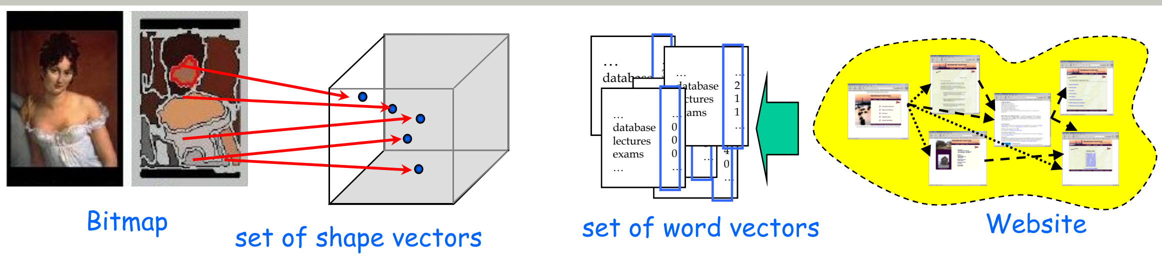
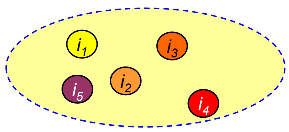
Multi-Instance Data

- Multi-Instance Data
- Aggregation-based Methods
- Distance and Similarity Measures
- Multi-Instance Learning and general Multi-Instance Classification
- Clustering Multi-Instance Objects

What is a Multi-Instance Data ?



Multi-Instance objects describe:
- multiple components (e.g. CAD data)
- various appearances (e.g. proteins)
- set-valued objects (e.g. market baskets, teams)



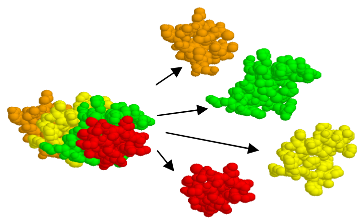
Differences to other structured objects:
1. All instances are elements of the same features space (vs. Multi-View)
2. Multi-Instance objects do not have an order (vs. time-series, sequences)

Examples for Multi-Instance Objects

s1

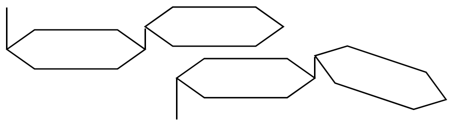
Proteins

- proteins consist of multiple amino acid sequences
- each sequences is an instance
- a protein is a set of its sequences



Macro-Molecules

- varying spatial conformations
- each conformation is an instance
- the molecule is described by a set of all possible conformations

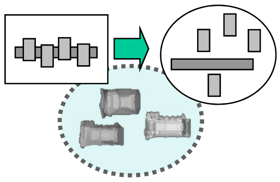


Folie 70

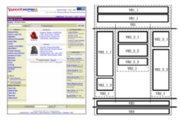
s1 schubert; 02.12.2014

Further Examples

- CAD-components: set of spatial primitives



- HTML documents: set of layout blocks
  (dom tree structure is dropped)



- Video data: videos can be described by
  sets of shots (order is dropped)

Formal:
Object *o* is part of the power set of R: $o = \{r_1,..,r_n\} \in 2^R$
where R is the feature space of instance.
(shortly instance space)

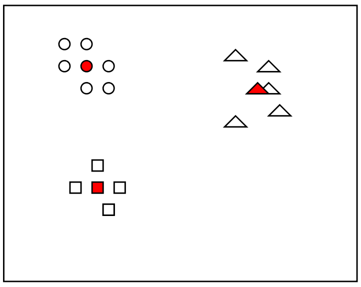
Aggregation-based Approaches

Idea: Reduce the multi-instance object into a single representative instance.

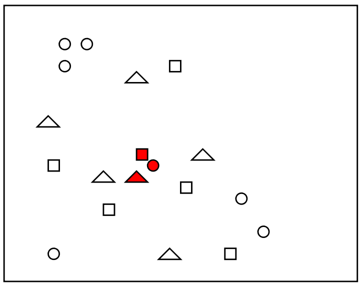
e.g. build the centroid
$\Rightarrow$ simple method describing a set by its componentwise means

problems:
- properties of the particular instances are lost
- cardinality of the set is lost
- outliers are not described well

Aggregation-based Approaches



1. case: aggregation on suitable data



2. case: aggregation in unsuitable data

Conclusion: Aggregation depends on the distribution of the objects.

