Knowledge Discovery in Databases II

Winter Term 2015/2016

Lecture 4 & 5:
Volume: High-Dimensional Data: Clustering in High Dimensional Data

Lectures : Dr Eirini Ntoutsi, PD Dr Matthias Schubert
Tutorials: PD Dr Matthias Schubert

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

Outline

1. Introduction and challenges of high dimensionality

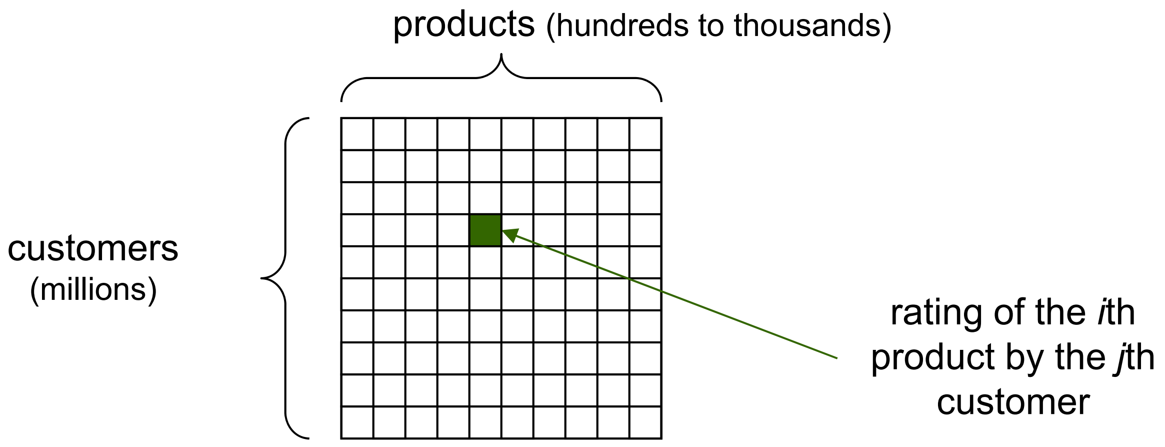
2. Feature Selection

3. Feature Reduction and Metric Learning

4. Clustering in High-Dimensional Data

Challenges for Clustering High-Dimensional Data

- Customer Recommendation / Target Marketing
  - Data
    - Customer ratings for given products
    - Data matrix:



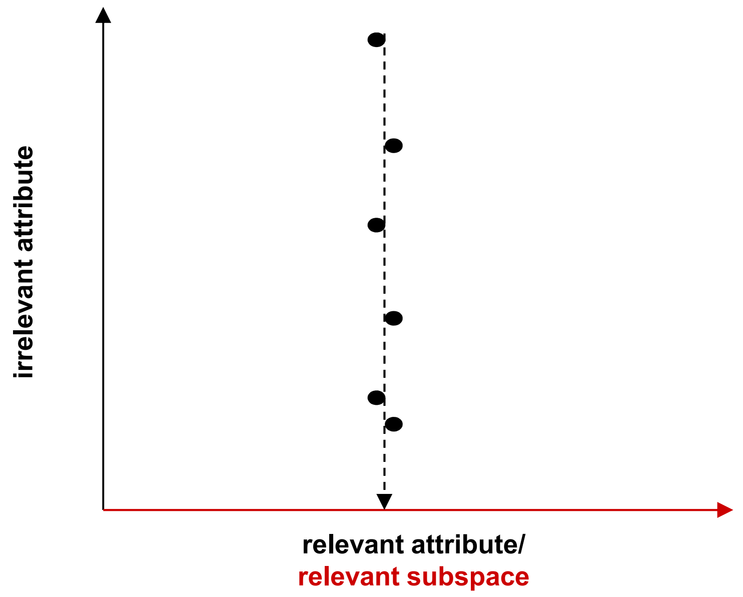
products (hundreds to thousands)
customers (millions)
rating of the *i*th product by the *j*th customer

  - Task: Cluster customers to find groups of persons that share similar preferences or disfavor (e.g. to do personalized target marketing)
    - *Challenge*:

      customers may be grouped differently according to different preferences/disfavors, i.e. different subsets of products

Challenges for Clustering High-Dimensional Data

- *Relevant* and *irrelevant* attributes
  - Not all features, but a subset of the features may be relevant for clustering
  - Groups of similar ("dense") points may be identified when considering only these features

