correlation

Correlation (sometimes called *Lift*) is a simple measure between two items A and B:

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)} = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \Rightarrow B)}{supp(B)}$$

- The two rules $A \Rightarrow B$ and $B \Rightarrow A$ have the same correlation coefficient
- Takes both $P(A)$ and $P(B)$ in consideration
- $corr_{A,B} > 1$: The two items A and B are positively correlated
- $corr_{A,B} = 1$: There is no correlation between the two items A and B
- $corr_{A,B} < 1$: The two items A and B are negatively correlated

Other Interestingness Measures: Correlation

Example 2

item	transactions							
X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

rule	support	confidence	correlation
$X \Rightarrow Y$	25%	50%	2
$X \Rightarrow Z$	37.5%	75%	0.86
$Y \Rightarrow Z$	12.5%	50%	0.57

- X and Y: positively correlated
- X and Z: negatively related
- Support and confidence of $X \Rightarrow Z$ dominates
- But: items X and Z are negatively correlated
- Items X and Y are positively correlated

Hierarchical Association Rules: Motivation

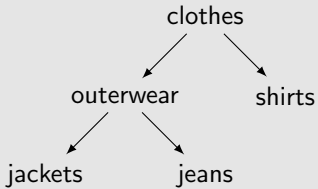
Problem

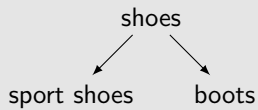
- High minSup: apriori finds only few rules
- Low minSup: apriori finds unmanagably many rules

Solution

Exploit item taxonomies (generalizations, is-a hierarchies) which exist in many applications

Example





Hierarchical Association Rules

New Task

Find all generalized association rules between generalized items, i.e. Body and Head of a rule may have items of any level of the hierarchy

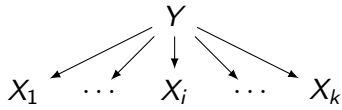
Generalized Association Rule

$X \Rightarrow Y$ with $X, Y \subset I, X \cap Y = \emptyset$ and no item in Y is an ancestor of any item in X

Example

- Jeans $\Rightarrow$ Boots; supp $<$ minSup
- Jackets $\Rightarrow$ Boots; supp $<$ minSup
- Outerwear $\Rightarrow$ Boots; supp $>$ minSup

Hierarchical Association Rules: Characteristics



Characteristics

Let $Y = \biguplus_{i=1}^{k} X_i$ be a generalisation.

- For all $1 \leq i \leq k$ it holds $supp(Y \Rightarrow Z) \geq supp(X_i \Rightarrow Z)$
- In general, $supp(Y \Rightarrow Z) = \sum_{i=1}^{k} supp(X_i \Rightarrow Z)$ does not hold (a transaction might contain elements from multiple low-level concepts, e.g. boots *and* sport shoes).

Mining Multi-Level Associations

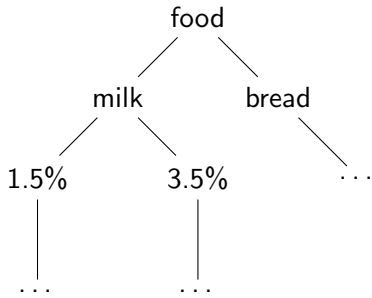
Top-Down, Progressive-Deepening Approach

1. First find high-level strong rules, e.g. milk $\Rightarrow$ bread [20%, 60%]
2. Then find their lower-level "weaker" rules, e.g. low-fat milk $\Rightarrow$ wheat bread [6%, 50%].

Support Threshold Variants

Different minSup threshold across multi-levels lead to different algorithms:

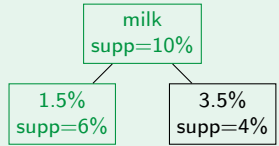
- adopting the same minSup across multi-levels
- adopting reduced minSup at lower levels



Minimum Support for Multiple Levels

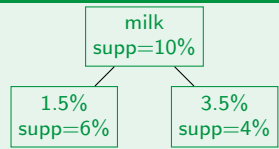
Uniform Support

- Search procedure is simplified (monotonicity)
- User only specifies one threshold



Reduced Support (Variable Support)

- Takes into account lower frequency of items in lower levels



Multilevel Association Mining using Reduced Support

Level-by-level independent method

Examine each node in the hierarchy, regardless of the frequency of its parent node.

Level-cross-filtering by single item

Examine a node only if its parent node at the preceding level is frequent.

Level-cross-filtering by k-itemset

Examine a k-itemset at a given level only if its parent k-itemset at the preceding level is frequent.

Multi-level Association: Redundancy Filtering

Some rules may be redundant due to "ancestor" relationships between items.

Example

- R_1: milk $\Rightarrow$ wheat bread [8%, 70%]
- R_2: 1.5% milk $\Rightarrow$ wheat bread [2%, 72%]

We say that rule 1 is an ancestor of rule 2.

Redundancy

A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

c

Interestingness of Hierarchical Association Rules: Notions

Let $X, X', Y, Y' \subseteq I$ be itemsets.

- X' is ancestor of X iff there exists ancestors $x'_1, \ldots, x'_k$ of $x_1, \ldots, x_k \in X$ and $x_{k+1}, \ldots, x_n$ with $n = |X|$ such that $X' = \{x'_1, \ldots, x'_k, x_{k+1}, \ldots, x_n\}$
- Let X' and Y' be ancestors of X and Y. Then we call the rules $X' \Rightarrow Y'$, $X \Rightarrow Y'$, and $X' \Rightarrow Y$ ancestors of the rule $X \Rightarrow Y$.
- The rule $X' \Rightarrow Y'$ is a direct ancestor of rule $X \Rightarrow Y$ in a set of rules if:
  1. Rule $X' \Rightarrow Y'$ is an ancestor of rule $X \Rightarrow Y$, and
  2. There is no rule $X'' \Rightarrow Y''$ being ancestor of $X \Rightarrow Y$ and $X' \Rightarrow Y'$ is an ancestor of $X'' \Rightarrow Y''$

R-Interestingness

R-Interestingness

A hierarchical association rule $X \Rightarrow Y$ is called R-interesting if:

- There are no direct ancestors of $X \Rightarrow Y$ or
- The actual support is larger than R times the expected support or
- The actual confidence is larger than R times the expected confidence

R-Interestingness: Expected Support

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$ the expected support of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{Z'}[P(Z)] = P(Z') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Z = X \cup Y = \{z_1, \ldots, z_n\}$, $Z' = X' \cup Y' = \{z'_1, \ldots, z'_j, z_{j+1}, \ldots, z_n\}$ and each $z'_i \in Z'$ is an ancestor of $z_i \in Z$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

R-Interestingness: Expected Confidence

Given the rule for $X \Rightarrow Y$ and its ancestor rule $X' \Rightarrow Y'$, then the expected confidence of $X \Rightarrow Y$ is defined as:

$$\mathbb{E}_{X' \Rightarrow Y'}[P(Y|X)] = P(Y' \mid X') \cdot \prod_{i=1}^{j} \frac{P(y_i)}{P(y_i)'}$$

where $Y = \{y_1, \ldots, y_n\}$ and $Y' = \{y'_1, \ldots, y'_j, y_{j+1}, \ldots, y_n\}$ and each $y'_i \in Y'$ is an ancestor of $y_i \in Y$.

R. Srikant, R. Agrawal: Mining Generalized Association Rules. In VLDB, 1995.

R-Interestingness: Example

Item	Support
clothes	20
outerwear	10
jackets	4

Let $R = 1.6$

No	Rule	Support	R-Interesting?
1	clothes $\Rightarrow$ shoes	10	yes: no ancestors
2	outerwear $\Rightarrow$ shoes	9	yes (wrt. rule 1): $supp(X \Rightarrow Y) = 9 > 1.6 \cdot \frac{10}{20} \cdot 10 = 8 = 1.6 \cdot \mathbb{E}(P(Z))$
3	jackets $\Rightarrow$ shoes	4	Not wrt. support: $\mathbb{E}(P(jackets \cup shoes)) = 3.2$ (wrt rule 1) $\mathbb{E}(P(jackets \cup shoes)) = 5.75$ (wrt rule 2)

Still need to check the confidence!

Summary Frequent Itemset & Association Rule Mining

- Frequent Itemsets
  - Mining: Apriori algorithm, hash trees, FP-tree
  - support, confidence
- Simple Association Rules
  - Mining: (Apriori)
  - Interestingness measures: support, confidence, correlation
- Hierarchical Association Rules
  - Mining: Top-Down Progressive Deepening
  - Multilevel support thresholds, redundancy, R-interestingness
- Further Topics (not covered)
  - Quantitative Association Rules (for numerical attributes)
  - Multi-dimensional association rule mining

Agenda

Motivation

Motivation

- In many applications the order matters, e.g. because the ordering encodes spatial or temporal aspects.
- In an ordered sequence, items are allowed to occur more than one time

Applications

Bioinformatics (DNA/protein sequences), Web mining, text mining, sensor data mining, process mining, ...

Sequential Pattern Mining: Basic Notions I

We now consider transactions having an order of the items. Define:

- *Alphabet* Σ is set symbols or characters (denoting items)
- *Sequence* $S = s_1 s_2 \dots s_k$ is an ordered list of a length $|S| = k$ items where $s_i \in \Sigma$ is an item at position i also denoted as $S[i]$
- A *k-sequence* is a sequence of length k
- *Consecutive subsequence* $R = r_1 r_2 \dots r_m$ of $S = s_1 s_2 \dots s_n$ is also a sequence in Σ such that $r_1 r_2 \dots r_m = s_j s_{j+1} \dots s_{j+m-1}$ with $1 \leq j \leq n - m + 1$. We say S contains R and denote this by $R \subseteq S$
- In a more general *subsequence* R of S we allow for gaps between the items of R, i.e. the items of the subsequence $R \subseteq S$ must have the same order of the ones in S but there can be some other items between them
- A *prefix* of a sequence S is any consecutive subsequence of the form $S[1:i] = s_1 s_2 \dots s_i$ with $0 \leq i \leq n$, $S[1:0]$ is the empty prefix

Sequential Pattern Mining: Basic Notions II

- A *suffix* of a sequence S is any consecutive subsequence of the form $S[i:n] = s_i s_{i+1} \dots s_n$ with $1 \leq i \leq n+1$, $S[n+1:n]$ is the empty suffix.
- *(Relative) support* of a sequence R in D: $supp(R) = |\{S \in D \mid R \subseteq S\}|/|D|$
- S is frequent (or sequential) if $supp(S) \geq minSup$ for threshold $minSup$.
- A frequent sequence is *maximal* if it is not a subsequence of any other frequent sequence
- A frequent sequence is *closed* if it is not a subsequence of any other frequent sequence with the same support

Sequential Pattern Mining

Task

Find all frequent subsequences occuring in many transactions.

Difficulty

The number of possible patterns is even larger than for frequent itemset mining!

Example

There are $|\Sigma|^k$ different k-sequences, where $k > |\Sigma|$ is possible and often encountered, e.g. when dealing with DNA sequences where the alphabet only comprises four symbols.

Sequential Pattern Mining Algorithms

Breadth-First Search Based

- GSP (Generalized Sequential Pattern) algorithm[2]
- SPADE[3]
- ...

Depth-First Search Based

- PrefixSpan[4]
- SPAM[5]
- ...

[2] Sirkant & Aggarwal: *Mining sequential patterns: Generalizations and performance improvements*. EDBT 1996

[3] Zaki M J. *SPADE: An efficient algorithm for mining frequent sequences*. Machine learning, 2001, 42(1-2): 31-60.

[4] Pei at. al.: *Mining sequential patterns by pattern-growth: PrefixSpan approach*. TKDE 2004

[5] Ayres, Jay, et al: *Sequential pattern mining using a bitmap representation*. SIGKDD 2002.

GSP (Generalized Sequential Pattern) algorithm

- Breadth-first search: Generate frequent sequences ascending by length
- Given the set of frequent sequences at level k, generate *all* possible sequence extensions or candidates at level $k + 1$
- Uses the Apriori principle (anti-monotonicity)
- Next compute the support of each candidate and prune the ones with $supp(c) < minSup$
- Stop the search when no more frequent extensions are possible

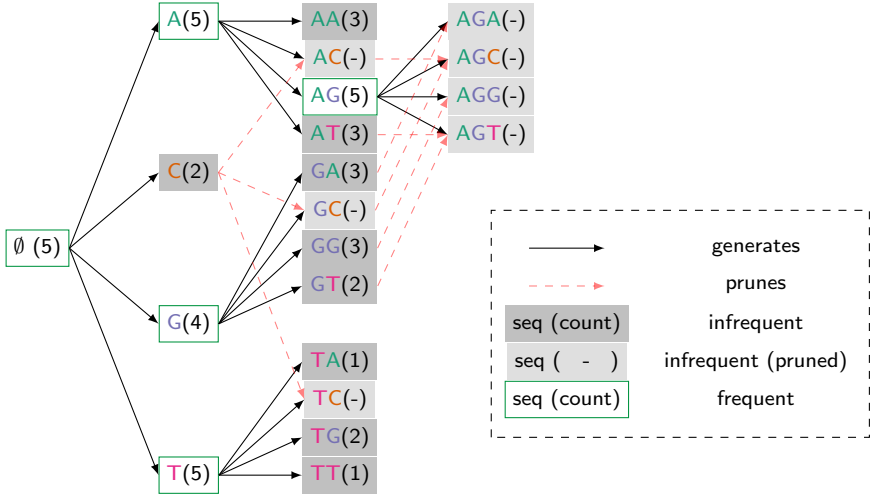
Projection-Based Sequence Mining: PrefixSpan: Representation

- The sequence search space can be organized in a prefix search tree
- The root (level 0) contains the empty sequence with each item $x \in \Sigma$ as one of its children
- A node labelled with sequence: $S = s_1 s_2 \ldots s_k$ at level k has children of the form $S' = s_1 s_2 \ldots s_k s_{k+1}$ at level $k+1$ (i.e. S is a prefix of S' or S' is an extension of S)

Prefix Search Tree: Example

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG

$minSup = .8$



Projected Database

- For a database D and an item $s \in \Sigma$, the projected database w.r.t. s is denoted D_s and is found as follows: For each sequence $S_i \in D$ do
  - Find the first occurrence of s in S_i, say at position p
  - $suff_{S_i,s} \leftarrow suffix(S_i)$ starting at position $p+1$
  - Remove infrequent items from $suff_{S_i,s}$
  - $D_s = D_s \cup suff_{S_i,s}$

Example

$minSup = .8$ (i.e. 4 transactions)

ID	Sequence	D_A	D_G	D_T
S_1	CAGAAGT	GAAGT	AAGT	$\emptyset$
S_2	TGACAG	AG	AAG	GAAG
S_3	GAG	G	AG	-
S_4	AGTT	GTT	TT	T
S_5	ATAG	TAG	$\emptyset$	AG

Projection-Based Sequence Mining: PrefixSpan Algorithm

- The *PrefixSpan* algorithm computes the support for only the individual items in the projected databased D_s
- Then performs recursive projections on the frequent items in a depth-first manner

1: Initialization: $D_R \leftarrow D, R \leftarrow \emptyset, \mathcal{F} \leftarrow \emptyset$
2: **procedure** PREFIXSPAN($D_R, R, minSup, \mathcal{F}$)
3: **for all** $s \in \Sigma$ such that $supp(s, D_R) \geq minSup$ **do**
4: $R_s \leftarrow R + s$ ▷ append s to the end of R
5: $\mathcal{F} \leftarrow \mathcal{F} \cup \{(R_s, sup(s, D_R))\}$ ▷ calculate support of s for each R_s within D_R
6: $D_s \leftarrow \emptyset$
7: **for all** $S_i \in D_R$ **do**
8: $S_i' \leftarrow$ projection of S_i w.r.t. item s
9: Remove all infrequent symbols from S_i'
10: **if** $S' \neq \emptyset$ **then**
11: $D_s \leftarrow D_s \cup S_i'$
12: **if** $D_s \neq \emptyset$ **then**
13: PrefixSpan($D_s, R_s, minSup, \mathcal{F}$)

PrefixSpan: Example

minSup = 0.8 (i.e. 4 transactions)

$D_\emptyset$

ID	Sequence
S_1	CAGAAGT
S_2	TGACAG
S_3	GAG
S_4	AGTT
S_5	ATAG

A(5)~~C~~(2)~~G~~(5)T(4)

D_G

ID	Sequence
S_1	AAGT
S_2	AAG
S_3	AG
S_4	TT
S_5	$\emptyset$

~~A~~(3)~~G~~(3)~~T~~(2)

D_T

ID	Sequence
S_1	$\emptyset$
S_2	GAAG
-	-
S_4	T
S_5	AG

~~A~~(2)~~G~~(2)~~T~~(1)

D_A

ID	Sequence
S_1	GAAGT
S_2	AG
S_3	G
S_4	GTT
S_5	TAG

~~A~~(3)G(5)~~T~~(3)

D_{AG}

ID	Sequence
S_1	G
S_2	$\emptyset$
S_3	$\emptyset$
S_4	$\emptyset$
S_5	$\emptyset$

~~G~~(1)

Hence, the frequent sequences are: $\emptyset$, A, G, T, AG

Interval-based Sequential Pattern Mining

Interval-Based Representation

- Deals with the more common interval-based items s (or events).
- Each event has a starting t_s^+ and an ending time point t_s^-, where $t_s^+ < t_s^-$

Application

Health data analysis, Stock market data analysis, etc.

Relationships

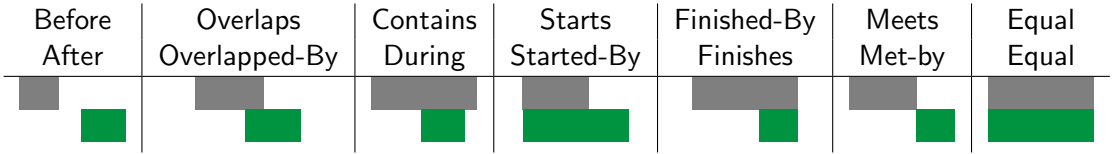
Predefined relationships between items are more complex.

- Point-based relationships: before, after, same time.
- Interval-based relationships: Allen's relations[6], End point representation[7], etc.

[6] Allen: Maintaining knowledge about temporal intervals. In Communications of the ACM 1983

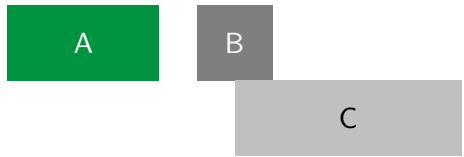
[7] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

Allen's Relations



Problem

- Allen's relationships only describe the relation between two intervals.
- Describing the relationship between k intervals unambiguously requires $\mathcal{O}(k^2)$ comparisons.



Interval-based Sequential Pattern Mining

- *TPrefixSpan*[8] converts interval-based sequences into point-based sequences:

$\{A^+\}, \{A^-\}, \{B^+\}, \{B^-\}$

$\{A^+\}, \{B^+\}, \{A^-\}, \{B^-\}$

$\{A^+\}, \{A^-, B^+\}, \{B^-\}$

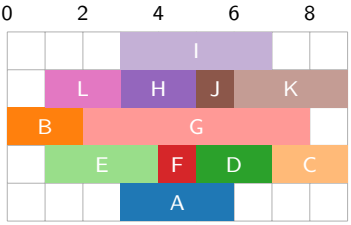
- Similar prefix projection mining approach as PrefixSpan algorithm.
- Validation checking is necessary in each expanding iteration to make sure that the appended time point can form an interval with a time point in the prefix.

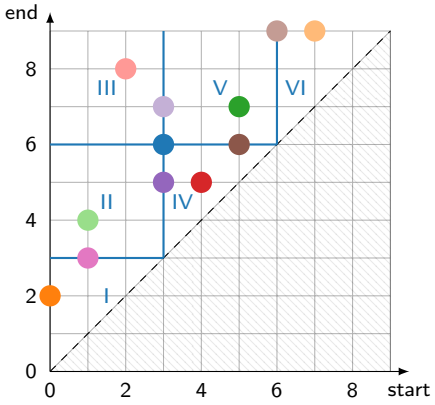
[8] Wu, Shin-Yi, and Yen-Liang Chen: Mining nonambiguous temporal patterns for interval-based events. TKDE 2007

An Open Issue: Considering Timing Information

Idea

Learn pattern from data by clustering, e.g. QTempIntMiner[9], Event Space Miner[10], PIVOTMiner[11]





[9] Guyet, T., & Quiniou, R.: *Mining temporal patterns with quantitative intervals.* ICDMW 2008

[10] Ruan, G., Zhang, H., & Plale, B.: *Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data.* IEEE Big Data 2014

[11] Hassani M., Lu Y. & Seidl T.: *A Geometric Approach for Mining Sequential Patterns in Interval-Based Data Streams.* FUZZ-IEEE 2016

Agenda

What is Clustering?

Clustering

Grouping a set of data objects into clusters (=collections of data objects).

- *Similar* to one another within the same cluster
- *Dissimilar* to the objects in other clusters

Typical Usage

- As a *stand-alone tool* to get insight into data distribution
- As a *preprocessing* step for other algorithms

General Applications of Clustering

- Preprocessing – as a data reduction (instead of sampling)
  - Image data bases (color histograms for filter distances)
  - Stream clustering (handle endless data sets for offline clustering)
- Pattern Recognition and Image Processing
- Spatial Data Analysis:
  - create thematic maps in Geographic Information Systems by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Business Intelligence (especially market research)
- WWW
  - Documents (Web Content Mining)
  - Web-logs (Web Usage Mining)
- Biology, e.g. Clustering of gene expression data

Application Example: Downsampling Images

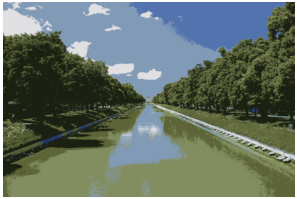
- Reassign color values to k distinct colors
- Cluster pixels using color difference, not spatial data

65536

256

16

8



4

2

Major Clustering Approaches

- Partitioning algorithms: Find k partitions, minimizing some objective function
- Probabilistic Model-Based Clustering (EM)
- Density-based: Find clusters based on connectivity and density functions
- Hierarchical algorithms: Create a hierarchical decomposition of the set of objects
- Other methods:
  - Grid-based
  - Neural networks (SOMs)
  - Graph-theoretical methods
  - Subspace Clustering

Agenda

Partitioning Algorithms: Basic Concept

Partition

Given a set D, a partitioning $\mathcal{C} = \{C_1, \ldots, C_k\}$ of D fulfils:

- $C_i \subseteq D$ for all $1 \leq i \leq k$
- $C_i \cap C_j = \emptyset \iff i \neq j$
- $\bigcup C_i = D$

(i.e. each element of D is in exactly one set C_i)

Goal

Construct a partitioning of a database D of n objects into a set of k ($k \leq n$) clusters minimizing an objective function.

Exhaustively enumerating all possible partitionings into k sets in order to find the global minimum is too expensive.

Partitioning Algorithms: Basic Concept

Popular Heuristic Methods

- Choose k representatives for clusters, e.g., randomly
- Improve these initial representatives iteratively:
  - Assign each object to the cluster it "fits best" in the current clustering
  - Compute new cluster representatives based on these assignments
  - Repeat until the change in the objective function from one iteration to the next drops below a threshold

Example

- k-means: Each cluster is represented by the center of the cluster
- k-medoid: Each cluster is represented by one of its objects

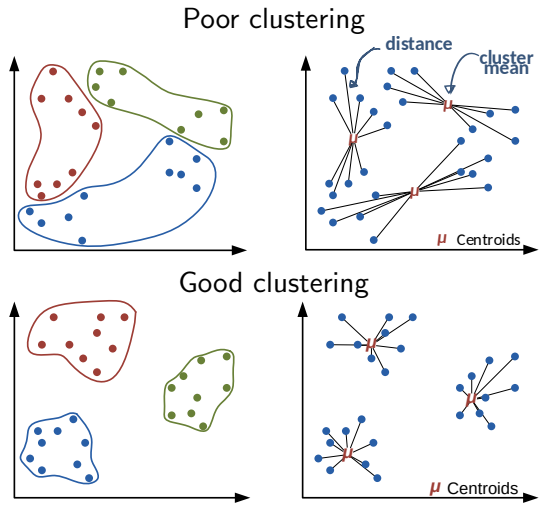
k-Means Clustering: Basic Idea

Idea[1]

Find a clustering such that the within-cluster variation of each cluster is small and use the centroid of a cluster as representative.

Objective

For a given k, form k groups so that the sum of the (squared) distances between the mean of the groups and their elements is minimal



[1] S.P. Lloyd: Least squares quantization in PCM. In IEEE Information Theory, 1982 (original version: technical report, Bell Labs, 1957)

k-Means Clustering: Basic Notions

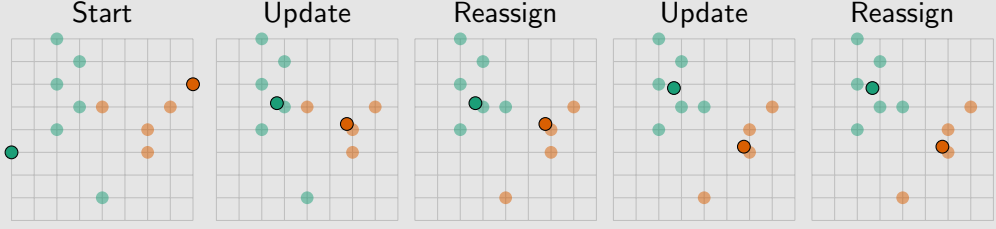
- Objects $p = (p_1, \ldots, p_d)$ are points in a d-dimensional vector space (the mean μ_S of a set of points S must be defined: $\mu_S = \frac{1}{|S|} \sum_{p \in S} p$)
- Measure for the compactness of a *cluster* C_j (sum of squared distances): $SSE(C_j) = \sum_{p \in C_j} ||p - \mu_{C_j}||_2^2$
- Measure for the compactness of a *clustering* $\mathcal{C}$: $SSE(\mathcal{C}) = \sum_{C_j \in \mathcal{C}} SSE(C_j) = \sum_{p \in D} ||p - \mu_{C(p)}||_2^2$
- Optimal Partitioning: $\operatorname{argmin}_{\mathcal{C}} SSE(\mathcal{C})$
- Optimizing the within-cluster variation is computationally challenging (NP-hard) $\rightsquigarrow$ use efficient heuristic algorithms

k-Means Clustering: Algorithm

k-Means Algorithm: Lloyd's algorithm

1: Given: k
2: Initialization: Choose k arbitrary representatives
3: **repeat**
4: Assign each object to the cluster with the nearest representative.
5: Compute the centroids of the clusters of the current partitioning.
6: **until** representatives do not change

Example



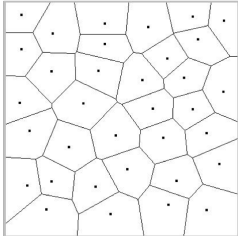
k-Means: Voronoi Model for Convex Cluster Regions

Voronoi Diagram

- For a given set of points $P = \{p_1, \ldots, p_k\}$ (here: cluster representatives), a *Voronoi diagram* partitions the data space into *Voronoi cells*, one cell per point
- The cell of a point $p \in P$ covers all points in the data space for which p is the nearest neighbors among the points from P

Observations

- The Voronoi cells of two neighboring points $p_i, p_j \in P$ are separated by the perpendicular hyperplane ("Mittelsenkrechte") between p_i and p_j.
- Voronoi cells are intersections of half spaces and thus convex regions



k-Means: Discussion

Strength

- Relatively efficient: $\mathcal{O}(tkn)$ (n: #obj., k: #clus., t: #it.; typically: $k, t \ll n$)
- Easy implementation

Weaknesses

- Applicable only when mean is defined
- Need to specify k, the number of clusters, in advance
- Sensitive to noisy data and outliers
- Clusters are forced to convex space partitions (Voronoi Cells)
- Result and runtime strongly depend on the initial partition; often terminates at a local optimum – however: methods for a good initialization exist

Variants: Basic Idea

One Problem of k-Means

Applicable only when mean is defined (vector space)

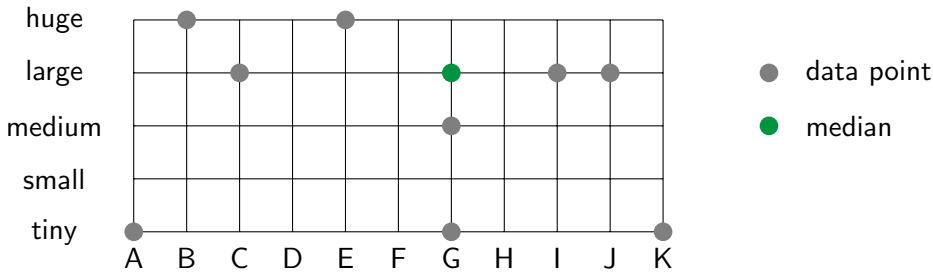
Alternatives for *Mean* representatives

- *Median*: (Artificial) Representative object "in the middle"
- *Mode*: Value that appears most often
- *Medoid*: Representative object "in the middle"

Objective

Find k representatives so that the sum of total distances (TD) between objects and their closest representative is minimal (more robust against outliers).