- If all instances are drawn from the same distribution aggregation makes sense.
- If instances might be drawn from different distributions, aggregation is not suitable.

Distance Measures and Kernels

Idea: Many data mining algorithms only need pairwise comparisons.

$\Rightarrow$ Define distances and kernel-functions on multi-instance objects

There are multiple ways to compare multi-instance objects:

- How many instances should be similar?
- Does there have to be a bijective mapping between the sets ?

=> There are multiple similarity measures which might make sense in varying application areas.

General Problem of comparing sets

Multi-instance comparisons yield an assignment task:

Which instance in object X has to be compared to which instance in object Y ?

Given two objects $X=\{x_1, x_2, x_3\}$ and $Y=\{y_1, y_2, y_3\}$:

- *Each x_i can be compared to each y_j.*
- *To how many y_j has x_i be similar?*
- *To how many x_i has y_j be similar?*
- *Usually: Each instance is assigned to at least one instance in the other object. (often the closest)*

	X_1	X_2	X_3
Y_1			
Y_2			
Y_3			

Hausdorff Distanz

Idea: Each instance is covered by the closest instance from the other. The maximum cover distance describes the distance of the two objects.

⇒ minimal distance = most similar instance (smallest radius to cover the instance)

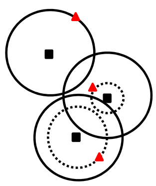
⇒ maximal distance over all row /columns (worst case cover)

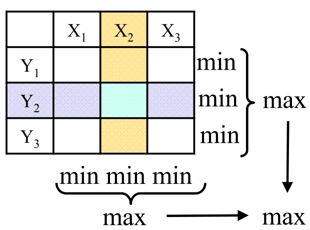
⇒ maximum of row and column maximums achieves symmetrie

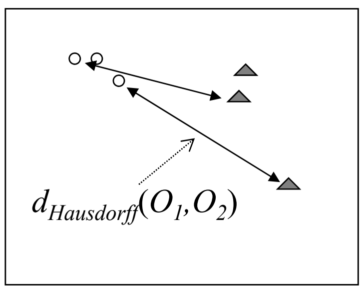
⇒ **Definition:** Hausdorff Distance

Let O_1, O_2 *be two* MI-objects and $d(x,y)$ an instance distance measure over the feature space R, then the Hausdorff distance is defined as follows:

$$d_{Hausdorff}(O_1,O_2) = \max\left(\max_{o_i \in O_1}\left(\min_{o_j \in O_2}\left(d(o_i,o_j)\right)\right), \max_{o_i \in O_2}\left(\min_{o_j \in O_1}\left(d(o_i,o_j)\right)\right)\right)$$







DATABASE SYSTEMS GROUP

Single Link or Minimal Hausdorff Distance

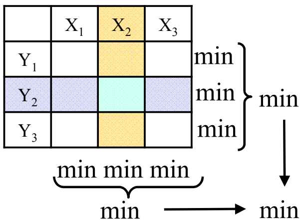
LMU

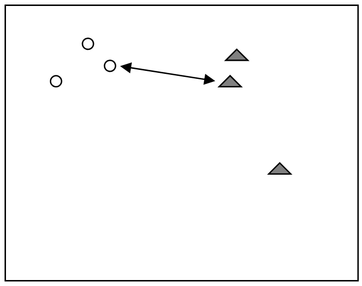
Idea: Used the closest pair of instances.

Definition: Minimal Hausdorff Distance or Single Link Distance

Let O_1, O_2 be two MI-objects and let $d(x,y)$ be an instance distance measure in the underlying feature space R, then the minimal Hausdorff or single link distance is defined as follows:

$$d_{\text{singlelink}}(O_1,O_2) = \min_{o_i \in O_1}\left(\min_{o_j \in O_2}\left(d(o_i,o_j)\right)\right)$$





DATABASE SYSTEMS GROUP

Sum of minimals distances

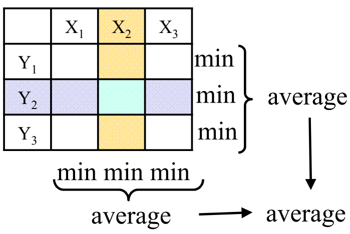
LMU

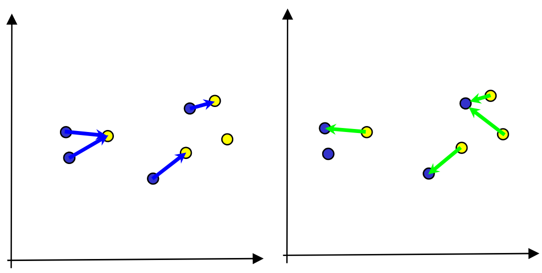
Idea: Use the average distance of closest pairs.