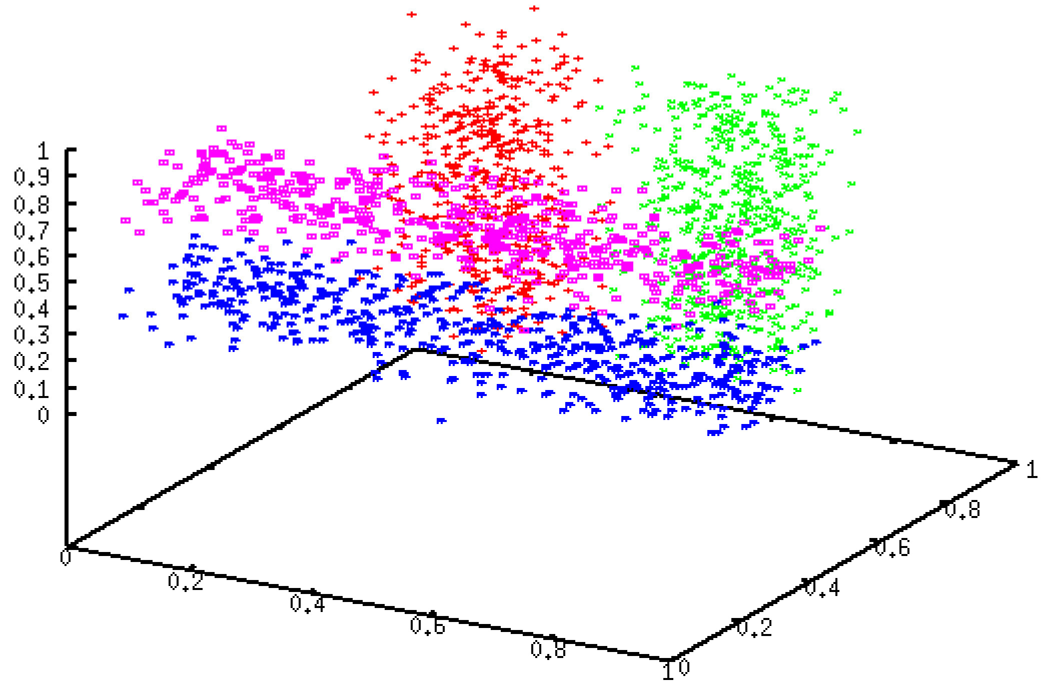


  - Different subsets of attributes may be relevant for different clusters

Challenges for Clustering High-Dimensional Data

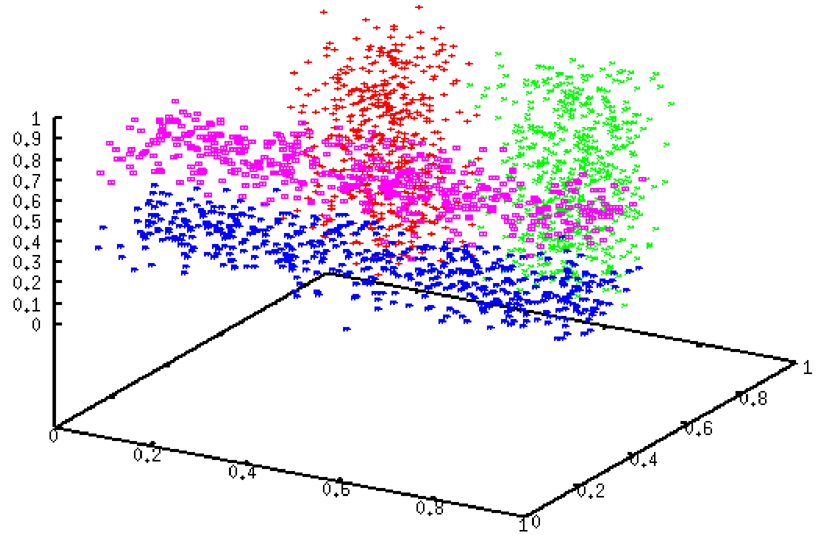
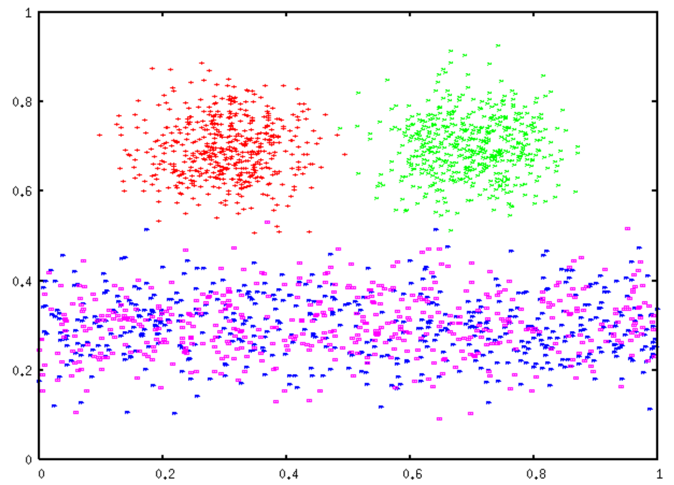
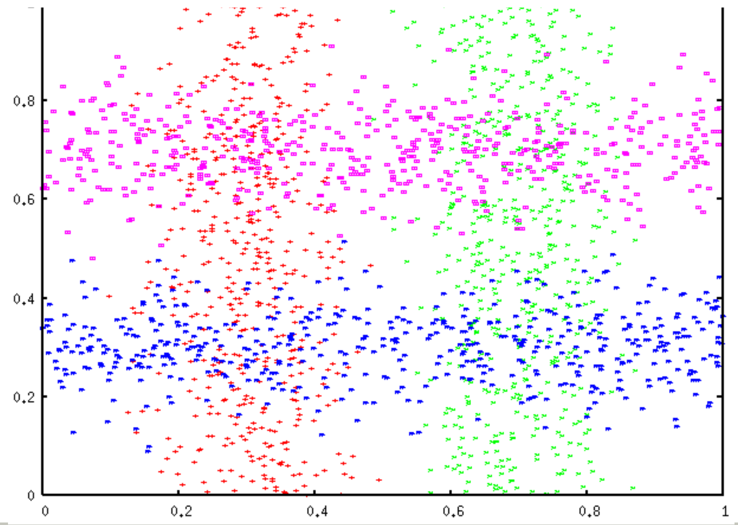
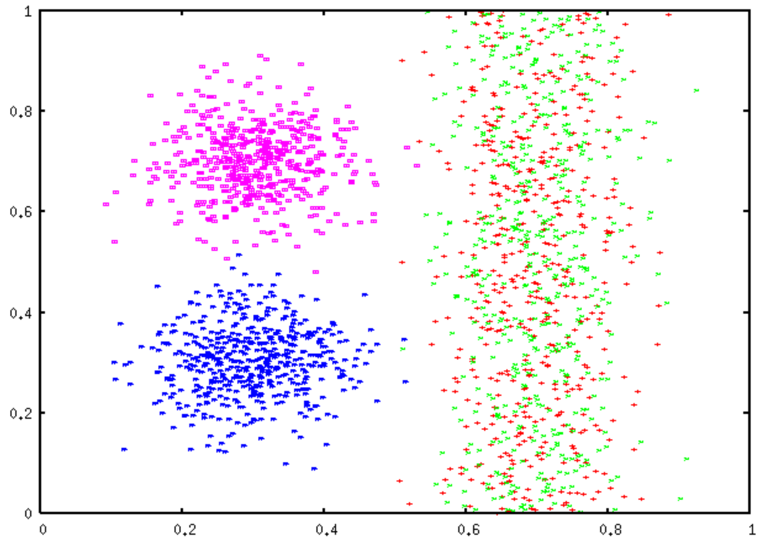
Effect on clustering:

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



Challenges for Clustering High-Dimensional Data

again: different attributes are relevant for different clusters

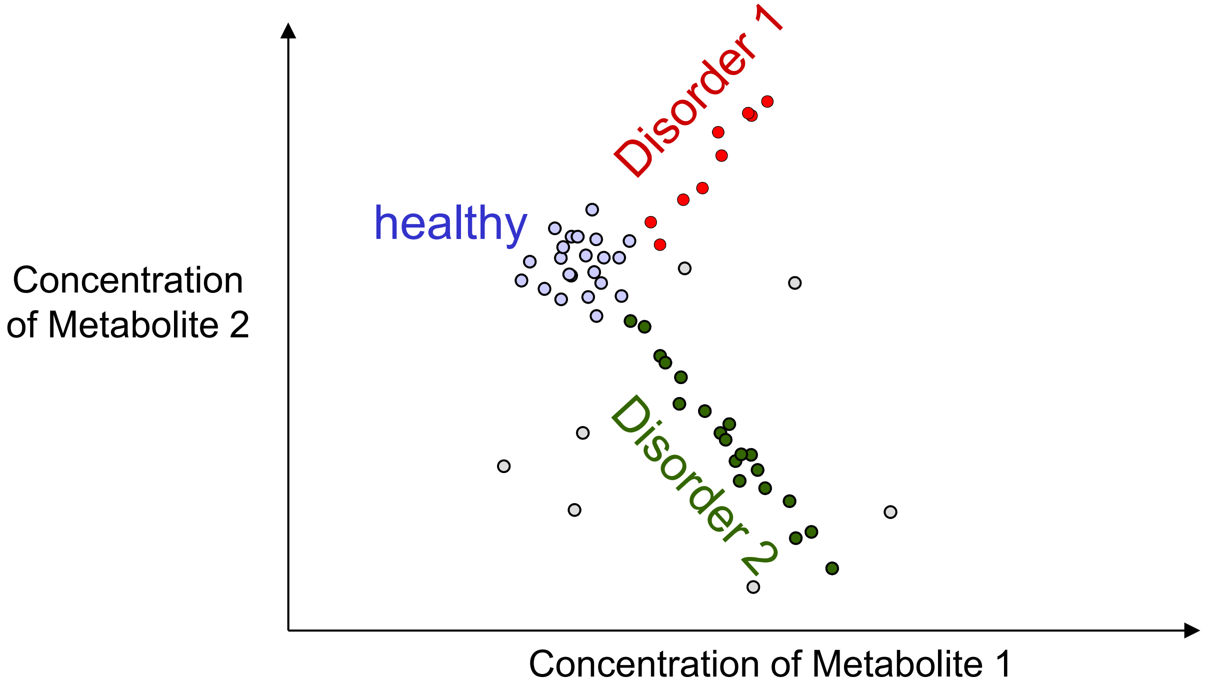





Challenges for Clustering High-Dimensional Data

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

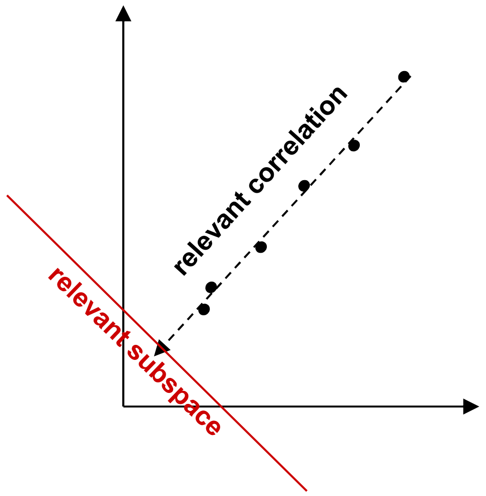
- *Challenge*:

  different metabolic disorders appear through different correlations of (subsets of) metabolites



Challenges for Clustering High-Dimensional Data

- *Correlation among attributes*
  - A subset of features may be correlated
  - Groups of similar ("dense") points may be identified when considering this correlation of features only

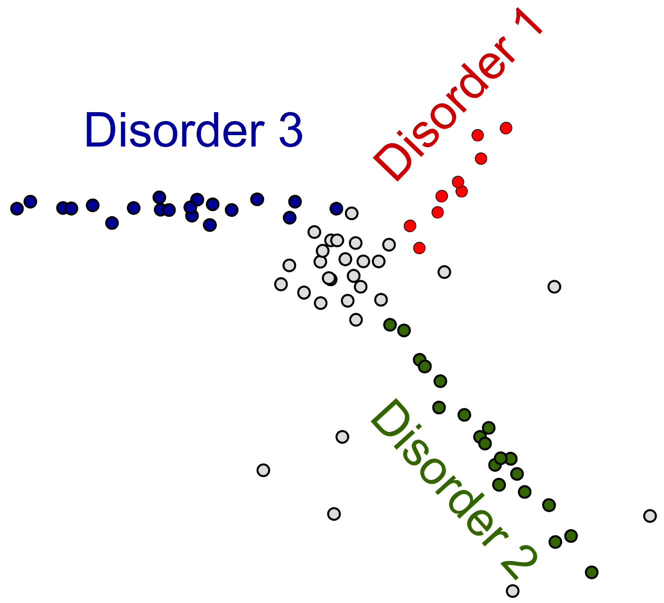


  - Different correlations of attributes may be relevant for different clusters

Challenges for Clustering High-Dimensional Data

Why not feature selection?

