OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

DKE Data & Knowledge Engineering Group

Adaptivity in Audio and Music Retrieval

Adaptive Hierarchies: Constrained Clustering and Utility

Andreas Nürnberger and Sebastian Stober

Data & Knowledge Engineering Group, Faculty of Computer Science

Otto-von-Guericke-Universität Magdeburg, Germany

Email: andreas.nuernberger@ovgu.de, sebastian.stober@ovgu.de

Outline

- Day 1: Adaptation and Personalization: Concepts and Challenges
- Day 2: Adaptive Music Retrieval: An Overview
- **Day 3: Adaptive Hierarchies: Constrained Clustering and Utility**
- Day 4: Adaptive Similarity
- Day 5: User Interfaces and Gamification: Design and Evaluation

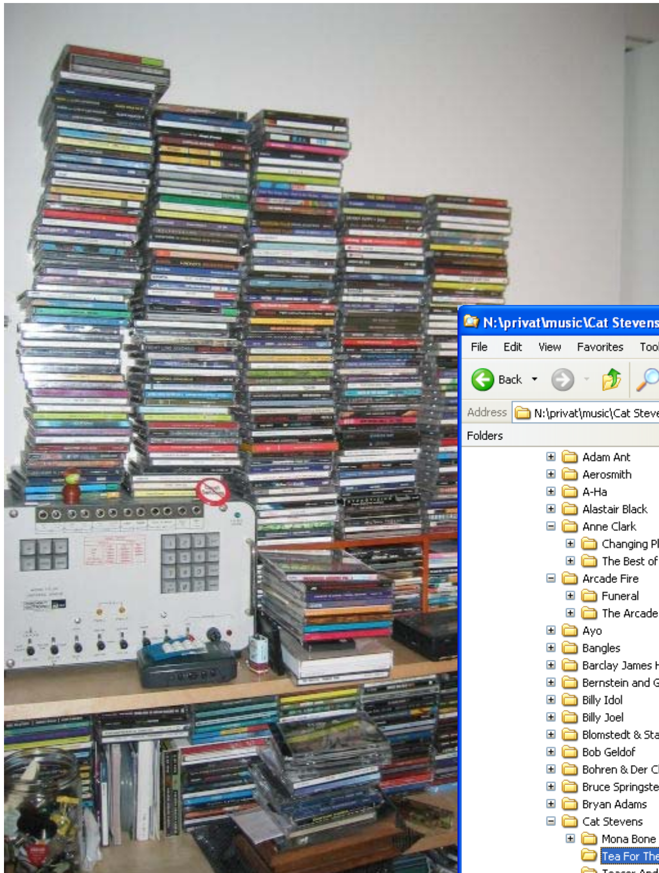
Overview (Day 3)

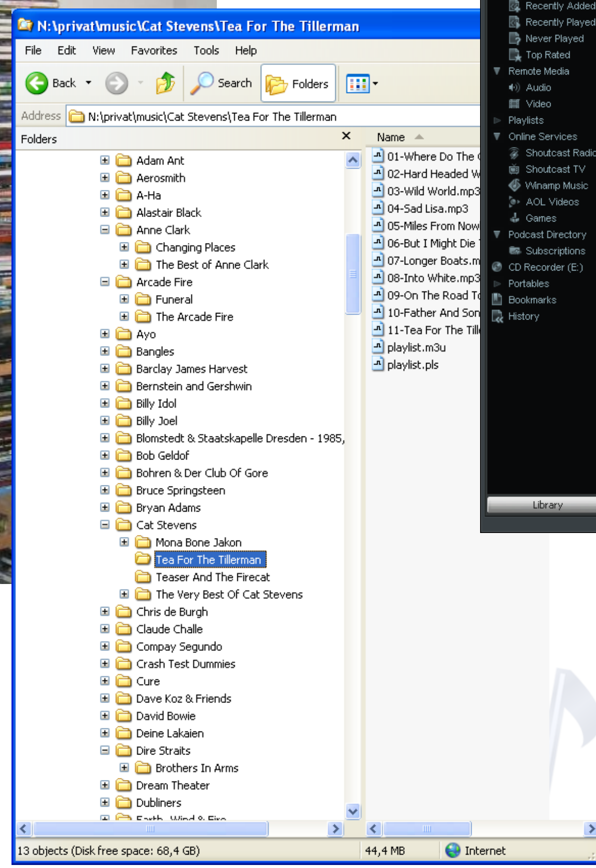
- **Motivation**
- Constrained Clustering
- A Utility based Approach

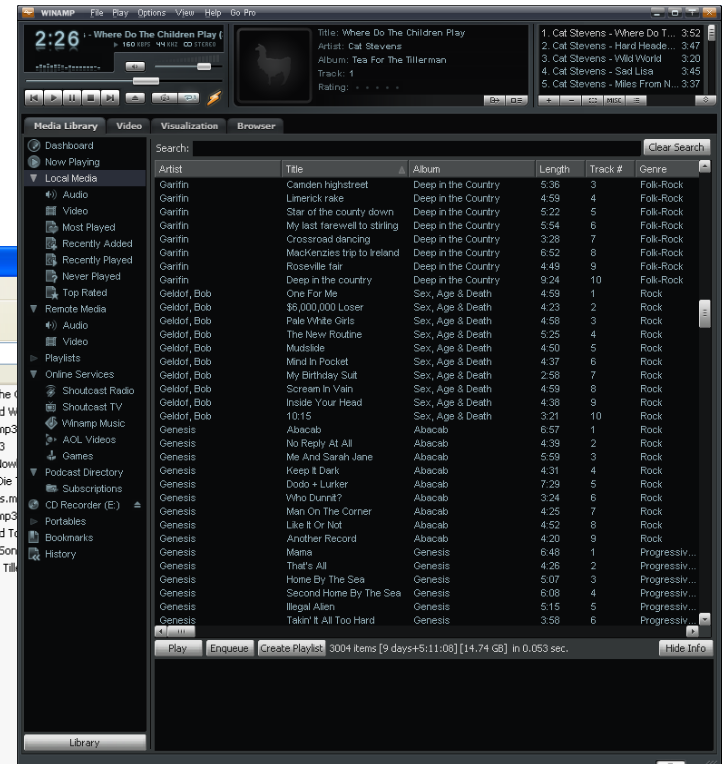
Personalized Structuring…

- How can we support a user in structuring a collection of objects, *such that the structure reflects the classification criteria of the user*?

Very often still...





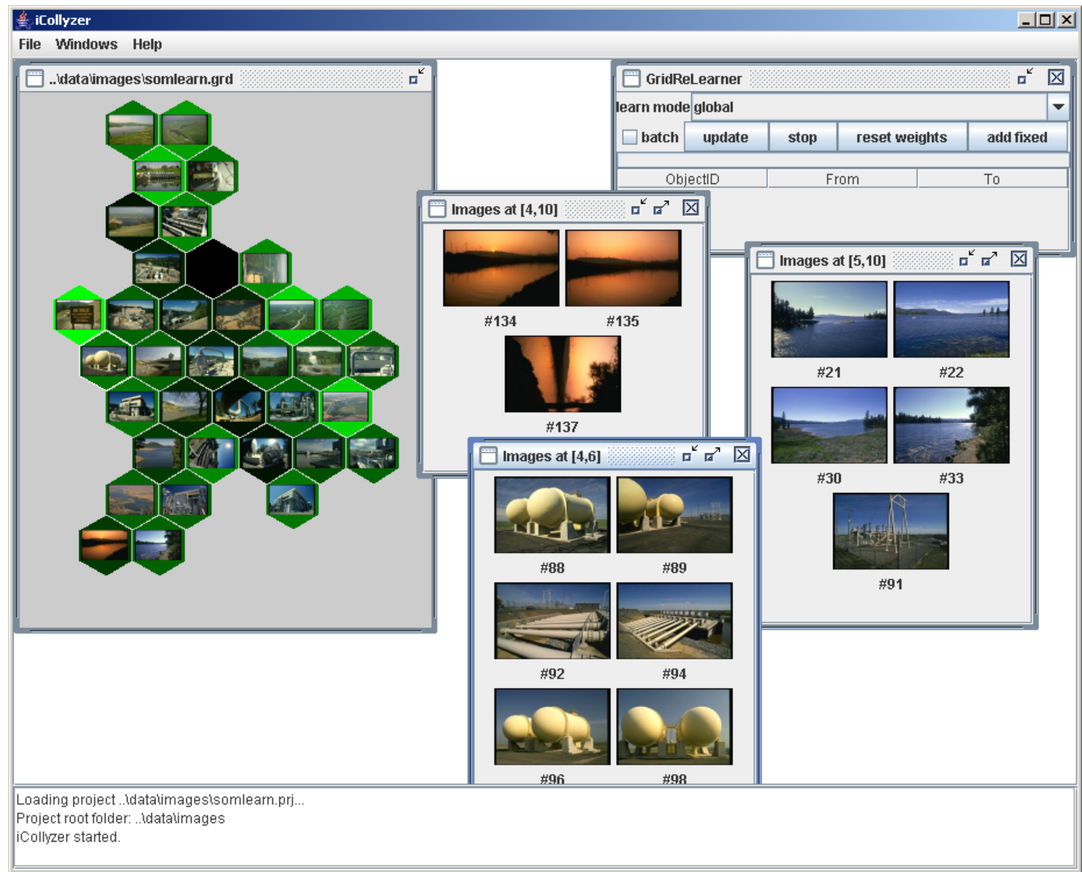


iCollyzer (Interactive COLLection organiZER)