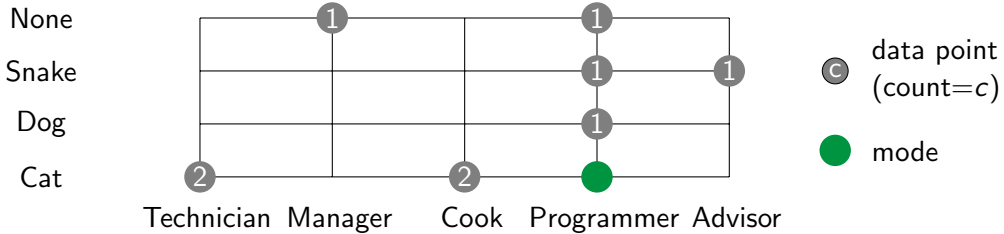
k-Median



Idea

- If there is an ordering on the data use median instead of mean.
- Compute median separately per dimension ($\rightsquigarrow$ efficient computation)

k-Mode



Mode

- Given: categorical data $D \subseteq \Omega = A_1 \times \cdots \times A_d$ where A_i are categorical attributes
- A *mode* of D is a vector $M = (m_1, \ldots, m_d) \in \Omega$ that minimizes $d(M, D) = \sum_{p \in D} d(p, M)$ where d is a distance function for categorical values (e.g. Hamming distance)
- Note: M is not necessarily an element of D

k-Mode

Theorem to determine a mode

Let $f(c, j, D) = \frac{1}{n}|\{p \in D \mid p[j] = c\}|$ be the relative frequency of category c of attribute A_j in the data, then:

$$d(M, D) \text{ is minimal } \Leftrightarrow \forall j \in \{1, \ldots, d\} \forall c \in A_j : f(m_j, j, D) \geq f(c, j, D)$$

- This allows to use the k-Means paradigm to cluster categorical data without losing its efficiency
- k-Modes algorithm[1] proceeds similar to k-Means algorithm
- Note: The mode of a dataset might be not unique

[1] Huang, Z. "A Fast Clustering Algorithm to Cluster very Large Categorical Data Sets in Data Mining" DMKD (1997)

k-Medoid

Potential problems with previous methods:

- Artificial centroid object might not make sense (e.g. education="high school" and occupation="professor")
- There might only be a distance function available but no explicit attribute-based data representations (e.g. Edit Distance on strings)

Partitioning Around Medoids [1]: Initialization

Given k, the *k-medoid* algorithm is initialized as follows:

- Select k objects arbitrarily as initial medoids (representatives)
- Assign each remaining (non-medoid) object to the cluster with the nearest representative
- Compute current $TD_{current}$

[1] Kaufman, Leonard, and Peter Rousseeuw. "Clustering by means of medoids." (1987)

k-Medoid

Partitioning Around Medoids (PAM) Algorithm

procedure PAM(Set D, Integer k)
  Initialize k medoids
  $\Delta_{TD} = -\infty$
  while $\Delta_{TD} < 0$ **do**
    Compute $TD_{N \leftrightarrow M}$ for each pair (medoid M, non-medoid N), i.e., TD after swapping M with N
    Choose pair (M, N) with minimal $\Delta_{TD} = TD_{N \leftrightarrow M} - TD_{current}$
    if $\Delta_{TD} < 0$ **then**
      Replace medoid M with non-medoid N
      $TD_{current} \leftarrow TD_{N \leftrightarrow M}$
      Store current medoids and assignments as best partitioning so far
  return medoids

- Problem with PAM: high complexity $\mathcal{O}\left(tk(n-k)^2\right)$
- Several heuristics can be employed, e.g. CLARANS [1]: randomly select (medoid, non-medoid)-pairs instead of considering all pairs

[1] Ng, Raymond T., and Jiawei Han. "CLARANS: A method for clustering objects for spatial data mining." IEEE TKDE (2002)

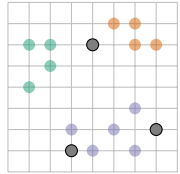
K-Means/Median/Mode/Medoid Clustering: Discussion

	k-**Means**	k-**Median**	k-**Mode**	k-**Medoid**
data	numerical (mean)	ordinal	categorical	metric
efficiency	high $\mathcal{O}(tkn)$			low $\mathcal{O}\left(tk(n-k)^2\right)$
sensitivity to outliers	high	low		

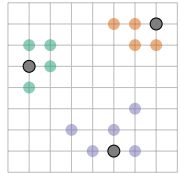
- Strength: Easy implementation (many variations and optimizations exist)
- Weaknesses
  - Need to specify k in advance
  - Clusters are forced to convex space partitions (Voronoi Cells)
  - Result and runtime strongly depend on the initial partition; often terminates at a local optimum – however: methods for good initialization exist

Initialization of Partitioning Clustering Methods

- Naive
  - Choose sample A of the dataset
  - Cluster A and use centers as initialization
- k-means++[1]
  - Select first center uniformly at random
  - Choose next point with probability proportional to the squared distance to the nearest center already chosen
  - Repeat until k centers have been selected
  - Guarantees an approximation ratio of $\mathcal{O}(\log k)$ (standard k-means can generate arbitrarily bad clusterings)
- In general: Repeat with different initial centers and choose result with lowest clustering error



Bad initialization



Good initialization

[1] Arthur, D., Vassilvitskii, S. "k-means++: The Advantages of Careful Seeding." ACM-SIAM Symposium on Discrete Algorithms (2007)

Choice of the Parameter k

- Idea for a method:
  - Determine a clustering for each $k = 2, \ldots, n-1$
  - Choose the "best" clustering
- But how to measure the quality of a clustering?
  - A measure should not be monotonic over k
  - The measures for the compactness of a clustering SSE and TD are monotonously decreasing with increasing value of k.

Silhouette-Coefficient [1]

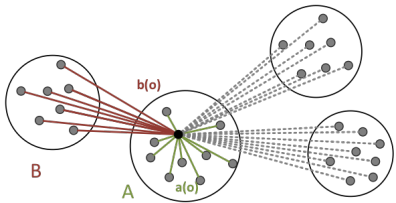
Quality measure for k-means or k-medoid clusterings that is not monotonic over k.

[1] Rousseeuw, P. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics (1987)

The Silhouette Coefficient

Basic idea

- How good is the clustering = how appropriate is the mapping of objects to clusters
- Elements in cluster should be "similar" to their representative
  - Measure the average distance of objects to their representative: $a(o)$
- Elements in different clusters should be "dissimilar"
  - Measure the average distance of objects to alternative clusters (i.e. second closest cluster): $b(o)$



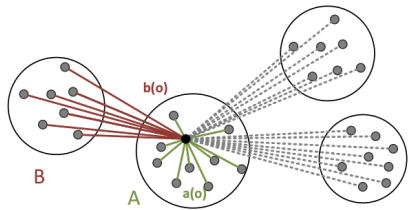
The Silhouette Coefficient

- $a(o)$ = "Avg. distance between o and objects in its cluster A."

$$a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} d(o, p)$$

- $b(o)$: "Smallest avg. distance between o and objects in other cluster."

$$b(o) = \min_{C_i \neq C(o)} \left\{ \frac{1}{|C_i|} \sum_{p \in C_i} d(o, p) \right\}$$



The Silhouette Coefficient

- The silhouette of o is then defined as

$$s(o) = \begin{cases} 0 & \text{if } a(o) = 0, \text{ e.g. } |C_i| = 1 \\ \frac{b(o)-a(o)}{max(a(o),b(o))} & \text{else} \end{cases}$$

- The value range of the silhouette coefficient is $[-1, 1]$
- The silhouette of a cluster C_i is defined as

$$s(C_i) = \frac{1}{|C_i|} \sum_{o \in C_i} s(o)$$

- The silhouette of a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ is defined as

$$s(\mathcal{C}) = \frac{1}{|D|} \sum_{o \in D} s(o)$$

where D denotes the whole dataset

The Silhouette Coefficient

- "Reading" the silhouette coefficient: Let $a(o) \neq 0$
  - $b(o) \gg a(o) \implies s(o) \approx 1$: good assignment of o to its cluster A
  - $b(o) \approx a(o) \implies s(o) \approx 0$: o is in-between A and B
  - $b(o) \ll a(o) \implies s(o) \approx -1$: bad, on average o is closer to members of B
- Silhouette coefficient $s(\mathcal{C})$ of a clustering: Average silhouette of all objects
  - $0.7 < s(\mathcal{C}) \leq 1.0$: strong structure
  - $0.5 < s(\mathcal{C}) \leq 0.7$: medium structure
  - $0.25 < s(\mathcal{C}) \leq 0.5$: weak structure
  - $s(\mathcal{C}) \leq 0.25$: no structure

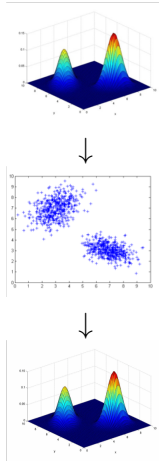
Silhouette Coefficient: Example

Image from Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Agenda

Expectation Maximization (EM)

- Statistical approach for finding maximum likelihood estimates of parameters in probabilistic models.
- Here: Using EM as clustering algorithm
- Approach: Observations are drawn from one of several components of a mixture distribution.
- Main idea:
  - Define clusters as probability distributions $\rightarrow$ each object has a certain probability of belonging to each cluster
  - Iteratively improve the parameters of each distribution (e.g. center, "width" and "height" of a Gaussian distribution) until some quality threshold is reached



Additional Literature: C. M. Bishop "Pattern Recognition and Machine Learning", Springer, 2009

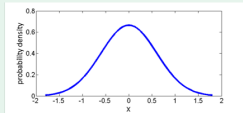
Excursus: Gaussian Mixture Distributions

Note: EM is not restricted to Gaussian distributions, but they will serve as example in this lecture.

Gaussian Distribution

- Univariate: single variable $x \in \mathbb{R}$:

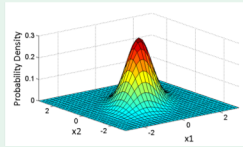
$$p(x \mid \mu, \sigma^2) = \mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



with *mean* $\mu \in \mathbb{R}$ and *variance* $\sigma^2 \in \mathbb{R}$

- Multivariate: d-dimensional vector $x \in \mathbb{R}^d$:

$$p(x \mid \mu, \Sigma) = \mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$



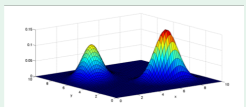
with *mean vector* $\mu \in \mathbb{R}^d$ and *covariance matrix* $\Sigma \in \mathbb{R}^{d \times d}$

Excursus: Gaussian Mixture Distributions

Gaussian mixture distribution with k components

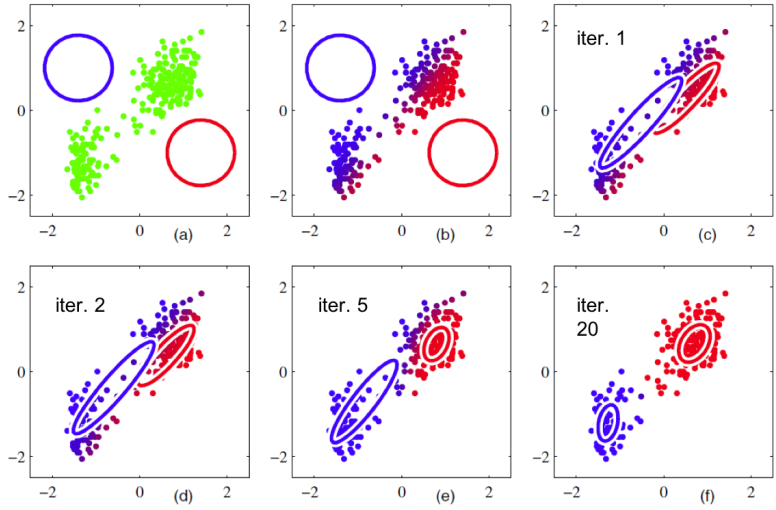
- For d-dimensional vector $x \in \mathbb{R}^d$:

$$p(x) = \sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x \mid \mu_l, \Sigma_l)$$



with *mixing coefficients* $\pi_l \in \mathbb{R}$, $\sum_l \pi_l = 1$ and $0 \leq \pi_l \leq 1$

EM: Exemplary Application



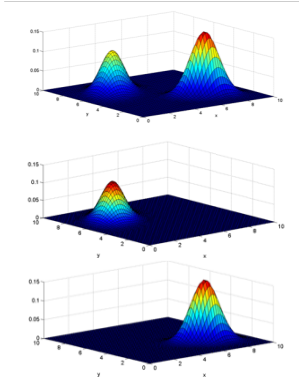
Example taken from: C. M. Bishop "Pattern Recognition and Machine Learning", 2009

EM: Clustering Model

Clustering

A clustering $\mathcal{M} = (C_1, \ldots, C_k)$ is represented by a mixture distribution with parameters $\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_k, \mu_k, \Sigma_k)$:

$$p(x \mid \theta) = \sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x \mid \mu_l, \Sigma_l)$$



Cluster

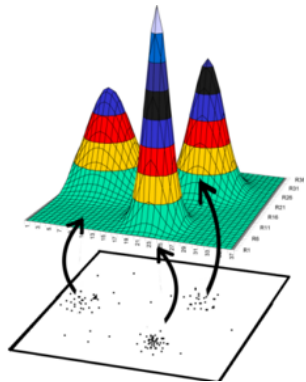
Each cluster is represented by one component of the mixture distribution:

$$p(x \mid \mu_l, \Sigma_l) = \mathcal{N}(x \mid \mu_l, \Sigma_l)$$

EM: Maximum Likelihood Estimation

- Given a dataset $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, the *likelihood* that all data points $x_i \in X$ are generated (independently) by the mixture model with parameters θ is given as:

$$p(X \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta)$$



Goal

Find the *maximum likelihood estimate (MLE)*, i.e., the parameters θ_{ML} with maximal likelihood:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \{p(X \mid \theta)\}$$

EM: Maximum Likelihood Estimation

- Goal: Find MLE. For convenience, we use the log-likelihood:

$$\begin{aligned}\theta_{ML} &= \operatorname*{argmax}_{\theta} \{p(X \mid \theta)\} \\ &= \operatorname*{argmax}_{\theta} \{\log p(X \mid \theta)\}\end{aligned}$$

- The log-likelihood can be written as

$$\begin{aligned}\log p(X \mid \theta) &= \log \prod_{i=1}^{n} \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l) \\ &= \sum_{i=1}^{n} \log \sum_{l=1}^{k} \pi_l \cdot p(x_i \mid \mu_l, \Sigma_l)\end{aligned}$$

EM: Maximum Likelihood Estimation

- Maximization w.r.t. the means:

$$\frac{\partial \log p(X \mid \theta)}{\partial \mu_j} = \sum_{i=1}^{n} \frac{\partial \log p(x_i \mid \theta)}{\partial \mu_j} = \sum_{i=1}^{n} \frac{\frac{\partial \log p(x_i|\theta)}{\partial \mu_j}}{p(x_i \mid \theta)} = \sum_{i=1}^{n} \frac{\frac{\partial \log p(x_i|\theta)}{\partial \mu_j}}{\sum_{l=1}^{k} p(x_i \mid \mu_l, \Sigma_l)}$$

$$= \sum_{i=1}^{n} \frac{\pi_j \cdot \Sigma_j^{-1}(x_i - \mu_j) \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} p(x_i \mid \mu_l, \Sigma_l)}$$

$$= \Sigma_j^{-1} \sum_{i=1}^{n} (x_i - \mu_j) \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_l \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_l)} \stackrel{!}{=} 0$$

- Use $\frac{\partial}{\partial \mu_j} \mathcal{N}(x_i \mid \mu_j, \Sigma_j) = \Sigma_j^{-1}(x_i - \mu_j) \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)$
- Define $\gamma_j(x_i) := \pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)$: Probability that component j generated x_i

EM: Maximum Likelihood Estimation

- Maximization w.r.t. the means yields

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i) x_i}{\sum_{i=1}^{n} \gamma_j(x_i)}$$

- Maximization w.r.t. the covariance matrices yields

$$\Sigma_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i)(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{n} \gamma_j(x_i)}$$

- Maximization w.r.t. the mixing coefficients yields

$$\pi_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i)}{\sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_l(x_i)}$$

EM: Maximum Likelihood Estimation

Problem with finding the optimal parameters θ_{ML}:

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_j(x_i) x_i}{\sum_{i=1}^{n} \gamma_j(x_i)} \quad \text{and} \quad \gamma_j(x_i) = \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_j \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_k)}$$

- Non-linear mutual dependencies
- Optimizing the Gaussian of cluster j depends on all other Gaussians.
- There is no closed-form solution!
- Approximation through iterative optimization procedures
- Break the mutual dependencies by optimizing μ_j and $\gamma_j(x_i)$ independently

EM: Iterative Optimization

Iterative Optimization

1. Initialize means μ_j, covariances Σ_j, and mixing coefficients π_j and evaluate the initial log-likelihood.
2. **E-step**: Evaluate the responsibilities using the current parameter values:

$$\gamma_j^{new}(x_i) = \frac{\pi_j \cdot \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \pi_j \cdot \mathcal{N}(x_i \mid \mu_l, \Sigma_l)}$$

3. **M-step**: Re-estimate the parameters using the current responsibilities:

$$\mu_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i) x_i}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}$$

$$\vdots$$

EM: Iterative Optimization

Iterative Optimization

$$\vdots$$

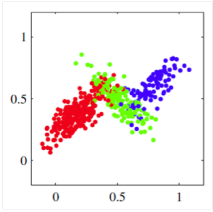
$$\Sigma_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)(x_i - \mu_j^{new})(x_i - \mu_j^{new})^T}{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}$$

$$\pi_j^{new} = \frac{\sum_{i=1}^{n} \gamma_j^{new}(x_i)}{\sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_l^{new}(x_i)}$$

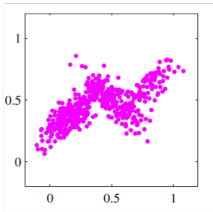
4. Evaluate the new log-likelihood $\log p(X \mid \theta^{new})$ and check for convergence of parameters or log-likelihood ($|\log p(X \mid \theta^{new}) - \log p(X \mid \theta)| \leq \epsilon$). If the convergence criterion is not satisfied, set $\theta = \theta^{new}$ and go to step 2.

EM: Turning the Soft Clustering into a Partitioning

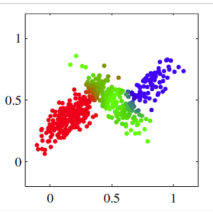
- EM obtains a soft clustering (each object belongs to each cluster with a certain probability) reflecting the uncertainty of the most appropriate assignment



original data



input for EM



soft clustering result of EM

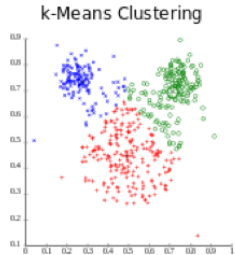
- Modification to obtain a partitioning variant: Assign each object to the cluster to which it belongs with the highest probability

$$C(x_i) = \underset{l \in \{1,\ldots,k\}}{\operatorname{argmax}} \{\gamma_l(x_i)\}$$

Example taken from: C. M. Bishop "Pattern Recognition and Machine Learning", 2009

EM: Discussion

- Superior to k-Means for clusters of varying size or clusters having differing variances
  - More accurate data representation
- Convergence to (possibly local) maximum
- Computational effort for t iterations: $\mathcal{O}(tnk)$
  - t is quite high in many cases
- Both, result and runtime, strongly depend on
  - the initial assignment
    - Do multiple random starts and choose the final estimate with highest likelihood
    - Initialize with clustering algorithms (e.g., k-Means): usually converges much faster
    - Local maxima and initialization issues have been addressed in various extensions of EM
  - a proper choice of k (next slide)



EM: Model Selection for Determining Parameter k

Problem

Classical trade-off problem for selecting the proper number of components k:

- If k is too high, the mixture may overfit the data
- If k is too low, the mixture may not be flexible enough to approximate the data

Idea

Determine candidate models θ_k for $k \in \{k_{min}, \ldots, k_{max}\}$ and select the model according to some quality measure *qual*:

$$\theta_{k^*} = \max_{k \in \{k_{min}, \ldots, k_{max}\}} \{qual(\theta_k)\}$$

- Silhouette Coefficient (as for k-Means) only works for partitioning approaches
- The likelihood is nondecreasing in k

EM: Model Selection for Determining Parameter k

Solution

Deterministic or stochastic *model selection* methods [1] which try to balance the goodness of fit with simplicity.

- Deterministic:

$$qual(\theta_k) = \log p(X \mid \theta_k) + \mathcal{P}(k)$$

  where $\mathcal{P}(k)$ is an increasing function penalizing higher values of k
- Stochastic: Based on Markov Chain Monte Carlo (MCMC)

[1] G. McLachlan and D. Peel. Finite Mixture Models. Wiley, New York, 2000.

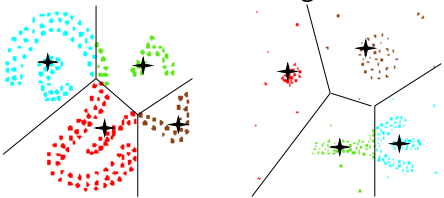
Agenda

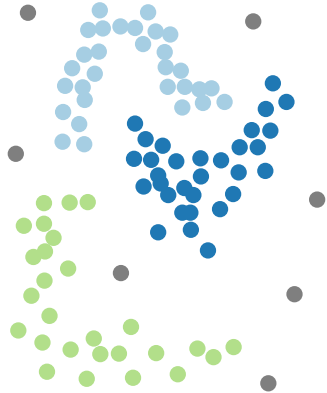
Density-Based Clustering

Basic Idea

Clusters are dense regions in the data space, separated by regions of lower density

Results of a k-medoid algorithm for $k = 4$:





Density-Based Clustering: Basic Concept

Note

Different density-based approaches exist in the literature. Here we discuss the ideas underlying the DBSCAN algorithm.

Intuition for Formalization

- For any point in a cluster, the local point density around that point has to exceed some threshold
- The set of points from one cluster is spatially connected

Density-Based Clustering: Basic Concept

Local Point Density

Local point density at a point q defined by two parameters:

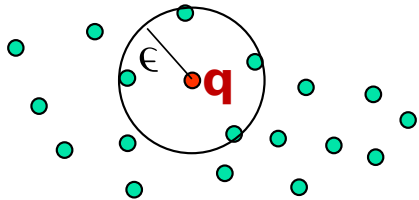
- ϵ-radius for the neighborhood of point q

$$N_\epsilon(q) = \{p \in D \mid dist(p, q) \leq \epsilon\} \tag{1}$$

In this chapter, we assume that $q \in N_\epsilon(q)$!

- *MinPts*: minimum number of points in the given neighbourhood $N_\epsilon(q)$.

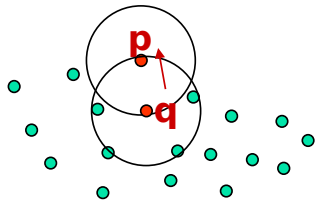
Density-Based Clustering: Basic Concept

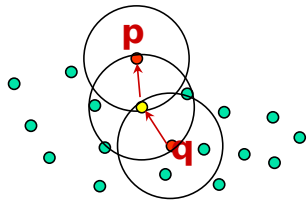


Core Point

q is called a core object (or core point) w.r.t. ϵ, *MinPts* if $|N_\epsilon(q)| \geq minPts$

Density-Based Clustering: Basic Definitions





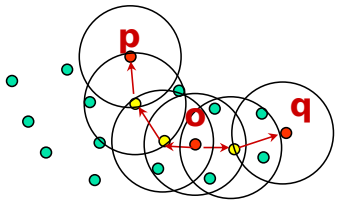
(Directly) Density-Reachable

p directly density-reachable from q w.r.t. ϵ, *MinPts* if:

1. $p \in N_\epsilon(q)$ and
2. q is core object w.r.t. ϵ, *MinPts*

Density-reachable is the transitive closure of directly density-reachable

Density-Based Clustering: Basic Definitions



Density-Connected

p is *density-connected* to a point q w.r.t. ϵ, *MinPts* if there is a point o such that both, p and q are density-reachable from o w.r.t. ϵ, *MinPts*

Density-Based Clustering: Basic Definitions

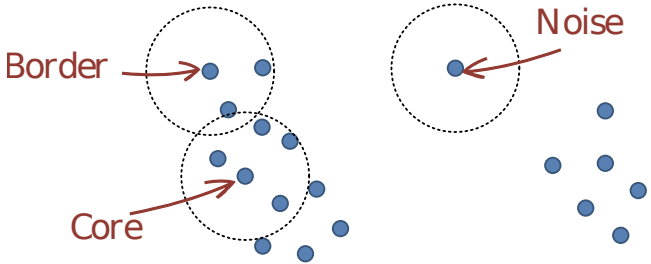
Density-Based Cluster

$\emptyset \subset C \subseteq D$ with database D satisfying:

Maximality: If $q \in C$ and p is density-reachable from q then $p \in C$

Connectivity: Each object in C is density-connected to all other objects in C

Density-Based Clustering: Basic Definitions



Density-Based Clustering

A partitioning $\{C_1, \ldots, C_k, N\}$ of the database D where

- $C_1, \ldots, C_k$ are all density-based clusters
- $N = D \setminus (C_1 \cup \ldots \cup C_k)$ is called the *noise* (objects not in any cluster)

Density-Based Clustering: DBSCAN Algorithm

Basic Theorem

- Each object in a density-based cluster C is density-reachable from any of its core-objects
- Nothing else is density-reachable from core objects.

Density-Based Clustering: DBSCAN Algorithm

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise[12]

1: **for all** $o \in D$ **do**
2: **if** o is not yet classified **then**
3: **if** o is a core-object **then**
4: Collect all objects density-reachable from o and assign them to a new cluster.
5: **else**
6: Assign o to noise N

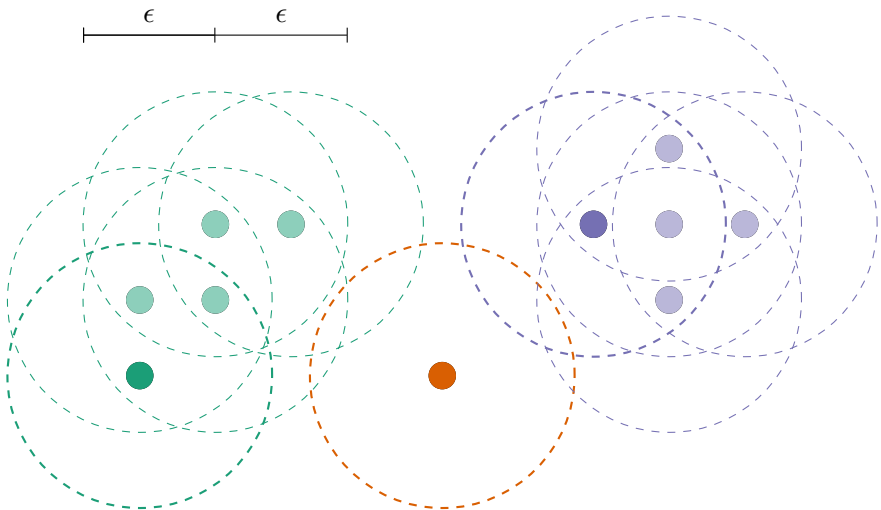
Note

Density-reachable objects are collected by performing successive ϵ-neighborhood queries.

[12] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In KDD 1996 , pp. 226-231.

DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1, C_2; Noise: N



Determining the Parameters ϵ and *MinPts*

Recap

Cluster: Point density higher than specified by ϵ and *MinPts*

Idea

Use the point density of the least dense cluster in the data set as parameters.

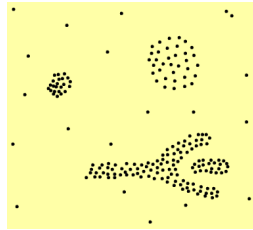
Problem

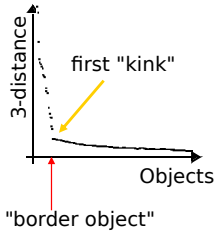
How to determine this?

Determining the Parameters ϵ and *MinPts*

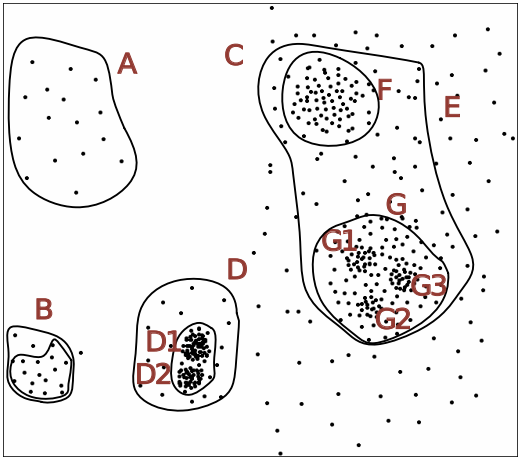
Heuristic

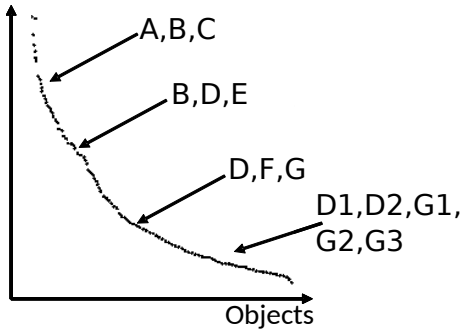
1. Fix a value for *MinPts* (default: $2d - 1$ where d is the dimension of the data space)
2. Compute the k-distance for all points $p \in D$ (distance from p to the its k-nearest neighbor), with $k = minPts$.
3. Create a k-distance plot, showing the k-distances of all objects, sorted in decreasing order
4. The user selects "border object" o from the *MinPts*-distance plot: ϵ is set to *MinPts*-distance(o).





Determining the Parameters ϵ and *MinPts*: Problematic Example





Database Support for Density-Based Clustering

Standard DBSCAN evaluation is based on recursive database traversal. Böhm et al.[13] observed that DBSCAN, among other clustering algorithms, may be efficiently built on top of similarity join operations.

ϵ-Similarity Join

An ϵ-*similarity join* yields all pairs of ϵ-similar objects from two data sets Q, P:

$$Q \bowtie_\epsilon P = \{(q, p) \in Q \times P \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM Q, P WHERE $dist(Q, P) \leq \epsilon$

[13] Böhm C., Braunmüller, B., Breunig M., Kriegel H.-P.: *High performance clustering based on the similarity join*. CIKM 2000: 298-305.

Database Support for Density-Based Clustering

ϵ-Similarity Self-Join

An ϵ-similarity *self* join yields all pairs of ϵ-similar objects from a database D.

$$D \bowtie_\epsilon D = \{(q, p) \in D \times D \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM D q, D p WHERE $dist(q, p) \leq \epsilon$

Database Support for Density-Based Clustering

The relation "directly ϵ, *MinPts*-density reachable" may be expressed in terms of an ϵ-similarity self join (abbreviate *minPts* with μ):

$$\begin{aligned} ddr_{\epsilon,\mu} &= \{(q,p) \in D \times D \mid q \text{ is } \epsilon,\mu\text{-core-point} \wedge p \in N_\epsilon(q)\} \\ &= \{(q,p) \in D \times D \mid dist(q,p) \leq \epsilon \wedge \exists_{\geq \mu} p' \in D : dist(q,p') \leq \epsilon\} \\ &= \{(q,p) \in D \times D \mid (q,p) \in D \bowtie_\epsilon D \wedge \exists_{\geq \mu} p'(q,p') \in D \bowtie_\epsilon D\} \\ &= \sigma_{|\pi_q(D \bowtie_\epsilon D)| \geq \mu}(D \bowtie_\epsilon D) =: D \bowtie_{\epsilon,\mu} D \end{aligned}$$

SQL Query

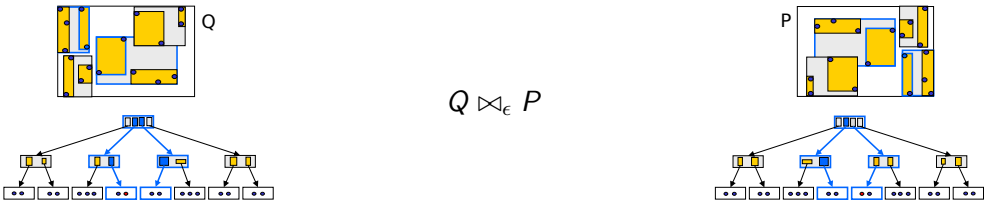
SELECT $*$ FROM D q, D p WHERE $dist(q,p) \leq \epsilon$ GROUP BY $q.id$ HAVING $count(q.id) \geq \mu$

Afterwards, DBSCAN computes the connected components of $D \bowtie_{\epsilon,\mu} D$.