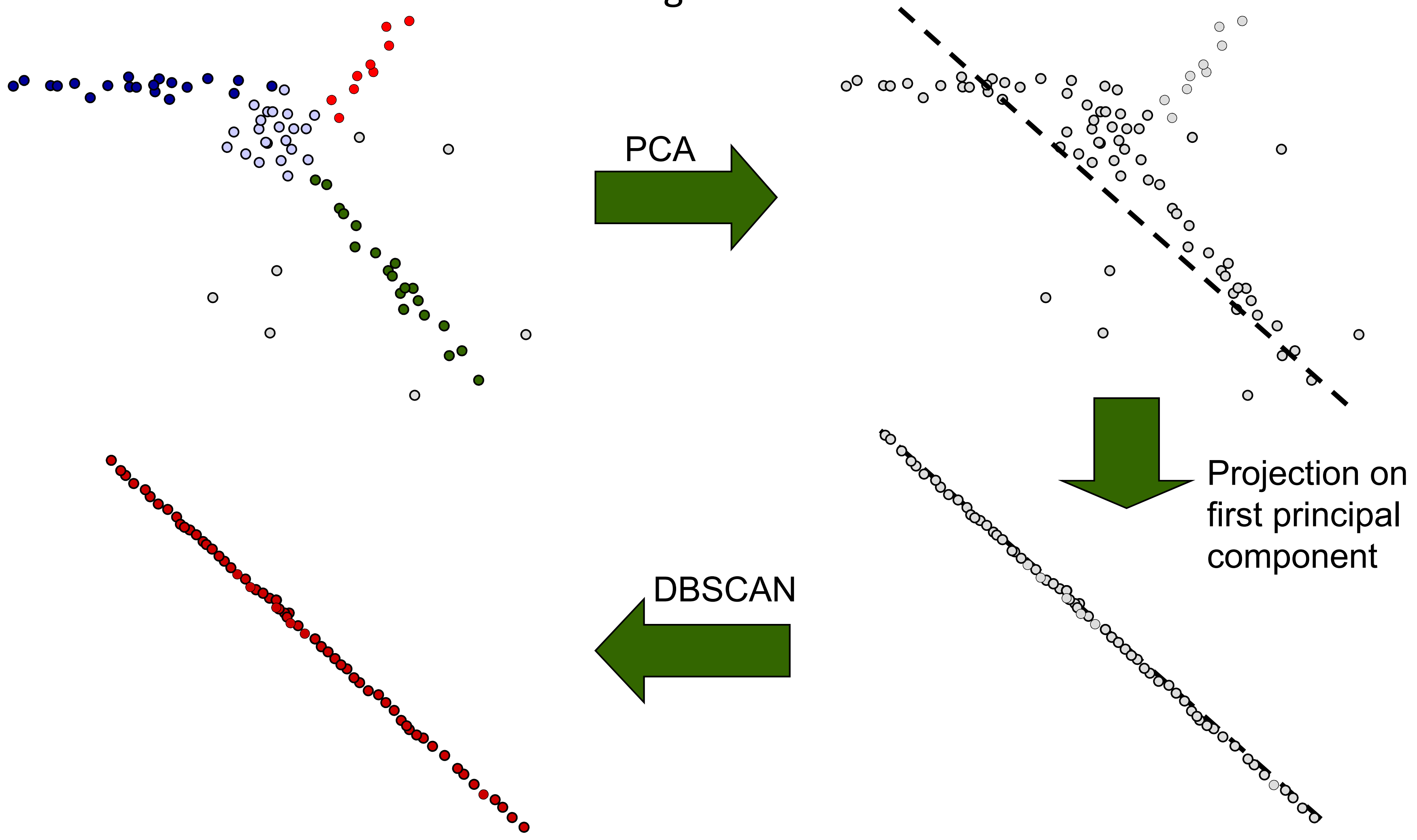
- (Unsupervised) feature selection is *global* (e.g. PCA)
- We face a local feature relevance/correlation: some features (or combinations of them) may be relevant for one cluster, but may be irrelevant for a second one



DATABASE SYSTEMS GROUP

Challenges for Clustering High-Dimensional Data

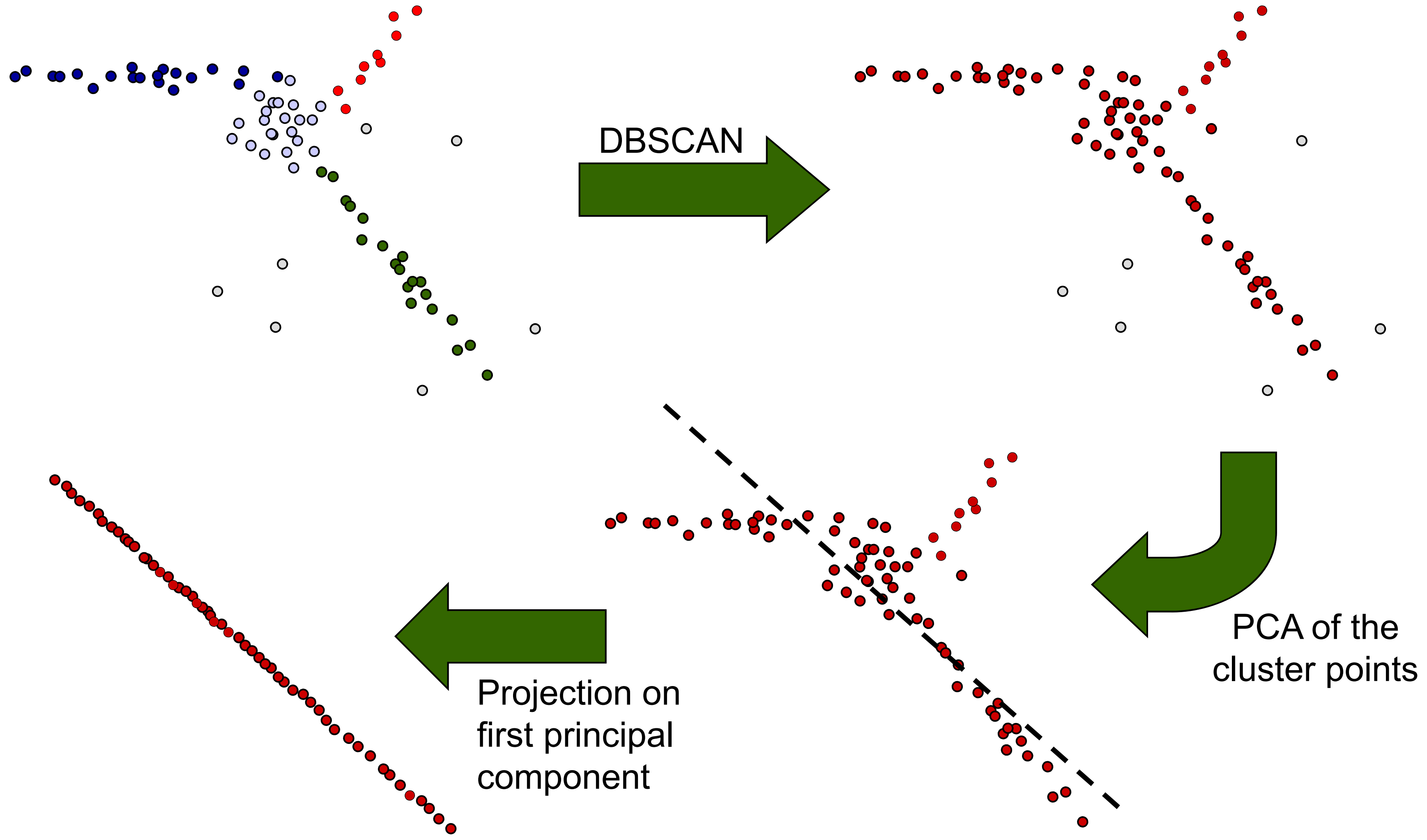
Use feature selection before clustering



DATABASE SYSTEMS GROUP

Challenges for Clustering High-Dimensional Data

Cluster first and then apply PCA


DBSCAN
PCA of the cluster points
Projection on first principal component

Challenges for Clustering High-Dimensional Data

Problem Summary

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

Challenges for Clustering High-Dimensional Data

- General problem setting of clustering high dimensional data

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

- Challenges:
  - Find the correct subspace of each cluster
    - Search space:
      - all possible arbitrarily oriented subspaces of a feature space
      - infinite
  - Find the correct cluster in each relevant subspace
    - Search space:
      - "Best" partitioning of points (see: minimal cut of the similarity graph)
      - NP-complete [SCH75]