- Interactive tool to explore image collections

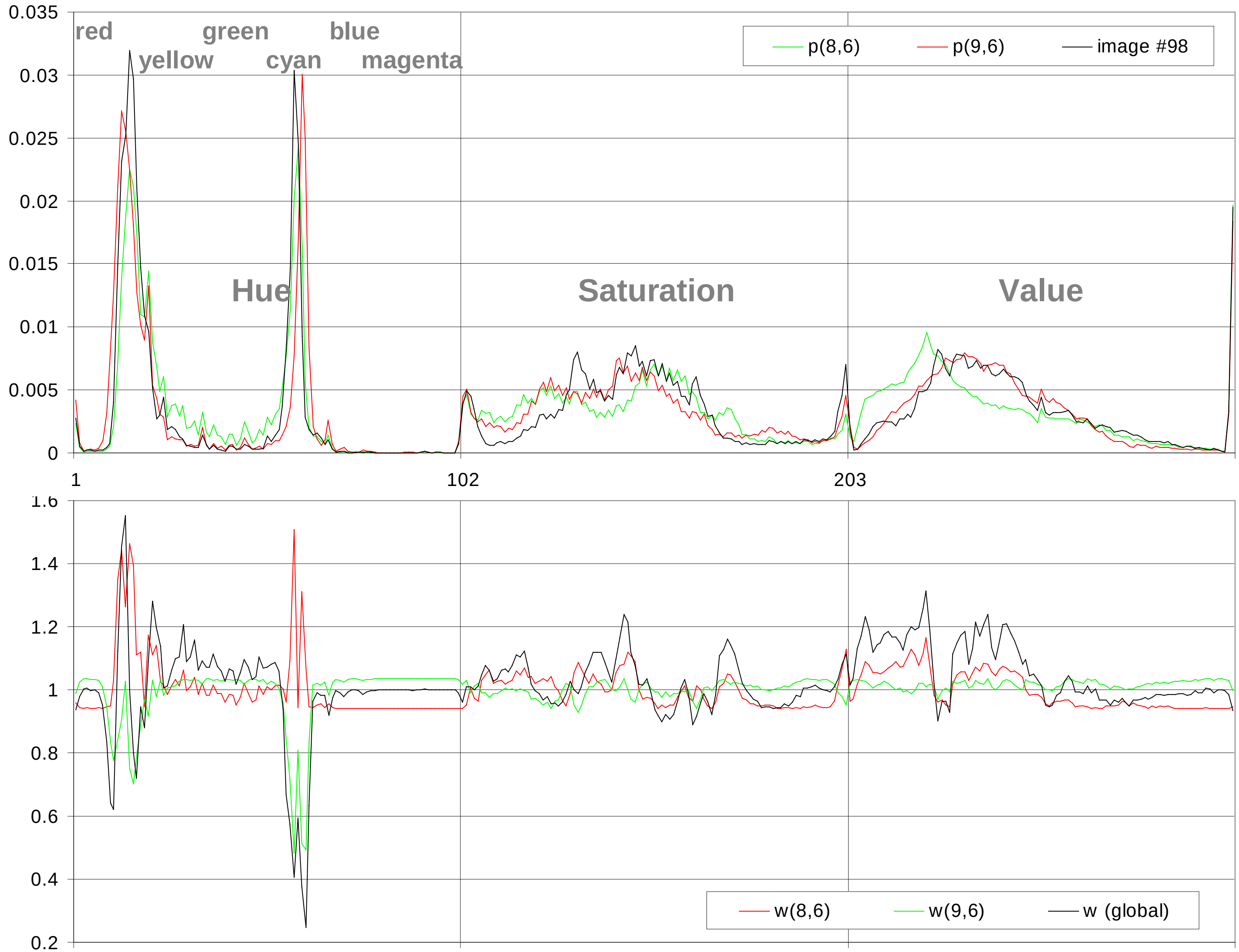
Basic concepts:

- Initial overview by clustering (SOM)
- Personalization during user interaction (move images inbetween groups)
- Learning: Adaptation of similarity measure



A. Nürnberger and A. Klose, Improving Clustering and Visualization of Multimedia Data Using Interactive User Feedback, in: *Proc. of 9th Intl. Conf. on Inform. Proc. and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, pp. 993-999, 2002.

iCollyzer (Interactive COLLection organiZER)



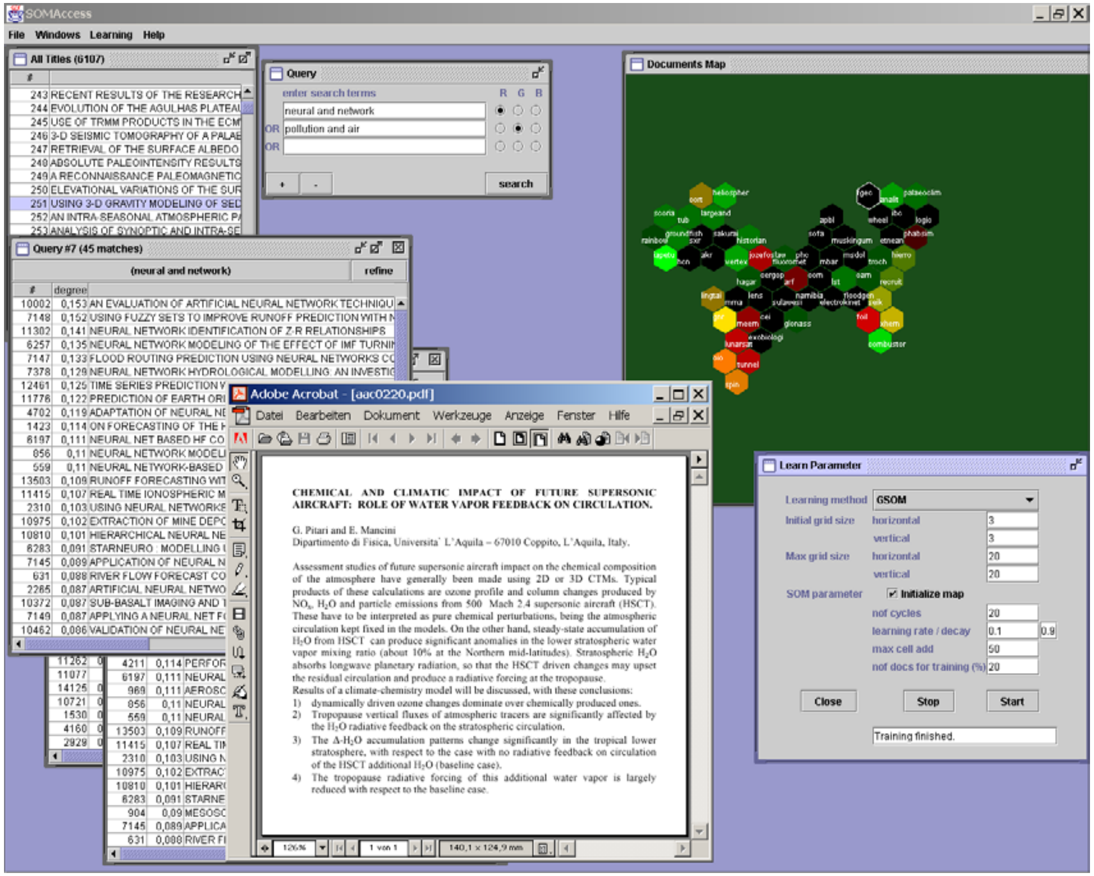
Extracted features and computed weights

iCollyzer (Interactive COLLection organiZER)

- Interactive tool to explore text collections

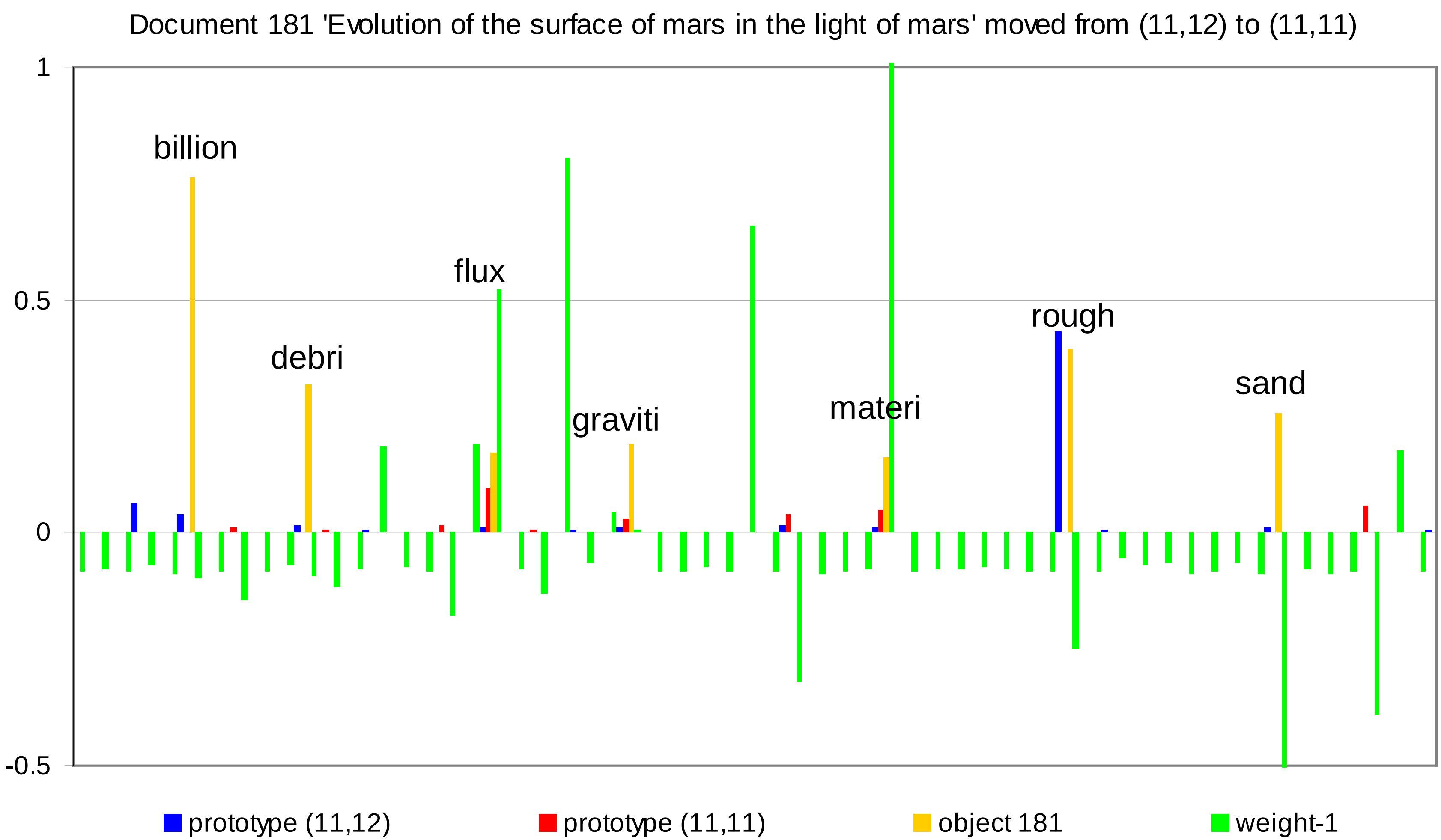
Basic concepts:

- Initial overview by clustering (SOM)
- Personalization of groups by remapping of documents
- Learning: Adaptation of similarity measure
- Search (retrieval)



A. Nürnberger and M. Detyniecki, Externally growing self-organizing maps and its application to e-mail database visualization and exploration, *Applied Soft Computing*, 6:4, pp. 357-371, Elsevier Science, 2006.

iCollyzer (Interactive COLLection organiZER)