Efficient Similarity Join Processing

For very large databases, efficient join techniques are available

- Block nested loop or index-based nested loop joins exploit secondary storage structure of large databases.
- Dedicated similarity join, distance join, or spatial join methods based on spatial indexing structures (e.g., R-Tree) apply particularly well. They may traverse their hierarchical directories in parallel (see illustration below).
- Other join techniques including sort-merge join or hash join are not applicable.



DBSCAN: Discussion

Advantages

- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Can separate clusters from surrounding noise
- Complexity: N_ϵ-query: $\mathcal{O}(n)$, DBSCAN: $\mathcal{O}(n^2)$.
- Can be supported by spatial index structures ($\rightsquigarrow$ N_ϵ-query: $\mathcal{O}(\log n)$)

Disadvantages

- Input parameters may be difficult to determine
- In some situations very sensitive to input parameter setting

Agenda

Iterative Mode Search

Idea

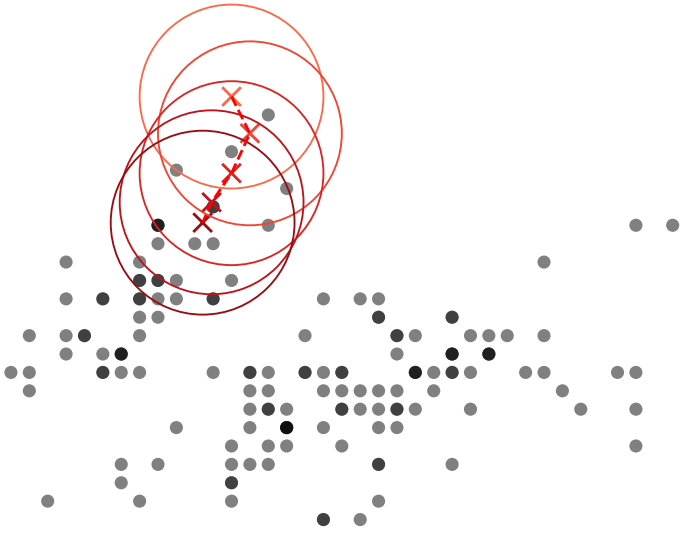
Find modes in the point density.

Algorithm[14]

1. Select a window size ϵ, starting position m
2. Calculate the mean of all points inside the window $W(m)$.
3. Shift the window to that position
4. Repeat until convergence.

[14] K. Fukunaga, L. Hostetler: *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition*, IEEE Trans Information Theory, 1975

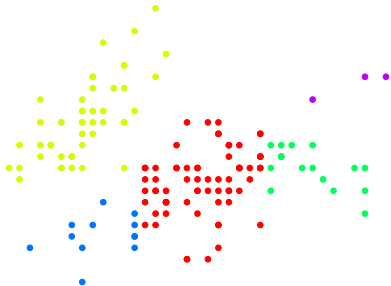
Iterative Mode Search: Example



Mean Shift: Core Algorithm

Algorithm[15]

Apply iterative mode search for each data point. Group those that converge to the same mode (called *Basin of Attraction*).



[15] D. Comaniciu, P. Meer. *Mean shift: A robust approach toward feature space analysis*. IEEE Trans. on pattern analysis and machine intelligence, 2002

Mean Shift: Extensions

Weighted Mean

Use different weights for the points in the window calculated by some kernel κ

$$m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} \kappa(x) x}{\sum_{x \in W(m^{(i)})} \kappa(x)}$$

Binning

First quantise data points to grid. Apply iterative mode seeking only once per bin.

Mean Shift: Discussion

Disadvantages

- Relatively high complexity: N_ϵ-query (=windowing): $\mathcal{O}(n)$. Algorithm: $\mathcal{O}(tn^2)$

Advantages

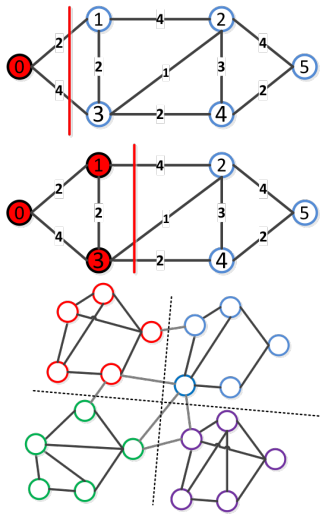
- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Robust to outliers
- Easy implementation and parallelisation
- Single parameter: ϵ
- Support by spatial index: N_ϵ-query (=windowing): $\mathcal{O}(\log n)$. Algorithm: $\mathcal{O}(tn \log n)$

Agenda

Clustering as Graph Partitioning

Approach

- Data is modeled by a similarity graph $G = (V, E)$
  - Vertices $v \in V$: Data objects
  - Weighted edges $\{v_i, v_j\} \in E$: Similarity of v_i and v_j
  - Common variants: ϵ-neighborhood graph, k-nearest neighbor graph, fully connected graph
- Cluster the data by partitioning the similarity graph
  - Idea: Find global minimum cut
    - Only considers inter-cluster edges, tends to cut small vertex sets from the graph
    - Partitions graph into two clusters
  - Instead, we want a *balanced multi-way partitioning*
  - Such problems are NP-hard, use approximations



Spectral Clustering

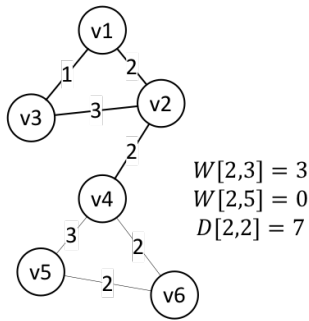
Given

Undirected graph G with weighted edges

- Let W be the (weighted) adjacency matrix of the graph
- And D its degree matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$; other entries are 0

Aim

Partition G into k subsets, minimizing a function of the edge weights between/within the partitions.



Spectral Clustering

Idea

- Consider the *indicator vector* f_C for the cluster C, i.e.

$$f_{C\,i} = \begin{cases} 1 & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$

  and the *Laplacian* matrix $L = D - W$
- Further, consider the function $fLf^T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(f_i - f_j)^2$ (derivation on next slide)
  - Small if f corresponds to a good partitioning
  - Given an indicator vector f_C, the function $f_C L f_C^T$ measures the weight of the inter-cluster edges!
  - Since L is positive semi-definite we have $fLf^T \geq 0$
  - Try to minimize fLf^T

Spectral Clustering

$$
\begin{aligned}
fLf^T &= fDf^T - fWf^T \\
&= \sum_i d_i f_i^2 - \sum_{ij} w_{ij} f_i f_j \\
&= \frac{1}{2}\left(\sum_i (\sum_j w_{ij}) f_i^2 - 2\sum_{ij} w_{ij} f_i f_j + \sum_j (\sum_i w_{ij}) f_j^2 \right) \\
&= \frac{1}{2}\left(\sum_{ij} w_{ij} f_i^2 - 2\sum_{ij} w_{ij} f_i f_j + \sum_{ij} w_{ij} f_j^2 \right) \\
&= \frac{1}{2} \sum_{ij} w_{ij} (f_i^2 - 2 f_i f_j + f_j^2) \\
&= \frac{1}{2} \sum_{ij} w_{ij} (f_i - f_j)^2
\end{aligned}
$$

Spectral Clustering: Example for Special Case

- Special case: The graph consists of k connected components (here: $k = 3$)
- The k components yield a "perfect" clustering (no edges between clusters), i.e. optimal clustering by indicator vectors $f_{C_1} = (1, 1, 1, 0, 0, 0, 0, 0, 0)$, $f_{C_2} = (0, 0, 0, 1, 1, 1, 0, 0, 0)$ and $f_{C_1} = (0, 0, 0, 0, 0, 0, 1, 1, 1)$

0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	0
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	0	1	1	0

Adjacency matrix W

2	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	2

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	2	-1	0	0	0	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	0	0	-1	3	-2	0	0	0
0	0	0	-1	-2	3	0	0	0
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	0	-1	-1	2

Laplacian matrix $L = D - W$

- Because of the block form of L, we get $f_C L f_C^T = 0$ for each component C

Connected Components and Eigenvectors

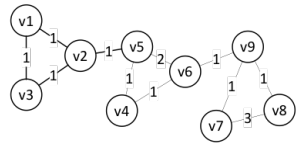
- General goal: find indicator vectors minimizing function fLf^T besides the trivial indicator vector $f_C = (1, \ldots, 1)$
- Problem: Finding solution is NP-hard (cf. graph cut problems)
- How can we relax the problem to find a (good) solution more efficiently?
- Observation: For the special case with k connected components, the k indicator vectors fulfilling $f_C L f_C^T = 0$ yield the perfect clustering
  - The indicator vector for each component is an eigenvector of L with eigenvalue 0
  - The k indicator vectors are orthogonal to each other (linearly independent)

Lemma

The number of linearly independent eigenvectors with eigenvalue 0 for L equals the number of connected components in the graph.

Spectral Clustering: General Case

- In general: L does not have zero-eigenvectors
  - One large connected component, no perfect clustering
  - Determine the (linear independent) eigenvectors with the *k smallest eigenvalues*!
- Example: The 3 clusters are now connected by additional edges



0	1	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	1	0	1	0	2	0	0	0
0	0	0	1	2	0	0	0	1
0	0	0	0	0	0	0	3	1
0	0	0	0	0	0	3	0	1
0	0	0	0	0	1	1	1	0

Adjacency matrix W

2	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	3

Degree matrix D

2	-1	-1	0	0	0	0	0	0
-1	3	-1	0	-1	0	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	-1	0	-1	4	-2	0	0	0
0	0	0	-1	-2	4	0	0	-1
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	-1	-1	-1	3

Laplacian matrix L

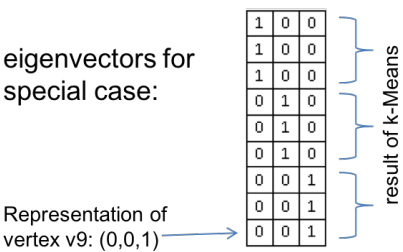
-1.3	3.3	0.4
-1	1	-1
-1.3	3.3	0.4
0	-6.6	0
-0.2	-4.3	-0.4
-0.2	-4.3	0.4
1.3	3.3	-0.4
1.3	3.3	-0.4
1	1	1

Eigenvectors of L

- Smallest eigenvalues of L: $(0.23, 0.70, 3.43)$

Spectral Clustering: Data Transformation

- How to find the clusters based on the eigenvectors?
  - Easy in special setting: 0-1 values; now: arbitrary real numbers
- Data transformation: Represent each vertex by a vector of its corresponding components in the eigenvectors
  - In the special case, the representations of vertices from the same connected component are equal, e.g. v_1, v_2, v_3 are transformed to $(1, 0, 0)$
  - In general case only *similar* eigenvector representations
- Clustering (e.g. k-Means) on transformed data points yields final result



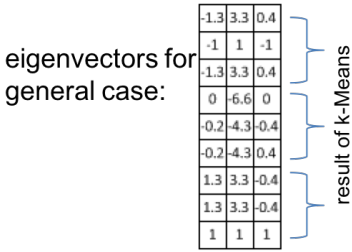
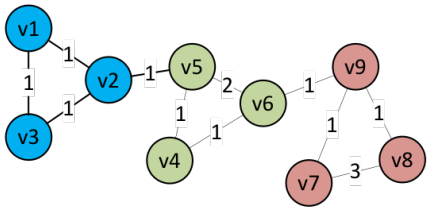
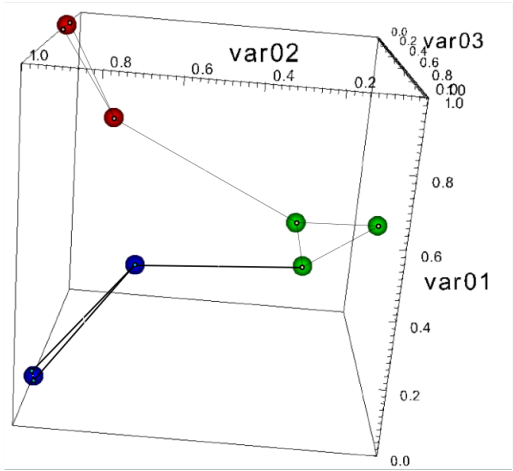


Illustration: Embedding of Vertices to a Vector Space

Spectral layout of previous example





Spectral Clustering: Discussion

Advantages

- No assumptions on the shape of the clusters
- Easy to implement

Disadvantages

- May be sensitive to construction of the similarity graph
- Runtime: k smallest eigenvectors can be computed in $\mathcal{O}(n^3)$ (worst case)
  - However: Much faster on sparse graphs, faster variants have been developed

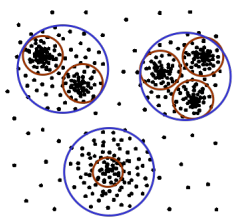
- Several variations of spectral clustering exist, using different Laplacian matrices which can be related to different graph cut problems [1]

[1] Von Luxburg, U.: A tutorial on spectral clustering, in Statistics and Computing, 2007

Agenda

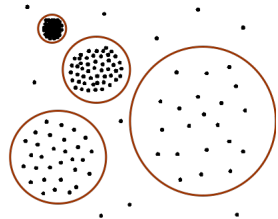
From Partitioning to Hierarchical Clustering

Global parameters to separate all clusters with a partitioning clustering method may not exist:



hierarchical cluster structure

and/or

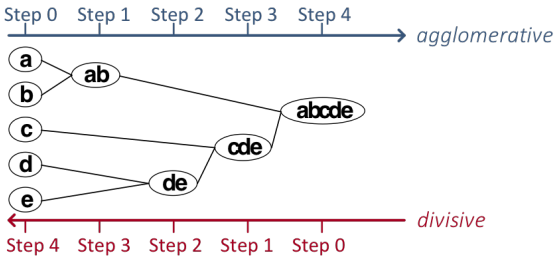


largely differing densities and sizes

Need a hierarchical clustering algorithm in these situations

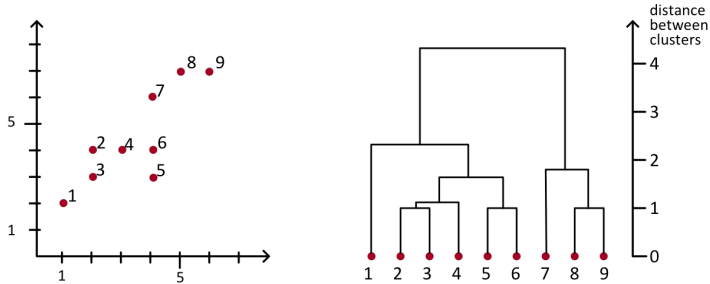
Hierarchical Clustering: Basic Notions

- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek $\delta\epsilon\nu\delta\rho o$ = tree)
  - Nodes in the dendrogram represent possible clusters
  - Dendrogram can be constructed bottom-up (agglomerative approach) or top down (divisive approach)



Hierarchical Clustering: Example

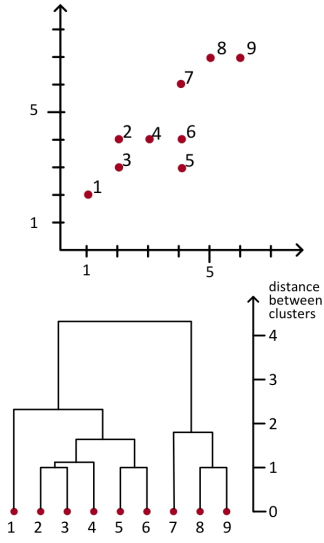
- Interpretation of the dendrogram
  - The root represents the whole data set
  - A leaf represents a single object in the data set
  - An internal node represents the union of all objects in its sub-tree
  - The height of an internal node represents the distance between its two child nodes



Agglomerative Hierarchical Clustering

Generic Algorithm

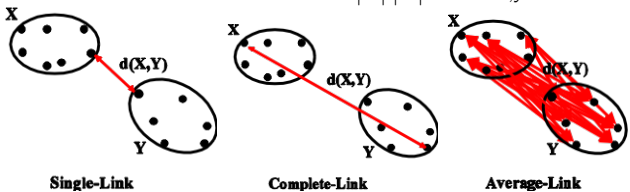
1. Initially, each object forms its own cluster
2. Consider all pairwise distances between the initial clusters (objects)
3. Merge the closest pair (A, B) in the set of the current clusters into a new cluster $C = A \cup B$
4. Remove A and B from the set of current clusters; insert C into the set of current clusters
5. If the set of current clusters contains only C (i.e., if C represents all objects from the database): STOP
6. Else: determine the distance between the new cluster C and all other clusters in the set of current clusters and go to step 3.



Single-Link Method and Variants

- Agglomerative hierarchical clustering requires a distance function for clusters
- Given: a distance function $dist(p, q)$ for database objects
- The following distance functions for clusters (i.e., sets of objects) X and Y are commonly used for hierarchical clustering:

$$\text{Single-Link:} \quad dist_{sl}(X, Y) = \min_{x \in X, y \in Y} dist(x, y)$$
$$\text{Complete-Link:} \quad dist_{cl}(X, Y) = \max_{x \in X, y \in Y} dist(x, y)$$
$$\text{Average-Link:} \quad dist_{al}(X, Y) = \frac{1}{|X| \cdot |Y|} \sum_{x \in X, y \in Y} dist(x, y)$$



Divisive Hierarchical Clustering

General Approach: Top Down

- Initially, all objects form one cluster
- Repeat until all clusters are singletons
  - Choose a cluster to split $\rightarrow$ *how?*
  - Replace the chosen cluster with the sub-clusters and split into two $\rightarrow$ *how to split?*

Example solution: DIANA

- Select the cluster C with largest diameter for splitting
- Search the most disparate object o in C (highest average dissimilarity)
  - Splinter group $S = \{o\}$
  - Iteratively assign the $o' \notin S$ with the highest $D(o') > 0$ to the splinter group until $D(o') \leq 0$ for all $o' \notin S$, where

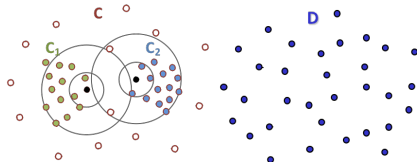
$$D(o') = \sum_{o_j \in C \setminus S} \frac{d(o', o_j)}{|C \setminus S|} - \sum_{o_i \in S} \frac{d(o', o_i)}{|S|}$$

Discussion Agglomerative vs. Divisive HC

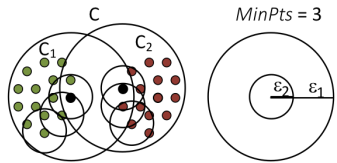
- Divisive and Agglomerative HC need $n-1$ steps
  - Agglomerative HC has to consider $\frac{n(n-1)}{2} = \binom{n}{2}$ combinations in the first step
  - Divisive HC potentially has $2^{n-1}-1$ many possibilities to split the data in its first step. Not every possibility has to be considered (DIANA)
- Divisive HC is conceptually more complex since it needs a second "flat" clustering algorithm (splitting procedure)
- Agglomerative HC decides based on local patterns
- Divisive HC uses complete information about the global data distribution $\rightsquigarrow$ able to provide better clusterings than Agglomerative HC?

Density-Based Hierarchical Clustering

- *Observation:* Dense clusters are completely contained by less dense clusters



- *Idea:* Process objects in the "right" order and keep track of point density in their neighborhood



Core Distance and Reachability Distance

Parameters: "generating" distance ϵ, fixed value *MinPts*

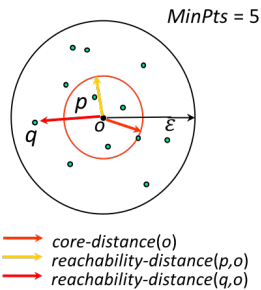
core-dist$_{\epsilon,MinPts}(o)$

- "smallest distance such that o is a core object"
- if core-dist $> \epsilon$: *undefined*

reach-dist$_{\epsilon,MinPts}(p, o)$

- "smallest dist. s.t. p is directly density-reachable from o"
- if reach-dist $> \epsilon$: ∞

$$\text{reach-dist}(p,o) = \begin{cases} dist(p,o) & , dist(p,o) \geq \text{core-dist}(o) \\ \text{core-dist}(o) & , dist(p,o) < \text{core-dist}(o) \\ \infty & , dist(p,o) > \epsilon \end{cases}$$

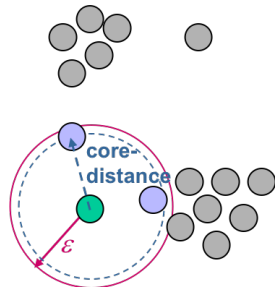


The Algorithm OPTICS

OPTICS[1]: Main Idea

"Ordering Points To Identify the Clustering Structure"

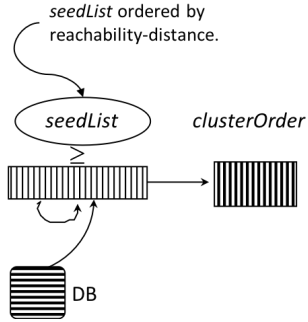
- Maintain two data structures
  - *seedList*: Stores all objects with shortest reachability distance seen so far ("distance of a jump to that point") in ascending order; organized as a heap
  - *clusterOrder*: Resulting cluster order is constructed sequentially (order of objects + reachability-distances)
- Visit each point
  - Always make a shortest jump



[1] Ankerst M., Breunig M., Kriegel H.-P., Sander J. "OPTICS: Ordering Points To Identify the Clustering Structure". SIGMOD (1999)

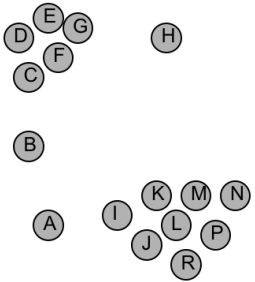
The Algorithm OPTICS

1: $seedList = \emptyset$
2: **while** there are unprocessed objects in DB **do**
3: **if** $seedList = \emptyset$ **then**
4: insert arbitrary unprocessed object into *clusterOrder* with reach-dist $= \infty$
5: **else**
6: remove first object from *seedList* and insert into *clusterOrder* with its current reach-dist
7: // *Let o be the last object inserted into clusterOrder*
8: mark o as processed
9: **for** $p \in range(o, \epsilon)$ **do**
10: // Insert/update p in *seedList*
11: compute reach-dist(p, o)
12: *seedList*.update$(p,$ reach-dist$(p, o))$



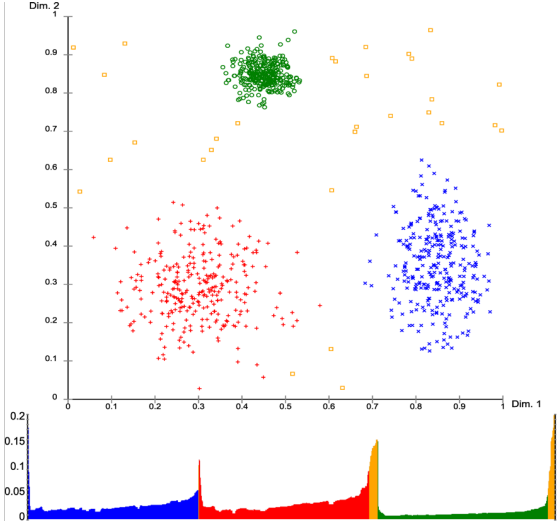
OPTICS: Example

$\epsilon = 44, MinPts = 3$



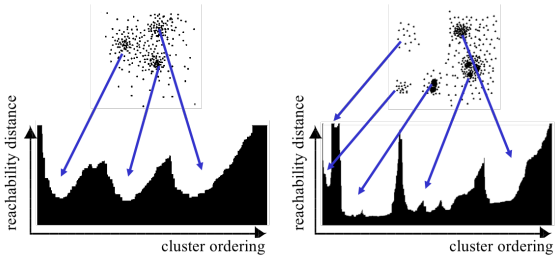
seed list:

OPTICS: The Reachability Plot



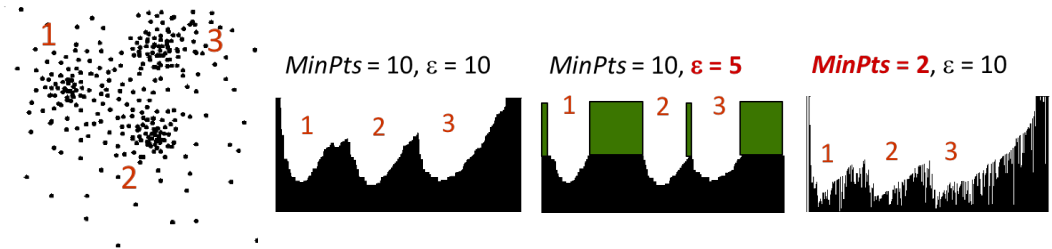
OPTICS: The Reachability Plot

- Plot the points together with their reachability-distances. Use the order in which they where returned by the algorithm
  - Represents the density-based clustering structure
  - Easy to analyze
  - Independent of the dimensionality of the data



OPTICS: Parameter Sensitivity

- Relatively insensitive to parameter settings
- Good result if parameters are just "large enough"



Hierarchical Clustering: Discussion

Advantages

- Does not require the number of clusters to be known in advance
- No (standard methods) or very robust parameters (OPTICS)
- Computes a complete hierarchy of clusters
- Good result visualizations integrated into the methods
- A "flat" partition can be derived afterwards (e.g. via a cut through the dendrogram or the reachability plot)

Disadvantages

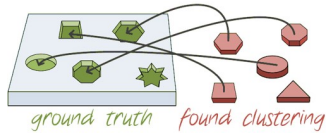
- May not scale well
  - Runtime for the standard methods: $\mathcal{O}(n^2 \log n^2)$
  - Runtime for OPTICS: without index support $\mathcal{O}(n^2)$
- User has to choose the final clustering

Agenda

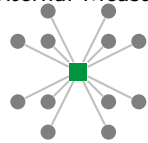
Evaluation of Clustering Results

Type	Positive	Negative
Expert's Opinion	may reveal new insight into the data	very expensive, results are not comparable
External Measures	objective evaluation	needs "ground truth"
Internal Measures	no additional information needed	approaches optimizing the evaluation criteria will always be preferred

Expert's Opinion



External Measure



Internal Measure

External Measures

Notation

Given a data set D, a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ and ground truth $\mathcal{G} = \{G_1, \ldots, G_l\}$.

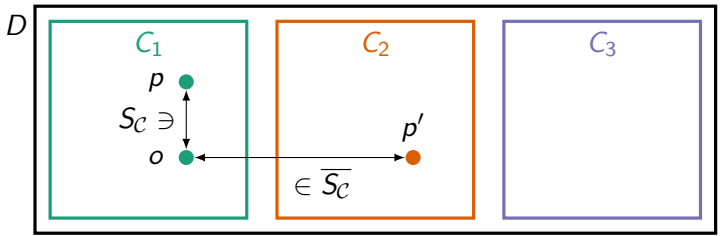
Problem

Since the cluster labels are "artificial", permuting them should not change the score.

Solution

Instead of comparing cluster and ground truth labels directly, consider all pairs of objects. Check whether they have the same label in $\mathcal{G}$ and if they have the same in $\mathcal{C}$.

Formalisation as Retrieval Problem



With $P = \{(o, p) \in D \times D \mid o \neq p\}$ define:

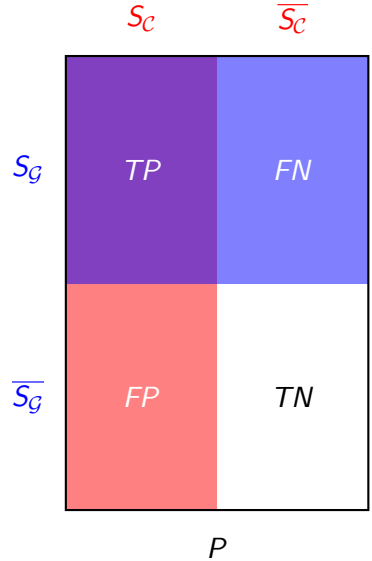
- *Same* cluster label: $S_C = \{(o, p) \in P \mid \exists C_i \in C : \{o, p\} \subseteq C_i\}$
- *Different* cluster label: $\overline{S_C} = P \setminus S_C$

and analogously for $\mathcal{G}$.

Formalisation as Retrieval Problem

Define

- $TP = |S_{\mathcal{C}} \cap S_{\mathcal{G}}|$
  (same cluster in both, "true positives")
- $FP = |S_{\mathcal{C}} \cap \overline{S_{\mathcal{G}}}|$
  (same cluster in $\mathcal{C}$, different cluster in $\mathcal{G}$, "false positives")
- $TN = |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|$
  (different cluster in both, "true negatives")
- $FN = |\overline{S_{\mathcal{C}}} \cap S_{\mathcal{G}}|$
  (different cluster in $\mathcal{C}$, same cluster in $\mathcal{G}$, "false negatives")



External Measures

- *Recall* ($0 \leq rec \leq 1$, larger is better)

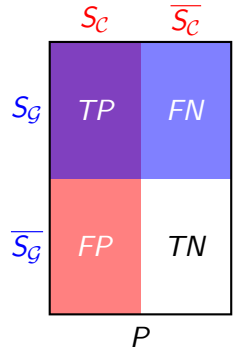
$$rec = \frac{TP}{TP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{G}}|}$$

- *Precision* ($0 \leq prec \leq 1$, larger is better)

$$prec = \frac{TP}{TP + FP} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{C}}|}$$

- F_1-*Measure* ($0 \leq F_1 \leq 1$, larger is better)

$$F_1 = \frac{2 \cdot rec \cdot prec}{rec + prec} = \frac{2|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|S_{\mathcal{C}}| + |S_{\mathcal{G}}|}$$



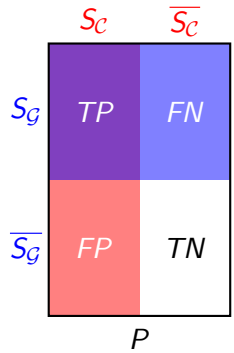
External Measures

- *Rand Index* ($0 \leq RI \leq 1$, larger is better):

$$RI(\mathcal{C} \mid \mathcal{G}) = \frac{TP + TN}{TP + TN + FP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}| + |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|}{|P|}$$

- *Adjusted Rand Index* (ARI): Compares $RI(\mathcal{C}, \mathcal{G})$ against expected $(\mathcal{R}, \mathcal{G})$ of random cluster assignment $\mathcal{R}$.
- *Jaccard Coefficient* ($0 \leq JC \leq 1$, larger is better):

$$JC = \frac{TP}{TP + FP + FN} = \frac{|S_{\mathcal{C}} \cap S_{\mathcal{G}}|}{|P| - |\overline{S_{\mathcal{C}}} \cap \overline{S_{\mathcal{G}}}|}$$



External Measures

- Confusion Matrix / Contingency Table $N \in \mathbb{N}^{k \times l}$ with $N_{ij} = |C_i \cap G_j|$

	G_1	$\dots$	G_l
C_1	$\lvert C_1 \cap G_1 \rvert$	$\dots$	$\lvert C_1 \cap G_l \rvert$
$\vdots$	$\vdots$	$\ddots$	
C_k	$\lvert C_k \cap G_1 \rvert$		$\lvert C_k \cap G_l \rvert$

- Define $N_i = \sum_{j=1}^{l} N_{ij}$ (i.e. $N_i = |C_i|$)
- Define $N = \sum_{i=1}^{k} N_i$ (i.e. $N = |D|$)

External Measures

- (Shannon) Entropy:

$$H(\mathcal{C}) = -\sum_{C_i \in \mathcal{C}} p(C_i) \log p(C_i) = -\sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|D|} \log \frac{|C_i|}{|D|} = -\sum_{i=1}^{k} \frac{N_i}{N} \log \frac{N_i}{N}$$

- Mutual Entropy:

$$\begin{aligned} H(\mathcal{C} \mid \mathcal{G}) &= -\sum_{C_i \in \mathcal{C}} p(C_i) \sum_{G_j \in \mathcal{G}} p(G_j \mid C_i) \log p(G_j \mid C_i) \\ &= -\sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|D|} \sum_{G_j \in \mathcal{G}} \frac{|C_i \cap G_j|}{|C_i|} \log \frac{|C_i \cap G_j|}{|C_i|} \\ &= -\sum_{i=1}^{k} \frac{N_i}{N} \sum_{j=1}^{l} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i} \end{aligned}$$

External Measures

- Mutual Information:

$$I(\mathcal{C}, \mathcal{G}) = H(\mathcal{C}) - H(\mathcal{C} \mid \mathcal{G}) = H(\mathcal{G}) - H(\mathcal{G} \mid \mathcal{C})$$

- Normalized Mutual Information (NMI) ($0 \leq NMI \leq 1$, larger is better):

$$NMI(\mathcal{C}, \mathcal{G}) = \frac{I(\mathcal{C}, \mathcal{G})}{\sqrt{H(\mathcal{C})H(\mathcal{G})}}$$

- Adjusted Mutual Information (AMI): Compares $MI(\mathcal{C}, \mathcal{G})$ against expected $MI(\mathcal{R}, \mathcal{G})$ of random cluster assignment $\mathcal{R}$.