Challenges for Clustering High-Dimensional Data

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

- How to solve the circular dependency problem?
  - Integrate subspace search into the clustering process
  - Thus, we need heuristics to solve
    - the clustering problem
    - the subspace search problem

    simultaneously

Overview of the discussed methods

- Bottom-Up approaches: Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]

Find all clusters in all subspaces.

Axis-parallel subspaces

- Top-Down Approaches: Projected Clustering
  - PROCLUS [APW+99]
  - PREDECON[BKKK04]

Each point is assigned to one subspace cluster or noise.

Axis-parallel subspaces

- Top-Down Approaches: Correlation Clustering
  - ORCLUS[AY00]
  - 4C [BKKZ04]

Each point is assigned to one subspace cluster or noise.

Arbitrary oriented subspaces

- Pattern based clustering

Overview of the discussed methods

- Bottom-Up approaches: Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]

Find all clusters in all subspaces.

Axis-parallel subspaces

- Top-Down Approaches: Projected Clustering
  - PROCLUS [APW+99]
  - PREDECON[BKKK04]

Each point is assigned to one subspace cluster or noise.

Axis-parallel subspaces

- Top-Down Approaches: Correlation Clustering
  - ORCLUS[AY00]
  - 4C [BKKZ04]

Each point is assigned to one subspace cluster or noise.

Arbitrary oriented subspaces

- Pattern based clustering

Bottom-up Algorithms

- Rational:
  - Start with 1-dimensional subspaces and merge them to compute higher dimensional ones.
  - Most approaches transfer the problem of subspace search into frequent item set mining.
    - The cluster criterion must implement the downward closure property
      - If the criterion holds for a *k*-dimensional subspace *S*, then it also holds for any (*k*–1)-dimensional projection of *S*
      - Use the reverse implication for pruning:

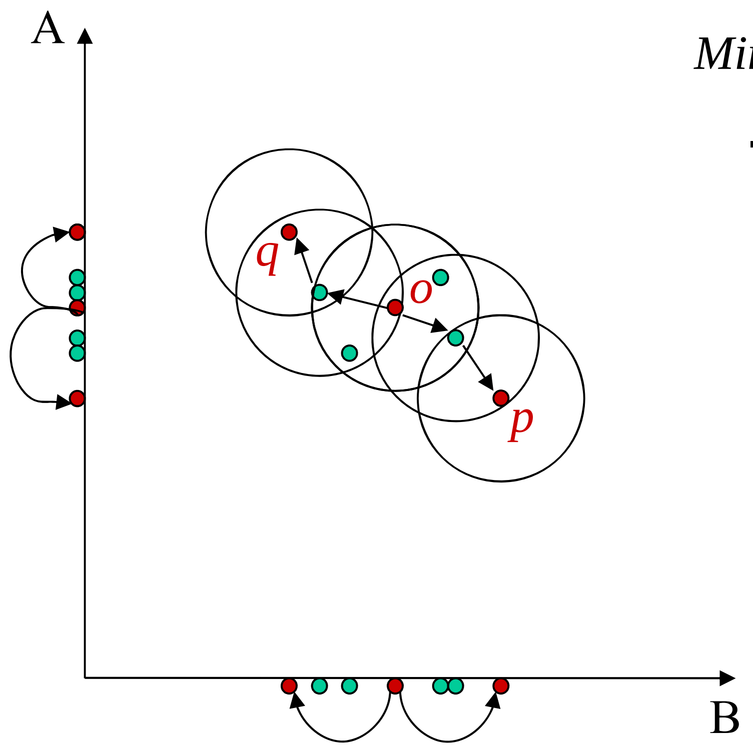
        If the criterion does not hold for a (*k*–1)-dimensional projection of *S*, then the criterion also does not hold for *S*
    - Apply any frequent itemset mining algorithm (e.g. APRIORI)
  - Some approaches use other search heuristics like best-first-search, greedy-search, etc.
    - Better average and worst-case performance
    - No guaranty on the completeness of results