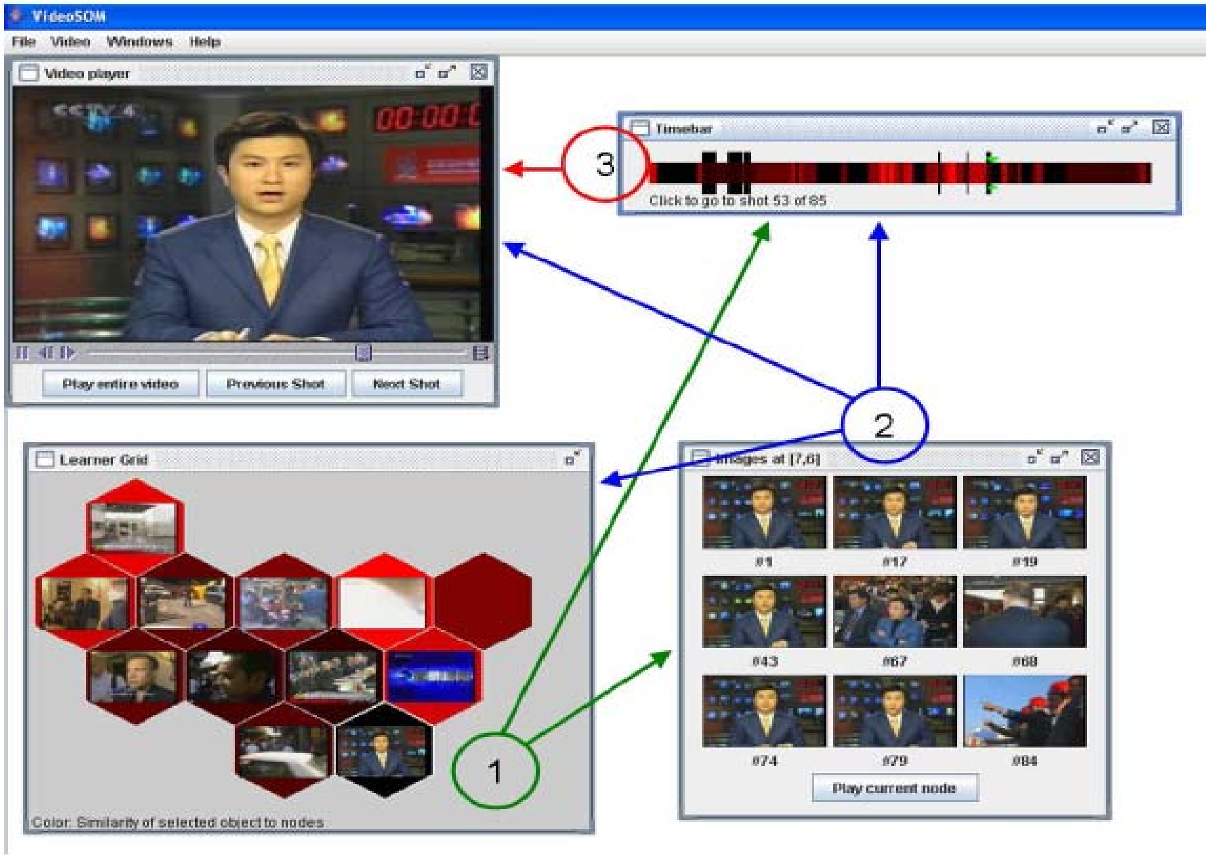
Extracted features and computed weights

VideoSOM (based on iCollyzer)

- Interactive tool to navigate in videos

Basic concepts:

- Initial overview by clustering of key frames (1)
- Navigation support by time bar that visualizes (adaptable) frame similarities (2)
- Direct navigation in video (3)



T. Bärecke, E. Kijak, A. Nürnberger and M. Detyniecki, Video Navigation based on Self-Organizing Maps, in: *Proc. of Intl. Conf. on Image and Video Retrieval (CIVR 2006)*, pp. 340-349, Springer Verlag, 2006.

AUCOMA (based on iCollyzer revision)

- Interactive tool to navigate in music collections

S. Stober and A. Nürnberger, AUCOMA - Adaptive Nutzerzentrierte Organisation von Musikarchiven, in: *Proc. of Deutsche Jahrestagung für Akustik (DAGA 2008),* 2008.

Adapting cluster structure

- Problem:
  - When user moves (reassigns) objects, this has to be considered by the system
- Idea:
  - Adapt similarity measure such that reassigned objects are correctly assigned / classified
  - Approaches:
    - Heuristics
    - Quadratic optimization

A. Nürnberger and S. Stober, User Modelling for Interactive User-Adaptive Collection Structuring, in: *Proc. of Intl. Workshop on Adaptive Multimedia Retrieval (AMR 2007)*, 2007.

Weighted similarity measure

- Feature weights w_l to "personalize" similarity:

$$sim(x_j, x_k) = \sum_{l=1}^{m} x_{jl} \cdot w_l \cdot x_{kl}$$

- Cluster assignment
  - Document *d* assigned to cluster c_s:

$$sim(c_s, d) \leq sim(c_i, d) \quad \forall i \neq s$$

  - Moving *d* to cluster c_t:

$$sim(c_t, d) \leq sim(c_i, d) \quad \forall i \neq t$$

- Problem:
  - change weights such that:

$$\sum_{l=1}^{m} x_{jl} \cdot w_l \cdot c_{sl} > \sum_{l=1}^{m} x_{jl} \cdot w_l \cdot c_{tl} \quad \forall s \neq t$$

Quadratic Optimization

- minimize change of weight vector *w*

$$\min_{w \in \Re^m} \sum_{l=1}^{m} (w_l - 1)^2$$

- weights should be non-negative

$$w_l \geq 0 \quad \forall 1 \leq l \leq m$$

- sum of the weights should be *m* (dictionary size)

$$\sum_{l=1}^{m} w_l = m$$

- keep all manually moved objects at their position

$$\sum_{l=1}^{m} x_{jl} \cdot w_l \cdot c_{sl} > \sum_{l=1}^{m} x_{jl} \cdot w_l \cdot c_{tl} \quad \forall s \neq t$$

Experimental Setup

- User study:
  - expensive, time consuming, not objective
- Alternative way: simulate user actions

- 2 datasets:
  - 1914 documents from a scientific news achive represented by 800 index terms
  - 10% (1000 documents) from the Banksearch dataset represented by 800 index terms

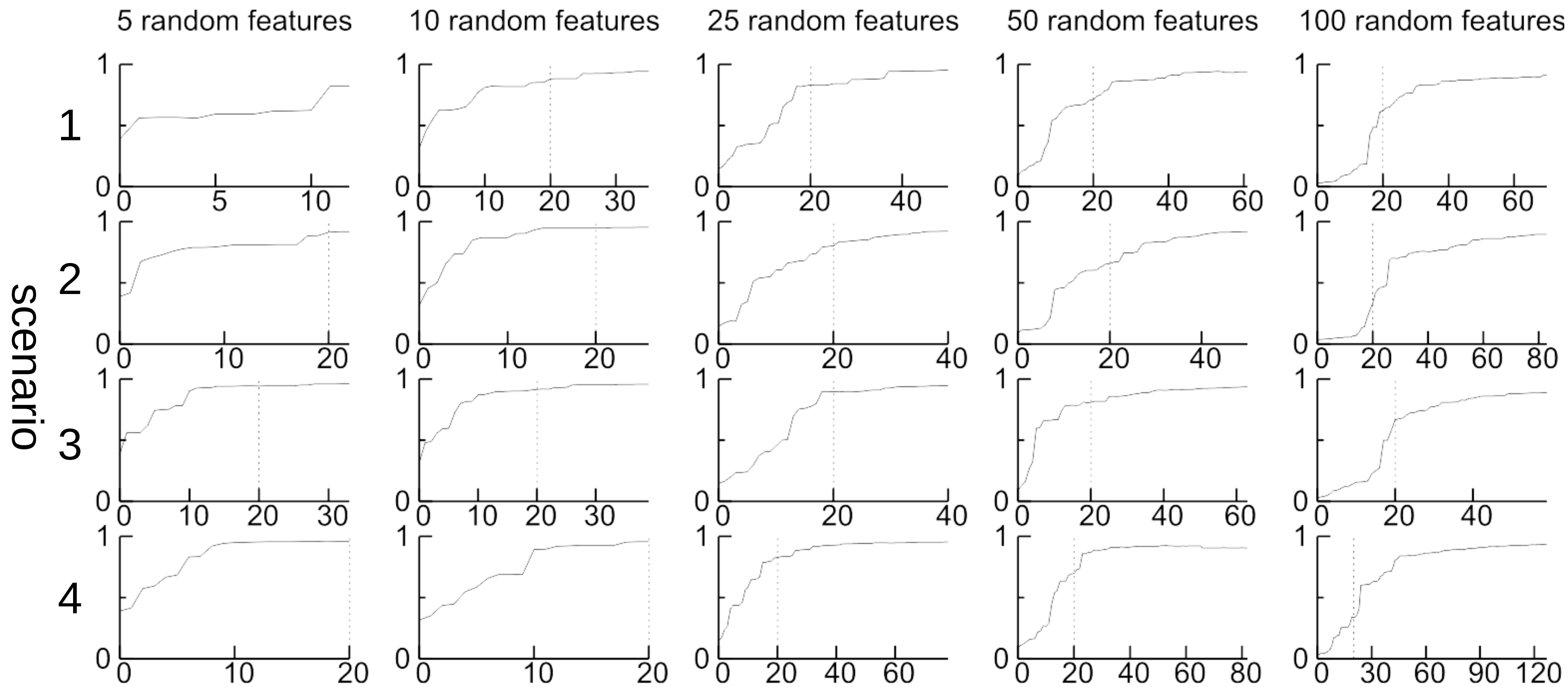
User Simulation

modify objects by adding random features
learn map on modified objects
repeat
    select an object o to be moved
    select most similar cell c for o according to user
    move o to c
until o could not be moved

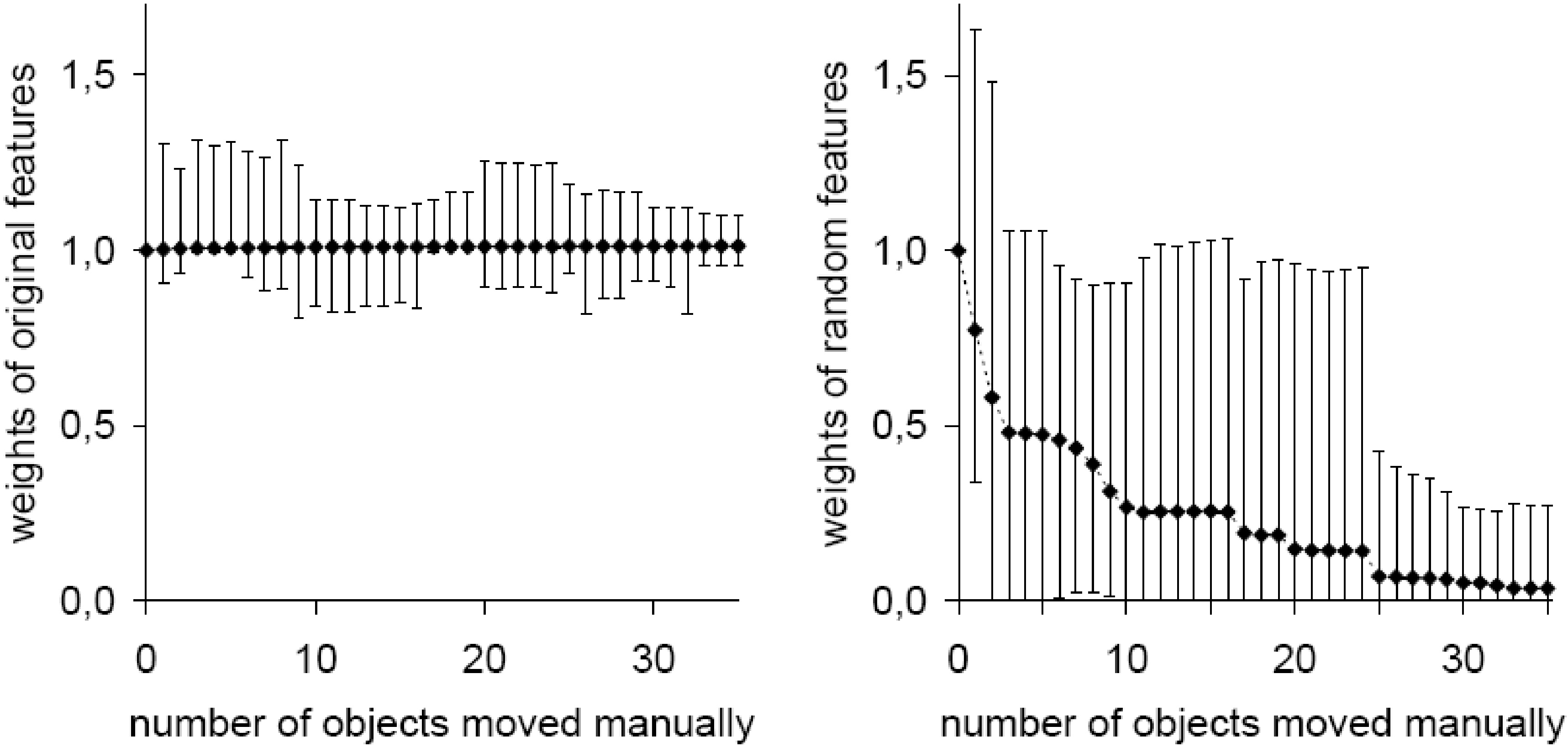
		cell selection	
		greedy	random
object selection	greedy	scenario 1	scenario 3
	random	scenario 2	scenario 4