Internal Measures: Cohesion

Notation

Let D be a set of size $n = |D|$, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a partitioning of D.

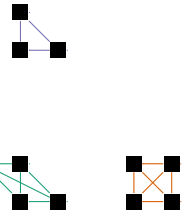
Cohesion

Average distance between objects of the same cluster.

$$coh(C_i) = \binom{|C_i|}{2}^{-1} \sum_{o,p \in C_i, o \neq p} d(o,p)$$

Cohesion of clustering is equal to weighted mean of the clusters' cohesions.

$$coh(\mathcal{C}) = \sum_{i=1}^{k} \frac{|C_i|}{n} coh(C_i)$$



Internal Measures: Separation

Separation

Separation between to clusters: Average distance between pairs

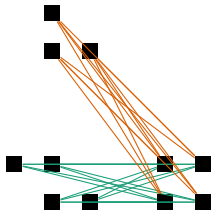
$$sep(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{o \in C_i, p \in C_j} d(o, p)$$

Separation of one cluster: Minimum separation to another cluster:

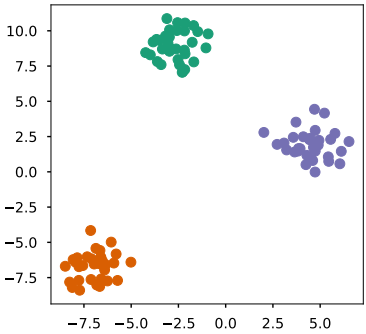
$$sep(C_i) = \min_{j \neq i} sep(C_i, C_j)$$

Separation of clustering is equal to weighted mean of the clusters' separations.

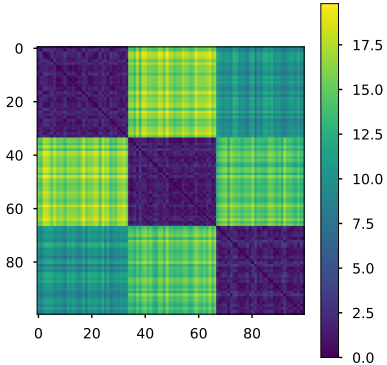
$$sep(\mathcal{C}) = \sum_{i=1}^{k} \frac{|C_i|}{n} sep(C_i)$$



Evaluating the Distance Matrix



dataset
(well separated)

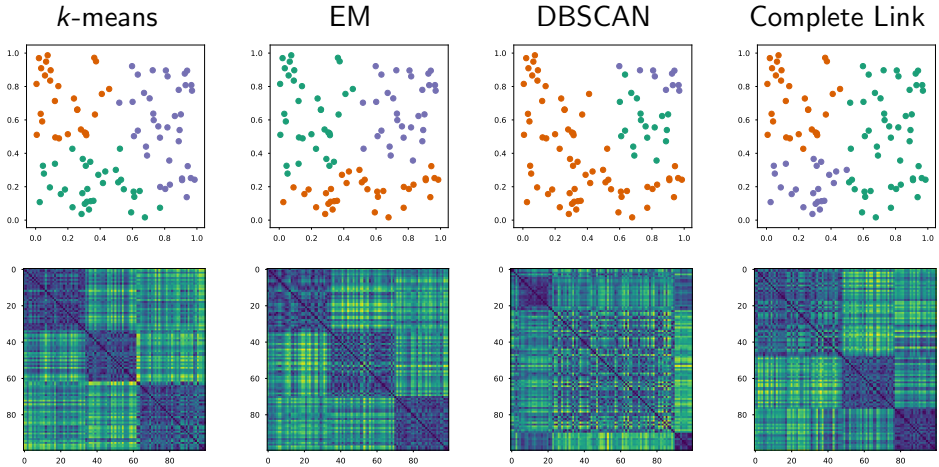


Distance matrix
(sorted by k-means cluster label)

after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Evaluating the Distance Matrix

Distance matrices differ for different clustering approaches (here on random data)

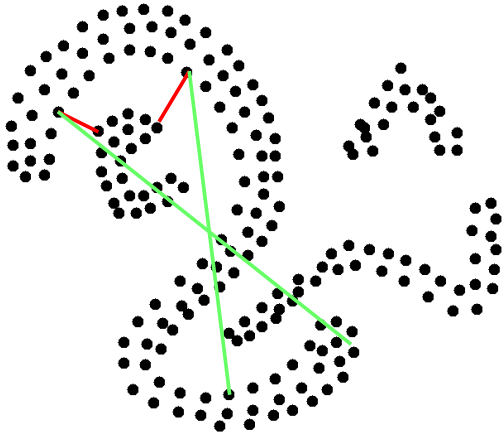


after: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

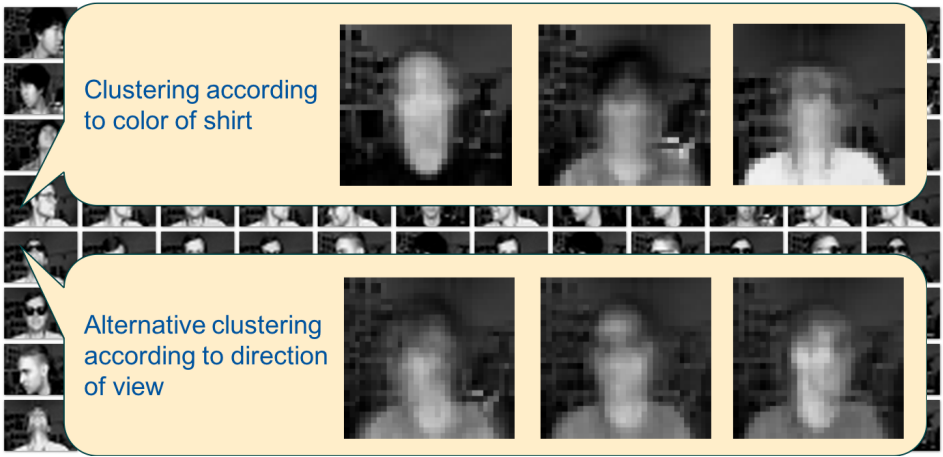
Cohesion and Separation

Problem

Suitable for convex cluster, but not for stretched clusters (cf. silhouette coefficient).



Ambiguity of Clusterings



- Clustering according to: Color of shirt, direction of view, glasses, ...

Ambiguity of Clusterings

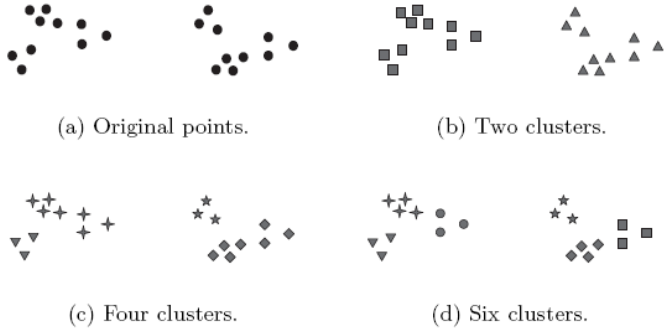


Figure 8.1. Different ways of clustering the same set of points.

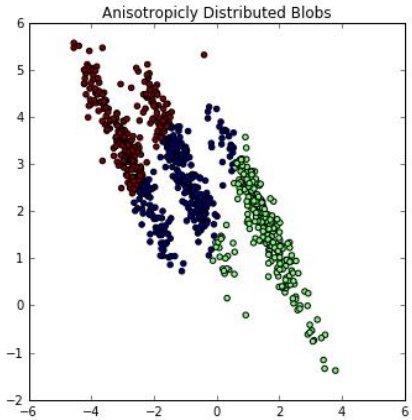
from: Tan, Steinbach, Kumar: Introduction to Data Mining (Pearson, 2006)

Ambiguity of Clusterings

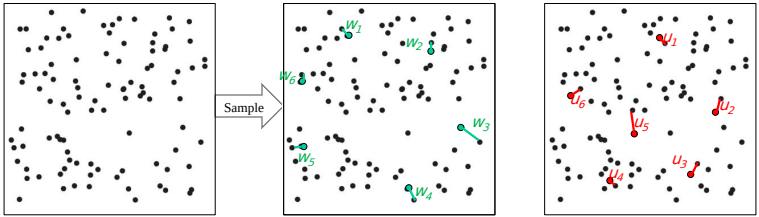
"Philosophical" Problem

"What is a correct clustering?"

- Most approaches find clusters in every dataset, even in uniformly distributed objects
- Are there clusters?
  - Apply clustering algorithm
  - Check for reasonability of clusters
- Problem: No clusters found $\neq$ no clusters existing
  - Maybe clusters exists only in certain models, but can not be found by used clustering approach



Hopkins Statistics

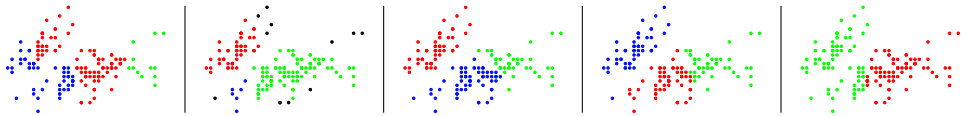


$$H = \frac{\sum_{i=1}^{m} u_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

- w_i: distance of selected objects to the next neighbor in dataset
- u_i: distances of uniformly distributed objects to next neighbor in dataset
- $0 \leq H \leq 1$;
  - $H \approx 0$: very regular data (e.g. grid);
  - $H \approx 0.5$: uniformly distributed data;
  - $H \approx 1$: strongly clusteredc

Agenda

Ensemble Clustering



Problem

- Many differing clustering models
- Different parameter choices, usually highly influences the result

What is a "good" clustering?

Idea

Find a consensus solution (also ensemble clustering) that consolidates multiple clustering solutions.

Ensemble Clustering: Benefits

- *Knowledge Reuse*: Possibility to integrate the knowledge of multiple known, good clusterings
- *Improved Quality*: Often ensemble clustering leads to "better" results than its individual base solutions.
- *Improved Robustness*: Combining several clustering approaches with differing data modeling assumptions leads to an increased robustness across a wide range of datasets.
- *Model Selection*: Novel approach for determining the final number of clusters
- *Distributed Clustering*: if data is inherently distributed (either feature-wise or object-wise) and each clusterer has only access to a subset of objects and/or features, ensemble methods can be used to compute a unifying result

Ensemble Clustering: Basic Notions

Given

A set of L clusterings $\mathfrak{C} = \mathcal{C}_1, \ldots, \mathcal{C}_L$ for dataset $D = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$.

Goal

Find a consensus clustering $\mathcal{C}^*$.

How to define a consensus clustering?

Two categories:

- Approaches based on pairwise similarity: Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ (ϕ is basically an external measure)
- Probabilistic approaches: Assume that the L labels for the objects $x_i \in D$ follow a certain distribution.

Similarity-Based Approaches

Goal

Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ is maximal.

Choices for ϕ

- Pair counting-based measures: Rand Index (RI), Adjusted RI, Probabilistic RI
- Information theoretic measures: Mutual Information (I), Normalized Mutual Information (NMI), Variation of Information (VI)

Problem

Minimising the objective for the above mentioned choices of ϕ in intractable.

Similarity-Based Approaches

Solutions

- Methods based on the co-association matrix (related to RI)
- Methods using cluster labels without co-association matrix (often related to NMI)
  - Mostly graph partitioning
  - Cumulative voting

Ensemble Clustering: Co-Association Matrix

Co-Association Matrix

The *co-association matrix* $S^{\mathfrak{C}} \in \mathbb{R}^{n \times n}$ represents the label similarity of object pairs:

$$S_{ij}^{\mathfrak{C}} = \sum_{\mathcal{C} \in \mathfrak{C}} \mathbb{I}[x_i \in \mathcal{C} \wedge x_j \in \mathcal{C}]$$

where $\mathbb{I}$ is the indicator function with $\mathbb{I}[\mathit{False}] = 0$, and $\mathbb{I}[\mathit{True}] = 1$.

Example

$D = \{1, 2, 3, 4, 5\}$ (i.e. $n = 5$),
$\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_2\}$,
$\mathcal{C}_1 = \{\{1, 2, 3\}, \{4, 5\}\}$,
$\mathcal{C}_2 = \{\{1, 2\}, \{3, 4, 5\}\}$.

$$S = \begin{pmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 2 & 2 \end{pmatrix}$$

Ensemble Clustering: Co-Association Matrix

Usage of Co-Association Matrix

- Use $S^{\mathfrak{C}}$ as similarity matrix to apply traditional clustering approach.
- By interpreting $S^{\mathfrak{C}}$ as weighted adjacency matrix, graph partitioning methods can be applied.

Co-Association Matrix and Rand Index

In [16] a connection of consensus clustering based on the co-association matrix and the optimization of the pairwise similarity based on the Rand Index has been proven:

$$C_{best} = \underset{\mathcal{C}^*}{\operatorname{argmax}} \sum_{\mathcal{C} \in \mathfrak{C}} RI(\mathcal{C}, \mathcal{C}^*)$$

[16] B. Mirkin: *Mathematical Classification and Clustering*. Kluwer, 1996.

Information-Theoretic Approaches

Setting

Find a consensus clustering $\mathcal{C}^*$ for which the similarity function $\Phi(\mathfrak{C}, \mathcal{C}^*) = \sum_{\mathcal{C} \in \mathfrak{C}} \phi(\mathcal{C}, \mathcal{C}^*)$ is maximal, with ϕ chosen as (Normalised) Mutual Information.

Problem

Usually a hard optimization problem!

Solution 1

Use meaningful optimization approaches (e.g. gradient descent) or heuristics to approximate the best clustering solution (e.g. [17])

[17] A. Strehl, J. Ghosh: *Cluster ensembles - a knowledge reuse framework for combining multiple partitions.* Journal of Machine Learning Research, 3, 2002, pp. 583-617.

Information-Theoretic Approaches

Solution 2

- Use a similar but solvable objective, e.g. [18]:
- Use as objective

$$C_{best} = \underset{\mathcal{C}^*}{\operatorname{argmax}} \sum_{\mathcal{C} \in \mathfrak{C}} I^s(\mathcal{C}, \mathcal{C}^*)$$

where I^s is the mutual information based on the generalized entropy of degree s:

$$H^s(X) = (2^{1-s} - 1)^{-1} \sum_{x_i \in X} (p_i^s - 1)$$

For $s = 2$, $I^s(\mathcal{C}, \mathcal{C}^*)$ is equal to the category utility function whose maximization is proven to be equivalent to the minimization of the square-error clustering criterion. $\implies$ Apply a simple label transformation and use e.g. K-Means

[18] A. Topchy, A.K. Jain, W. Punch. *Combining multiple weak clusterings*. In ICDM, pages 331-339, 2003

Probabilistic Approach

Assumptions

- All clusterings $\mathcal{C} \in \mathfrak{C}$ are partitionings of the dataset D.
- There are K^* consensus clusters.
- With $\mathcal{C}(x)$ denoting the cluster label assigned to x in clustering $\mathcal{C}$, the following dataset Y given by

$$Y = \{y_i \in \mathbb{N}_0^L \mid x_i \in D, \forall 1 \leq j \leq L : (y_i)_j = \mathcal{C}_i(x_i)\}$$

(labels of base clusterings) follows a multivariate mixture distribution:

$$p(Y \mid \Theta) = \prod_{i=1}^{n} \sum_{k=1}^{K^*} \alpha_k p_k(y_i \mid \theta_k) \overset{cond.ind.}{=} \prod_{i=1}^{n} \sum_{k=1}^{K^*} \alpha_k \prod_{j=1}^{L} p_{kl}(y_{ij} \mid \theta_{kl})$$

with $p_{kl}(y_{ij} \mid \theta_{kl}) \sim M(1, (p_{kl1}, \ldots, p_{kl|\mathcal{C}_l|}))$, i.e. $p_{kl}(y_{ij} \mid \theta_{kl}) = \prod_{k'=1}^{|\mathcal{C}_l|} p_{klk'}^{\mathbb{I}(y_{nl}=k')}$

Probabilistic Approach

Goal

Find the parameters $\Theta = (\alpha_1, \theta_1, \ldots, \alpha_{K*}, \theta_{K*})$ such that the likelihood $p(Y \mid \Theta)$ is maximized.

Solution [19]

Optimize the parameters via the EM approach

[19] Topchy, Jain, Punch: *A mixture model for clustering ensembles*. In ICDM, pp. 379-390, 2004.

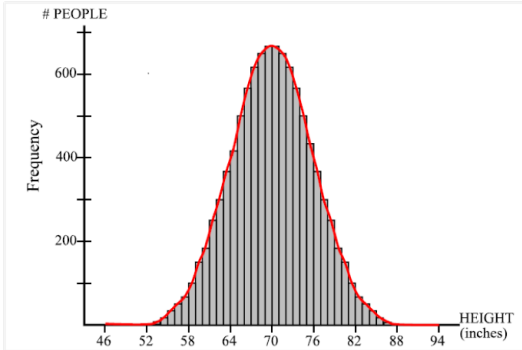
Agenda

Introduction

What is an outlier?

Hawkins (1980) "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

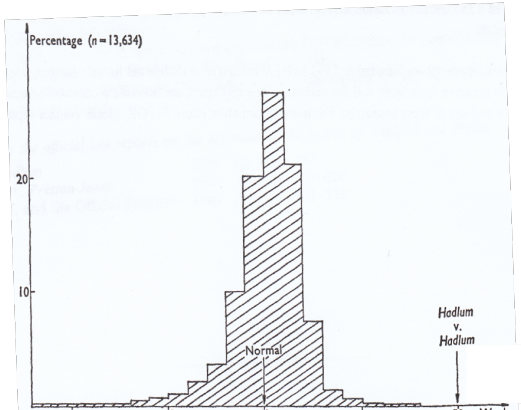
- Statistics-based intuition:
  - Normal data objects follow a "generating mechanism", e.g. some given statistical process
  - Abnormal objects deviate from this generating mechanism



Introduction

Example: Hadlum vs. Hadlum (1949) [Barnett 1978]

- The birth of a child to Mrs. Hadlum happened 349 days after Mr. Hadlum left for military service.
- Average human gestation period is 280 days (40 weeks).
- Statistically, 349 days is an outlier.

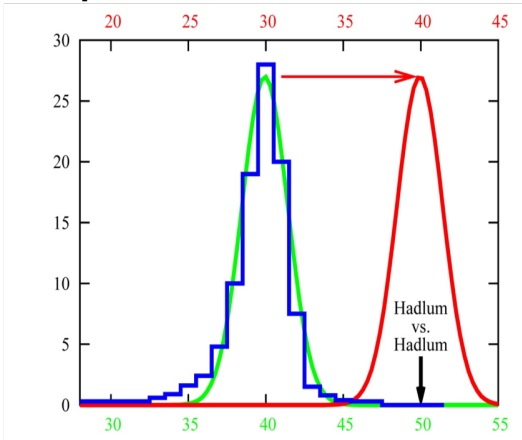


Distribution of human gestation periods.

Introduction

Example: Hadlum vs. Hadlum (1949) [Barnett 1978]

- Blue: statistical basis (13634 observations of gestation periods)
- Green: assumed underlying Gaussian process
  - Very low probability for the birth of Mrs. Hadlums child being generated by this process
- Red: assumption of Mr. Hadlum (another Gaussian process responsible for the observed birth, where the gestation period starts later)



Introduction

Applications

- Fraud detection
  - Purchasing behavior of a credit card owner usually changes when the card is stolen
  - Abnormal buying patterns can characterize credit card abuse
- Medicine
  - Whether a particular test result is abnormal may depend on other characteristics of the patients (e.g. gender, age, ...)
  - Unusual symptoms or test results may indicate potential health problems of a patient
- Public health
  - The occurrence of a particular disease, e.g. tetanus, scattered across various hospitals of a city indicate problems with the corresponding vaccination program in that city
  - Whether an occurrence is abnormal depends on different aspects like frequency, spatial correlation, etc.

Applications (cont'd)

- Sports statistics
  - In many sports, various parameters are recorded for players in order to evaluate the players' performances
  - Outstanding (in a positive as well as a negative sense) players may be identified as having abnormal parameter values
  - Sometimes, players show abnormal values only on a subset or a special combination of the recorded parameters
- Detecting measurement errors
  - Data derived from sensors (e.g. in a given scientific experiment) may contain measurement errors
  - Abnormal values could provide an indication of a measurement error
  - Removing such errors can be important in other data mining and data analysis tasks
  - *"One person's noise could be another person's signal."*

Introduction

Important Properties of Outlier Models

- Global vs. local approach
  - "Outlierness" regarding whole dataset (global) or regarding a subset of data (local)?
- Labeling vs. Scoring
  - Binary decision or outlier degree score?
- Assumptions about "Outlierness"
  - What are the characteristics of an outlier object?

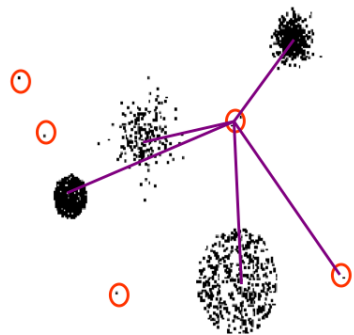
Agenda

Clustering-based Outliers

An object is a cluster-based outlier if it does not strongly belong to any cluster.

Basic Idea

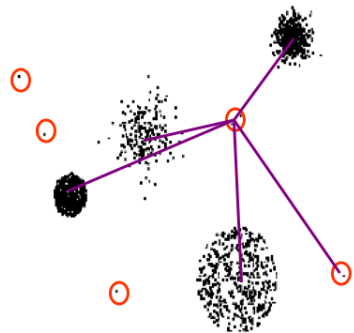
- Cluster the data into groups
- Choose points in small clusters as candidate outliers.
- Compute the distance between candidate points and non-candidate clusters.
- If candidate points are far from all other non-candidate points and clusters, they are outliers



Clustering-based Outliers

More Systematic Approaches

- Find clusters and then assess the degree to which a point belongs to any cluster
  - E.g. for k-Means, use distance to the centroid
- If eliminating a point results in substantial improvement of the objective function, we could classify it as an outlier
  - Clustering creates a model of the data and the outliers distort that model.
  - Applicable to clustering algorithms optimizing some objective function (e.g. k-means)

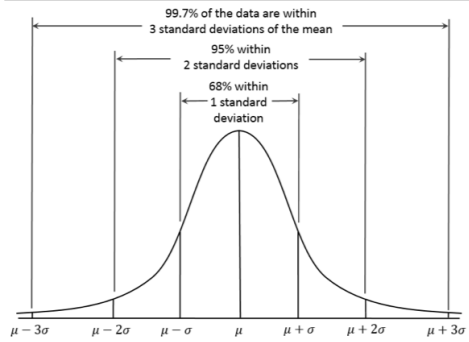


Agenda

Statistical Tests

General Idea

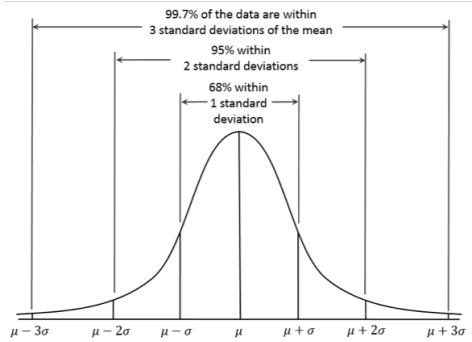
- Given a certain kind of statistical distribution (e.g., Gaussian)
- Compute the parameters assuming all data points have been generated by such a statistical distribution (e.g., mean and standard deviation)
- Outliers are points that have a low probability to be generated by the overall distribution (e.g., deviate more than 3 times the standard deviation from the mean)



Statistical Tests

Basic Assumption

- Normal data objects follow a (known) distribution and occur in a high probability region of this model
- Outliers deviate strongly from this distribution



Statistical Tests

A huge number of different tests are available differing in

- Type of data distribution (e.g. Gaussian)
- Number of variables, i.e., dimensions of the data objects (univariate/multivariate)
- Number of distributions (mixture models)
- Parametric versus non-parametric (e.g. histogram-based)

Example on the Following Slides

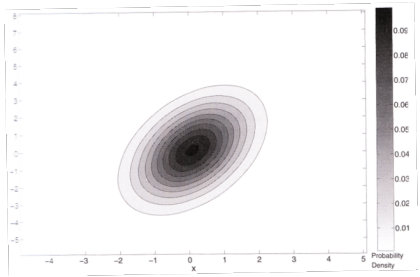
- Gaussian distribution
- Multivariate
- Single model
- Parametric

Statistical Outliers: Gaussian Distribution

Probability Density Function of a Multivariate Normal Distribution

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

- μ is the mean value of all points (usually data are normalized such that $\mu = 0$)
- Σ is the covariance matrix from the mean



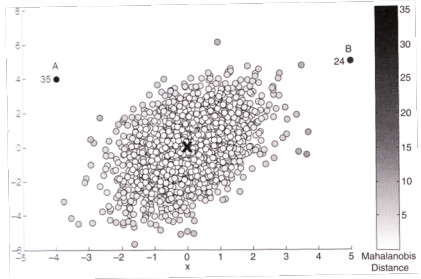
Statistical Outliers: Mahalanobis Distance

Mahalanobis Distance

Mahalanobis distance of point x to μ :

$$MDist(x, \mu) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

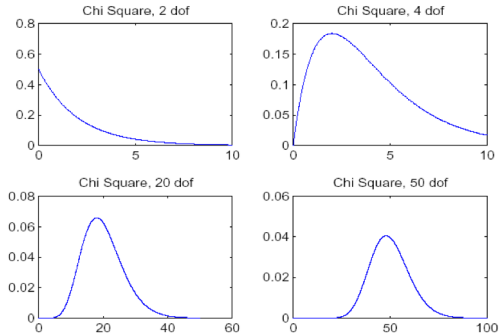
- *MDist* follows a χ^2-distribution with d degrees of freedom (d = data dimensionality)
- Outliers: All points x, with $MDist(x, \mu) > \chi^2(0.975)$ ($\approx 3\sigma$)



Statistical Outliers: Problems

Problems

- Curse of dimensionality: The larger the degree of freedom, the more similar the *MDist* values for all points

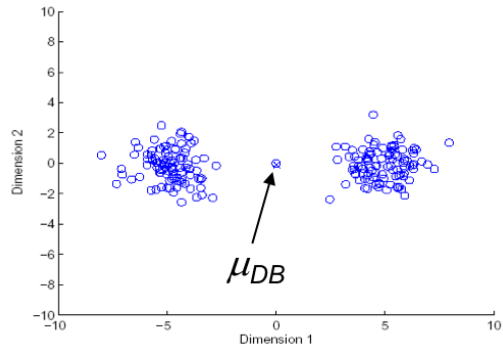


- x-axis = observed *MDist* values
- y-axis = frequency of observation

Statistical Outliers: Problems

Problems (cont'd)

- Robustness
  - Mean and standard deviation are very sensitive to outliers
  - These values are computed for the complete data set (including potential outliers)
  - The *MDist* is used to determine outliers although the *MDist* values are influenced by these outliers



Statistical Outliers: Problems

Problems (cont'd)

- Data distribution is fixed
- Low flexibility (if no mixture models)
- Global method

Agenda

Distance-Based Approaches

General Idea

Judge a point based on the distance(s) to its neighbors (Several variants proposed)

Basic Assumption

- Normal data objects have a dense neighborhood
- Outliers are far apart from their neighbors, i.e., have a less dense neighborhood

Distance-Based Approaches

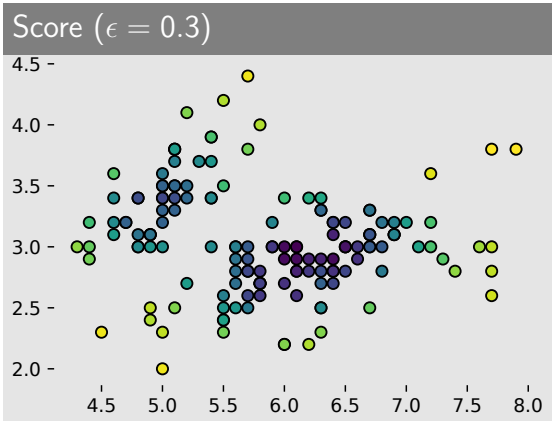
D(ϵ, π) Outliers [20]

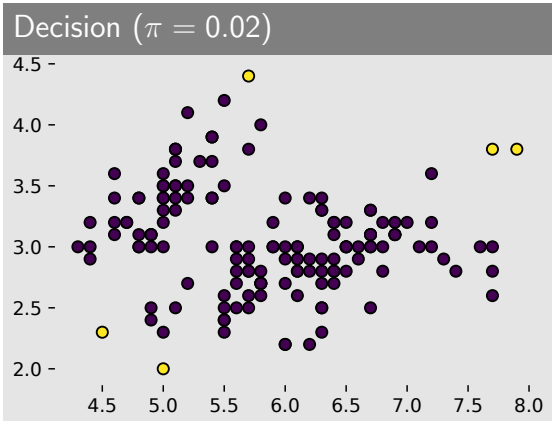
- Given: radius ϵ, percentage π
- A point p is considered an outlier if at most π percent of all points (including p) have a distance to p less than ϵ.

$$OutlierSet(\epsilon, \pi) = \left\{ p \in D \;\middle|\; \frac{|\{q \in D \mid dist(p,q) < \epsilon\}|}{|D|} \leq \pi \right\}$$

[20] E. Knorr, R. Ng. *A Unified Notion of Outliers: Properties and Computation.* KDD'97

Distance-Based Approaches: $D(\epsilon, \pi)$ Example





Distance-Based Approaches: kNN

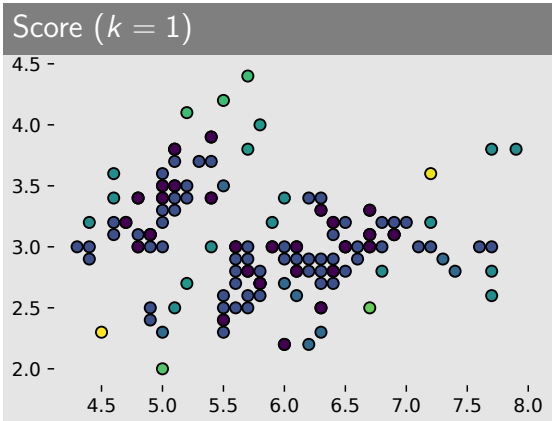
Outlier scoring based on kNN distances

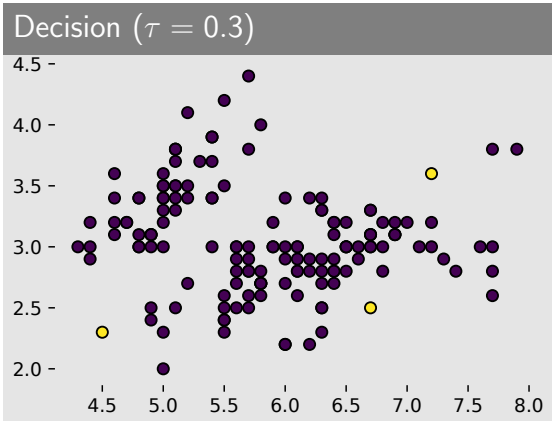
General models: Take the kNN distance of a point as its outlier score

Decision

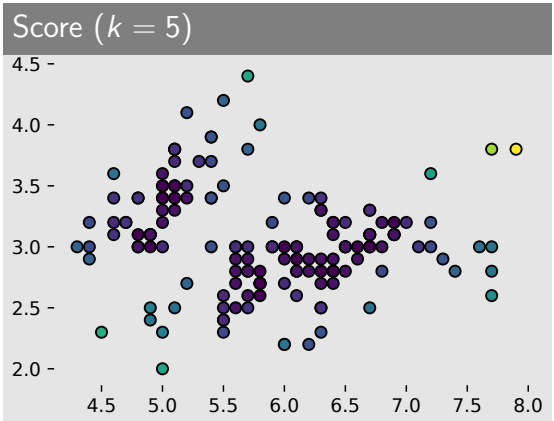
k-distance above some threshold $\tau \iff$ Outlier.