Bottom-up Algorithms

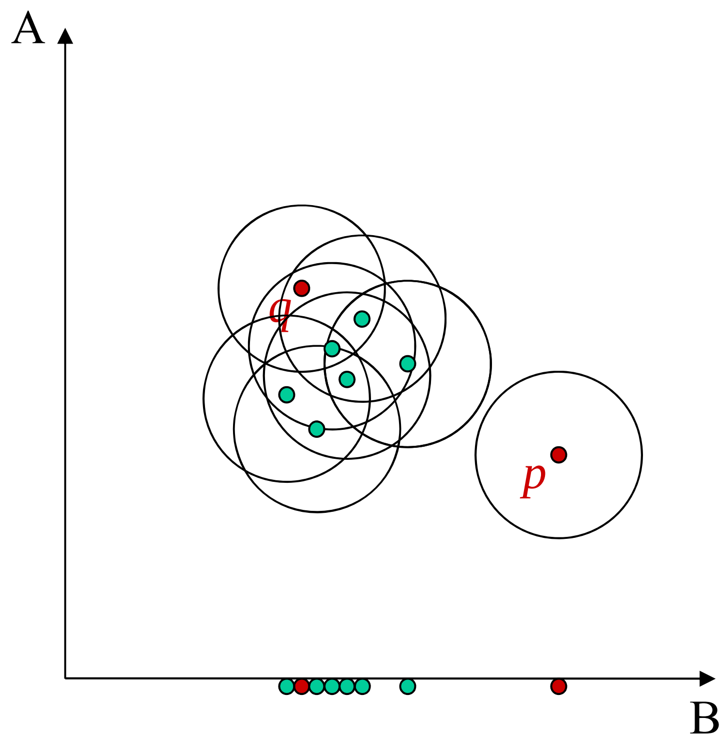
Downward-closure property

if C is a dense set of points in subspace S,

then C is also a dense set of points in any subspace $T \subset S$



p and *q* density-connected in {A,B}, {A} and {B}

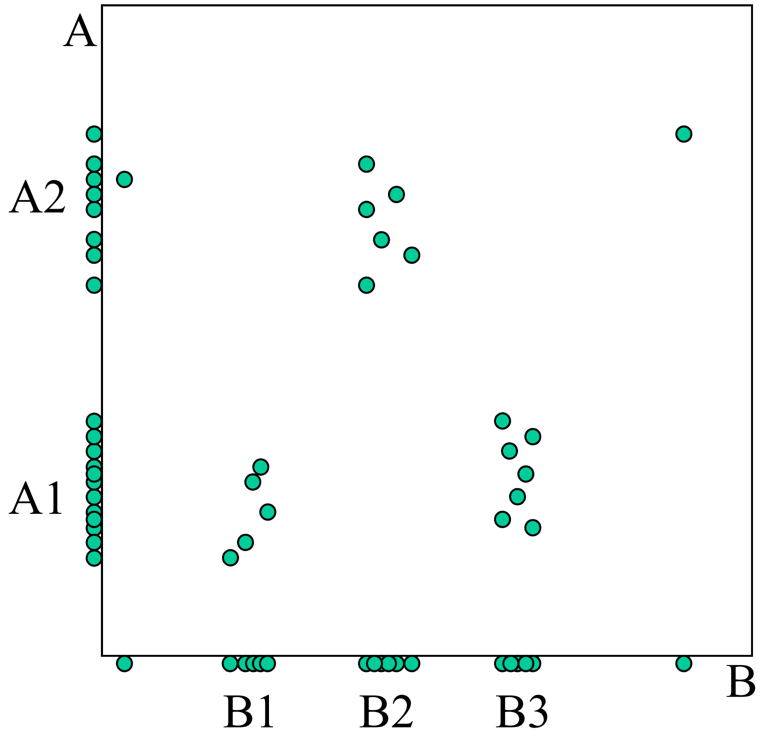


p and *q* not density-connected in {B} and {A,B}

Bottom-up Algorithms

Downward-closure property

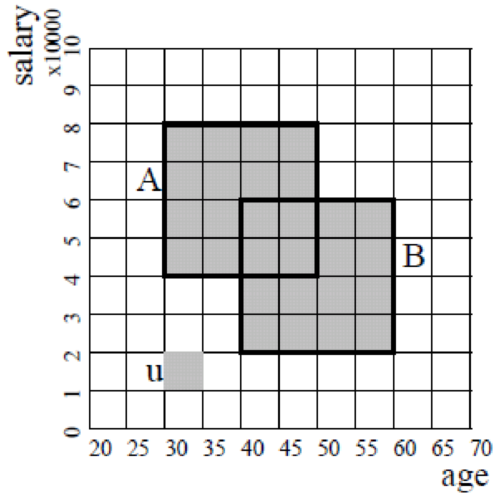
the reverse implication does not hold necessarily



CLIQUE [AGGR98] 1/6

CLIQUE serves two purposes:

1. Identify subspaces containing clusters
2. Identify the clusters



Approach

- Clusters are "dense regions" in the feature space
- Partition the feature space into ξ equal sized parts in each dimension.
- A *unit* is the intersection of one interval from each dimension
- *Dense* unit: If unit *u* contain more than τ objects, $\tau =$ density threshold
- *Clusters* are maximal sets of connected dense units (e.g., *A U B*)

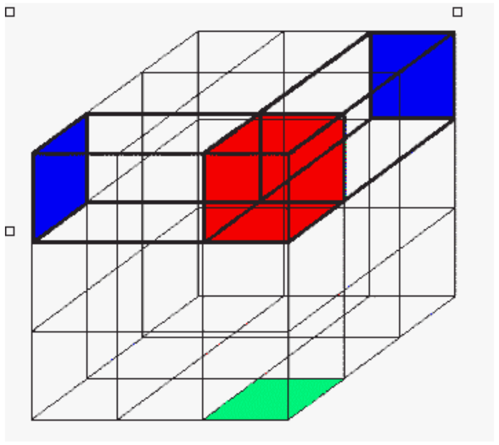
CLIQUE: 1. Identify subspaces containing clusters 2/6

Task: Find dense units

- Greedy approach (Bottom-Up), comparable to APRIORI for finding frequent itemsets (Downward Closure):
  - Determine 1-dimensional dense units D_1
  - Candidate generation procedure:
    - Based on D_{k-1}, the set of (k-1) dimensional dense units
    - Generate candidate set C_k by self joining D_{k-1}
      - Join condition: units should share first k-2 dimensions.
    - Discard those candidates which have a k-1 projection not included in D_{k-1}
- Downward Closure for dense grid units
  - If unit u is dense in a k-dimensional space then each projection of the unit into a k-1 dimensional subspace has to be dense as well.
  - **Inversion**: If any (k-1) dimensional projection of u is not dense, then u cannot be dense in the k-dimensional feature space

DATABASE SYSTEMS GROUP

CLIQUE: Example 3/6



🟦	2-dim. dense unit
🟥	3-dim. candidate unit
🟩	2-dim. unit which has to be checked

- If all ξ k-1 dimensional units are dense
  => check candidate on data set

- heuristics reduction of uninteresting subspace
  => prevents the exponential growth of interesting subspaces