Results for Experiment 1

- Top-N precision increased to 0.82-0.97 (mean 0.93)
- Moving ~1% of the collection was sufficient
- Random selection did not yield worse results

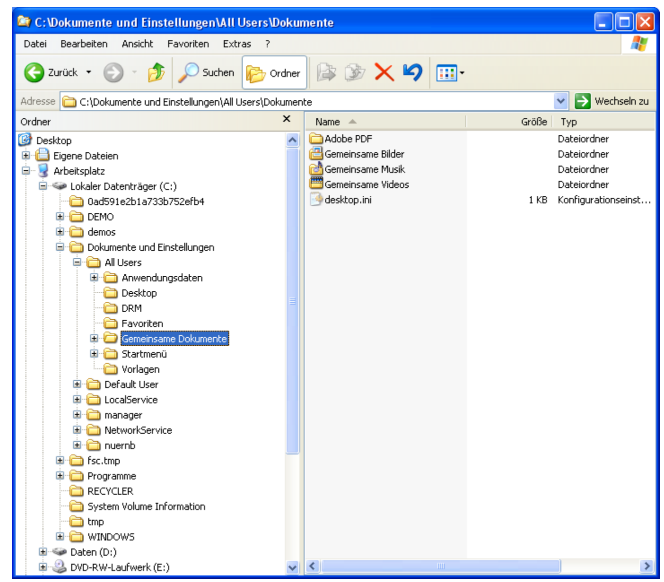


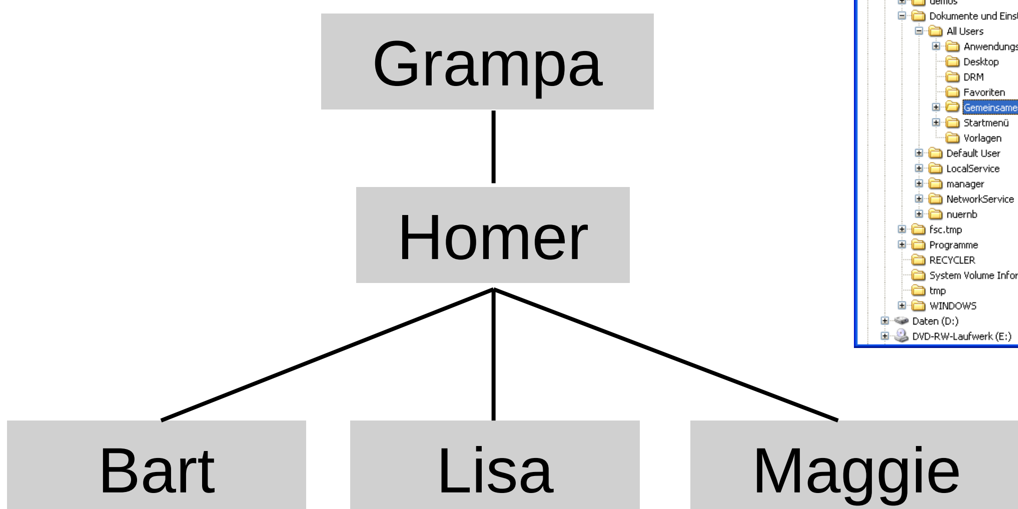
Weight changes



Hierarchical Organization

- Flat structures very often not sufficient
- Many systems/methods used for information organization can be mapped to hierarchies:
  - file systems, bookmark structures, (book)shelves by tree maps, …





Hierarchical Organization

- Hierarchies can support a user in
  - Search, Exploration and Browsing
- Problems of “manual” organization
  - Categorization of objects very time consuming
  - Creation of the hierarchy
    - might even depend on user and/or context
- Machine learning approaches that can be used
  - Hierarchical classification/categorization:
    - structure known
    - automatic insertion of documents
  - Hierarchical clustering:
    - structure not or only partially known
    - automatic creation of hierarchy

Personalization…

- How can we use information about the way a *user* is *structuring collections* in order to support him in *structuring yet unknown collections*?

Hierarchical clustering

- Problem:
  - Automatically created structure of a data collection should reflect (as good as possible) the structure that would be created by a specific user,

  but
  - desired structure is only partially (or not at all) known

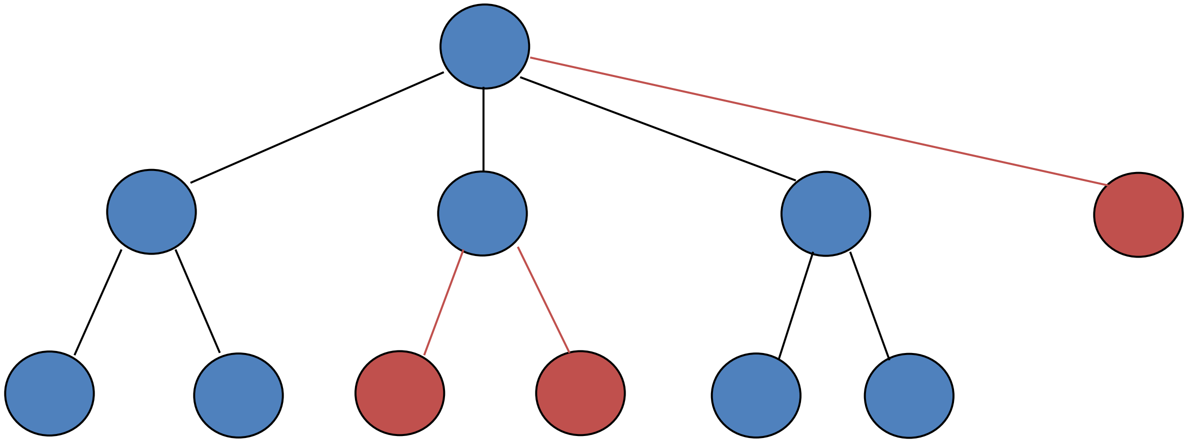
- Approach:
  - Modeling structuring criteria of a user by *constraints*
  - Transfer ideas of “flat” *constrained-based clustering* approaches to hierarchical agglomerative clustering
  - Use *constraints* in order to learn feature weights

K. Bade and A. Nürnberger, Personalized Hierarchical Clustering, in: *Proc. of IEEE/WIC/ACM Intl. Conference on Web Intelligence (WI-06)*, pp. 181-187, IEEE Computer Society Press, 2006.

K. Bade und A. Nürnberger, Creating a Cluster Hierarchy under Constraints of a Partially Known Hierarchy, in: *Proc. of 2008 SIAM International Conference on Data Mining*, 2008.

Task

- Given user hierarchy is a “guideline”
- Hierarchy refinement with
  - New classes (sibling nodes to existing ones)
  - New sub-classes (refinement in depth)



Learning Scenario

- Classes $C = C_k \bigcup C_u$
- Relations between classes (tree structure)

$$R_{Hk} \subset R_H = \{(c_1, c_2) \in C \times C / c_1 \geq_H c_2\}$$

- Documents

$$D = D_k \bigcup D_u$$

$$T_k = \{(d,c) \in D_k \times C_k\} \qquad T_u = \{(d,c) \in D_u \times C_u\}$$

- Two-fold notion of semi-supervised
  - Partially labeled training data
  - Not all classes are known
- Approach: Constrained Clustering…

Overview (Day 3)

- Motivation
- **Constrained Clustering**
- A Utility based Approach

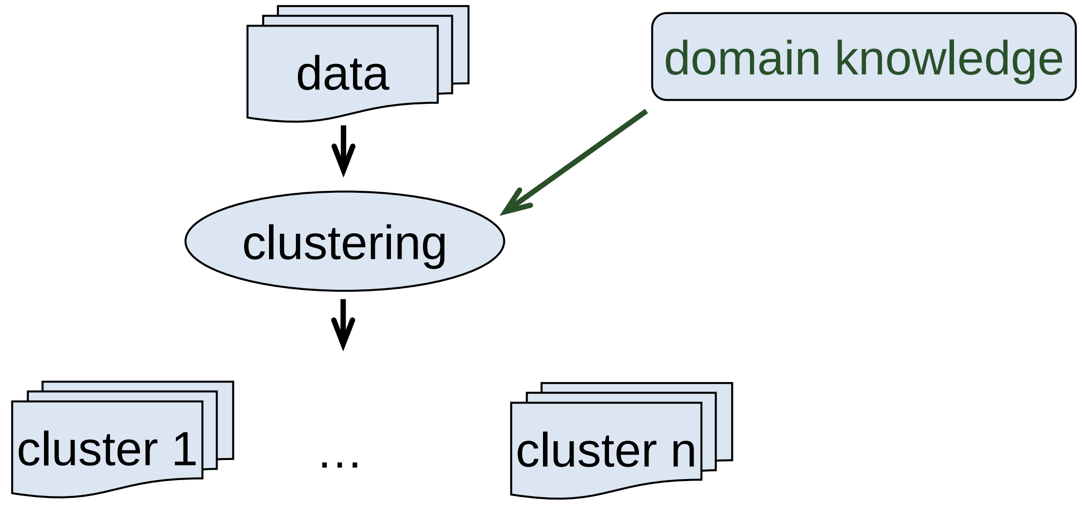
Constrained Clustering

- Clustering aims at uncovering a structure that could be be used to organize data
- Problem: There are often several different meaningful structures
- Examples:
  - Movies structured by genre, actors, director, …
  - Photos structured by time, people shown, places, moods, motives, …