Definition: Sum of Minimum distances (SMD)

Let O_1, O_2 *be two* MI-objects and $d(x,y)$ an instance distance measure over the feature space R, then the SMD distance is defined as follows:

$$d_{SMD}(O_1,O_2) = \frac{1}{2}\left(\frac{1}{|O_1|}\sum_{o_i \in O_1}\left(\min_{o_j \in O_2}\left(d(o_i,o_j)\right)\right) + \frac{1}{|O_2|}\sum_{o_j \in O_2}\left(\min_{o_i \in O_1}\left(d(o_i,o_j)\right)\right)\right)$$





Complexity of Hausdorff, Single Link and SMD distances

All distance measures so far have the complexity $O(|O_1| \cdot |O_2| \cdot d)$

- assuming that $d(x,y)$ is computable in $O(d)$
- reason: for each instance in O_1 the distance to each instance in O_2 must be compute.

Metric properties:

- Hausdorff distance is a metric:
  symmetry, reflexivity, triangular inequality hold.
- single link not even reflexive
- SMD is symmetric and reflexive, but the triangular inequality does not hold.

Minimal Matching Distance (MMD)

Idea: The distance between two sets is described by a cost-minimal bijection.

Definition*:*

Let O_1, O_2 *be two* MI-objects and let $d(x,y)$ be an instance distance measure over the feature space R, then the Minimal Matching Distance is defined as follows:

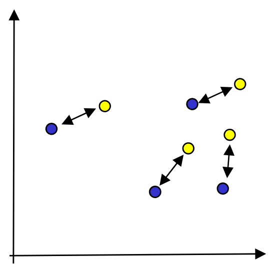
$$d_{MM}(O_1,O_2) = \min_{\pi_i \in \Pi(O_1)} \left(\sum_{k=1}^{|O_2|} d\left(o_{1,\pi(k)}, o_{2,k}\right) + \sum_{l=|O_2|+1}^{|O_1|} w\left(o_{1,\pi(l)}\right) \right)$$

w.l.o.g. let $|O_1| > |O_2|$. $\Pi(O_1)$ *represents the set of all* permutations of the instances in O_1 und $w(o_{i,j})$ is a weighting term penalizing matched instances without a match.

Remark:

MMD is metric if $w(o_{i,j})$ is large enough to prevent unmatched instances, i.e. $w(o_{i,j})$ *has to be larger than any distance to* any other instance.

=> Not matching any object is always worse than matching it



Computing MMD

Method: Solve a minimal weight perfect matching problem, e.g. with the Ungarian method (runtime complexity $O(n^3)$).

Algorithm:

1. Compute the bipartite graph between the instances
2. Fill up missing row and columns until the matrix is quadratic (use $w(o_{i,j})$ as values)
3. Subtract the minimum from each row
4. Subtract the minimum from each column
5. Find a minimal set of marks for rows and columns until all 0 elements are covered
6. If the minimal set of marks equals n then permutate the matrix in a way that the zero elements occupy the main diagonal
7. If the number of marked rows and columns < n
   a. Search the minimal value among all unmarked objects
   b. Subtract this minimal value from all unmarked elements
   c. Add the minimum value to the elements where two marks (1 row and 1 column mark) overlap
   d. Goto step 5

Example: Computing MMD

Matrix of pairwise distances

10	12	20	21
10	12	21	24
14	17	28	30
16	20	30	35

Subtract row min

0	2	10	11
0	2	11	13
0	3	14	16
0	4	14	19

subtract column min

0	0	0	0
0	0	1	2
0	1	4	5
0	2	4	8

Mark all 0s

0	0	0	0
0	0	1	2
0	1	4	5
0	2	4	8

Add and subtract unmarked m in

1	1	0	0
0	0	0	1
0	1	3	4
0	2	3	7

Mark

1	1	0	0
0	0	0	1
0	1	3	4
0	2	3	7

add and subtract min

2	1	0	0
1	0	0	1
0	0	2	3
0	1	2	6

compute result

2	1	0	0
1	0	0	1
0	0	2	3
0	1	2	6

$D_{mmd}(O_1,O_2)=21+21+17+16 = 75$

10	12	20	21
10	12	21	24
14	17	28	30
16	20	30	35

Kernel Function for Multi-Instance Objects

Idea: Compare two MI-objects by adding up pairwise similarities where the similarity is described by a kernel.