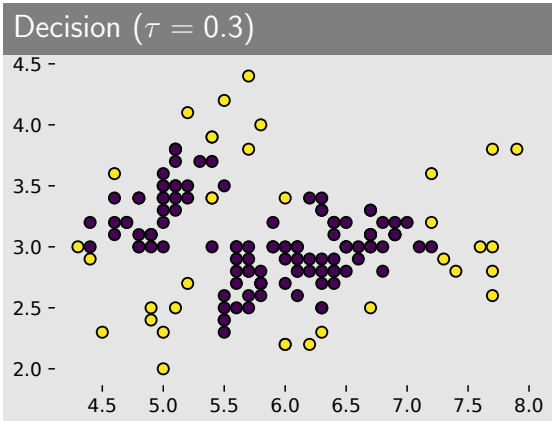
Distance-Based Approaches: kNN Example





Distance-Based Approaches: kNN Example





kNN: Problems

Problems

- Highly sensitive towards k:
  - Too small k: small number of close neighbors can cause low outlier scores.
  - Too large: all objects in a cluster with less than k objects might become outliers.
- cannot handle datasets with regions of widely different densities due to the global threshold

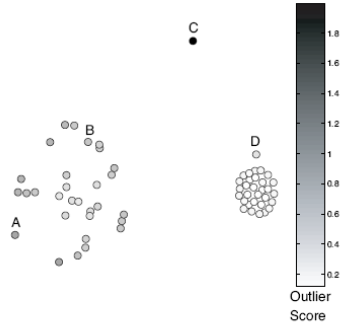


Figure 10.7. Outlier score based on the distance to the fifth nearest neighbor. Clusters of differing density.

Image Source: P. Tan, M. Steinbach, V. Kumar (2006). *Classification: basic concepts, decision trees, and model evaluation. Introduction to data mining*, 1, 145-205.

Agenda

Density-Based Approaches

General Idea

- Compare the density around a point with the density around its local neighbors.
- The relative density of a point compared to its neighbors is computed as an outlier score.
- Approaches also differ in how to estimate density.

Basic Assumption

- The density around a normal data object is similar to the density around its neighbors.
- The density around an outlier is considerably different to the density around its neighbors.

Density-Based Approaches

Problems

- Different definitions of density: e.g., #points within a specified distance ϵ from the given object
- The choice of ϵ is critical (too small $\Longrightarrow$ normal points considered as outliers; too big $\Longrightarrow$ outliers considered normal)
- A global notion of density is problematic (as it is in clustering); fails when data contain regions of different densities

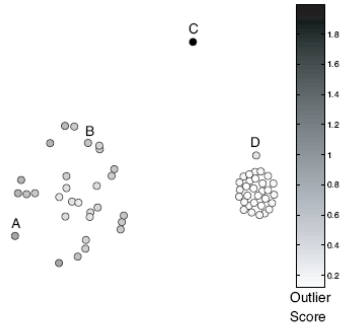


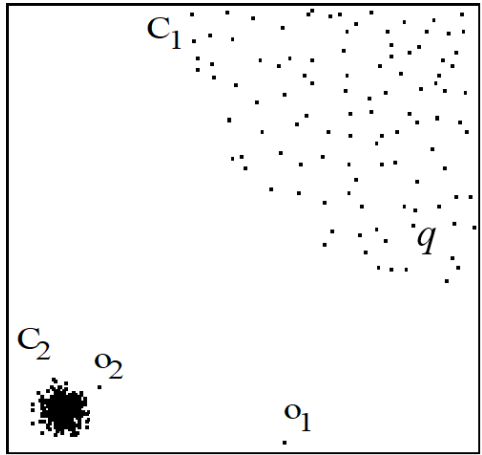
Figure 10.7. Outlier score based on the distance to the fifth nearest neighbor. Clusters of differing density.

D has a higher absolute density than A but compared to its neighborhood, Ds density is lower.

Density-Based Approaches

Failure Case of Distance-Based

- $D(\epsilon, \pi)$: parameters ϵ, π cannot be chosen s.t. o_2 is outlier, but none of the points in C_1 (e.g. q)
- kNN-distance: kNN-distance of objects in C_1 (e.g. q) larger than the kNN-distance of o_2.



Density-Based Approaches

Solution

Consider the relative density w.r.t. to the neighbourhood.

Model

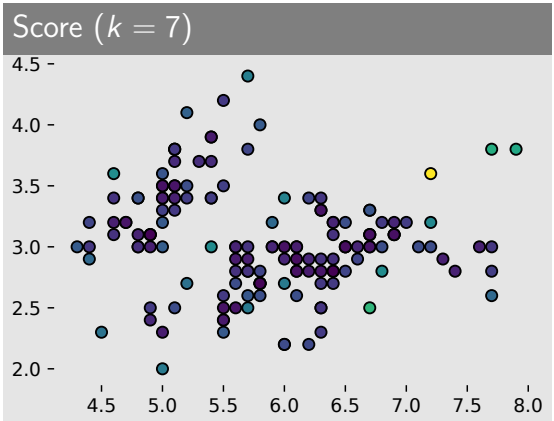
- Local Density (ld) of point p (inverse of avg. distance of kNNs of p)

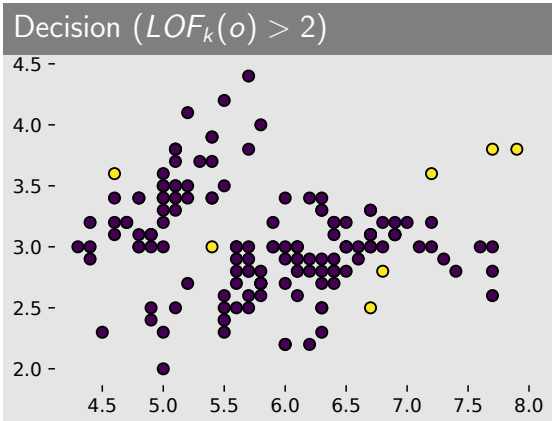
$$ld_k(p) = \left(\frac{1}{k} \sum_{o \in kNN(p)} dist(p, o) \right)^{-1}$$

- Local Outlier Factor (LOF) of p (avg. ratio of lds of kNNs of p and ld of p)

$$LOF_k(p) = \frac{1}{k} \sum_{o \in kNN(p)} \frac{ld_k(o)}{ld_k(p)}$$

Density-Based Approaches





Density-Based Approaches

Extension (Smoothing factor)

- Reachability "distance"

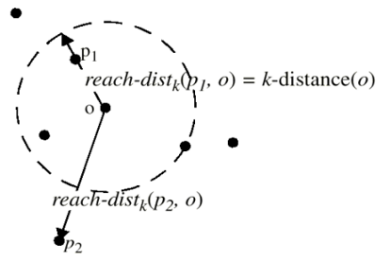
$$rd_k(p, o) = \max\{kdist(o), dist(p, o)\}$$

- Local reachability distance lrd_k

$$lrd_k(p) = \left(\frac{1}{k} \sum_{o \in kNN(p)} rd(p, o) \right)^{-1}$$

- Replace ld by lrd

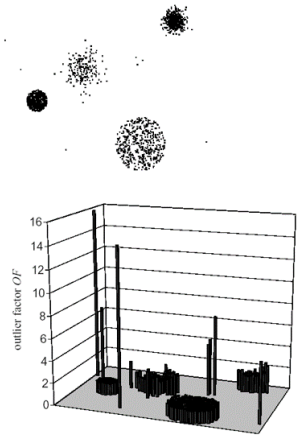
$$LOF_k(p) = \frac{1}{k} \sum_{o \in kNN(p)} \frac{lrd_k(o)}{lrd_k(p)}$$



Density-Based Approaches

Discussion

- $LOF \approx 1 \implies$ point in cluster
- $LOF \gg 1 \implies$ outlier.
- Choice of k defines the reference set

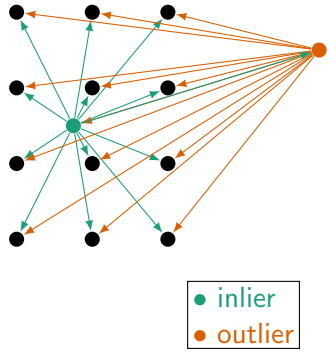


Agenda

Angle-Based Approach

General Idea

- Angles are more stable than distances in high dimensional spaces
- *o outlier* if most other objects are located in similar directions
- *o no outlier* if many other objects are located in varying directions



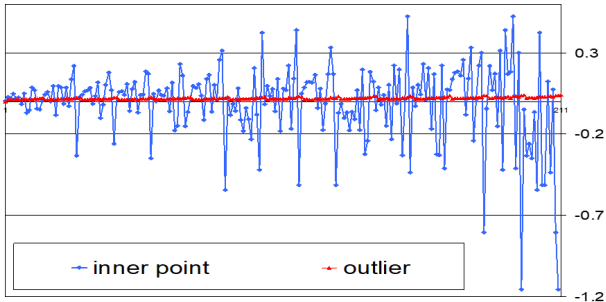
Basic Assumption

- Outliers are at the border of the data distribution
- Normal points are in the center of the data distribution

Angle-Based Approach

Model

- Consider for a given point p the angle between $\overrightarrow{px}$ and $\overrightarrow{py}$ for any two x, y from the database
- Measure the variance of the angle spectrum



Angle-Based Approach

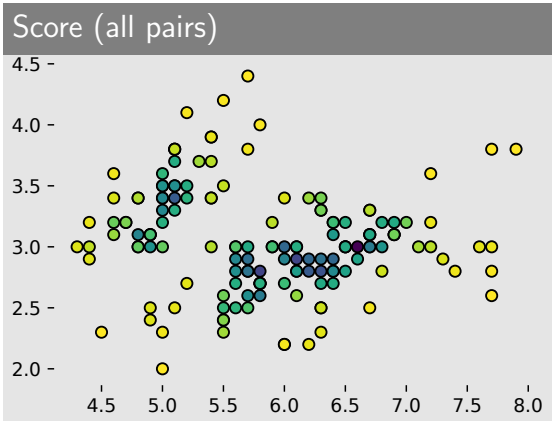
Model (cont'd)

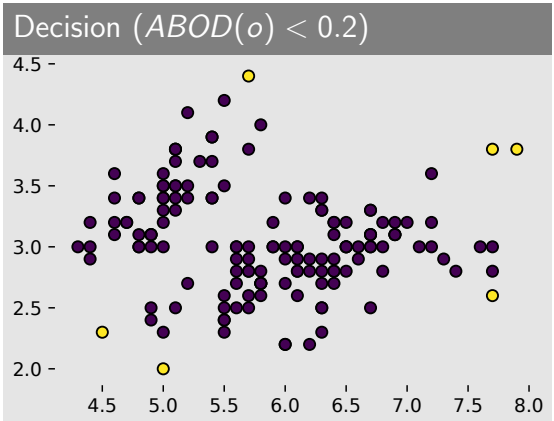
- Weighted by the corresponding distances (for lower dimensional data sets where angles are less reliable)

$$ABOD(p) = \mathrm{VAR}_{x,y \in D}\left(\frac{1}{\|\overrightarrow{xp}\|_2 \|\overrightarrow{yp}\|_2} \cos\left(\overrightarrow{xp}, \overrightarrow{yp}\right)\right) = \mathrm{VAR}_{x,y \in D}\left(\frac{\langle \overrightarrow{xp}, \overrightarrow{yp} \rangle}{\|\overrightarrow{xp}\|_2^2 \|\overrightarrow{yp}\|_2^2}\right)$$

- Small ABOD $\iff$ outlier

Angle-Based Approaches





Agenda

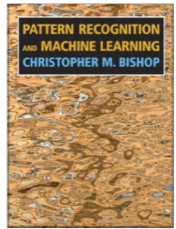
Summary

- Properties: global vs. local, labeling vs. scoring
- *Clustering-Based* Outliers: Identification as non-(cluster-members)
- *Statistical* Outliers: Assume probability distribution; outliers = unlikely to be generated by distribution
- *Distance-Based* Outliers: Distance to neighbors as outlier metric
- *Density-Based* Outliers: Relative density around the point as outlier metric
- *Angle-Based* Outliers: Angles between outliers and random point pairs vary only slightly

Agenda

Additional Literature for this Chapter

Christopher M. Bishop: *Pattern Recognition and Machine Learning*. Springer, Berlin 2006.



Introduction: Example

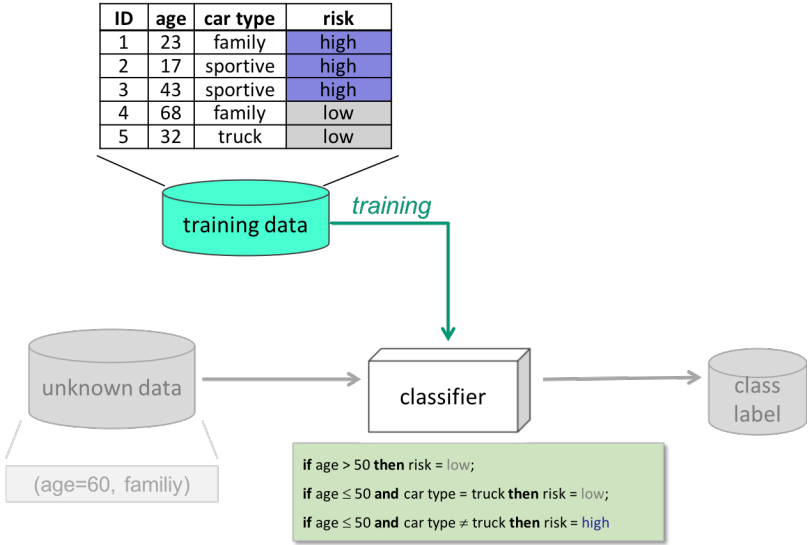
- Training data

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

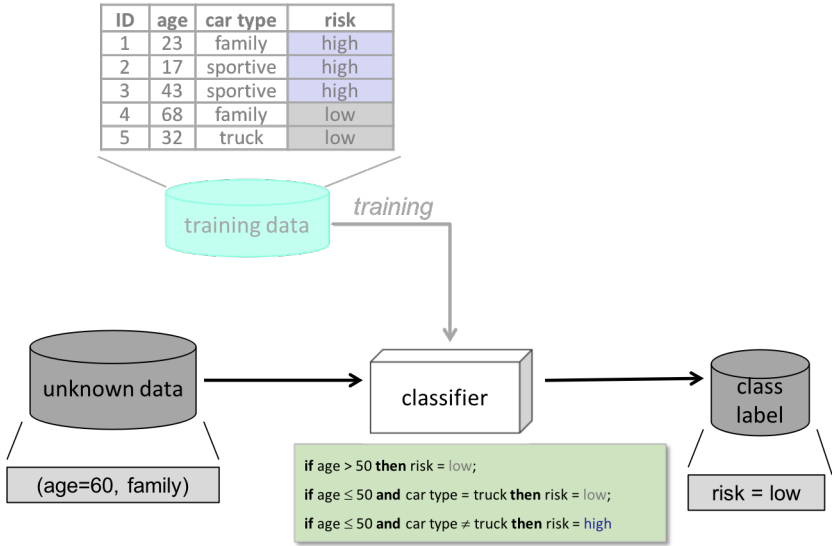
- Simple classifier

if age > 50 **then** risk $=$ low
if age ≤ 50 **and** car type $=$ truck **then** risk $=$ low
if age ≤ 50 **and** car type $\neq$ truck **then** risk $=$ high

Classification: Training Phase (Model Construction)



Classification: Prediction Phase (Application)



Classification

The systematic assignment of new observations to known categories according to criteria learned from a training set.

Formal Setup

- A *classifier* K for a *model* $M(\theta)$ is a function $K_{M(\theta)} : D \to Y$, where
  - D: data space
    - Often d-dim. space with attributes $a_1, \ldots, a_d$ (not necessarily a vector space)
    - Some other space, e.g. metric space
  - $Y = \{y_1, \ldots, y_k\}$: set of k distinct *class labels*
  - $O \subseteq D$: set of *training objects* o with known class labels $y \in Y$
- *Classification*: Application of classifier K on objects from $D \setminus O$
- Model $M(\theta)$ is the "type" of the classifier, and θ are the model parameters
- *Supervised learning*: find/learn optimal parameters θ for $M(\theta)$ given O

Supervised vs. Unsupervised Learning

Unsupervised Learning (clustering)

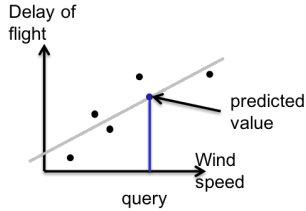
- The class labels of training data are unknown
- Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
  - Classes (=clusters) are to be determined

Supervised Learning (classification)

- Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - Classes are known in advance (a priori)
- New data is classified based on information extracted from the training set

Numerical Prediction

- Related problem to classification: numerical prediction
  - Determine the numerical value of an object
  - Method: e.g., regression analysis
  - Example: Prediction of flight delays

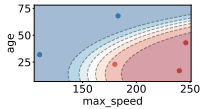


- Numerical prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Numerical prediction models continuous-valued functions
- Numerical prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown value
  - Major method for numerical prediction is regression:
    - Linear and multiple regression
    - Non-linear regression

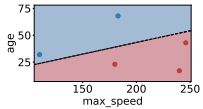
Goals for this Section

1. Introduction of different classification models
2. Learning techniques for these models

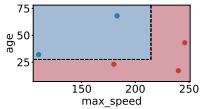
age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low



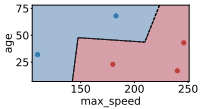
Bayes classifier



Linear classifier



Decision tree



k-NN classifier

Quality Measures for Classifiers

- Classification accuracy or classification error (complementary)
- Compactness of the model
  - Decision tree size, number of decision rules, ...
- Interpretability of the model
  - Insights and understanding of the data provided by the model
- Efficiency
  - Time to generate the model (training time)
  - Time to apply the model (prediction time)
- Scalability for large databases
  - Efficiency in disk-resident databases
- Robustness
  - Robust against noise or missing values

Evaluation of Classifiers: Notions

- Using training data to build a classifier and to estimate the model's accuracy may result in misleading and overoptimistic estimates
  - $\rightsquigarrow$ Overspecialization of the learning model to the training data
- *Train-and-Test*: Decomposition of labeled data set O into two partitions
  - Training data is used to train the classifier
    - Construction of the model by using information about the class labels
  - Test data is used to evaluate the classifier
    - Temporarily hide class labels, predict them anew and compare with original class labels
- Train-and-Test is not applicable if the set of objects for which the class label is known is very small

Evaluation of Classifiers: Cross Validation

m-fold Cross Validation

- Decompose data set evenly into m subsets of (nearly) equal size
- Iteratively use $(m-1)$ partitions for training data and the remaining single partition as test data
- Combine the m classification accuracy values to an overall classification accuracy

Leave-one-out: Special case of cross validation ($m = n$)

- For each of the objects o in the data set O:
  - Use set $O \setminus \{o\}$ as training set
  - Use the singleton set $\{o\}$ as test set
  - Compute classification accuracy by dividing the number of correct predictions through the database size $|O|$
- Particularly well applicable to nearest-neighbor classifiers

Quality Measures: Accuracy and Error

- Let K be a classifier
- Let $C(o)$ denote the correct class label of an object o
- Measure the quality of K:
  - Predict the class label for each object o from a data set $T \subseteq O$
  - Determine the fraction of correctly predicted class labels

Classification Accuracy of K

$$G_T(K) = \frac{|\{o \in T \mid K(o) = C(o)\}|}{|T|}$$

Classification Error of K

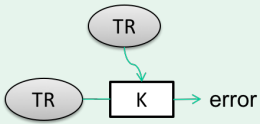
$$F_T(K) = \frac{|\{o \in T \mid K(o) \neq C(o)\}|}{|T|} = 1 - G_T(K)$$

Quality Measures: Accuracy and Error

- Let K be a classifier
- Let $TR \subseteq O$ be the training set: Used to build the classifier
- Let $TE \subseteq O$ be the test set: Used to test the classifier

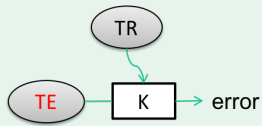
Resubstitution Error of K

$$F_{TR}(K) = \frac{|\{o \in TR \mid K(o) \neq C(o)\}|}{|TR|}$$



(True) Classification Error of K

$$F_{TE}(K) = \frac{|\{o \in TE \mid K(o) \neq C(o)\}|}{|TE|}$$



Quality Measures: Confusion Matrix

- Results on the test set: Confusion matrix

		classified as ...				
		class 1	class 2	class 3	class 4	class 5
correct label	class 1	35	1	1	1	4
	class 2	0	31	1	1	5
	class 3	3	1	50	1	2
	class 4	1	0	1	10	2
	class 5	3	1	9	16	13

(correctly classified in green)

- Based on the confusion matrix, we can compute several accuracy measures, including:
  - Classification Accuracy/Error
  - Precision and Recall

Quality Measures: Precision and Recall

Recall

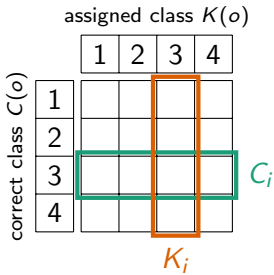
Fraction of test objects of class i, which have been identified correctly.

$$Recall_{TE}(K, i) = \frac{|\{o \in C_i \mid K(o) = C(o)\}|}{|C_i|}$$

Precision

Fraction of test objects assigned to class i, which have been identified correctly.

$$Precision_{TE}(K, i) = \frac{|\{o \in C_i \mid K_i(o) = C(o)\}|}{|K_i|}$$



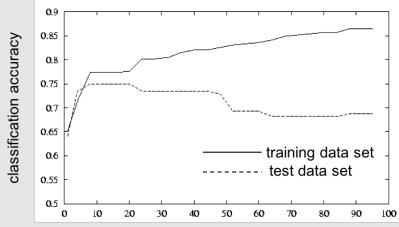
$$C_i = \{o \in TE \mid C(o) = i\}$$
$$K_i = \{o \in TE \mid K(o) = i\}$$

Overfitting

Characterization of Overfitting

The classifier adapts too closely to the training dataset and may therefore fail to accurately predict class labels for test objects unseen during training.

Example: Decision Tree



Generalization ← ***classifier*** → **specialization**

"overfitting"

Overfitting

Overfitting

- Occurs when the classifier is too optimized to the (noisy) training data
- As a result, the classifier yields worse results on the test data set
- Potential reasons:
  - Bad quality of training data (noise, missing values, wrong values)
  - Different statistical characteristics of training data and test data

Overfitting Avoidance

- Removal of *noisy/erroneous/contradicting* training data
- Choice of an appropriate *size* of the training set
  - Not too small, not too large
- Choice of appropriate sample
  - Sample should describe all aspects of the domain and not only parts of it

Underfitting

Underfitting

- Occurs when the classifiers model is too simple, e.g. trying to separate classes linearly that can only be separated by a quadratic surface
- Happens seldomly

$\rightsquigarrow$ *Trade-off*: Usually one has to find a good balance between over- and underfitting.

Agenda

Bayes Classification

- Probability based classification
  - Based on likelihood of observed data, estimate explicit probabilities for classes
  - Classify objects depending on costs for possible decisions and the probabilities for the classes
- Incremental
  - Likelihood functions built up from classified data
  - Each training example can incrementally increase/decrease the probability that a hypothesis (class) is correct
  - Prior knowledge can be combined with observed data.
- Good classification results in many applications

Bayes' Theorem

Probability Theory

- Conditional probability: $P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$ ("prob. of A given B")
- Product Rule: $P(A \wedge B) = P(A \mid B) \cdot P(B)$

Bayes' Theorem

- $P(A \wedge B) = P(A \mid B) \cdot P(B)$
- $P(B \wedge A) = P(B \mid A) \cdot P(A)$
- Since $P(A \wedge B) = P(B \wedge A)$, $P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$, and thus

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Bayes Classifier

- Bayes' rule: $P(c_j \mid x) = \frac{P(x|c_j) \cdot P(c_j)}{p(x)}$ for object x and class $c_j \in \mathcal{C}$.
- We are interested in maximizing this, i.e.

$$\underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \left(P(c_j \mid x)\right) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \left(\frac{P(x \mid c_j) \cdot P(c_j)}{p(x)} \right) \overset{(*)}{=} \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \left(P(x \mid c_j) \cdot P(c_j)\right)$$

  where $(*)$ assumes the value of $p(x)$ is constant and hence does not change the result.
- Final decision rule:

$$K(x) = c_{\text{max}} = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} \left(P(x \mid c_j) \cdot P(c_j)\right)$$

- But how to obtain $P(c_j)$ and $P(x \mid c_j)$.

Bayes Classifier: Density Estimation

A-Priori Class Probabilities

Estimate the a-priori probabilities $P(c_j)$ of classes $c_j \in \mathcal{C}$ by using the observed relative frequency of the individual class labels c_j in the training set, i.e.,

$$P(c_j) = \frac{N_{c_j}}{N}$$

Conditional Probabilities

- *Non-parametric* methods: Kernel methods Parzen's window, Gaussian kernels, etc.
- *Parametric* methods, e.g.
  - Single Gaussian distribution: Computed by maximum likelihood estimators (MLE)
  - *Mixture* models: e.g. Gaussian Mixture Model computed by EM algorithm

Bayes Classifier: Density Estimation

Problem

Curse of dimensionality often lead to problems in high dimensional data $\rightsquigarrow$ Density functions become too uninformative

Solution

- Dimensionality reduction
- Usage of statistical independence of single attributes (extreme case: naïve Bayes)

Naïve Bayes Classifier

Assumptions

- Objects are given as d-dimensional vectors, $x = (x_1, \ldots, x_d)$
- For any given class c_j the attribute values x_i are conditionally independent, i.e.

$$P(x_1, \ldots, x_d \mid c_j) = \prod_{i=1}^{d} P(x_i \mid c_j) = P(x_1 \mid c_j) \cdot \ldots \cdot P(x_d \mid c_j)$$

Decision Rule

$$K_{\text{naïve}}(x) = \operatorname*{argmax}_{c_j \in \mathcal{C}} \left(P(c_j) \cdot \prod_{i=1}^{d} P(x_i \mid c_j) \right)$$

Naïve Bayes Classifier

Categorical Attribute x_i

$P(x_i \mid c_j)$ can be estimated as the relative frequency of samples having value v_i as the ith attribute in class c_j in the training set.

Continuous Attribute x_i

$P(x_i \mid c_j)$ can, for example, be estimated through a Gaussian distribution determined by μ_{ij}, σ_{ij}.

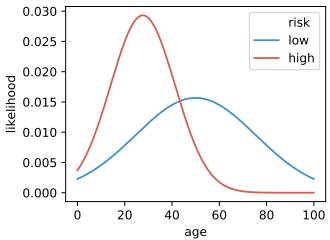
$\rightsquigarrow$ Computationally easy in both cases.

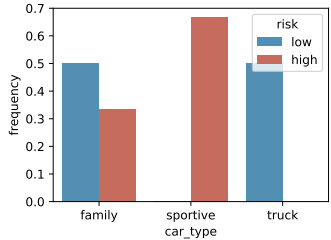
Naïve Bayes Classifier: Example

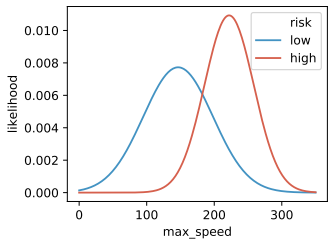
age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

Model Setup

- $Age \sim N(\mu, \sigma^2)$ (normal distribution)
- $car\_type \sim$ relative frequencies
- $max\_speed \sim N(\mu, \sigma^2)$ (normal distribution)







Naïve Bayes Classifier: Example (cont'd)

Query

$q = (age = 60; car\_type = family; max\_speed = 190)$

Example

We have:

- $P(high) = \frac{3}{5}$
- $\mu_{age,high} = \frac{83}{3}, \sigma^2_{age,high} = \frac{1112}{3} \implies P(age = 60 \mid high) \approx 0.00506$
- $P(car\_type = family \mid high) = \frac{1}{3}$
- $\mu_{max\_speed,high} = 222, \sigma^2_{max\_speed,high} = 2664 \implies P(max\_speed = 190 \mid high) \approx 0.00638$

and hence

$$\begin{aligned} P(high)P(q \mid high) &= P(high)P(age = 60 \mid high)P(car\_type = family \mid high)P(max\_speed = 190 \mid high) \\ &\approx 6.45166 \cdot 10^{-6} \end{aligned}$$

Analogously, we obtain $P(low)P(q \mid low) = 15.72290 \cdot 10^{-6} \implies K_{\ddot{\text{naive}}}(q) = low$.