- The same or a different cluster?
  - “Sleepless in Seattle” – “You’ve Got Mail”
    - Same cluster, because both films are very similar,
    - e.g. romantic comedies with same director and same stars
  - “Charlie and the Chocolate Factory” – “Nightmare on Elm Street”
    - Different clusters, because both films are from different genres
    - Same cluster, because both are directed by Johnny Depp

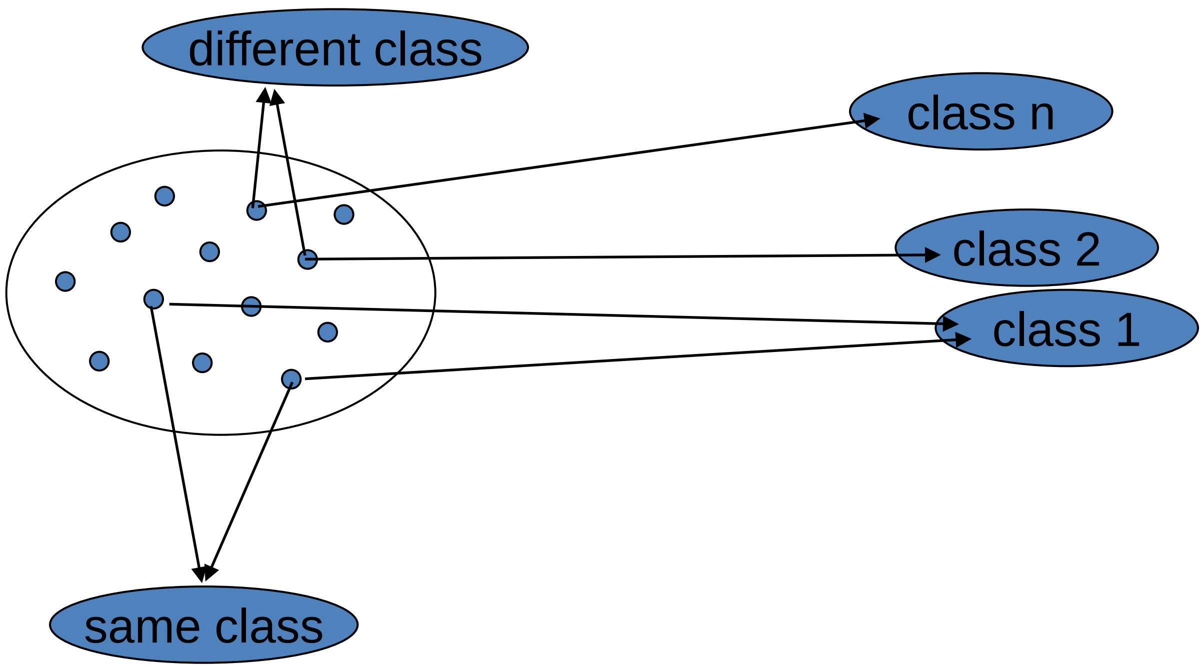
Constrained Clustering

- Goal: Find the structure intended by the user
  → "Need"-driven clustering
- "Need" might depend on
  - User's knowledge and background
  - Current context / task
- Constrained clustering integrates domain knowledge into the clustering process.



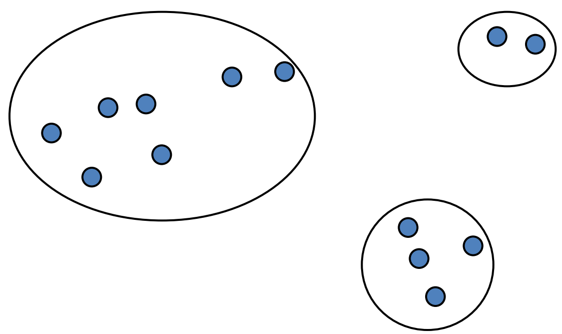
Available Domain Knowledge

- Some labeled training data
- Relations between objects
  - Similar and distinct items

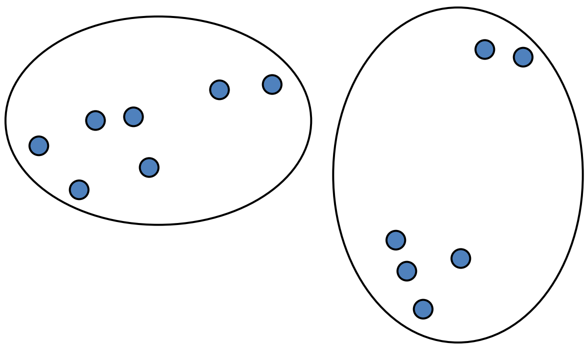


Available Domain Knowledge

- Cluster size/shape/number
  - Number of clusters

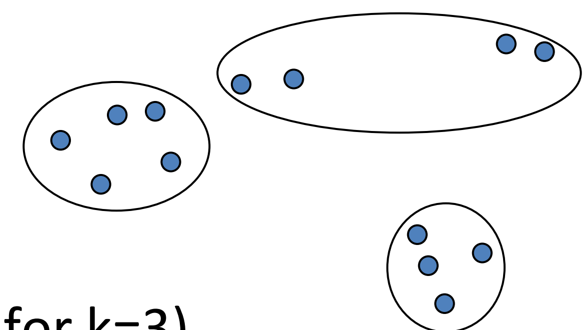


k=3



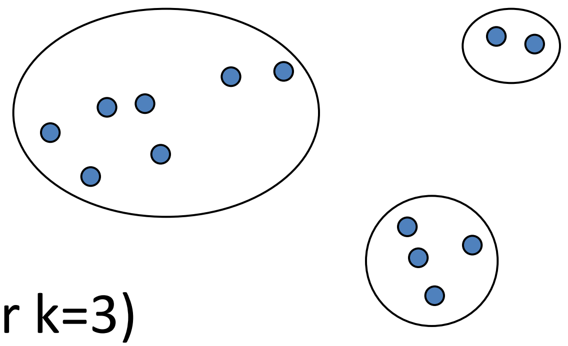
k=2

  - Balanced clusters



(for k=3)

  - Minimum cluster variance



(for k=3)

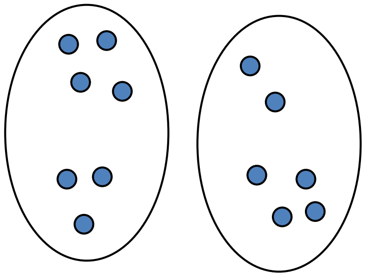
  - Minimum/maximum cluster size

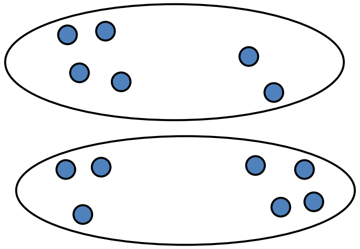
Available Domain Knowledge

- Negative background information
  - Find a structure that is different from a given one
  - Example for k=2

Given:

Found:

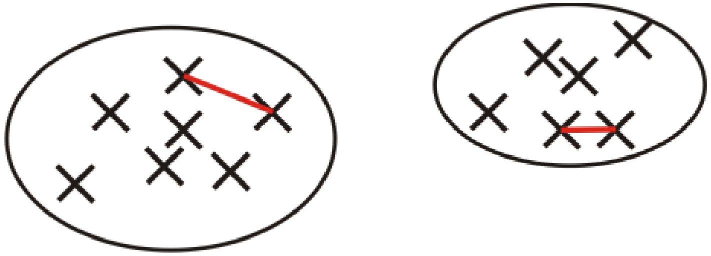




Pairwise Constraints

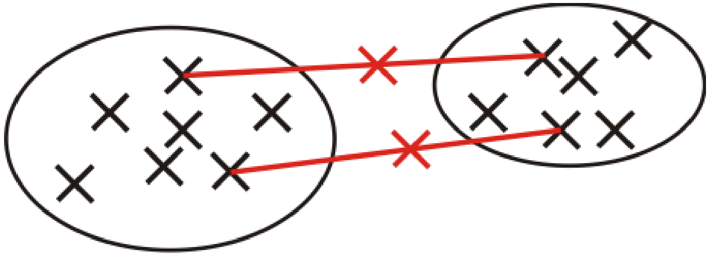
- Pairwise specification of relations between objects
- Must-link constraints
  - Pairs of items that belong to the same cluster

$$con_{=}(x,y)$$



- Cannot-link constraints
  - Pairs of items that belong to different clusters

$$con_{\neq}(x,y)$$



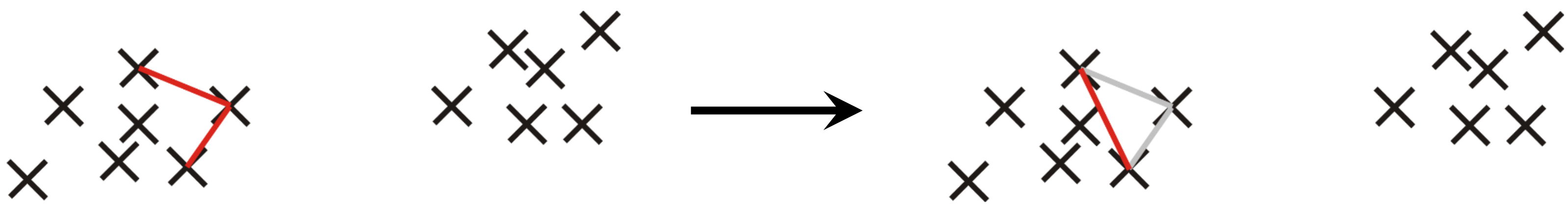
Pairwise Constraints

- Constraints are symmetric

$$\forall x,y : con(x,y) \rightarrow con(y,x)$$

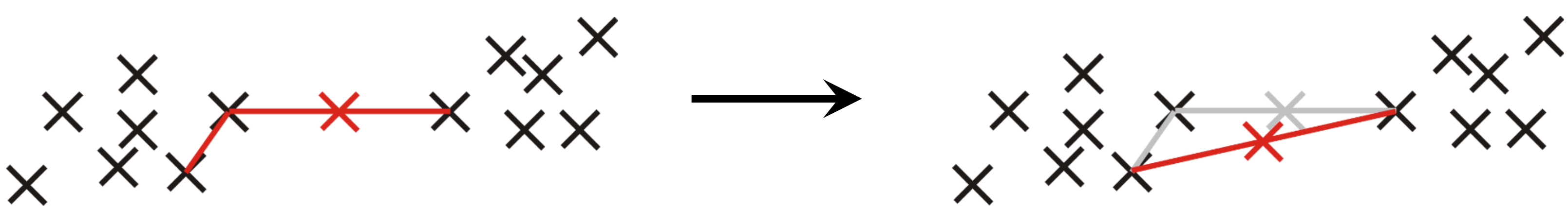
- Must-link constraints are transitive

$$\forall x,y,z : con_{=}(x,y) \wedge con_{=}(y,z) \rightarrow con_{=}(x,z)$$



- Cannot-link constraints are not transitive, but

$$\forall x,y,z : con_{=}(x,y) \wedge con_{\neq}(y,z) \rightarrow con_{\neq}(x,z)$$