Definition: Convolution Kernel

Let O_1,O_2 be two MI-Objects and let $K(x,y)$ be a kernel function in feature space R. Then the convolution kernel is defined as follows:

$$K_{Convolution}(O_1, O_2) = \sum_{o_{1,i} \in O_1, o_{2,j} \in O_2} K(o_{1,i}, o_{2,j})$$

Remarks:

- Basic ides is similar to the average-link distance (average value of pairwise distances)
- Convolution kernels are Mercer kernels and can be used for kernel-based learners like SVMs, Kernel-PCA, etc.

Multi-Instance Classification

Setting: $DB = 2^F$ where F is a feature space.
training set (O,c) where $O \in DB$ and $c \in C$.

Challenge:
Which instances $\{o_i, .., o_j\} \subseteq O$ *are responsible for the membership of O in class c?*

classic multi instance learning:

- two classes 1 and 0
- object O belongs to class c if there is at least on instance $o_i \in O$ relevant to 1

general multi instance learning:

- arbitrary amount of classes
- instances can be relevant to multiple classes
- class membership might depend on any subset of O

General Multi Instance Classification

DATABASE SYSTEMS GROUP

LMU

Problem:

MI objects from the same class need not be completely similar (similar w.r.t to each instance). => Classes can be described in mutliple different ways

general approach to multi-instance classifiers:

- classes can be defined by „concepts“ on the instances
  (football team 1 g**oal keeper** and 10 **regular players**)
- each concept describes a „class“ of instance
- concepts might occur in a class or be completely absent
- the cardinality of the concepts in the class might play a role
  (5 goal keeper and 1 regular player is not a football team)

General Mult-Instance Classification

DATABASE SYSTEMS GROUP

LMU

Classification of Multi-Instance objects with given concepts

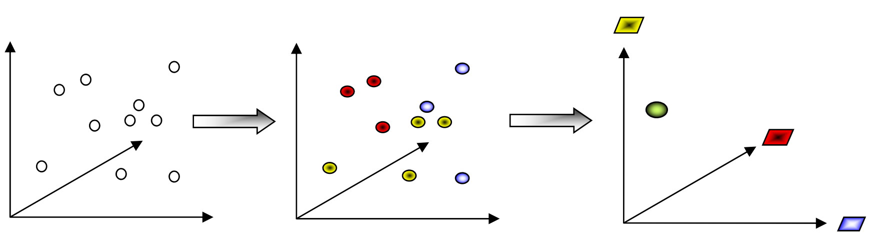
Input: Let C be a multi-instance class set, let K be a set of instance concepts *K* and let DB be a set of multi-instance objects *DB being labelled with elements from C*. Furthermore, let $CL(O) = c_i \in C$ *describe the mapping of object O to the elements of C and let* $KL(o_j) = K_l \in K$ *describe the mapping of instance* o_j *to K.*

Idea: Two Stage Classification.

- Learn a mapping of instance o_j to concepts K_l

=> Each multi-instance object can be mapped to a distribution over K

- Learn a classifier mapping concept distribution to multi-instance classes C.



General Multi-Instance Classification

Classification of multi-instance objects with unknown concepts

Input: Let C be a multi-instance class set and let DB be a set of multi-instance objects *DB being labelled with elements from C*.
Furthermore, let $CL(O) = c_i \in C$ *describe the mapping of object O to the elements of C.*

Problem: The concepts for defining a class are unknown
=> training a classifier to predict instance concepts is not possible

Solution approaches:

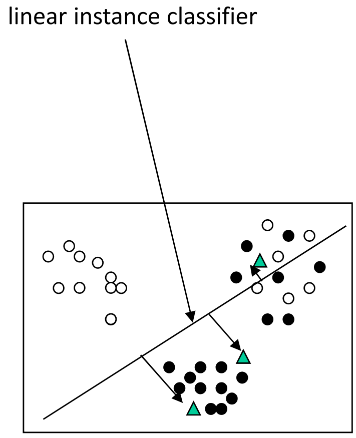
- train an instance classifier predicting the likelihood that instance o_i *is element of any multi-instance object O having a class* c_j.
- Aggregate the prediction over all instances in O
  (assumption: O was generated by drawing n times with replacement)

Remark:

- methods depends on reliability of the confidence values
- method assumes the independency of the instances (multinomial distribution)

General Multi-Instance Classification

Example: 2 classes, 3 „unknown“ concepts



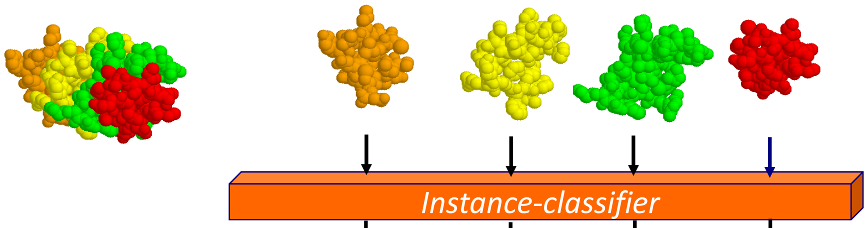
- Trainings set for instance classifier

$$TR_A = \bigcup_{O_i \in DB} \{o_j \in O_i \wedge CL(O_i) = A\}$$

- instances in concepts being typical for a class should be classified with a high confidence
- instances in ambiguous concepts should be classified with smaller confidence values
- the classifier often needs rather complex class borders (small bias but larger likelihood of overfitting)

General Multi-Instance Classification

Example: Combination of the instance predictions



conf. for instance	C1	0.6	0.05	0.4	0.2	
	C2	0.4	0.95	0.6	0.8	0.987 C2
conf. for cmpl. object	C1	**0.6**	**0.073**	**0.05**	**0.013**	
	C2	**0.4**	**0.927**	**0.95**	**0.987**	

Confidence of *O* for class C_k: $\Pr[C_k \mid O] = \dfrac{\Pr[C_k]\cdot P[O \mid C_k]}{\sum_{i\in C}\Pr[C_i]\cdot P[O \mid C_i]}$ (Bayes theorem)

where $\Pr[W \mid O_k] = \prod_{p_i \in W} \Pr[I_i \mid C_k]$

Classical Multi-Instance Learning

Setting: There is one relevant concept K_{rel}. All objects containing at least one instance $o_i \in O$ *with* $K(o_i)=K_{rel}$ *belong to class „relevant“.*

Examples:

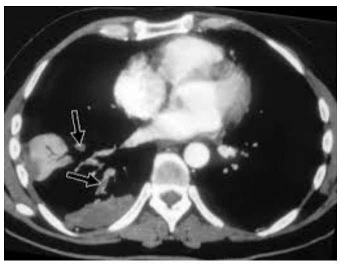
1- Does a molecule smell like musk? [Dietterich et al. 1998]

Molecules are described as sets of spatial conformations. If the molecule has a spatial conformation matching the musk receptor, it has a musky smell.

2- Search for lung embolisms

CT scanner generates a set of suspicious areas in the lung. If a least one of them is a lung embolism the patient needs treatment.

http://medicalpicturesinfo.com/pulmonary-embolism/



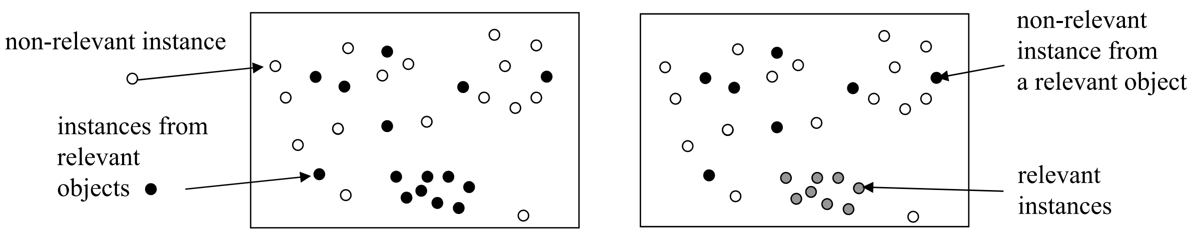
Classical Multi-Instance Learning

Approach: Classify all single instances
=> if one is relevant, the complete object is relevant as well.