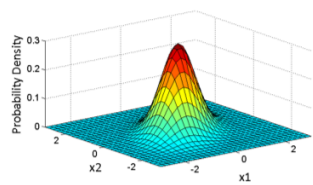
Bayesian Classifier

- Assuming dimensions of $x = (x_1, \ldots, x_d)$ are *not* independent
- Assume multivariate normal distribution (i.e. Gaussian)

$$P(x \mid C_j) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma_j)}} \exp\left(-\frac{1}{2}(x - \mu_j)\Sigma_j^{-1}(x - \mu_j)^T\right)$$

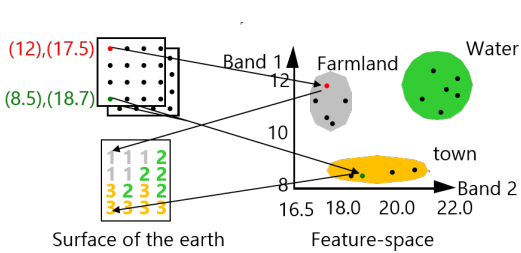
with

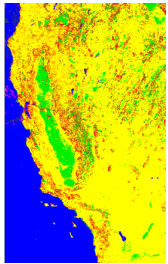
- μ_j: mean vector of class C_j
- Σ_j is the $d \times d$ covariance matrix
- $\det(\Sigma_j)$ is the determinant of Σ_j, and Σ_j^{-1} its inverse



Example: Interpretation of Raster Images

- Scenario: Automated interpretation of raster images
  - Take an image from a certain region (in d different frequency bands, e.g., infrared, etc.)
  - Represent each pixel by d values: $(x_1, \ldots, x_d)$
- Basic assumption: different surface properties of the earth ("landuse") follow a characteristic reflection and emission pattern



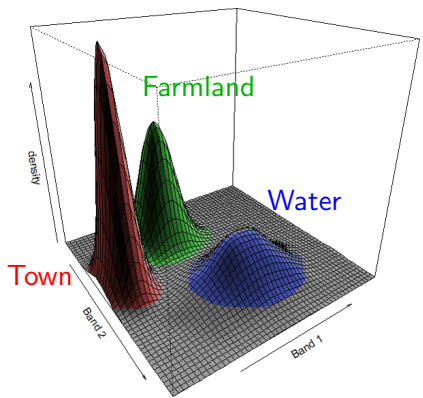


Example: Interpretation of Raster Images

Application of the Bayes classifier:

- Estimation of the $P(x \mid c)$ without assumption of conditional independence
- Assumption of d-dimensional normal (= Gaussian) distributions for the value vectors of a class

Probability of class membership



Example: Interpretation of Raster Images

Method

Estimate the following measures from training data

- μ_j: d-dimensional mean vector of all feature vectors of class C_j
- Σ_j: $d \times d$ covariance matrix of class C_j

Problems

- if likelihood of respective class is very low
- if several classes share the same likelihood

$\rightsquigarrow$ Mitigate e.g. by applying some minimum likelihood threshold; do not classify regions below.

Bayesian Classifiers: Discussion

Pro

- High classification accuracy for many applications if density function defined properly
- Incremental computation: many models can be adopted to new training objects by updating densities
  - For Gaussian: store count, sum, squared sum to derive mean, variance
  - For histogram: store count to derive relative frequencies
- Incorporation of expert knowledge about the application in the prior $P(C_i)$

Contra

- Limited applicability: often, required conditional probabilities are not available
- Lack of efficient computation: in case of a high number of attributes (particularly for Bayesian belief networks)

The Independence Hypothesis

The Independence Hypothesis . . .

- . . . makes efficient computation possible
- . . . yields optimal classifiers when satisfied
- . . . but is seldom satisfied in practice, as attributes (variables) are often correlated.

Attempts to overcome this limitation

- *Bayesian networks*, that combine Bayesian reasoning with causal relationships between attributes
- *Decision trees*, that reason on one attribute at the time, considering most important attributes first

Agenda

Linear Discriminant Function Classifier

Idea

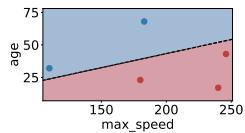
Separate points of two classes by a hyperplane

- I.e., classification model is a hyperplane
- Points of one class in one half space, points of second class in the other half space

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

Questions

- How to formalize the classifier?
- How to find optimal parameters of the model?



Basic Notions

Recall some general algebraic notions for a vector space V:

- $\langle x, y \rangle$denotes an inner product of two vectors $x, y \in V$
  - E.g., the scalar product $\langle x, y \rangle = x^T y = \sum_{i=1}^{d} x_i y_i$
- $H(w, w_0)$ denotes a hyperplane with normal vector w and constant term w_0:

$$x \in H \Leftrightarrow \langle x, w \rangle + w_0 = 0$$

- The normal vector w may be normalized to w':

$$w' = \frac{1}{\sqrt{\langle w, w \rangle}} w \implies \langle w', w' \rangle = 1$$

- Distance of a point x to the hyperplane $H(w', w_0)$:

$$dist(x, H(w', w_0)) = |\langle w', x \rangle + w_0|$$

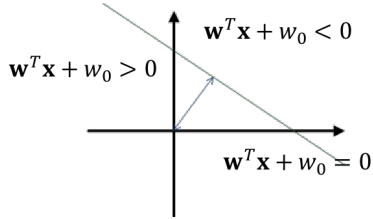
Formalization

- Consider a two-class example (generalizations later on):
  - D: d-dimensional vector space with attributes $a_1, \ldots, a_d$
  - $Y = \{-1, 1\}$ set of 2 distinct class labels y_j
  - $O \subseteq D$: Set of objects $o = (o_1, \ldots, o_d)$ with known class labels $y \in Y$ and cardinality $|O| = N$
- A hyperplane $H(w, w_0)$ with normal vector w and constant term w_0

$$x \in H \Leftrightarrow w^T x + w_0 = 0$$

Classification Rule

$$K_{H(w,w_0)}(x) = \text{sign}(w^T x + w_0)$$



Optimal Parameter Estimation

How to estimate optimal parameters w, w_0?

1. Define an objective/loss function $L(\cdot)$ that assigns a value (e.g. the error on the training set) to each parameter-configuration
2. Optimal parameters minimize/maximize the objective function

How does an objective function look like?

- Different choices possible
- Most intuitive: Each misclassified object contributes a constant (loss) value $\rightsquigarrow$ 0-1 loss

Optimal Parameter Estimation

0-1 Loss Objective for Linear Classifier

- $L(w, w_0) = \sum_{i=1}^{N} I(y_i \neq K_{H(w,w_0)}(x_i))$
- $\min_{w,w_0} L(w, w_0)$

where $I(condition) = 1$ if condition holds, 0 otherwise

- Minimize the overall number of training errors, but ...
  - NP-hard to optimize in general (non-smooth, non-convex)
  - Small changes of w, w_0 can lead to large changes of the loss

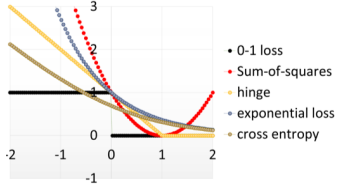
Loss Functions

Alternative Convex Loss Functions

Sum-of-squares loss	$(w^T x_i + w_0 - y_i)^2$	
Hinge loss	$\max\{0, (1 - y_i(w^T x_i + w_0)\}$	(SVM)
Exponential loss	$\exp(-y_i(w^T x_i + w_0))$	(AdaBoost)
Cross-entropy error	$-y_i \log g(x_i) + (1 - y_i) \log(1 - g(x_i))$	(Logistic Regression)
	where $g(x_i) = \frac{1}{1+\exp(-(w^T x_i + w_0))}$	

... and many more

- Optimizing different loss function leads to several classification algorithms
- Next, we derive the optimal parameters for the sum-of-squares loss



Optimal Parameters for SSE loss

Objective Function

$$SSE(w, w_0) = \frac{1}{2} \sum_{i=1}^{N} (w^T x_i + w_0 - y_i)^2$$

- Minimize the error function for getting optimal parameters
- Use standard optimization technique:
  1. Calculate first derivative
  2. Set derivative to zero and compute the global minimum (SSE is a convex function)

Optimal Parameters for SSE Loss

- Transform the problem for simpler computations:
  - $w^T x + w_0 = \sum_{i=1}^{d} w_i x_i + w_0 = \sum_{i=0}^{d} w_i x_i$ with $x_0 = 1$
  - For w let $\tilde{w} = (w_0, \ldots, w_d)^T$
- Combine the values to matrices

$$\tilde{X} = \begin{pmatrix} 1 & x_{1,1} & \ldots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \ldots & x_{N,d} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

- Then the sum-of-squares error is equal to

$$SSE(\tilde{w}) = \frac{1}{2}(\tilde{X}\tilde{w} - Y)^T(\tilde{X}\tilde{w} - Y)$$

Optimal Parameters for SSE Loss

- Take the derivative:

$$\frac{\partial}{\partial \tilde{w}} SSW(\tilde{w}) = \tilde{X}^T(\tilde{X}\tilde{w} - Y)$$

- Solve $\frac{\partial}{\partial \tilde{w}} SSE(\tilde{w}) = 0$:

$$\begin{aligned} \tilde{X}^T(\tilde{X}\tilde{w} - Y) &= 0 \\ \Leftrightarrow \tilde{X}^T\tilde{X}\tilde{w} &= \tilde{X}^T Y \\ \Leftrightarrow \tilde{w} &= \underbrace{(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T}_{\substack{\text{Left-inverse of } \tilde{X} \\ \text{(”Moore-Penrose Inverse”)}}} Y \end{aligned}$$

Optimal Parameters for SSE Loss

- Set $\hat{w} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T Y$
- Classify new point x with $x_0 = 1$ using

Classification Rule

$$K_{H(\hat{w})}(x) = \text{sign}(\hat{w}x)$$

Example SSE

- Data (consider only age and max. speed):

$$\tilde{X} = \begin{pmatrix} 1 & 23 & 180 \\ 1 & 17 & 240 \\ 1 & 43 & 246 \\ 1 & 68 & 183 \\ 1 & 32 & 110 \end{pmatrix}, \qquad Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low

- Encode classes as $\{\text{high} = 1, \text{low} = -1\}$

$$(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T = \begin{pmatrix} 0.7491 & -0.0836 & -0.8603 & -0.4736 & 1.6684 \\ -0.0087 & -0.0114 & 0.0049 & 0.0194 & -0.0043 \\ -0.0012 & 0.0036 & 0.0046 & -0.0002 & -0.0068 \end{pmatrix}$$

$$\implies \hat{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T Y = \begin{pmatrix} w_0 \\ w_{age} \\ w_{maxspeed} \end{pmatrix} = \begin{pmatrix} -1.3896 \\ -0.0302 \\ 0.0141 \end{pmatrix}$$

Example SSE

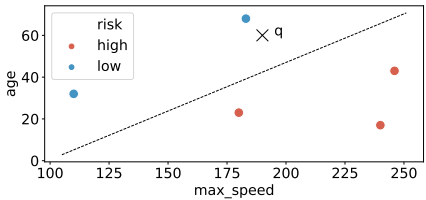
- Model parameters:

$$\hat{w} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T Y = \begin{pmatrix} w_0 \\ w_{age} \\ w_{maxspeed} \end{pmatrix} = \begin{pmatrix} -1.3896 \\ -0.0302 \\ 0.0141 \end{pmatrix}$$

$$\implies K_{H(\hat{w})}(x) = \text{sign}\left(\begin{pmatrix} -0.0302 \\ 0.0141 \end{pmatrix}^T x - 1.3896 \right)$$

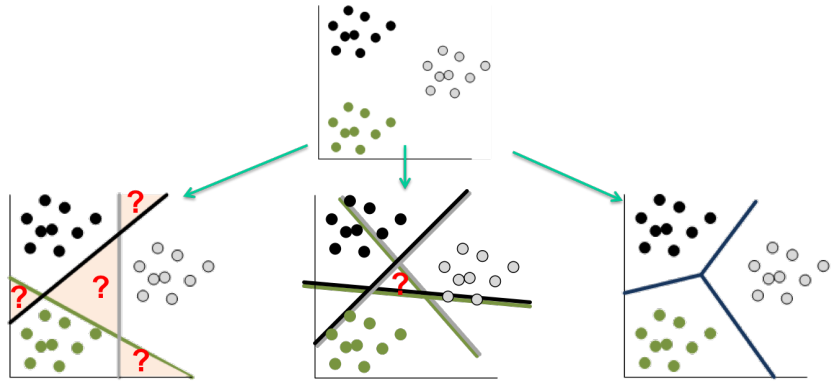
- Query: $q = (age = 60; max\_speed = 190)$

$$\text{sign}(\hat{w}^T \tilde{q}) = \text{sign}(-0.5323) = -1$$
$$\implies class = low$$



Extension to Multiple Classes

Assume we have more than two ($k > 2$) classes. What to do?



Extension to Multiple Classes

Idea of Multiclass Linear Classifier

- Take k linear functions of the form $H_{w_j,w_{j,0}}(x) = w_j^T x + w_{j,0}$
- Decide for class y_j:

$$y_j = \underset{j=1,\ldots,k}{\operatorname{argmax}} H_{w_j,w_{j,0}}(x)$$

- Advantage: No ambiguous regions except for points on decision hyperplanes
- The optimal parameter estimation is also extendable to k classes $y_1, \ldots, y_k$

Discussion (SSE)

Advantages

- Simple approach
- Closed form solution for parameters
- Easily extendable to non-linear spaces (later on)

Disadvantages

- Sensitive to outliers $\rightsquigarrow$ not a stable classifier
  - How to define and efficiently determine the maximum stable hyperplane?
- Only good results for linearly separable data
- Expensive computation of selected hyperplanes

$\rightsquigarrow$ Approach to solve problems: Support Vector Machines (SVMs) [Vapnik 1979, 1995]

Agenda

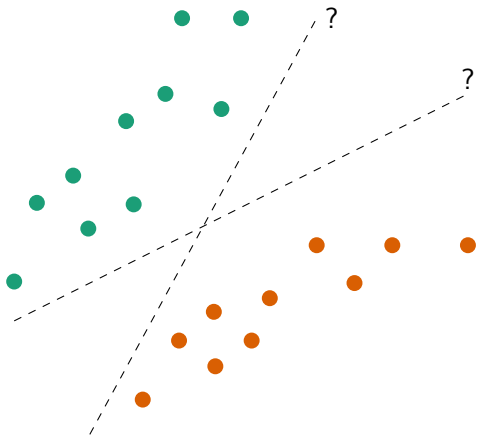
Maximum Margin Hyperplane

Question

How to define the notion of the "best" hyperplane differently?

Criteria

- Stability at insertion
- Distance to the objects of both classes

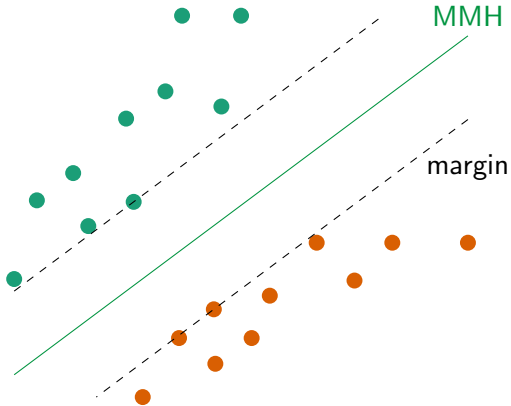


Support Vector Machines: Principle

Basic Idea

Linear separation with the *Maximum Margin Hyperplane (MMH)*:

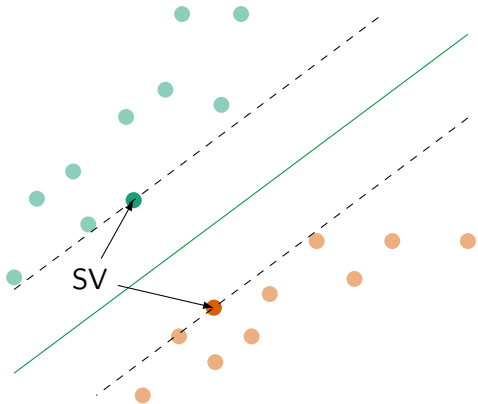
- Distance to points from any of the two sets is maximal, i.e., at least ξ
- Minimal probability that the separating hyperplane has to be moved due to an insertion
  $\rightsquigarrow$ Best generalization behavior; MMH is "maximally stable"



Support Vector Machines: Principle

Support Vectors

MMH only depends on points p_i whose distance to the hyperplane is exactly ξ. These p_i are called *support vectors* (SV).



Formalisation

- Let $\mathbf{x_i} \in \mathbb{R}^d$ denote the data points, and $y_i = +1$, if first class, else $y_i = -1$.
- A hyperplane in Hesse normal form is represented by a normal vector $\mathbf{w} \in \mathbb{R}^d$ of unit length (i.e., $\|\mathbf{w}\|_2 = 1$), and a (signed) distance from the origin $b \in \mathbb{R}$.
- In the following slides, we will define the requirements which the MMH shall fulfil.

Requirements of the MMH

The parameters $(\mathbf{w}, b)$ of the MMH shall fulfil the following two requirements:

No Error

The classification is accurate for all points, i.e.

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0 \iff \begin{cases} y_i = -1 & \langle \mathbf{w}, \mathbf{x_i} \rangle + b < 0 \\ y_i = +1 & \langle \mathbf{w}, \mathbf{x_i} \rangle + b > 0 \end{cases}$$

Requirement: Maximal Margin

Let $\xi = \min_{\mathbf{x_i} \in TR} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b|$ denote the minimum distance of any training object $\mathbf{x_i}$ to the hyperplane $H(\mathbf{w}, b)$. The margin ξ should be as large as possible.

Computation of the MMH

- *Task*: Maximise ξ subject to $y_i \cdot (\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > \xi$ for all $i \in \{1, \ldots, n\}$.
- Scaling the constraints by ξ^{-1} yields $y_i \cdot (\langle \xi^{-1}\mathbf{w}, \mathbf{x_i} \rangle + \xi^{-1} b) > 1$ for all $i \in \{1, \ldots, n\}$.
- Define $\mathbf{w}' = \xi^{-1}\mathbf{w}$, and $b' = \xi^{-1} b$.
- Maximizing ξ corresponds to minimizing $\langle \mathbf{w}', \mathbf{w}' \rangle = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{\xi^2}$.

Primary Optimization Problem

$$\begin{aligned} \min \quad & \|\mathbf{w}'\|_2^2 \\ s.t. \quad & y_i \cdot (\langle \mathbf{w}', \mathbf{x_i} \rangle + b') > 1 \qquad i \in \{1, \ldots, n\} \end{aligned}$$

Computation of the MMH

Primary Optimization Problem

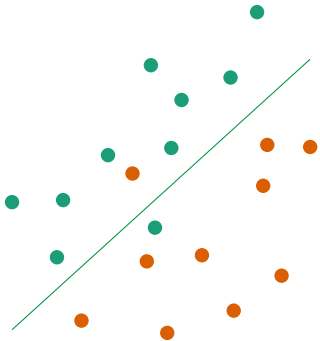
$$\begin{aligned} \min \quad & \|\mathbf{w}'\|_2^2 \\ s.t. \quad & y_i \cdot (\langle \mathbf{w}', \mathbf{x_i} \rangle + b') > 1 \qquad i \in \{1, \ldots, n\} \end{aligned}$$

- Convex optimization problem: Quadratic programming problem with linear constraints
  $\Longrightarrow$ Solution can be obtained by Lagrangian Theory.

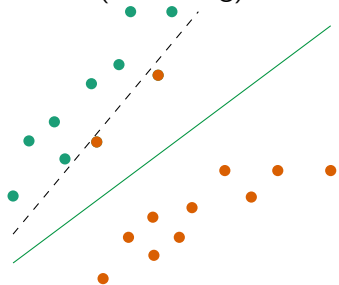
Soft Margin Optimization

- Problem of MMH optimization: How to treat non-(linearly separable) data?
- Two typical problems:

data points not linearly separable



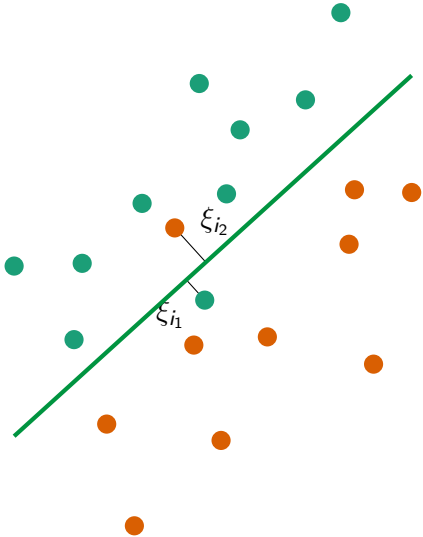
complete separation not optimal (overfitting)



- Trade-off between training error and size of margin

Soft Margin Optimization

- Additionally regard the number of training errors when optimizing:
  - ξ_i is the distance from $\mathbf{x_i}$ to the margin (often called *slack variable*):
    - $\xi_i = 0 \implies \mathbf{x_i}$ on correct side
    - $\xi_i > 0 \implies \mathbf{x_i}$ on wrong side
- Introduce parameter C to weight the misclassification against the size of the margin.



Soft Margin Optimization

Primary Optimization Problem With Soft Margin

$$
\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}'\|_2^2 + C\sum_{i=1}^{n}\xi_i & \\
s.t. \quad & y_i \cdot \left(\left\langle \mathbf{w}', \mathbf{x_i} \right\rangle + b'\right) > 1 - \xi_i & i \in \{1, \ldots, n\} \\
& \xi_i \geq 0 & i \in \{1, \ldots, n\}
\end{aligned}
$$

Soft Margin Optimization

Wolfe-Dual with Lagrange Multipliers

$$\max \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \qquad i \in \{1, \ldots, n\}$$

- $\alpha_i = 0$: $\mathbf{x_i}$ is not a support vector
- $\alpha_i = C$: $\mathbf{x_i}$ is support vector with $\xi_i > 0$
- $0 < \alpha_i < C$: $\mathbf{x_i}$ is support vector with $\xi_i = 0$

Soft Margin SVM

Decision Rule

$$H(x) = sign\left(\sum_{\mathbf{x_i} \in SV} \alpha_i y_i \langle \mathbf{x_i}, \mathbf{x} \rangle + b\right)$$

where SV denotes the set of support vectors.

SVM: Discussion

Pro

- generate classifiers with a high classification accuracy
- relatively weak tendency to overfitting (generalization theory)
- efficient classification of new objects due to often small number of support vectors
- compact models

Contra

- training times may be long (appropriate feature space may be very high-dimensional)
- expensive implementation

Agenda

Non-Linearly Separable Data Sets

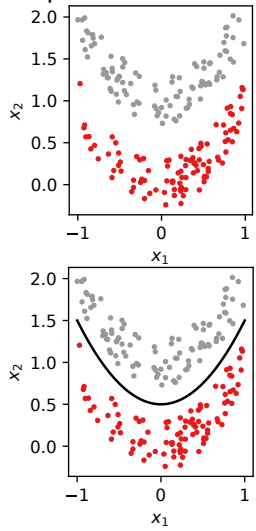
Problem

For real data sets, a linear separation with a high classification accuracy often is not possible.

Idea

Transform the data non-linearly into a new space, and try to separate the data in the new space linearly (extension of the hypotheses space)

Example for quadratically separable data set



Extension of the Hypotheses Space

Principle

$$\text{input space} \xrightarrow{\phi} \text{extented feature space}$$

$\rightsquigarrow$ Try to linearly separate in the extended feature space.

Example

$$\phi(x, y) = \left(1, x, y, x^2, xy, y^2\right)$$

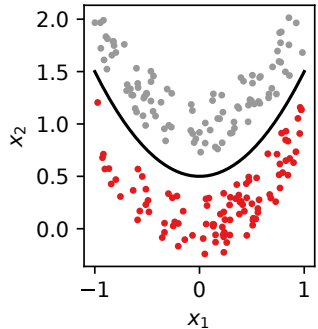
Here: A hyperplane in the extended feature space is a polynomial of degree 2 in the input space

Extension of the Hypotheses Space: Example (1)

Input Space (2 attributes):

$$x = (x_1, x_2)$$

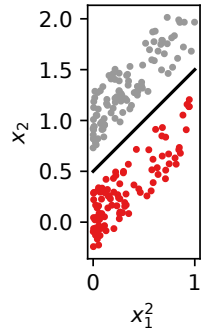
$$x_2 = x_1^2 + 0.5$$



Extended Space (6 attributes):

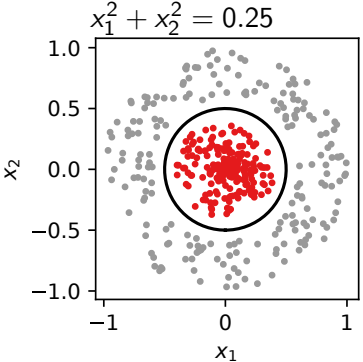
$$\phi(x) = \left(1, x, y, x^2, xy, y^2\right)$$

$$x_2 = (x_1^2) + 0.5$$

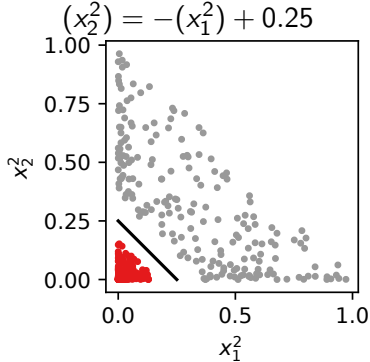


Extension of the Hypotheses Space: Example (2)

Input Space (2 attributes):
$x = (x_1, x_2)$



Extended Space (3 attributes):
$\phi(x) = (x_1^2, x_2^2, x_1 x_2)$



Extension of Linear Discriminant Function Classifier

- Linear classifier can be easily extended to non-linear spaces
- Recap: linear classifier $K_{H(w,w_0)}(x) = sign(w^T x + w_0)$
- Extend to non-linear case:
  - Transform all data points x to new feature space $\phi(x)$
  - Data Matrix X becomes a matrix Φ
  - The optimal hyperplane vector becomes ...

$$\tilde{w}_{opt,\phi} = (\Phi^T \Phi)^{-1} \Phi^T Y$$

  - ... and that's all!
- New classification rule: $K_{H(w_\phi, w_{0,\phi})}(x) = sign(w_\phi^T \phi(x) + w_{0,\phi})$
- SVM can be extended in a similar way

Non-linear Classification: Discussion

Pro

- By explicit feature transformation a much richer hypotheses space
- Simple extension of existing techniques
- Efficient evaluation, if transformed feature space not too high-dimensional

Contra

- Explicit mapping to other feature spaces can become problematic
- Meaningful transformation is usually not known a-priori
- Complex data distributions may require very high-dimensional features spaces $\rightsquigarrow$ High memory consumption, High computational costs

Implicit Mappings: Kernel Methods

Explicit Mapping

Explicit mapping of the data into the new feature space:

- After transformation, any vector-based distance is applied
- Resulting feature space may be very high dimensional $\rightsquigarrow$ Potential problems: Inefficient calculation, storage overhead

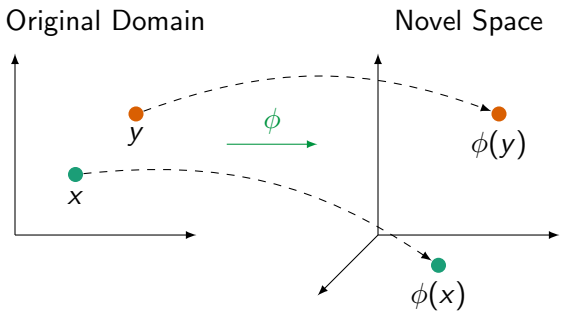
Often, we do not need the transformed data points themselves, but just the distances between them!

Implicit Mappings: Kernel Methods

"Kernel Trick"

Just *implicitly* map the data to a feature space: Determine a function K_ϕ, which computes the distance in the kernel space without explicitly computing $\phi(\cdot)$

$$K_\phi(x, y) = \langle \phi(x), \phi(y) \rangle$$



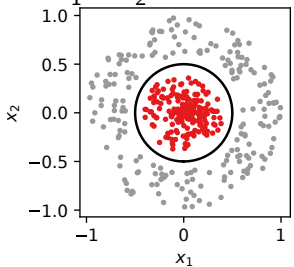
Kernel: Example

- Assume the original domain is $\mathcal{X} = \mathbb{R}^2$
- We transform a point $x = (x_1, x_2)$ to $\phi(x) = \left(x_1^2, x_2^2, x_1 x_2\right)$, i.e. the novel feature space is $\mathcal{H} = \mathbb{R}^3$, and $\kappa : \mathcal{X} \to \mathcal{H}$.

Input Space (2 attributes):

$$x = (x_1, x_2)$$

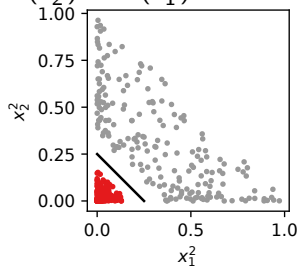
$x_1^2 + x_2^2 = 0.25$



Extended Space (3 attributes):

$$\phi(x) = \left(x_1^2, x_2^2, x_1 x_2\right)$$

$(x_2^2) = -(x_1^2) + 0.25$



Kernel: Example

- Original point $x = (x_1, x_2)$; transformed point $\phi(x) = \left(x_1^2, x_2^2, \sqrt{2} \cdot x_1 x_2\right)$
- We want to calculate the dot product in the novel feature space $\mathcal{H}$:

$$\begin{aligned}
\langle \phi(x), \phi(y) \rangle &= \left\langle \left(x_1^2, x_2^2, \sqrt{2} \cdot x_1 x_2\right), \left(y_1^2, y_2^2, \sqrt{2} \cdot y_1 y_2\right) \right\rangle \\
&= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \\
&= (x_1 y_1 + x_2 y_2)^2 \\
&= \langle x, y \rangle^2
\end{aligned}$$

- We do not have to explicitly map the points to the feature space $\mathcal{H}$!
- Simply calculate squared dot product in the original domain $\mathcal{X}$!
- $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, (x, y) \mapsto \langle x, y \rangle^2$ is called a (valid) kernel

Why is the dot product useful?