CLIQUE: 2. Identify clusters 4/6

Task: Find maximal sets of connected dense units

Given: a set of dense units D in the same k-dimensional subspace S

Output: A partition of D into clusters $D_1, \ldots, D_k$

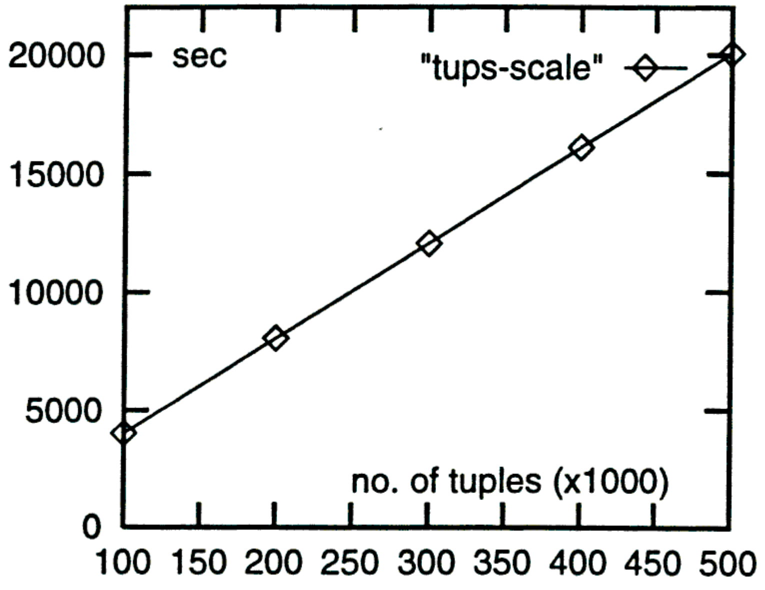
- The problem is equivalent to finding connected components in a graph
  - nodes: dense units
  - edges: there is an edge if the corresponding dense units have a common face (neighboring units)
  - Depth-first search algorithm: Start with a unit u in D, assign it to a new cluster ID and find all the units it is connected to. Repeat if there are nodes not yet visited.
- Time complexity: Assuming the dense units fit in memory (e.g. in a hash tree)

For each unit, we have to check 2k neighbors to find connected units

$\Rightarrow$ number of tree accesses: O $(2kn)$, where n: #dense units in S

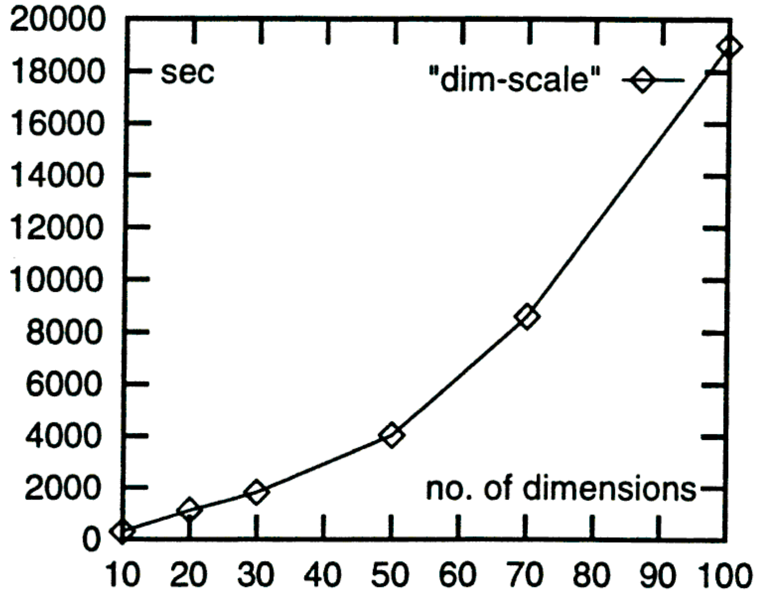
CLIQUE: Experimental Evaluation 5/6

runtime vs. number of objects *n*



QUE

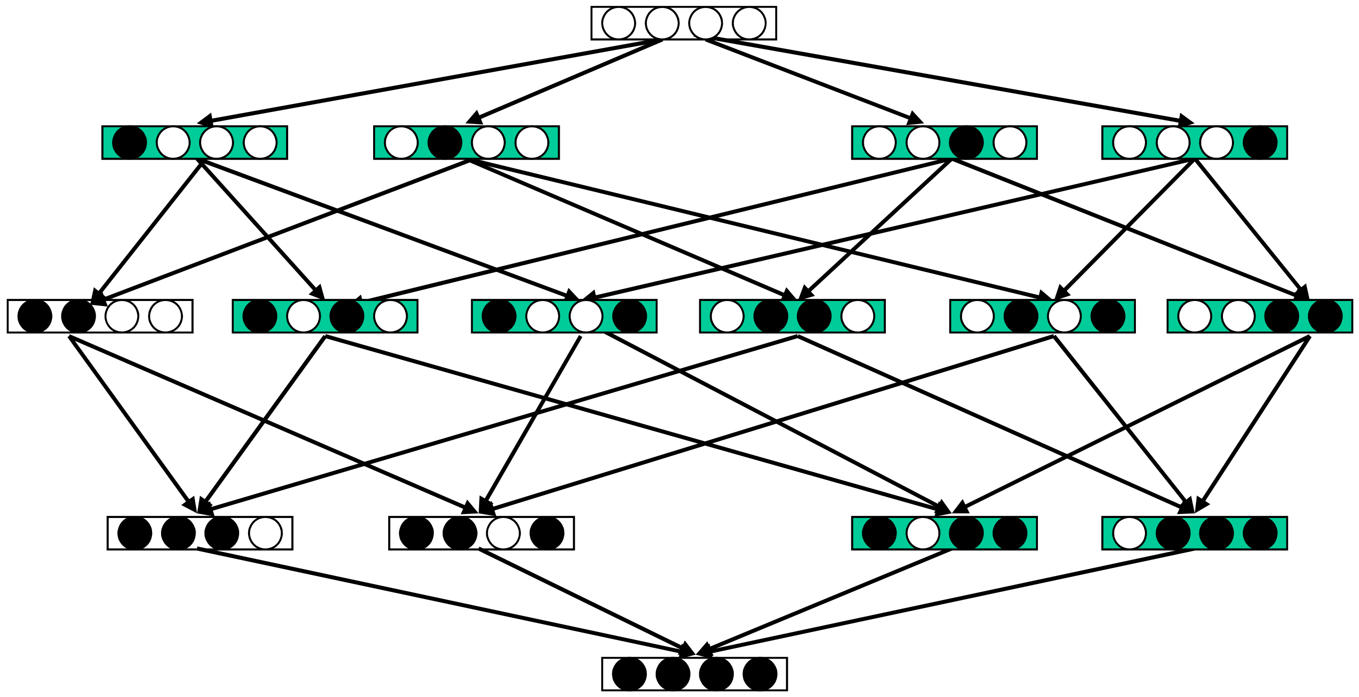
runtime vs. dimensionality *d*



linear in *n*, quadratic in *d*

CLIQUE: Discussion *6/6*

- Input: ξ and τ specifying the density threshold
- Output: *all* clusters in *all* subspaces, clusters may overlap
- Uses a fixed density threshold for all subspaces (in order to ensure the downward closure property)
- Simple but efficient cluster model