Problem: Labeled instances are only reliable for the non-relevant class.

Remark: Multi-instance learning corresponds to learning a classifier for the relevant concept

- all instances of objects in the non-relevant class cannot be part of the relevant concept
- instances of objects from the relevant class can belong to both concepts
- at least one instance for each object has to belong to the relevant concept

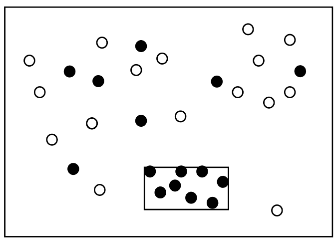


Classical Multi-Instance Learning

Approaches to classical multi-instance learning

Find a region in the feature space which contains only relevant instances (no negative samples) and contains at least one instance from each relevant object.

- Solution space is constructed by all sets of instances containing one instance from each objects. (assume: k objects having n instances => n^k solutions)
- Each solution can be used to demark the relevant area of the feature space
- It cannot be guaranteed that there is one area without any non-relevant samples
- Irrelevant features, learning bias etc. also influence the quality



Expectation Maximization Diverse Density classification (EM-DD)

Idea: Describe the relevant concept by an instance *h* and weights s_d for weighting the influence of the features $D=\{d_1,..,d_m\}$.

Predicting the object class is done by the max confidence of any instance in O:

$$Label(O_1 \mid h, \vec{s}) = \max_j \left\{ \exp\left[-\sum_{i=1}^{m} \left(s_i\left(o_{j,i} - h_i\right)\right)^2 \right] \right\}$$

where l=0 codes „relevant" and l=1 codes „irrelevant"

The Quality of the classifier for set DB can be described by the Negativ Logarithmic Diverse Density (*NLDD*) :

$$NLDD(h, \vec{s}, DB) = \sum_{i=1}^{|DB|} \left(-\log\left(\left| l_i - Label\left(O_i \middle| h, \vec{s}\right) \right| \right) \right)$$

DATABASE SYSTEMS GROUP

Classical Multi-Instance Learning

LMU

EM-DD training algorithm:

init *h* //e.g. centroid of a samples of the relevant instances, s_i = 0.1

While($NLDD_{new} < NLDD_{old}$)

FOR ALL O_i in DB mit CL(O_i) = „relevant" DO

$$o_i^* = \arg\max_{o_{ij} \in O_i} \left(Label\left(O_i \middle| h, \vec{s}\right) \right)$$

$$h' = \arg\max_{h \in H} \prod_{i=1}^{n} \Pr\left(l_i \middle| h, \vec{s}, o_i^*\right)$$

// optimization of weights
// by gradient descent

$NLDD_{old} = NLDD_{new}$

$NLDD_{new} = NLDD(h',D)$

h = h'

return *h*

Remark: $\Pr\left(l_i \middle| h, \vec{s}, o_i^*\right) = \exp\left[-\sum_{i=1}^{m} \left(s_i\left(o_i^* - h_i\right)\right)^2 \right]$

Multi-Instance Classifier

Conclusions:

general Multi-Instance Classification

- only a view dedicated approaches are published
- most approaches are based on distance measures or kernels

Classical Multi-Instance Learning

- Large effort in the research community
  - Citation-kNN and Bayes-kNN (nearest neighbor-based approaches)
  - Multi-Instance decision trees and rule-based classifiers
  - Neural Networks for multi-instance objects

  ⇒ EM-DD (showed most promising results without any meta-learning)
- General benchmark is the musk use case !!
  More practical results showed good results for general MI-learners

Multi-Instance Clustering

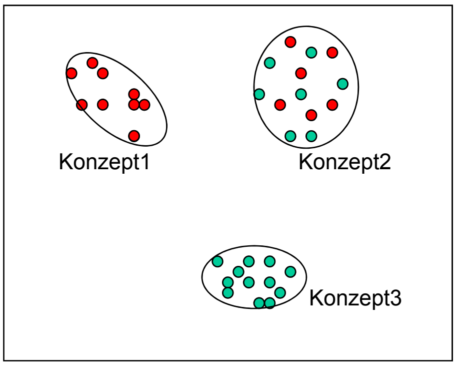
- MI-Objects can be clustered based on distance-based methods such as k-Medoid, DBSCAN, OPTICS, etc.
  - selecting a well-suited distance measure is very important
  - only applicable to purely distance-based methods (cluster model ?)
    (e.g., k-Means cannot be used due to the lack of centroids)