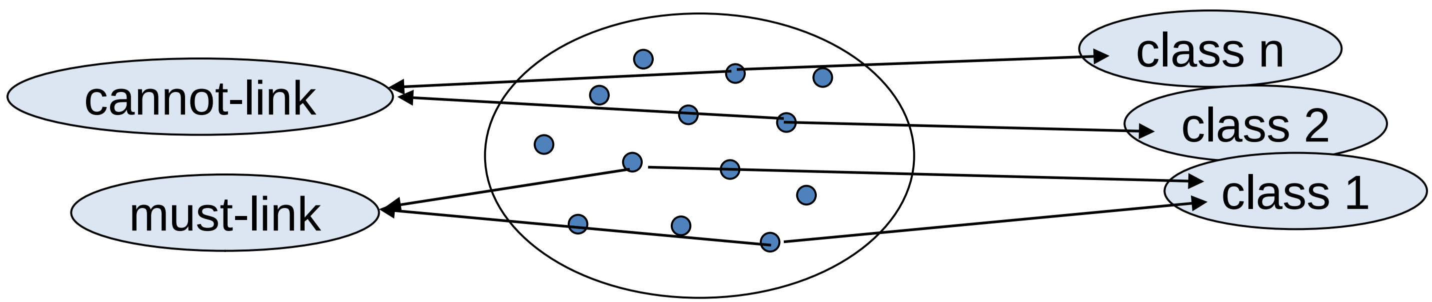
Hard vs. Soft Constraints

- Hard constraints
  - Must be satisfied
  - All constraints are equally important

- Soft constraints
  - Additional weight: $con(x,y,w)$, $w \in [-1,1]$
  - Cannot-link constraint: $w = -1$
  - Must-link constraint: $w = 1$
  - Specify the importance of constraint satisfaction

Sources of Pairwise Constraints (1)

- Partially labeled data
  - Pairs of elements of the same class form a set of must-links constraints
  - Pairs of elements from different classes form a set of cannot-link constraints



Sources of Pairwise Constraints (2)

- User feedback
  - Relative assignment of objects without knowing the true class labels
  - Indication of cluster errors → critiquing a solution
  - Active learning → ask the user for difficult objects before presenting a solution

- Automatically through knowledge about the task

Approaches

- Two main categories of methods to integrate pairwise constraints in the clustering process can be distinguished:
  - Instance-based
    - Approaches enforcing constraints
    - Approaches that allow violations
  - Metric-based

Instance-based Approaches

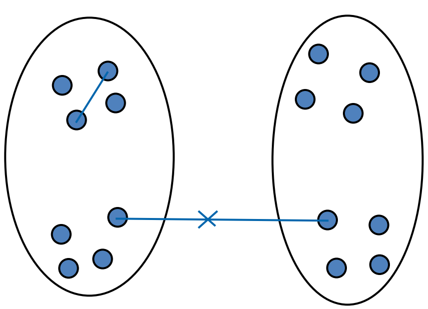
- Direct use of constraints (similar to lazy learning)
- During initialization, e.g.
  - Use all components that are connected by must-link constraints as starting points for HAC
  - Initialize cluster centers of *k*-means with the centroids of the connected must-link components
- Enforce constraints during clustering, e.g.
  - Do not cluster together objects that cannot link
  - Do not separate objects that must link
- Integrate constraints in objective function, e.g.
  - Compute trade-off between constraint violations

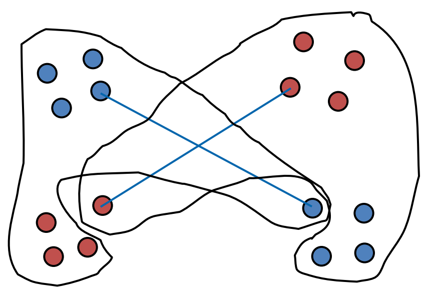
Metric-based Approaches

- Idea: Generalize knowledge from constraints
- Learning a metric
  - Identification of important features reflecting the intended structure
  - Distort the similarity/distance space accordingly, e.g. by feature weighting
- Apply the metric during clustering
  - Usually metric is learned based on the constraints before clustering
- Alternatives:
  - Adapt metric during clustering
  - Advantage: integration of information about the distribution of unlabeled objects

Instance-based vs. Metric-based

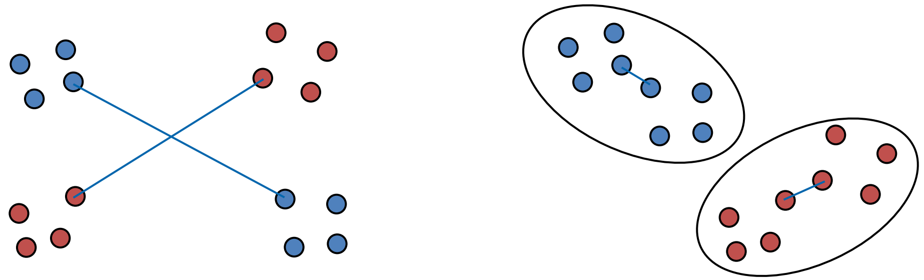
- Instance based approaches usually have rather local effects
- Strength of impact depends on the clustering algorithm and the method of integration
- Constraint enforcement might not lead to a global benefit or even decreased performance (although often performance increase is reported)
  - Non-informative constraints
  - Constraints violating the clustering objective





Instance-based vs. Metric-based

- Metric based approaches have usually a more global influence
  → Change in the underlying similarity measure
- Equal points should be handled equally



- Sufficient number of constraints is needed for a good generalization
  → overfitting

Instance-based vs. Metric-based

Re-using constraints:

- Metric learning
  - Can make use of an independent training set of constraints
  - Metric can be applied to any future dataset
- Instance-based approaches
  - Data points in the constraint set must always be clustered together with unlabeled data
  - Future datasets must be added to the constraint objects

Constrained Clustering Approaches

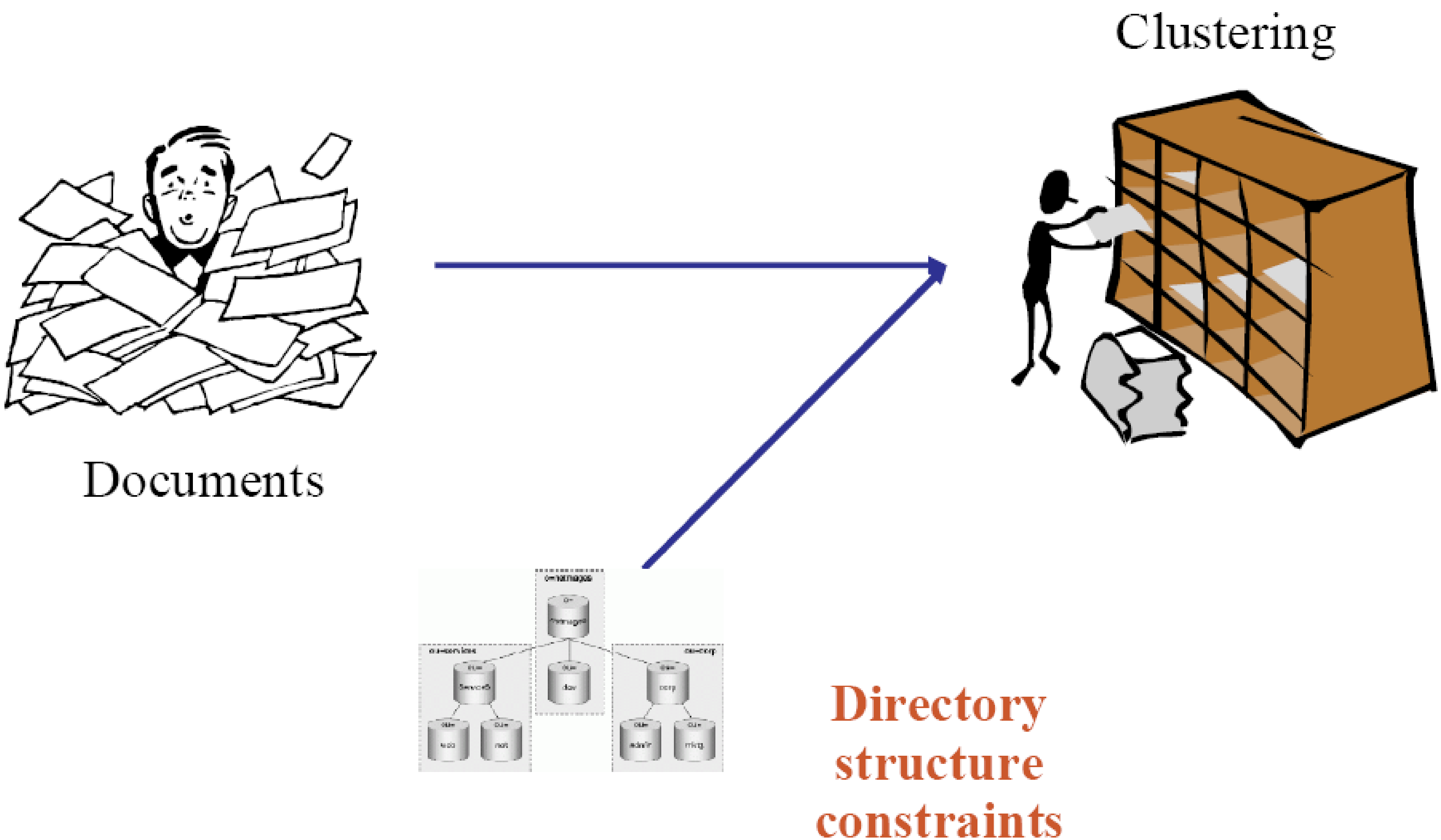
- Algorithms based on k-Means
  - COP-k-Means
  - PCK-Means
  - MPCK-Means