- Kernels correspond to dot products in some feature space
- With the dot product we are able to compute:
  - The norm/length of a vector $\|x\| = \sqrt{\langle x, x \rangle}$
  - The distance between two vectors:

$$\|x - y\|^2 = \langle x - y, x - y \rangle = \langle x, x \rangle + \langle y, y \rangle - 2 \langle x, y \rangle$$

  - The angle between two vectors:

$$\angle(x, y) = \arccos \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|}$$

Formal Definitions

Definition: Kernel Function

A *kernel function* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function, i.e., $\kappa(x, y) = \kappa(y, x)$, mapping pairs of objects $x, y \in \mathcal{X}$ to real numbers.

Definition: Mercer Kernel

For all finite $\{x_1, \ldots, x_n\} = X \subseteq \mathcal{X}$, let $\kappa(X) := (\kappa(x_i, x_j))_{i,j} \in \mathbb{R}^{n \times n}$. A kernel function κ is called *Mercer kernel*, valid kernel, admissible kernel, or positive semi-definite, if for all such finite X, the matrix $\kappa(X)$ is positive semi-definite, i.e. for all $c \in \mathbb{R}^n$, it holds

$$c^T \kappa(X) c \geq 0$$

Formal Definitions (cont'd)

Definition: Dot Product

A dot product in a vector space $\mathcal{H}$ is a function $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ satisfying:

- $\langle x, x \rangle = 0$ for $x = 0$
- $\langle x, x \rangle > 0$ for $x \neq 0$
- $\langle x, y \rangle = \langle y, x \rangle$ (Symmetry)
- $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$ (Bi-linearity)

Definition: Hilbert Space

A vector space $\mathcal{H}$ endowed with a dot product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ for which the induced norm gives a complete metric space, is termed *Hilbert Space*.

Interpretation of Kernel Functions

Theorem

Let $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a valid kernel on $\mathcal{X}$. There exists a possibly infinite-dimensional Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $\kappa(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$ where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the dot product in a Hilbert space $\mathcal{H}$.

$\rightsquigarrow$ every kernel κ can be seen as a dot product in some feature space $\mathcal{H}$.

Advantages

- Feature space $\mathcal{H}$ can be infinite-dimensional
- Not really necessary to know which feature space $\mathcal{H}$ we have
- Computation of kernel is done in original domain $\mathcal{X}$

Kernel SVM

Wolfe-Dual Optimization Problem with Lagrange Multipliers

$$\max \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \qquad i \in \{1, \ldots, n\}$$

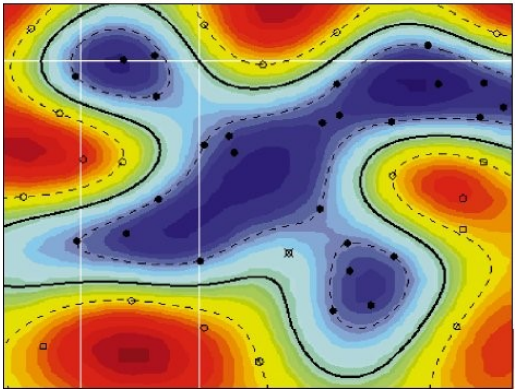
Decision Rule

$$H(x) = sign\left(\sum_{x_i \in SV} \alpha_i y_i \kappa(x_i, x) + b \right)$$

Example for Mercer Kernels

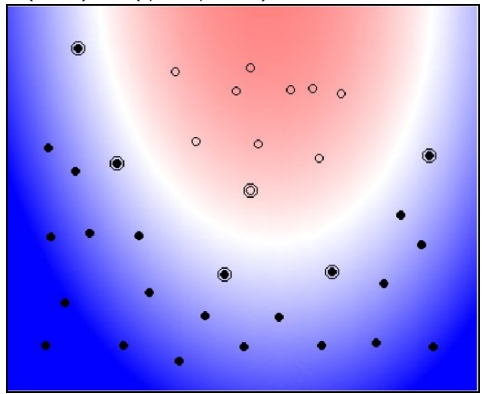
Radial Basis Kernel

$\kappa(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$



Polynomial Kernel (degree 2)

$\kappa(x, y) = (\langle x, y \rangle + 1)^d$



Discussion

Pro

- Kernel methods provide a simple method for dealing with non-linearity
- Implicit mapping allows for mapping to arbitrary-dimensional spaces:c
  Computational effort depends on the number of training examples, but not on the feature space dimensionality

Contra

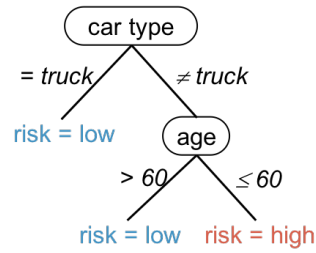
- Resulting models rarely provide an intuition
- Choice of kernel can be difficult

Agenda

Decision Tree Classifiers

- Approximating discrete-valued target function
- Learned function is represented as a tree:
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Tree can be transformed into decision rules:
  if age $>$ 60 **then** risk $=$ low
  if age $\leq$ 60 **and** car type $=$ truck **then** risk $=$ low
  if age $\leq$ 60 **and** car type $\neq$ truck **then** risk $=$ high

age	car_type	max_speed	risk
23	family	180	high
17	sportive	240	high
43	sportive	246	high
68	family	183	low
32	truck	110	low



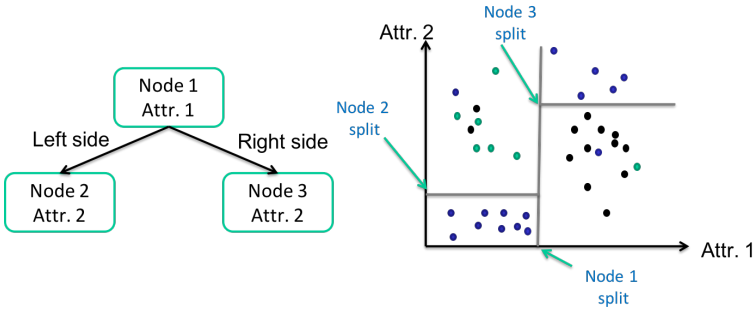
Advantages

- Decision trees represent explicit knowledge
- Decision trees are intuitive to most users

Decision Tree Classifier: Splits

Goal

- Each tree node defines an axis-parallel $(d-1)$-dimensional hyperplane, that splits the data space.
- *Find such splits which lead to as homogenous groups as possible.*



Decision Tree Classifiers: Basics

- Decision tree generation (training phase) consists of two phases
  1. Tree construction
     - At start, all the training examples are at the root
     - Partition examples recursively based on selected attributes
  2. Tree pruning
     - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Traverse the tree and test the attribute values of the sample against the decision tree
  - Assign the class label of the respective leaf to the query object

Algorithm for Decision Tree Construction

- Basic algorithm (a greedy algorithm)
  - Tree is created in a *top-down recursive divide-and-conquer* manner
  - Attributes may be categorical or continuous-valued
  - At the start, all the training examples are assigned to the root node
  - Recursively partition examples at each node and push them down to the new nodes
  - Select test attributes and determine split points or split sets for the respective values based on a heuristic or statistical measure (*split strategy*, e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Algorithm for Decision Tree Construction

- Most algorithms are versions of this basic algorithm (greedy, top-down)
  - E.g.: ID3, or its successor C4.5

ID3 Algorithm

procedure ID3(*Examples*, *TargetAttr*, *Attributes*) ▷ specialized to learn boolean-valued functions
  Create *Root* node for the tree
  if all *Examples* are positive **then return** *Root* with *label* $= +$
  else if all *Examples* are negative **then return** *Root* with *label* $= -$
  else if *Attributes* $= \emptyset$ **then return** *Root* with *label* $=$ most common value of *TargetAttr* in *Examples*
  else
    $A =$ "best" decision attribute for next node ▷ how to determine the "best" attribute?
    Assign A as decision attribute for *Root*
    for each possible value v_i of A **do** ▷ how to split the possible values?
      Generate branch corresponding to test $A = v_i$
      $Examples_{v_i} =$ examples that have value v_i for A
      if $Examples_{v_i} = \emptyset$ **then**
        Add leaf node with *label* $=$ most common value of *TargetAttr* in *Examples*
      else
        Add subtree ID3($Examples_{v_i}$, *TargetAttr*, *Attributes* $\setminus \{A\}$)

Example: Decision for "playing_tennis"

- Query: How about playing tennis today?
- Training data:

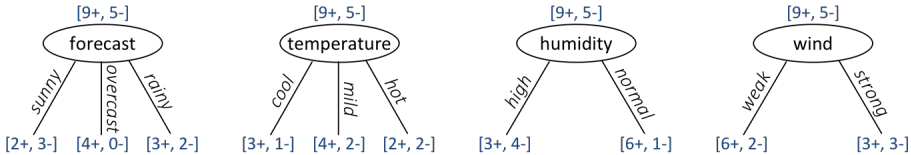
day	forecast	temperature	humidity	wind	tennis decision
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

- Build decision tree ...

Split Strategies: Quality of Splits

Given

- A set T of training objects
- A (disjoint, complete) partitioning $T_1, \ldots T_m$ of T
- The relative frequencies p_i of class c_i in T and in the partitions $T_1, \ldots T_m$



Wanted

- A measure for the heterogeneity of a set S of training objects with respect to the class membership
- A split of T into partitions $\{T_1, \ldots, T_m\}$ such that the heterogeneity is minimized

$\rightsquigarrow$ Proposals: *Information gain*, *Gini index*, *Misclassification error*

Attribute Selection Measures: Information Gain

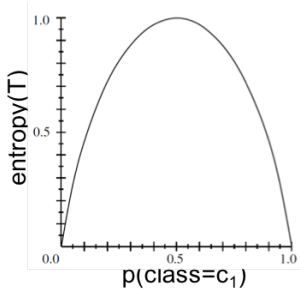
- Used in ID3/C4.5

Entropy

- Minimum number of bits to encode a message that contains the class label of a random training object
- The entropy of a set T of training objects is defined as

$$entropy(T) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

  for k classes with frequencies p_i
- $entropy(T) = 0$ if $p_i = 1$ for any class c_i
- $entropy(T) = 1$ if $p_i = \frac{1}{k}$ for all classes c_i



$k = 2$

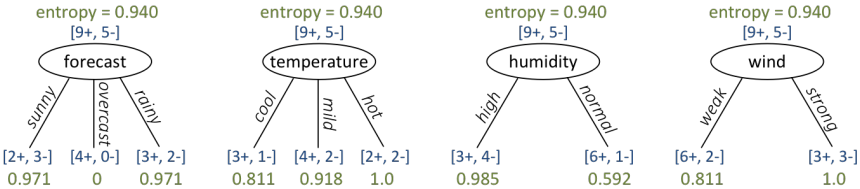
Attribute Selection Measures: Information Gain

Information Gain

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The information gain of attribute A w.r.t. T is defined as

$$information\_gain(T, A) = entropy(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} entropy(T_i)$$

Attribute Selection: Example (Information Gain)

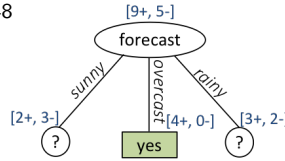


$$information\_gain(T, forecast) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.246$$

$$information\_gain(T, temperature) = 0.94 - \frac{4}{14}0.811 - \frac{6}{14}0.981 - \frac{4}{14}1 = 0.029$$

$$information\_gain(T, humidity) = 0.94 - \frac{7}{14}0.985 - \frac{7}{14}0.592 = 0.151$$

$$information\_gain(T, wind) = 0.94 - \frac{8}{14}0.811 - \frac{6}{14}1 = 0.048$$

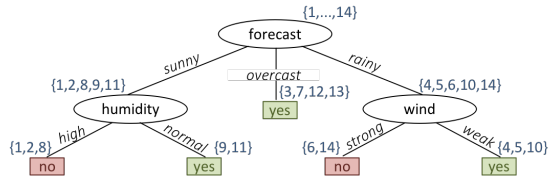


Result: "forecast" yields the highest information gain

Example: Decision Tree for "playing_tennis"

day	forecast	temperature	humidity	wind	tennis decision
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

Final decision tree:



Attribute Selection Measures: Gini Index

- Used in IBM's IntelligentMiner

Gini Index

The Gini index for a set T of training objects is defined as

$$gini(T) = 1 - \sum_{i=1}^{k} p_i^2$$

- Small value of Gini index $\equiv$ low heterogeneity
- Large value of Gini index $\equiv$ high heterogeneity

Gini Index (of an attribute A)

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The Gini index of attribute A w.r.t. T is defined as

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} gini(T_i)$$

Attribute Selection Measures: Misclassification Error

Misclassification Error

The Misclassification Error for a set T of training objects is defined as

$$Error(T) = 1 - \max_{c_i} p_i$$

- Small value of Error $\equiv$ low heterogeneity
- Large value of Error $\equiv$ high heterogeneity

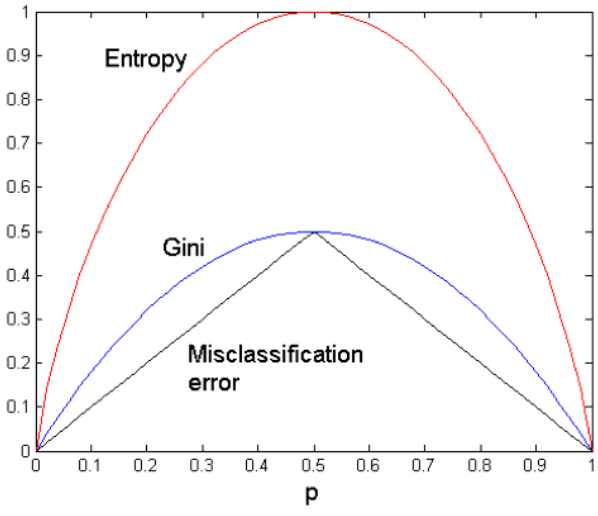
Misclassification Error (of an attribute A)

Let A be the attribute that induced the partitioning $\{T_1, \ldots, T_m\}$ of T. The Misclassification Error of attribute A w.r.t. T is defined as

$$Error_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} Error(T_i)$$

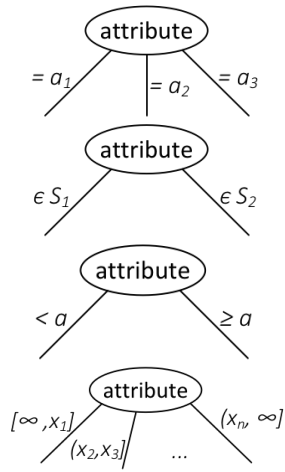
Attribute Selection Measures: Comparison

For two-class problems:



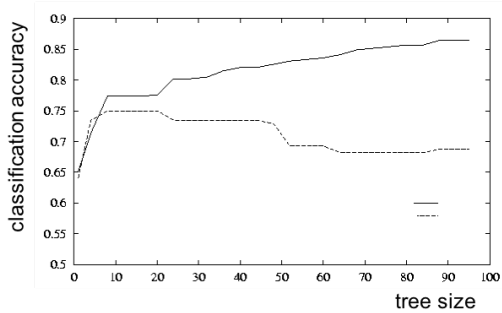
Split Strategies: Types of Splits

- Categorical attributes
  - Split criteria based on equality "$attribute = a$"
  - Based on subset relationships "$attribute \in set$"
    $\rightsquigarrow$ many possible choices (subsets)
    - Choose the best split according to, e.g., gini index
- Numerical attributes
  - Split criteria of the form "$attribute < a$"
    $\rightsquigarrow$ many possible choices for the split point
    - One approach: Order test samples w.r.t. their attribute value; consider every mean value between two adjacent samples as possible split point; choose best one according to, e.g., gini index
  - Partition the attribute value into a discrete set of intervals (Binning)



Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result has poor accuracy for unseen samples



- Two approaches to avoid overfitting for decision trees:
  1. Post-pruning = pruning of overspecialized branches
  2. Pre-pruning = halt tree construction early

Avoid Overfitting in Classification

Post-pruning

Pruning of overspecialized branches:

- Remove branches from a "fully grown" tree and get a sequence of progressively pruned trees
- Use a set of data different from the training data to decide which is the "best pruned tree"

Avoid Overfitting in Classification

Pre-pruning

Halt tree construction early, do not split a node if this would result in the goodness measure falling below a threshold.

- Choice of an appropriate value for *minimum support*
  - Minimum support: minimum number of data objects a leaf node contains
  - In general, *minimum support* $\gg 1$
- Choice of an appropriate value for *minimum confidence*
  - Minimum confidence: minimum fraction of the majority class in a leaf node
  - Typically, *minimum confidence* $\ll 100\%$
  - Leaf nodes can absorb errors or noise in data records
- Discussion
  - With Pre-pruning it is difficult to choose appropriate thresholds
  - Pre-pruning has less information for the pruning decision than post-pruning $\rightsquigarrow$ can be expected to produce decision trees with lower classification quality
  - Tradeoff: tree construction time vs. classification quality

Pruning of Decision Trees: Approach Post-pruning

Reduced-Error Pruning [1]

- Decompose classified data into training set and test set
- Create a decision tree E for the training set
- Prune E using the test set T
  - Determine the interior node v of E whose pruning reduces the number of misclassified data points on T the most (i.e., replace the subtree S with root v by a leaf. Determine the value of the leaf by majority voting)
  - Prune
  - Finish if no such interior node exists
- Only applicable if a sufficient number of classified data is available

[1] J.R. Quinlan. Rule induction with statistical data – a comparison with multiple regression. In Journal of the Operational Research Society, 38, pages 347-352, 1987

Pruning of Decision Trees: Approach Post-pruning

Minimal Cost Complexity Pruning [1]

- Does not require a separate test set
  - Applicable to small training sets as well
- Pruning of the decision tree by using the training set
  - Classification error is no appropriate quality measure
- New quality measure for decision trees:
  - Trade-off of classification error and tree size
  - Weighted sum of classification error and tree size
- General observation
  - The smaller decision trees yield the better generalization

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984

Minimal Cost Complexity Pruning: Notions

- Size $|E|$ of a decision tree E: number of leaf nodes
- Cost-complexity quality measure of E with respect to training set T, classification error F_T and complexity parameter $\alpha \geq 0$:

$$CC_T(E, \alpha) = F_T(E) + \alpha|E|$$

- For the *smallest minimal subtree* $E(\alpha)$ of E w.r.t. α, it is true that:
  1. There is no subtree of E with a smaller cost complexity
  2. If $E(\alpha)$ and B both fulfill (1), then is $E(\alpha)$ a subtree of B
- $\alpha = 0$: $E(\alpha) = E$
  - Only error matters
- $\alpha \to \infty$: $E(\alpha) =$ root node of E
  - Only tree size matters
- $0 < \alpha < \infty$: $E(\alpha)$ is a proper substructure of E
  - The root node or more than the root node

Extracting Classification Rules from Trees

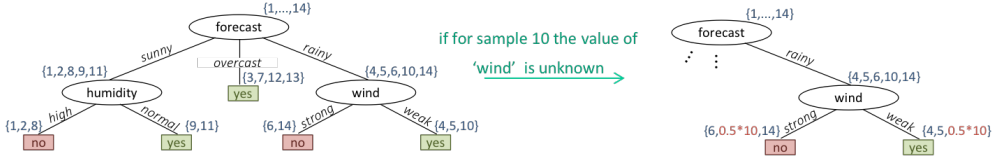
- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand

Example

if forecast = "overcast" **then** playing_tennis = "yes"
if forecast = "sunny" **and** humidity = "high" **then** playing_tennis = "no"
if forecast = "sunny" **and** humidity = "normal" **then** playing_tennis = "yes"
if forecast = "rainy" **and** wind = "strong" **then** playing_tennis = "no"
if forecast = "rainy" **and** wind = "weak" **then** playing_tennis = "yes"

Enhancement: Handle Missing Attribute Values

- If node n tests attribute A:
  - Assign most common value of A among other examples sorted to node n
  - Assign the most common value of the attribute among other examples with the same target value sorted to node n
  - Assign probability p_i to each of the possible values v_i of attribute A among other examples sorted to node n
    - Assign fraction p_i of example to each descendant in tree
    - Classify new examples in the same fashion: Classification decision is the one with the highest probability (sum over all instance fragments of each class decision)



Decision Tree Classifiers: Summary

Pro

- Relatively fast learning speed (in comparison to other classification methods)
- Fast classification speed
- Convertible to simple and easy to understand classification rules
- Often comparable classification accuracy with other classification methods

Contra

- Not very stable, small changes of the data can lead to large changes of the tree

Agenda

Nearest Neighbor Classifiers

Motivation

- Assume data in a non-vector representation: graphs, forms, XML-files, etc.
- No simple way to use linear classifiers or decision trees

Solutions

- Use appropriate kernel function for kernel machines (e.g. kernel SVM)
  $\rightsquigarrow$ Not always clear how to define a kernel
- Transformation of objects to some vector space (e.g. representation learning)
  $\rightsquigarrow$ Difficult to determine appropriate transformation & vector space
- *Here*: Nearest neighbor classifier
  $\rightsquigarrow$ Direct usage of the similarity of objects for classification

Nearest Neighbor Classifiers

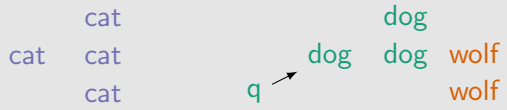
Procedure

Assign query object q to the class c_j of the closest training object $x \in D$:

$$class(q) = class(NN(q)) \qquad NN(q) = \{x \in D \mid \forall x' \in D : d(q,x) \leq d(q,x')\}$$

$\rightsquigarrow$ Instance-Based Learning

Example



Classifier decides that query object q is a dog.

Instance-Based Methods

Eager Evaluation

- Create models from data (training phase) and then use these models for classification (test phase)
- Examples: Decision tree, Bayes classifier

Instance-Based Learning

- Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical Approaches: k-nearest neighbor approach:
  - Instances represented as points in a metric space (e.g. Euclidean)
  - Classification by label of NN $\rightsquigarrow$ requires fast NN queries
  - $\rightsquigarrow$ Training phase = Index construction (e.g. R-tree)

Nearest Neighbor Classifiers: Notions

Notions

- Distance Function: Defines the (dis-)similarity for pairs of objects
- *Decision Set*: The set of k nearest neighboring objects to be used in the decision rule

Decision Rule

Given the class labels of the objects from the decision set, how to determine the class label to be assigned to the query object?

Decision Rules

Majority Vote (Default)

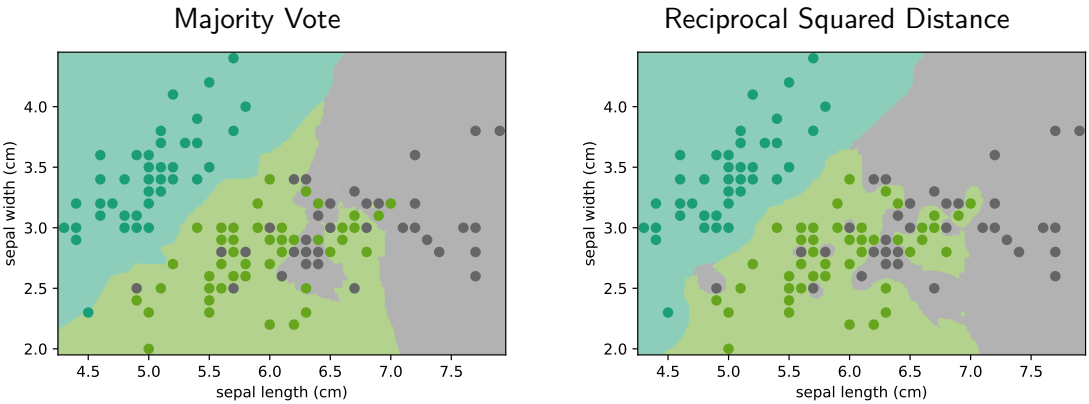
Choose majority class in the decision set, i.e. the class with the most representatives in the decision set

Weighted Decision Rules

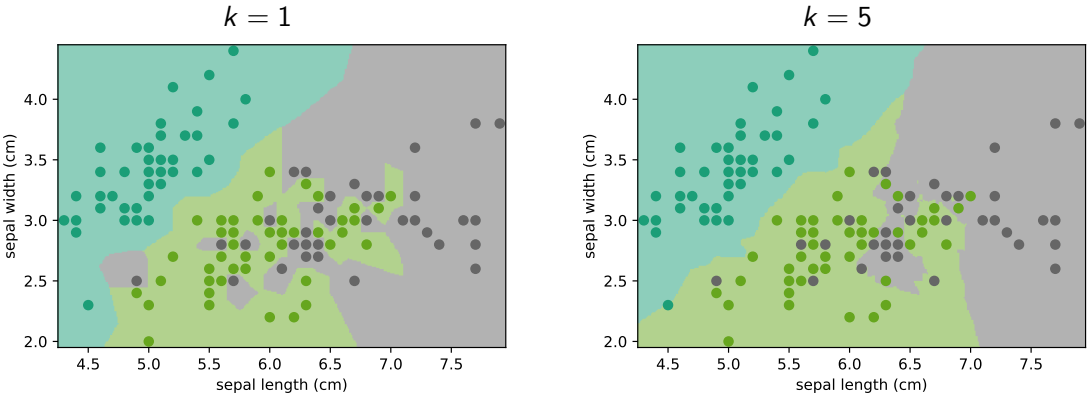
Choose *weighted* majority of decision set class labels. Weight variants:

- *Reciprocal squared distance*: $d(q, x)^{-2}$
- *Inverse A-Priori Probability*: Use inverse frequency of classes in the training set

Example: $k = 5$ – Influence of Weighting



Example: Majority Vote – Influence of k



NN Classifier: Parameter k

Choosing an appropriate k: Tradeoff between *overfitting* and *generalization*:

Influence of k

- k too small: High sensitivity against outliers
- k too large: Decision set contains many objects from other classes

Rules of Thumb

- Based on theoretical considerations: Choose k, such that it grows slowly with n, e.g. $k \approx \sqrt{n}$ or $k \approx \log n$
- Empirically, $1 \ll k < 10$ yields a high classification accuracy in many cases

NN Classifier: Variants

- *k-NN* Classifier: Consider the k nearest neighbors for the class assignment decision
- *Weighted k-NN* Classifier: Use weights for the classes of the k nearest neighbors
- *Mean-based NN* Classifier: Determine mean vector m_i for each class c_j (in training phase); Assign query object to the class c_j of the nearest mean vector m_i
- *Generalization*: Representative-based NN Classifier; Use more than one representative per class

NN Classifier: Discussion

Pro

- *Applicability*: Training data and distance function required only
- High classification *accuracy* in many applications
- Easy *incremental* adaptation to new training objects useful also for prediction
- *Robust* to noisy data by averaging k-nearest neighbors

Contra

- Naïve implementation is inefficient: Requires k-nearest neighbor query processing $\rightsquigarrow$ support by database techniques may help to reduce from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$
- Does not produce explicit knowledge about classes, *but* provides some explanation information
- Curse of dimensionality: Distance between neighbors could be dominated by irrelevant attributes $\rightsquigarrow$ Apply dimensionality reduction first

Agenda

Ensemble Classification

Problem

- No single classifier performs good on every problem
- For some techniques, small changes in the training set lead to very different classifiers

Idea

Improve performance by combining different classifiers $\rightsquigarrow$ ensemble classification.
Different possibilities exist. Discussed here:

- Bagging (Bootstrap aggregation)
- Boosting

Bagging

How to obtain different classifiers?

Easiest way: Train the *same classifier* K on *different datasets*

Bagging (or Bootstrap Aggregation)

- Randomly select m different subsets from the training set
- On each subset, independently train a classifier K_i $(i = 1, \ldots, m)$
- Overall decision:

$$K(x) = sign\left(\frac{1}{m}\sum_{i=1}^{m} K_i(x)\right)$$

Boosting

Boosting

- Linear combination of several *weak* learners (different classifiers)
- Given m weak learners K_i and weights α_i for $i = 1, \ldots, m$
- Overall decision

$$K(x) = sign\left(\sum_{i=1}^{m} \alpha_i K_i(x)\right)$$

- *Important difference*: classifiers are trained in sequence!
- Repeatedly misclassified points are weighted stronger

AdaBoost

Widely used boosting method: AdaBoost [21]: Meta-algorithm that iteratively generates a chain of weak learners

General Idea

- Assume $(t-1)$ weak learners are already given. The tth learner should focus on instances that were previously misclassified.
- Assign a weight w_i to each instance x_i to represent its importance
- Start with equal weight for each instance, adapt weights according to the performance of previously trained classifiers

[21] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, 1996.

AdaBoost – Algorithm

Given: n data points $x_1, \ldots, x_n$, labels $y_1, \ldots, y_n$
Initialize $w_1 = \ldots = w_n = \frac{1}{n}$
for $i = 1, \ldots, m$ **do**
Fit a classifier $K_i(x)$ to the training data by minimizing weighted error function

$$J_i = \sum_{j=1}^{n} w_j \mathbb{I}(K_i(x_j) \neq y_j)$$

where $\mathbb{I}$ is the indicator function
Compute weighting coefficient

$$\alpha_i = \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right) \qquad \text{where } \epsilon_i = \frac{J_i}{\sum_{j=1}^{n} w_j}$$

Update all data weights:

$$w_j := w_j \exp\left(\alpha_i \mathbb{I}(K_i(x_j) \neq y_j)\right)$$

Final Model

$$K(x) = \mathit{sign}\left(\sum_{i=1}^{m} \alpha_i K_i(x)\right)$$

Classification: Summary

	Model	Compactness	Model Interpretability	Decision Interpretability
Linear Model	hyperplane	compact (# dims)	medium/low	low
SVM	hyperplane/non-linear(kernel)	compact (# SV)	medium/low	low
Decision Tree	set of (axis-parallel) hyperplanes	compact (pruned)	good	good (rules)
***k*NN**	no model	no model	no model	medium/good (example)
Bayes	statistical density distribution	model dependent	model dependent	medium/good (probabilities)

	Data Types	Robustness	Training Time	Test Time
Linear Model	arbitrary (kernel)	low	high	low/high (for high-dim)
SVM	arbitrary (kernel)	high	medium	low/medium
Decision Tree	categorical & vector	low	low/medium	low
***k*NN**	arbitrary (distance)	high	no training	low (index)/ very high
Bayes	arbitrary (probability distribution)	high	model-dependent	model-dependent; often low

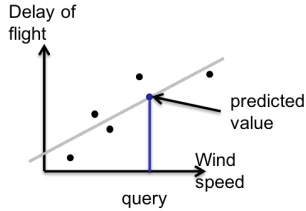
Classification: Conclusion

- Classification is an extensively studied problem (mainly in statistics and machine learning)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Scalability is an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of complex data, e.g., text, spatial, multimedia, etc.;
  Example: kNN-classifiers rely on distances but do not require vector representations of data
- Results can be improved by ensemble classification

Agenda

Numerical Prediction

- Related problem to classification: numerical prediction
  - Determine the numerical value of an object
  - Method: e.g., regression analysis
  - Example: Prediction of flight delays



- Numerical prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Numerical prediction models continuous-valued functions
- Numerical prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown value
  - Major method for numerical prediction is regression:
    - Linear and multiple regression
    - Non-linear regression

Examples

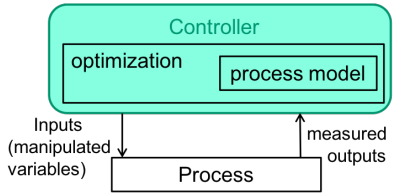
Example: Housing values in suburbs of Boston

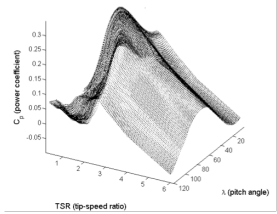
- Inputs:
  - Number of rooms
  - Median value of houses in the
  - Neighborhood
  - Weighted distance to five Boston employment centers
  - Nitric oxides concentration
  - Crime rate per capita
  - ...
- Goal: Compute a model of the housing values, which can be used to predict the price for a house in that area.