- concept-based multi-instance clustering
  use the concept model from classification:
  1. instances belong to certain concepts
  2. multi-instance objects can be described by distribution over the concepts and their cardinality

  => clusters can be composed by objects having similar concept distributions and size

Concept-Based Multi-Instance Clustering

Idea: Each instance $o_i \in O$ represents a concept.
Multi-instance (MI-)cluster are distributions over the set of concepts.



MI-Cluster1 contains instances from concept1 and concept 2.

MI-Cluster2 contains instances from concept2 and concept 3.

Description of a MI-cluster = cluster description of the contributing concepts

DATABASE SYSTEMS GROUP

Concept-Based Multi-Instance Clustering

LMU

Example: Video Data

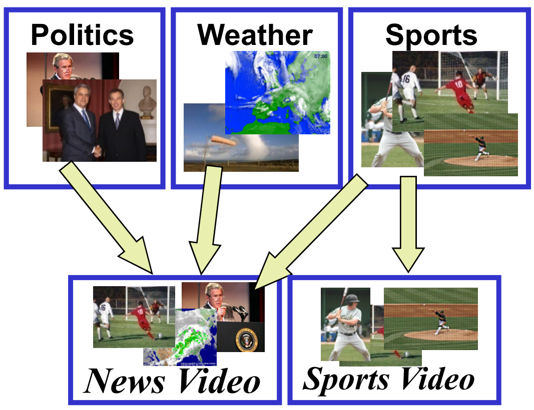
- Videos are represented as sets of Shots/Scenes
- Shots belong to a concept (e.g. sports, weather,..)
- videos are sets of shots (MI objects)

⇒ MI-cluster contain video with shots belonging to the same concepts

⇒ Sport-videos contain sports shots

⇒ News videos contains sports, weather, politics,...-shots

Concepts



MI-Cluster

An MI-EM Algorithms

Definition 1: Instance Set

- ***DB*** a set of MI-objects $o = \{i_1,\dots,i_k\}$

- the instance set I_{DB} of DB: $$I_{DB} = \bigcup_{DB} o$$

Definition 2: Instance Model

An Instance Model *IM* for the instance set I_{DB} is defined as follows:

- *k* distributions describing concepts e.g. Gaussian with mean μ_j and covariance matrix Σ_j.

- a prior distribution on the concepts Pr[k_j].

An MI-EM Algorithm

Definition 3: Multi-Instance Cluster Model

- a set *C* of clusters over the instance model *IM*.
- All MI-Cluster $c \in$ C are described as follows:
  - apriori probability *Pr* [*c*],
  - a cardinality distribution *Pr* [*Card(o)* | *c*]
  - an conditional distribution of concepts *Pr* [$i \in k$ | $i \in$ o $\in c$] (shortly: *Pr* [*k* | *c*]) for each concept *k* in *IM.*

The total probability of object *o* is computed as follows:

$$\Pr[o] = \sum_{c \in C} \Pr[c] \cdot \Pr[Card(o) \mid c] \cdot \prod_{i \in o} \prod_{k \in IM} \Pr[k \mid c]^{\Pr[k|i]}$$

the a-posteriori probability of *o* and cluster *c* is given as:

$$\Pr[c \mid o] = \frac{1}{\Pr[o]} \Pr[c] \cdot \Pr[Card(o) \mid c] \cdot \prod_{i \in o} \prod_{k \in IM} \Pr[k \mid c]^{\Pr[k|i]}$$