  Not discussed here…

- Hierarchical constrained clustering
  - iHAC
  - mHAC

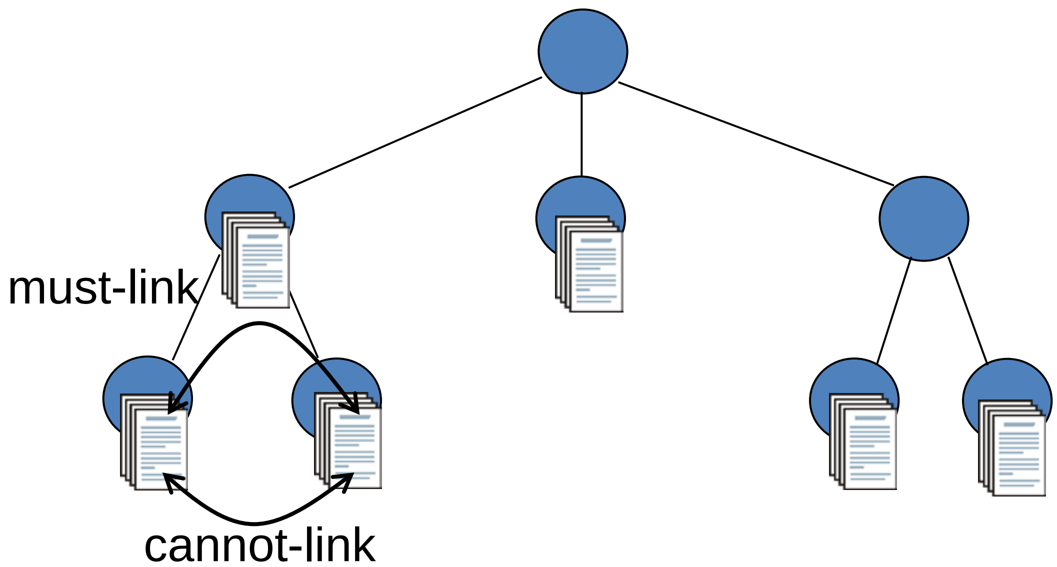
  In the following…

Constrained Hierarchical Clustering



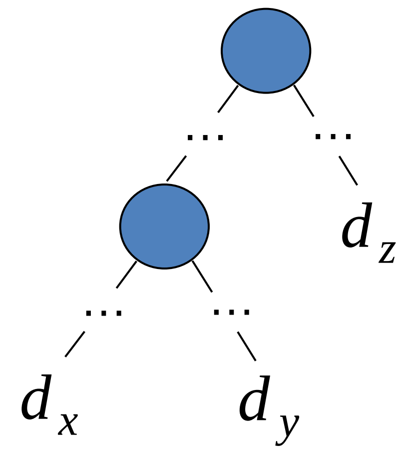
Constraints

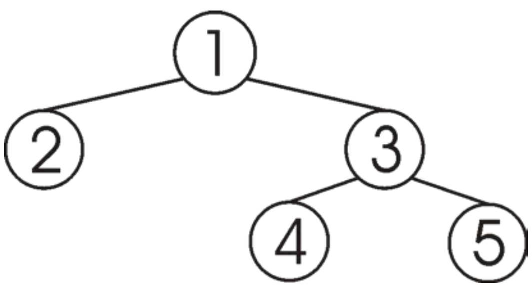
- Absolute constraint formulation of must-link and cannot-link constraints not appropriate
- In hierarchical clustering
  - Items are linked over different hierarchy levels
  - Constraints differ on different hierarchy levels



Must-link-before Constraints

- Ordering of item linkage
- Triples instead of pairs: (d_x, d_y, d_z)
- d_x and d_y should be linked on a lower hierarchy level than d_x and d_z





$$\left(d_x \in C_4, d_y \in C_3, d_z \in C_1\right)$$

$$\left(d_x \in C_2, d_y \in C_2, d_z \in C_3\right)$$

Must-link-before Constr. (Properties)

- Every cluster containing d_x and d_z also contains d_y

$$\forall c : d_x \in c \wedge d_z \in c \rightarrow d_y \in c$$

- There is at least one cluster, which contains d_x and d_y but not d_z

$$\exists c : d_x \in c \wedge d_y \in c \wedge d_z \notin c$$

- Symmetry

$$(d_x, d_y, d_z) \rightarrow (d_y, d_x, d_z)$$

Must-link-before Constr. (Properties)

- Transitivity inside a sub-tree

$$(d_x, d_y, d_z) \wedge (d_y, d_w, d_z) \rightarrow (d_x, d_w, d_z)$$

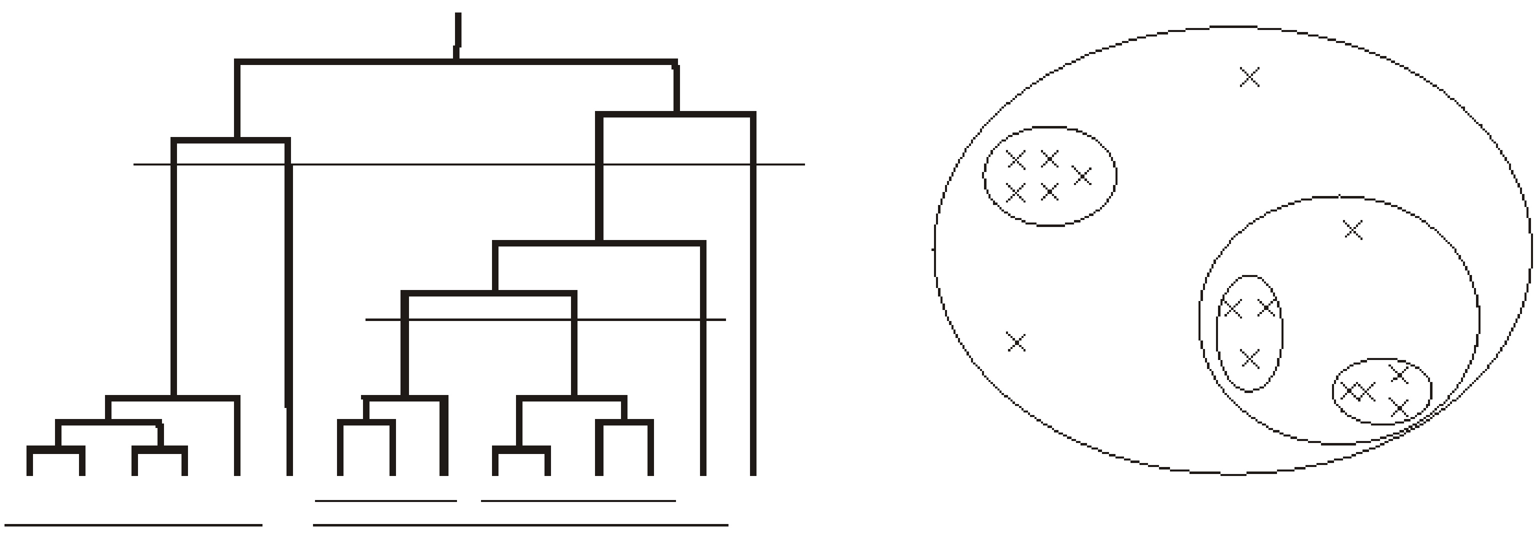
- Transitivity between different hierarchy levels

$$(d_x, d_y, d_z) \wedge (d_y, d_z, d_w) \rightarrow (d_x, d_y, d_w)$$

Hierarchical Agglomerative Cluster

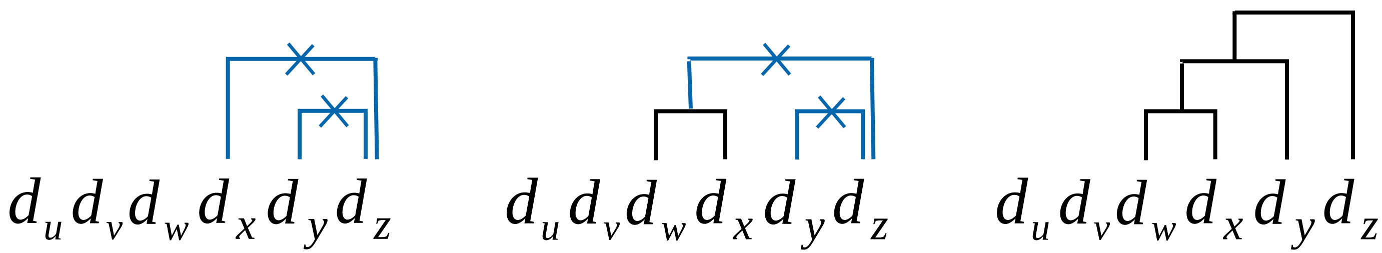
HAC Algorithm:

- Initialize lowest dendrogram level with each item as a separate cluster
- Repeat until a single cluster is left:
  1. Merge closest two clusters from the current top-most partition
  2. Add the new partition to the dendrogram



Instance-based Constr. HAC (iHAC)

- Cluster merges only in accordance to constraints
- For all (d_x,d_y,d_z): the cluster containing d_z can only be merged with a cluster containing both, d_x and d_y, or neither



- If no merge is possible due to constraints
  - Stop early or
  - Merge clusters violating the least constraints

Instance-based Constr. HAC (iHAC)

iHAC(D)

Initialize dendrogram DG by adding the lowest level $C_0 = \{c_1, \ldots, c_n\}$ with $\forall d_i \in D : c_i = \{d_i\}$

for $l = 1$ **to** n **do**

Choose two clusters $c_1, c_2 \in C_{l-1}$ and merge them: $c_m = c_1 \cup c_2$, whereby the merge of c_1 and c_2 violates the fewest constraints and from all cluster pairs satisfying this condition c_1 and c_2 are closest

$C_l = (C_{l-1} \setminus \{c_1, c_2\}) \cup \{c_m\}$

Add C_l to DG

end for

return DG

Metric-based Constr. HAC (mHAC)

- Parameterized cosine similarity

$$sim(d_i, d_j, W) = \frac{\vec{d}_i^T W \vec{d}_j}{\left|\vec{d}_i\right|_W \left|\vec{d}_j\right|_W} \qquad \left|\vec{d}\right|_W = \sqrt{\vec{d}^T W \vec{d}}$$

- W – symmetric positive definite matrix
- Here: diagonal matrix

$$W = \vec{w}^T \cdot I \cdot \vec{w} = \left(\sqrt{w_1} \ldots \sqrt{w_n}\right) \cdot I \cdot \begin{pmatrix} \sqrt{w_1} \\ \vdots \\ \sqrt{w_n} \end{pmatrix}$$

- Required properties:

$$\forall w_i : w_i \geq 0 \qquad \sum_i w_i = n$$

- All weights 1 is standard similarity

Metric-based Constr. HAC (mHAC)

- Constraints interpretation

$$(d_x, d_y, d_z) \Rightarrow \mathrm{sim}(d_x, d_y) > \mathrm{sim}(d_x, d_z)$$

- Weight learning through gradient descent approach
  - Initialize weights with 1 (standard similarity)
  - Repeat until convergence:

    For each violated constraint:

$$w_i \leftarrow w_i + \eta \frac{\partial\left(\mathrm{sim}(d_x, d_y) - \mathrm{sim}(d_x, d_z)\right)}{\partial w_i}$$

    i.e. make d_x and d_y more similar and d_x and d_z more dissimilar

Metric-based Constr. HAC (mHAC)

- Metric is learned before clustering
- Similarity matrix for HAC is initialized with new metric

- HAC clustering can be replaced by iHAC
→ Combination of both approaches

References

- **Motivational Examples:**

  - R. Yan, J. Zhang, J. Yang, A. Hauptmann, A Discriminative Learning Framework with Pairwise Constraints for Video Object Classification, 2004.