Examples

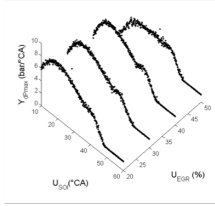
Control engineering

- Control the inputs of a system in order to lead the outputs to a given reference value
- Required: A model of the process





Wind turbine

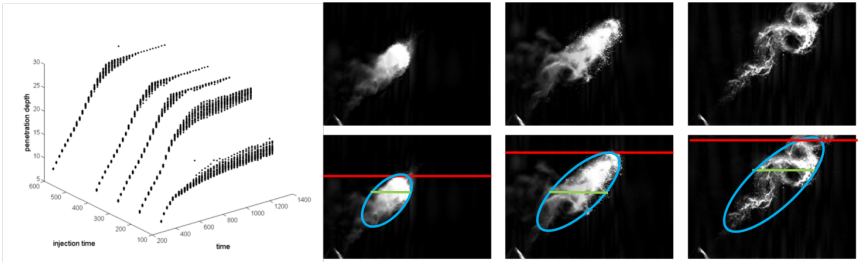


Diesel engine

Examples

Fuel injection process

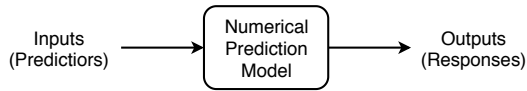
- Database of spray images
- Inputs: Settings in the pressure chamber
- Outputs: Spray features, e.g., penetration, depth, spray, width, spray area
- Goal: Compute a model which predicts the spray features, for input settings which have not been measured.



Numerical Prediction

Numerical Prediction

- Given: A set of observations
- Compute: A generalized model of the data which enables the prediction of the output as a continuous value



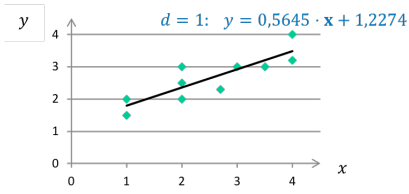
Quality Measures

- Accuracy of the model
- Compactness of the model
- Interpretability of the model
- Runtime efficiency (training, prediction)

Linear Regression

- Given a set of N observations with inputs of the form $x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$ and outputs $y \in \mathbb{R}$
- Approach: Minimize the *Sum of Squared Errors (SSE)*
- Numerical Prediction: Describe the outputs y as a linear equation of the inputs

$$\hat{y} = f(x) = w_0 + \sum_{i=1}^{d} w_i x_i = (1, x_1, \ldots, x_d)^T w$$



- Train the parameters $w = (w_0, w_1, \ldots, w_d)^T$ according to

$$\min_w \frac{1}{2} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

Linear Regression

- Matrix notation: Let $X \in \mathbb{R}^{N \times (d+1)}$ be the matrix containing the inputs, $Y \in \mathbb{R}^N$ the outputs, and w the resulting coefficients:

$$X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \dots & x_{N,d} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \qquad w = \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix}$$

- Goal: Find the coefficients w, which minimize the SSE

$$\begin{aligned} \min_w \frac{1}{2} \sum_{i=1}^{N} (y_i - f(x_i))^2 &= \min_w \frac{1}{2} ||Xw - Y||_2^2 \\ &= \min_w \frac{1}{2} (Xw - Y)^T (Xw - Y) \end{aligned}$$

Linear Regression

- The optimal coefficients can be derived by setting the first derivative to zero (cf. classification with linear discriminant functions)

$$w = (X^T X)^{-1} X^T Y$$

- For $d = 1$, the regression coefficients w_0 and w_1 can be computed as

$$w_1 = \frac{Cov(x, y)}{Var(x)} = \frac{\tilde{x}^T \tilde{y}}{\tilde{x}^T \tilde{x}}, \qquad w_0 = \bar{y} - w_1 \bar{x}$$

  where $\tilde{x} = x - \bar{x}$ and $\tilde{y} = y - \bar{y}$

- Note: If $\bar{x} = \bar{y} = 0$ (i.e., the data is centered), then $w_1 = \frac{x^T y}{x^T x}$ and $w_0 = 0$

Polynomial Regression

- Second order polynomial for $d = 1$:

$$\hat{y} = f(x) = w_0 + w_1 x_1 + w_2 x_2^2 = (1, x_1, x_2^2)^T w$$

with

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,1}^2 \\ \vdots & \vdots & \vdots \\ 1 & x_{N,1} & x_{N,1}^2 \end{pmatrix} \quad \text{and} \quad w = (X^T X)^{-1} X^T Y$$

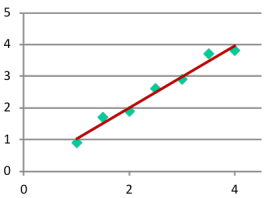
- Second order polynomial for $d = 2$:

$$\hat{y} = f(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

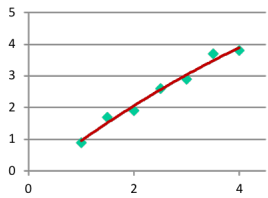
Polynomial Regression

- The number of coefficients increases exponentially with k and d
- Model building strategies: Forward selection, backward elimination
- The order of the polynomial should be as low as possible, high order polynomials tend to overfit the data

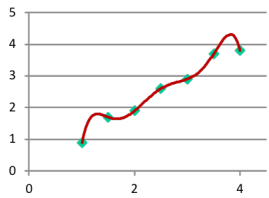
Linear model



Polynomial model 2^{nd} order



Polynomial model 6^{th} order



Nonlinear Regression

- Different nonlinear functions can be approximated
- Transform the data to a linear domain

$$\hat{y} = \alpha e^{\gamma x} \implies \ln \hat{y} = \ln \alpha + \gamma x$$
$$\implies \hat{y}' = w_0 + w_1 x$$

(for $\hat{y}' = \ln \hat{y}$, $w_0 = \ln \alpha$ and $w_1 = \gamma$)

- The parameters w_0 and w_1 are estimated with SSE
- The parameters α and γ are obtained, describing an exponential curve which passes through the original observations
- Problem: SSE determines normally distributed errors in the transformed space $\rightsquigarrow$ skewed error distribution in the original space

Nonlinear Regression

- Different nonlinear functions can be approximated
- Outputs are estimated by a function with nonlinear parameters, e.g., exponential, trigonometric
- Example type of function:

$$\hat{y} = w_0 + w_1 e^{w_2 x} + \sin(w_3 x)$$

- Approach: The type of nonlinear function is chosen and the corresponding parameters are computed
- No closed form solution exists $\rightsquigarrow$ numerical approximation:
  - Gauss Newton, Gradient descent, Levenberg-Marquardt

Linear and Nonlinear Regression

Problems

- Linear regression: Most of the real world data has a nonlinear behavior
- Polynomial regression: Limited, cannot describe arbitrary nonlinear behavior
- General nonlinear regression: The type of nonlinear function must be specified in advance

Agenda

Piecewise Linear Functions

Piecewise Linear Functions

For a partitioning of the input space $\mathcal{C} = \{C_1, \ldots, C_k\}$, a piecewise linear function is defined by coefficient vectors $\mathbf{w_i}$, and offset β_i for $i = 1, \ldots, k$.

$$f(\mathbf{x}) = \begin{cases} \mathbf{w}_1^T \mathbf{x} + \beta_1, & \mathbf{x} \in C_1 \\ \vdots & \\ \mathbf{w}_k^T \mathbf{x} + \beta_k, & \mathbf{x} \in C_k \end{cases}$$

Piecewise Linear Functions

Properties

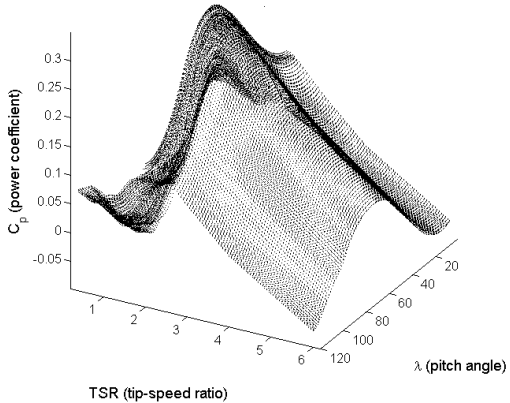
- Simple approach
- Can approximate any function
- *Accuracy increases* with increasing number of partitions
- *Compactness & Interpretability decreases* with increasing number of partitions

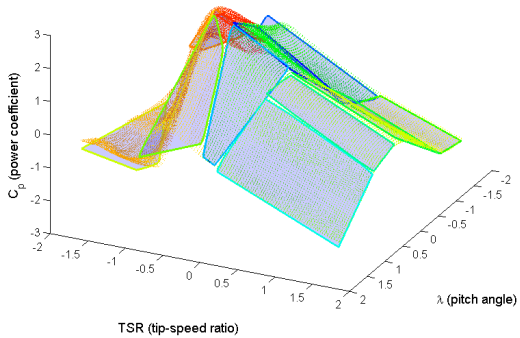
Challenge

Find an appropriate partitioning of the input space

Regression Tree

Greedy divide and conquer: recursive partioning of the input space.



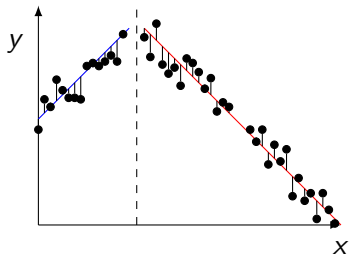


Regression Tree

General Approach

- Given: Set of observations $T = (X, Y)$ with $X = \{x_1, \ldots, x_n\}$, and $Y = \{y_1, \ldots, y_n\}$
- Find a split of T into T_1, T_2 with minimal summed impurity $imp(T_1) + imp(T_2)$.
- If the stopping criterion is not reached: repeat for T_1 and T_2
- If the stopping criterion is reached: undo the split

Impurity Measure



Variance of the Residuals

$$imp(T) = \frac{1}{|T|} \sum_{(\mathbf{x},y)\in T} (y - f(\mathbf{x}))^2$$

where f is the approximator function, i.e. here a linear function. For constant f, this measure coincides with the variance of the output.

Stopping Criterion: Impurity Ratio

Impurity Ratio Stopping Criterion

The recursive splitting is stopped if one of the following holds

- The sample size of a node is below some specified threshold
- The split is *not significant* for threshold τ_0, when

$$\tau = \frac{imp(T_1) + imp(T_2)}{imp(T)} \geq \tau_0$$

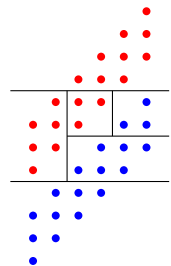
Choosing τ_0

τ_0 controls the overfitting/underfitting trade-off:

- τ_0 too large $\implies$ stopping too early $\implies$ model not accurate enough
- τ_0 too small $\implies$ stopping too early $\implies$ model overfits to observations

Split Strategy

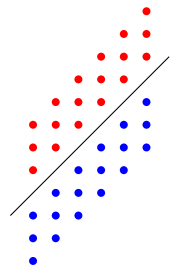
- The split strategy determines how the training samples are partitioned, whether the split is actually performed is decided by the stopping criterion.
- The most common splits are *axis parallel*:
  - Split = a value in one input dimension
  - Compute the impurity of all possible splits in all input dimensions and choose at the end the split with the lowest impurity
  - For each possible split compute the two corresponding models and their impurity $\rightsquigarrow$ expensive to compute



Five axis-parallel splits to separate red from blue samples.

Strategy for Oblique Splits

- More intuitive to use oblique splits
- An oblique split is a linear separator in the input space instead of a split value in an input dimension
- The optimal split (with minimal impurity measure) cannot be efficiently computed $\rightsquigarrow$ Heuristic approach required

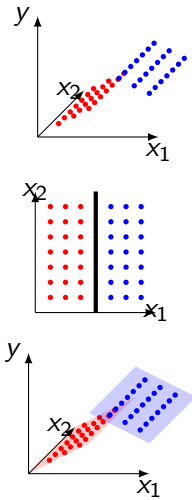


A single oblique split can separate red from blue samples.

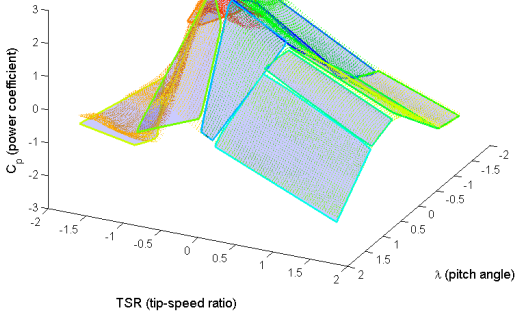
Strategy for Oblique Splits

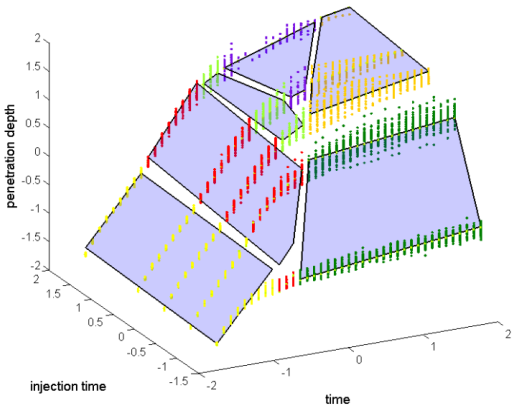
Heuristic Approach

1. Compute a clustering in the full (input + output) space, such that the samples are as well as possible described by linear equations
2. Project the clusters onto the input space
3. Use the clusters to train a linear classifier in the input space. Split = separating hyperplane in input space
4. Compute linear models for the two linearly separated clusters



Example: Oblique Splits





Continuous Splits

Motivation

- At the boundaries of the partitions the prediction may be discontinuous.
- In some applications this might by undesirable, e.g. when using the prediction to control engine speed, a rapid jump may cause damage to the engine

Solution

Hinging Hyperplane Models (not detailed in this lecture).

$$f(\mathbf{x}) = \sum_{i=1}^{k} h_i(\mathbf{x}) \qquad h_i(\mathbf{x}) = \begin{cases} \left\langle \mathbf{w}^{(i,+)}, \tilde{\mathbf{x}} \right\rangle, & \left\langle \mathbf{w}^{(i)}, \tilde{\mathbf{x}} \right\rangle > 0 \\ \left\langle \mathbf{w}^{(i,-)}, \tilde{\mathbf{x}} \right\rangle, & \left\langle \mathbf{w}^{(i)}, \tilde{\mathbf{x}} \right\rangle \leq 0 \end{cases} \qquad \begin{array}{l} \tilde{\mathbf{x}} = (1, x_1, \ldots, x_d) \\ \mathbf{w}^{(i)} = \mathbf{w}^{(i,+)} - \mathbf{w}^{(i,-)} \end{array}$$

Announcements

First Exam Inspection

- Time: Tu, 12.03.19, 10:00 - 11:30
- Place: Oettingenstr. 67, room 157

Second Exam

- Time: Mo, 18.03.19, 16:00 - 18:00
- Place: M218 A240 (HGB, Geschw.-Scholl-Pl. 1)

For further announcements, please check the course website:
http://www.dbs.ifi.lmu.de/cms/studium_lehre/lehre_master/kdd1819/index.html

Agenda

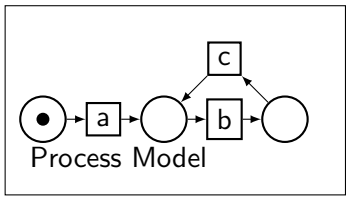
Motivation

supports/controls

configures

log events

model analyses



discovery

conformance

enhancement

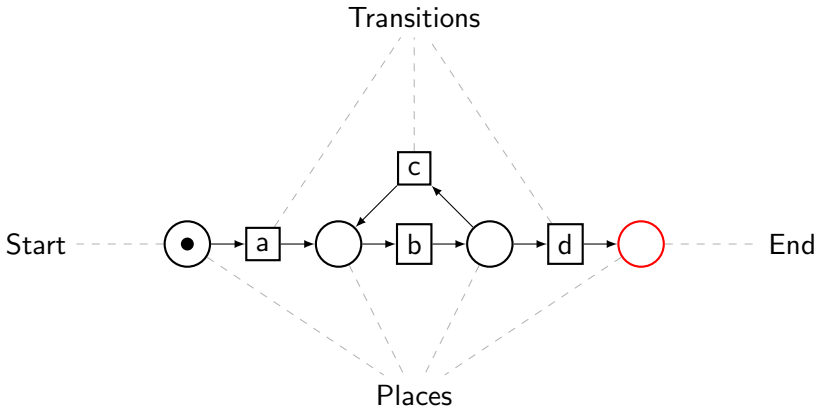
ID	Time	Location	Piece
12	14.01.2018 10:32	MUNICH	1
42	14.01.2018 11:40	MUNICH	2
5	14.01.2018 15:17	MUNICH	3
12	14.01.2018 22:12	LEIPZIG	1
...			

Notions

- Process: System of actions, movements (e.g. sign document, customer call, financial transaction, delivery of goods)
- Different instances/cases should follow a common process description
- Each case contains actions as events (their sequence is called *trace*)
- An event is represented by at least
  - A case identifier
  - An activity label
  - A timestamp

  but may also comprise additional (meta-)information (e.g. involved (work) resources)

Petri Nets as Process Model



Tasks

Main Tasks

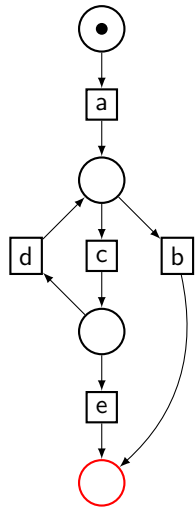
1. *Process Discovery*:
   Mine multiple sequences of actions to derive a workflow pattern
2. *Conformance Checking*:
   Use previously mined model to judge the validity of a new case
3. *Process Enhancement*:
   Evolve models with new data, find deviations

Process Discovery

Input	
#	trace
2048	ace
1234	acdce
404	acdcdce
120	acdcdcdce
42	ab
5	acdb

Quality Measures	
Fitness	ability to replay the log
Simplicity	simplified as much as possible
Generalization	no underfitting of log
Precision	no overfitting of log

Output



Example Discovery Algorithm: α-Miner[22]

1. Scan the log for all activities
2. For each pair of activities and , we define the relations
   - $a > b$ if for some case a is immediately followed by b (direct succession)
   - $a \parallel b$ if $a > b$ and $b > a$ (parallelism)
   - $a \rightarrow b$ if $a > b$ and not $b > a$ (causality)
   - $a \# b$ if not $a > b$ and not $b > a$
3. All activities, having only $\#$ or $\rightarrow$ in their row are starting activities. They are collected in T_{in}.
4. Analogously, $\#$ or $\leftarrow$ determine T_{out}.

Example: $\{abcd, acbd, acd\}$

	a	b	c	d
a		$\rightarrow$	$\rightarrow$	$\#$
b	$\leftarrow$		$\parallel$	$\rightarrow$
c	$\leftarrow$	$\parallel$		$\rightarrow$
d	$\#$	$\leftarrow$	$\leftarrow$	

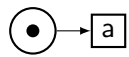
$T_{in} = \{a\}$, $T_{out} = \{d\}$

[22] van der Aalst, Weijters, Maruster (2003). "Workflow Mining: Discovering process models from event logs", IEEE Transactions on Knowledge and Data Engineering, vol 16

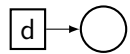
Example Discovery Algorithm: α-Miner

1. Prepare a Petri net. The set of transitions is equal to activities
2. A starting place is created and connected to each node in T_{in}
3. Also, a final place is created and each node in T_{out} is connected to it
4. Determine all pairs of sets A and B, such that
   - $\forall a_1, a_2 \in A : a_1 \# a_2$
   - $\forall b_1, b_2 \in B : b_1 \# b_2$
   - $\forall a \in A, b \in B : a \rightarrow b$
5. A place is added in between A and B and connected accordingly

2.

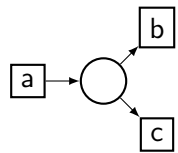


3.



4. $A = \{a\}, B = \{b, c\}$

5.



Conformance Checking

Use previously mined model to judge the validity of a new case (similar to binary classification: valid vs. invalid)

Input

- Model
- Trace

Aims

- Model reasoning
- auditing
- security (fraud detection)

Example Conformance Checking Algorithm: Token-Replay

Replay the event in the model. Count:

- the number of produced tokens (p)
- the number of consumed tokens (c)
- the number of missing tokens (m)
- the number of remaining tokens (r)

Output a *fitness* value

$$f = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

The fitness value ranges between 0 and 1, where 1 is a perfect match.

Agenda

Further Machine Learning Methods

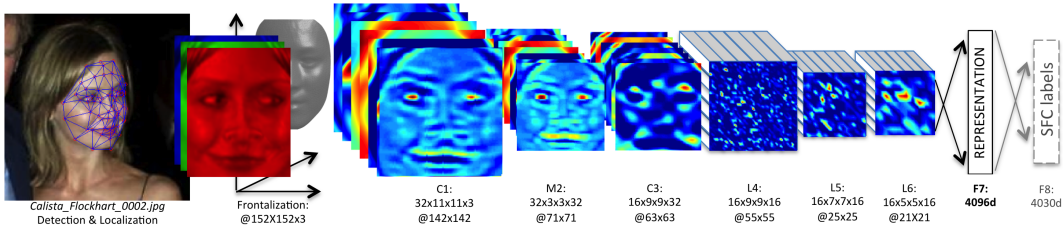


Image Source: Taigman, et al. "Deepface: Closing the gap to human-level performance in face verification." CVPR'14.

- Graphical Models
- Generative Models
- Neural Networks
- Deep Learning

$\rightsquigarrow$ Machine Learning (SS), Deep Learning and Artificial Intelligence (WS)

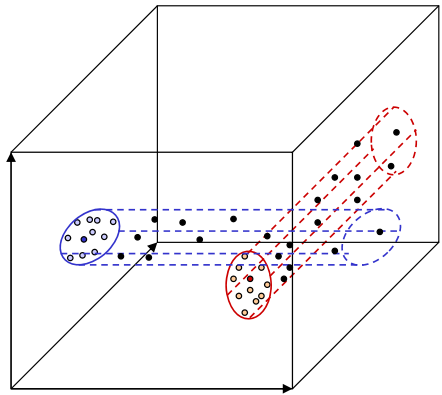
Decision Making / Planning

- Setting:
  - Agents are in some environment, observe, and have to take actions that influence the environment.
- Methods:
  - Deterministic/Stochastic Planning
  - A^*-Search
  - Model-Free Reinforcement Learning
  - Q-Learning
  - Adversarial Search (e.g. Alpha-Beta Pruning)

$\rightsquigarrow$ Deep Learning (WS), Managing Massive Multiplayer Online Games (SS)

High-Dimensional Data

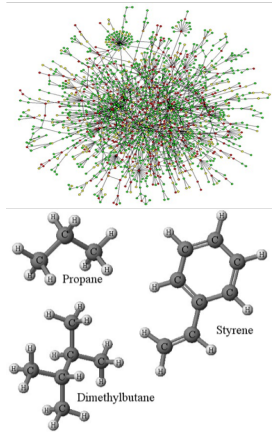
- Challenges:
  - *Curse of dimensionality*: distances become more and more similar
  - Datasets become sparse.
  - Expensive distance measures
  - Degeneration of index structures
  - Unintuitive properties in high dimensions.
- Tasks
  - Feature Selection
  - Feature Reduction / Metric Learning
  - Clustering in High-Dimensional Spaces



$\rightsquigarrow$ Knowledge Discovery in Databases II (SS), Big Data Management and Analytics (WS)

Graph Data

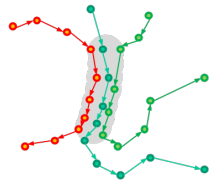
- Graphs are everywhere!
  - Chemical data analysis, proteins
  - Biological pathways/networks
  - Program control flow, traffic flow
  - Web graph, social network analysis
- Typical tasks
  - Measure similarity between graphs
  - Find frequent patterns in graphs
  - Generate "realistic" synthetic graphs
  - Identify groups in social networks
  - Integrate additional information



$\rightsquigarrow$ Knowledge Discovery in Databases II (SS), Big Data Management and Analytics (WS)

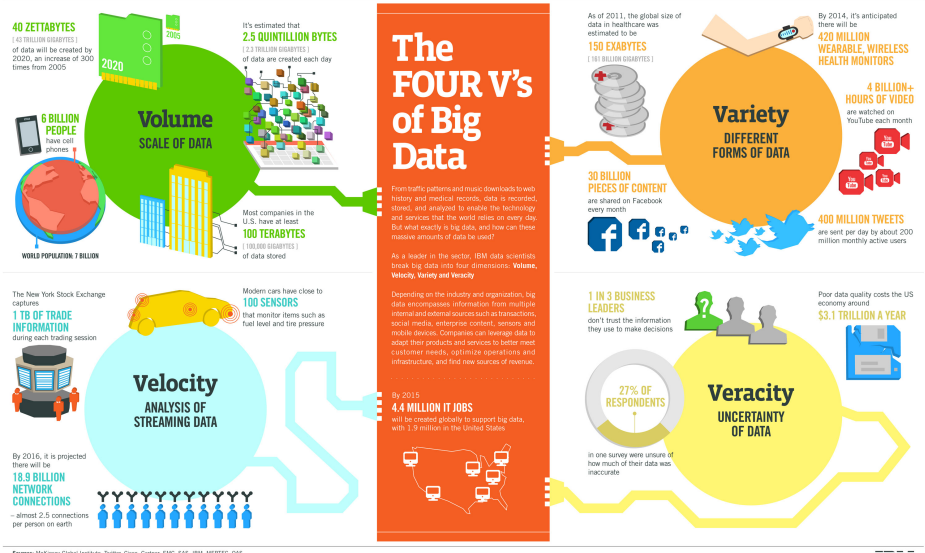
Spatial Data

- Mining spatial data
  - Spatial clustering, outlier detection, prediction, rule mining, ...
- Spatial data management
  - Process spatial queries without scanning the whole database
  - Spatial index structures: BSP-tree, R-tree, Quad-tree, ...
- Mining trajectory data
  - Similarity models for trajectories
  - Trajectory compression
  - Mining patterns in trajectories (encounters, flocks, ...)



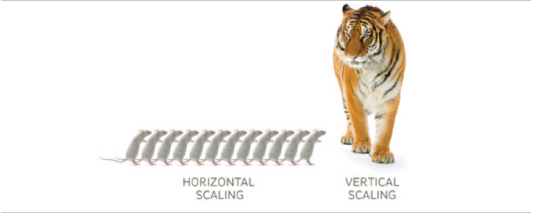
⇝ Managing Massive Multiplayer Online Games (SS)

Big Data

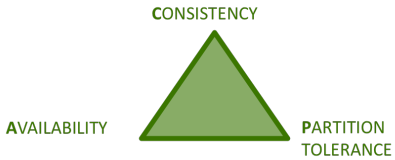

40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005
2005
2020
Volume
SCALE OF DATA
6 BILLION PEOPLE
have cell phones
WORLD POPULATION: 7 BILLION
It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day
Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored
The FOUR V's of Big Data
From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?
As a leader in the sector, IBM data scientists break big data into four dimensions: Volume, Velocity, Variety and Veracity
Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.
By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States
As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]
Variety
DIFFERENT FORMS OF DATA
30 BILLION PIECES OF CONTENT
are shared on Facebook every month
By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS
4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month
400 MILLION TWEETS
are sent per day by about 200 million monthly active users
The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session
Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure
Velocity
ANALYSIS OF STREAMING DATA
By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth
1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions
27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate
Veracity
UNCERTAINTY OF DATA
Poor data quality costs the US economy around
$3.1 TRILLION A YEAR
Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS
IBM.

Big Data Management

- Vertical scaling limited and expensive
  ⇝ Distributed storage
- NoSQL databases
  - Redis
  - MongoDB
  - Cassandra
  - Neo4J
- Distributed file systems
  - GFS (Google)
  - HDFS (Hadoop)
  - S3 (Amazon)



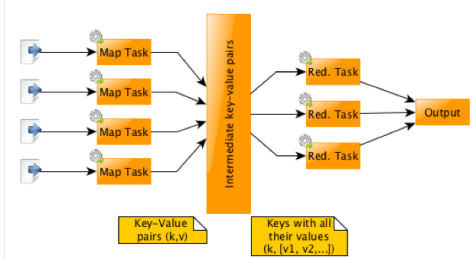
https://www.greentree.com/latest-news/avoiding-cumulus-congestus



⇝ Big Data Management and Analytics (WS)

Distributed Data Processing

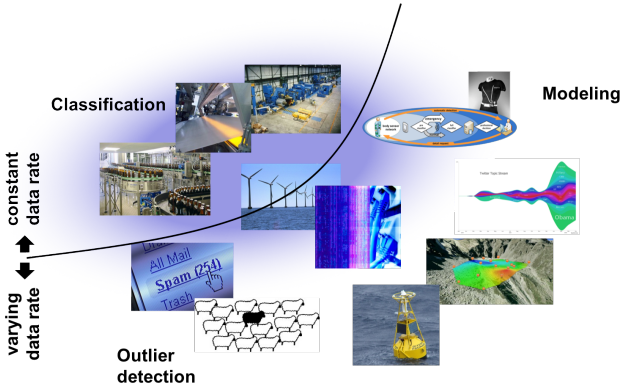
- Processing and analyzing big data
- Map-Reduce: Programming model for distributed processing of large datasets
  - Algorithms are specified as sequences of map and reduce functions
  - Programs are automatically parallelized and executed on a cluster
  - System is tolerant to hardware faults
- Frameworks
  - Apache Spark (batch processing)
  - Apache Flink (stream processing)



⇝ Big Data Management and Analytics (WS)

Stream Data

- Data objects arrive over time in a continuous data stream
- Challenges
  - Infinite stream
  - Limited time and memory
  - Evolving distribution
  - Varying data rates
  - Concept drift
- Typical tasks
  - Sampling and buffering
  - Stream statistics
  - Aging mechanisms



⇝ Knowledge Discovery in Databases II (SS), Big Data Management and Analytics (WS)

Seminars, Practicals, Theses

Dive deeper into specific topics and get hands-on experience:

- Master Seminar "Recent Developments in Data Science" (SS)
- Master Practical "Big Data Science" (SS)
- Master Practical "Applied Reinforcement Learning" (SS)
- Individual Bachelor and Master Theses