DATABASE SYSTEMS GROUP

An MI-EM Algorithm

LMU

Exampe: 2 MI-Cluster

Cluster 1: ▲

50 % apriori probability

expected number if instances: 2 3

concept1	concept2	concept3
0.2 1	0.01	0.79 2

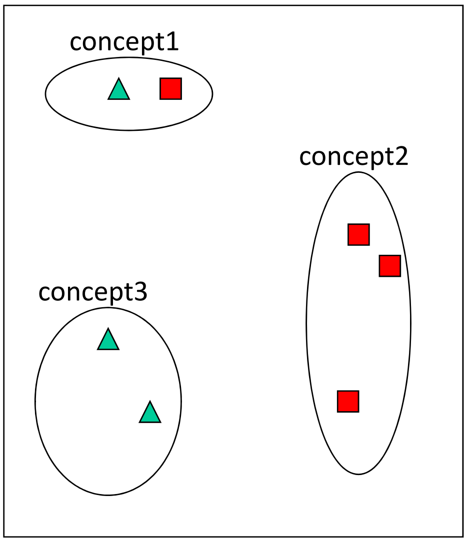
Cluster 2: ■

50 % apriori probability

expected number if instances : 5 4

concept1	concept2	concept3
0.1 1	0.89 3	0.01

Instance Model *IM*



DATABASE SYSTEMS GROUP

An MI-EM Algorithm

LMU

Overview of the algorithm:

1- Compute a mixture model (*IM*) on the instance set *I*
(build concepts by instance clustering using EM)

2- Compute an initial model for clustering MI objects

3- Use an EM based on the multinomial distribution to optimize the cluster model

An MI-EM Algorithm

Step(1):

Build I_{DB} and use EM-clustering to derive *IM*.

Step(2):

- For each MI-object *O* in DB build a "Confidence Summary Vector" *CSV*(*o*).
  - each dimension is a concept
  - the *i*-th component of CSV (*o*) is defined as:

$$CSV_j(o) = \sum_{i \in o} \Pr[k_j] \cdot \Pr[i \mid k_j]$$

- use k-means to group the *CSVs to an initial cluster model*

An MI-EM Algorithm

Step 3: MI-EM

E-Step: Compute the Log-Likelihood of the current model M.

$$E(M) = \sum_{o \in DB} \log \sum_{c_i \in C} \Pr[c_i \mid o]$$

M-Step: apply the following updates:

update apriori probability: $W_{c_i} = \Pr[c_i] = \frac{1}{Card(DB)} \sum_{o \in DB} \Pr[c_i \mid o]$

update cardinality distribution: $l_{c_i} = \frac{\sum_{o \in DB} \Pr[c_i \mid o] \cdot Card(o)}{Card(DB)} \cdot \frac{1}{MAXLENGTH}$

update concept distribution: $P_{k_j,c_i} = \Pr[k_j, c_i] = \frac{\sum_{o \in DB} \left(\Pr[c_i \mid o] \cdot \sum_{u \in o} \Pr[u \mid k_j] \right)}{\sum_{o \in DB} \Pr[c_i \mid o]}$

Multi-Instance Data Mining

Conclusions:

- aggregation is useful for homogeneous sets
- multiple distance and similarity function for MI objects
- distance measures can be plugged into various algorithms
- selecting the right distance measure is essential to the success
- concept-based mining abstracts sets to concepts and applies data mining to the concept distribution
- concept-based rely on a suitable set of concepts and methods to assign instances to this concept

Literatur

- Kriegel H.-P, Pryakhin A., Schubert M. : ***An EM-Approach for Clustering Multi-Instance Objects,*** Proc. 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD 2006), Singapore, 2006.
- Dietterich T.G., Lathrop R.H., Lozano-Perez T. : ***Solving the Multiple Instance Problem with Axis-Parallel Rectangles,*** Artificial Intelligence, vol. 89, num.1-2, Seiten 31-71, 1997
- Weidmann N., Frank E., Pfahringer B. : ***A Two-Level Learning Method for Generalized Multi-instance Problems***. ECML 2003: S. 468-479
- Gärtner T., Flach P.A., Kowalczyk A., Smola A.j. : ***Multi-Instance Kernels***, Proceedings of the 19th International Conference on Machine Learning, p. 179-186, 2002
- Zhang Q., Goldman S. : ***EM-DD: An improved multiple-instance learning technique***. Neural Information Processing Systems 14, 2001.
- Eiter T., Mannila H. : ***Distance Measures for Point Sets and Their Computation***. Acta Informatica, 34(2):103-133, 1997.
- Brecheisen S, Kriegel H.-P., Kröger P., Pfeifle M., Schubert M. : ***Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects***, Proc. ACM SIGMOD Int. Conf. on Management of Data (**SIGMOD'2003**), San Diego, CA, 2003