- **Hierarchical Approaches:**

  - Korinna Bade und Andreas Nürnberger, Creating a Cluster Hierarchy under Constraints of a Partially Known Hierarchy, In: *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM'08)*, 2008.
  - Korinna Bade und Andreas Nürnberger, Constraint Based Hierarchical Clustering for Text Documents, In: *Proceedings of the LWA 2007 Workshop*, 2007.
  - Korinna Bade, Marcel Hermkes und Andreas Nürnberger, User Oriented Hierarchical Information Organization and Retrieval, In: *Proceedings of the 2007 European Conference on Machine Learning (ECML'07),* 2007.
  - Sebastian Stober, Andreas Nürnberger. AUCOMA - Adaptive Nutzerzentrierte Organisation von Musikarchiven. In Ute Jekosch & Rüdiger Hoffmann (eds.): Fortschritte der Akustik: Plenarvorträge und Fachbeiträge der 34. Deutschen Jahrestagung für Akustik DAGA 2008, Pages 547-548, German Acoustical Society (DEGA), 2008. (in German)

Overview (Day 3)

- Motivation
- Constrained Clustering
- **A Utility based Approach**

Hierarchical classification

- Idea:
  - Improving performance of hierarchical classifier depending on the way a user is accessing the hierarchy, i.e.
  - objects are inserted/classified such that a user is still able to retrieve them even if the classification is highly uncertain

- Realization:
  - Representation of user behavior (its way to access the structure) by a *utility-function*
  - Classification as decision problem

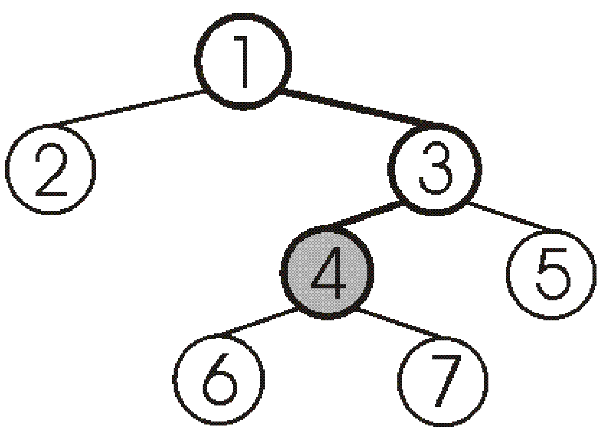
K. Bade, E. Hüllermeier and A. Nürnberger, Hierarchical Classification by Expected Utility Maximization, in: *Proc. of IEEE International Conference on Data Mining (ICDM 2006)*, pp. 43-52, IEEE Computer Society Press, 2006.

Modeling of utility-function (1)

- Utility function defines utility of assigning an object to a specific class (even a wrong class!)
- Here: In a hierarchy a "wrong" classification on the users search/retrieval path might still have a high utility!
- Definition for hierarchy H and classes c_i:
  - Retrieval path:

$$rp_c = \{c_i \in H \mid c_i \geq_H c\}$$

  - Hierarchical distance $dist_H(c_i, c_j)$: Number of nodes from c_i to c_j



Modeling of utility-function (2)

- Different utilities of classes (depending on user):
  - Correct class: Maximal utility *util* = 1
  - Class in retrieval path: Usage decreases on the way to the root node: $util \in [0;1]$
  - Class not in the retrieval path: *util* = 0
- Hierarchical utility:

$$util(\hat{c} \mid c) = \begin{cases} \exp(-\gamma \cdot dist_H(\hat{c}, c)) & \text{if } \hat{c} \geq_H c, \\ L & \text{otherwise.} \end{cases}$$

  - γ defines "laziness" of a user
    - $\gamma \to \infty$: only the correct class has a utility *util* > 0
    - $\gamma = 0$: all nodes have similar (maximal) utility (Remark: in this case it would be best to predict always the root node)

Decision problem (1)

- Given:
  - Hierarchy *H*
  - Training data *D*
    - Remark: inner nodes can be empty!
  - Utility function *util* as defined above
- Still required:
  - Probability estimates $\rho_j = P(c_j | d)$
    $\rightarrow$ train probabilistic classifier *C*
  - Here: Naïve Bayes and SVM

Decision problem (2)

- Decision is prediction of class c_i, undefined state is c_j and utility is defined by $util(c_i|c_j)$:

$$
\begin{array}{c|cccc}
{}_{i}\backslash{}^{j} & p_1 & p_2 & \cdots & p_n \\
 & c_1 & c_2 & \cdots & c_n \\
\hline
c_1 & u_{11} & u_{12} & \cdots & u_{1n} \\
c_2 & u_{21} & u_{22} & \cdots & u_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_m & u_{m1} & u_{m2} & \cdots & u_{mn}
\end{array}
$$

- Prediction of class c_{best} for a document d if the expected utility EU is maximal:

$$c_{best} = \underset{c_i \in H}{\arg\max}\, EU(c_i \mid d) = \underset{c_i \in H}{\arg\max} \sum_{c_j \in H} P(c_j \mid d) \cdot util(c_i \mid c_j)$$

Estimation of parameters

- How to obtain parameter γ and L?
- Idea:
  - Use of a second utility function!
  - First utility function *models User*
    - used to train and evaluate the model
    - fixed parameters
  - Parameters of second utility function are learned:
    - utility function adapts to data
    - possibility to adapt to poor classifiers

Algorithm HUClass

HUClass(d, $classifier$, γ, L)

    For each $c_i \in H$:

        Compute probability estimate $P(c_i|d)$ by $classifier$

    c_{best} = null

    For each $c_i \in H$:

        Compute expected utility $\mathrm{EU}(c_i|d)$

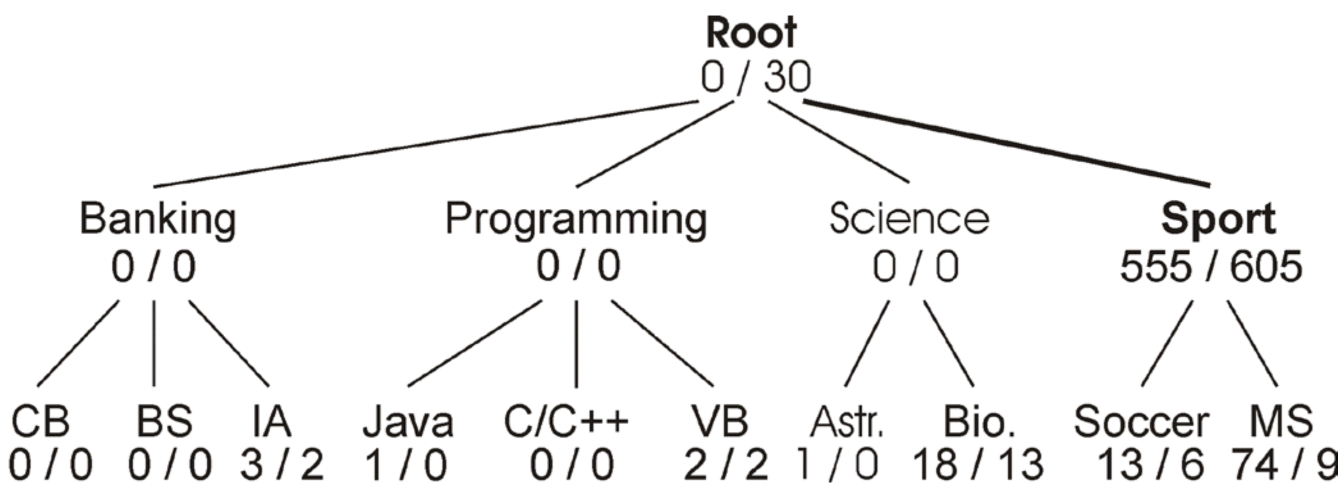
        If $\mathrm{EU}(c_i|d) > \mathrm{EU}(c_{best}|d)$

            $c_{best} = c_i$

    Return c_{best}

Evaluation (1)

- Data: Banksearch Data
  - 11,000 classified web pages
  - 11 classes, 2 level hierarchy
- Training:
  - 300 documents from each class



classification of banksearch "sports documents"
with naïve Bayes / HUClass(NB)

Evaluation (1)

- Evaluation measures:
  - Hierarchical Accuracy, Precision, Recall und F-Measure
  - Number of documents classified correctly
  - Number of documents in and aside of retrieval path
- In case of NB classification a user would have almost 700 (searched!) documents more in retrieval path

	Naïve Bayes	HUClass(NB, 3.4, −350)	SVM	HUClass(SVM, 0.6, 0)
acc_h	0.8278 ± 0.0055	0.8460 ± 0.0031	0.9301 ± 0.0014	0.9323 ± 0.0014
$prec_h$	0.8469 ± 0.0057	0.9135 ± 0.0049	0.9305 ± 0.0013	0.9435 ± 0.0019
rec_h	0.8277 ± 0.0056	0.8462 ± 0.0032	0.9300 ± 0.0014	0.9323 ± 0.0014
f_h	0.8372 ± 0.0051	0.8786 ± 0.0039	0.9303 ± 0.0013	0.9379 ± 0.0016
$\#n_c$	6281.4 ± 41.46	5839.2 ± 34.64	7079.6 ± 10.05	6994.8 ± 11.65
$\overline{\#n_c}\ (\overline{ml(n_c)})$	42.8 ± 3.43 (1.0 ± 0.0)	1198.2 ± 103.16 (1.42 ± 0.04)	12.6 ± 3.38 (1.0 ± 0.0)	209.6 ± 13.75 (1.36 ± 0.04)
$\underline{\#n_c}\ (\underline{ml(n_c)})$	1295.4 ± 41.75 (1.40 ± 0.03)	582.2 ± 72.10 (1.31 ± 0.37)	527.4 ± 9.50 (1.40 ± 0.02)	415.2 ± 12.17 (1.39 ± 0.02)

Evaluation (2)

- Open Directory Dump
  - 8123 web pages
  - Up to 4 levels in hierarchy, 2-17 leave nodes
- Results:
  - Bigger performance improvements than for banksearch ds
  - Reasons: more complex structure and noisy classes

	Naïve Bayes	HUClass(NB, 0.6, −615 000)	SVM	HUClass(SVM, 1.0, 0)
acc_h	0.4556 ± 0.0096	0.5078 ± 0.0096	0.6632 ± 0.0047	0.6786 ± 0.0082
$prec_h$	0.5753 ± 0.0243	0.6986 ± 0.0170	0.7146 ± 0.0148	0.7820 ± 0.0075
rec_h	0.3119 ± 0.0089	0.4007 ± 0.0096	0.5225 ± 0.0061	0.5508 ± 0.0038
f_h	0.4044 ± 0.0120	0.5092 ± 0.0106	0.6035 ± 0.0069	0.6463 ± 0.0050
$\#n_c$	1118.2 ± 25.69	898.6 ± 8.89	1725.0 ± 17.15	1524.4 ± 34.49
$\overline{\#n_c}$ $(\overline{ml(n_c)})$	195.0 ± 23.57 (1.25 ± 0.04)	1196.4 ± 76.06 (2.11 ± 0.08)	104.2 ± 10.61 (1.21 ± 0.05)	543.0 ± 25.98 (1.23 ± 0.02)
$\underline{\#n_c}$ $(\underline{ml(n_c)})$	1376.4 ± 32.04 (1.95 ± 0.03)	594.6 ± 78.72 (1.87 ± 0.18)	859.6 ± 10.21 (1.73 ± 0.01)	621.4 ± 12.71 (1.78 ± 0.02)

Personalized hierarchical classification

- Open questions:
  - What are possible user (access) models?
  - What user models really make sense?
  - How to obtain parameters of user model?
    - Parameter currently strongly dependent on underlying data (above: optimized for data!)
    - Is it possible to learn parameters during user interaction?
    - Is it possible to detect (automatically) typical user classes?
  - What visualization methods are appropriate?

The End

THANKS A LOT FOR LISTENING!