SUBCLU [KKK04] 1/6

Motivation:

Drawbacks of a grid-based regions:

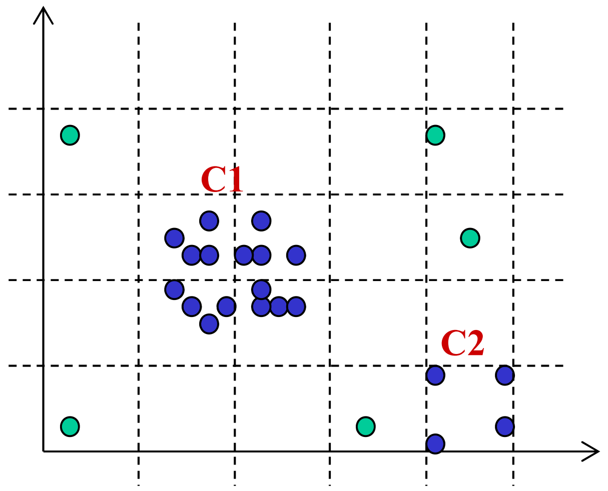
- Positioning of the grid influences the clustering
- Only rectangular regions
- Selection of ξ and τ is very sensitive
  Example:

Cluster for $\tau = 4$

(is C_2 a cluster?)

for $\tau > 4$: no cluster

(C_1 is lost)



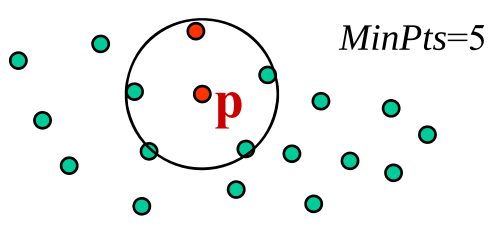
$\Rightarrow$ define regions based on the neighborhood of data points

$\Rightarrow$ use density-based clustering

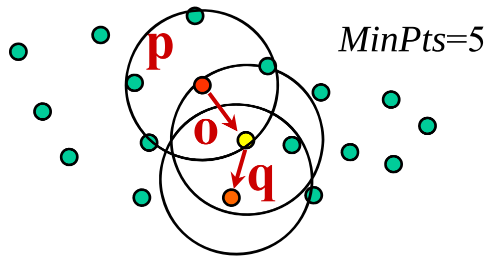
DATABASE SYSTEMS GROUP

SUBCLU: Cluster model 2/6

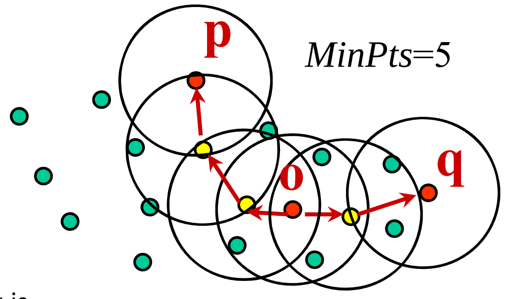
- Density-based cluster model of DBSCAN
- Clusters are *maximal* sets of *density-connected* points
- Density connectivity is defined based on *core points*
- Core points have at least *MinPts* points in their ε-neighborhood



p is core



p is core; o is directly density reachable from *p*; q is density reachable from o is o core?



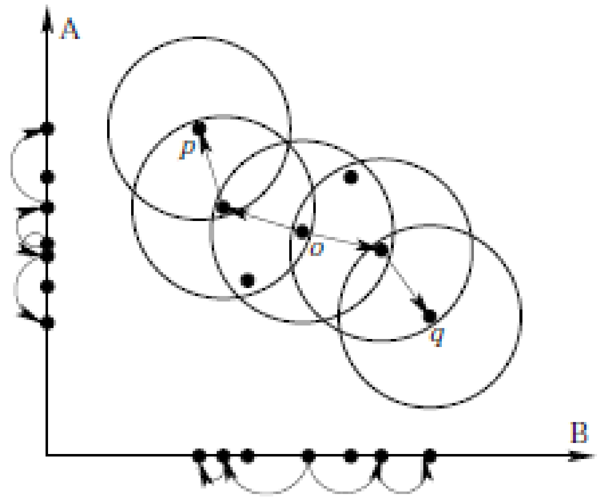
p and *q* are density connected

- Detects clusters of arbitrary shapes and positionings (in the corresponding subspaces)
- Naïve approach: Apply DBSCAN in all possible subspaces → exponential (2^d)
- Idea: Exploit clustering information from previous step (subspaces)
  - Density-connected clusters are not monotonic
  - But, density connected sets are monotonic!

SUBCLU: Downward closure of density connected sets 3/6

If C is a density connected set in subspace S then C is a density connected set in any subspace $T \subset S$.

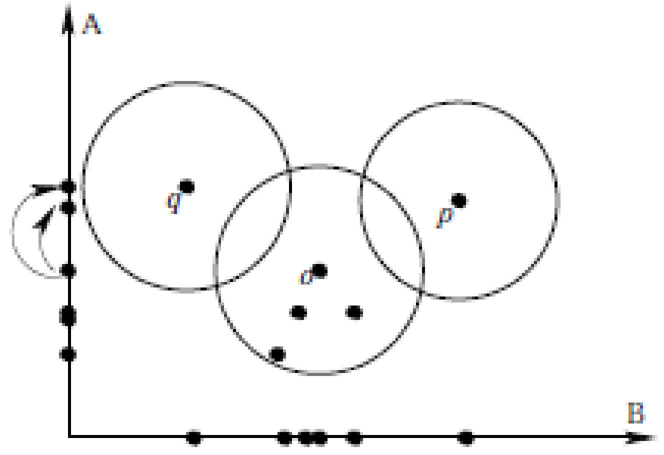
- But, if C is a cluster in S, does not need to be a cluster in $T \subset S$ – maximality might be violated
- All clusters in a higher-dimensional subspace will be subsets of the clusters detected in this first clustering.



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

ε: circles indicate

MinPts = 4



(b) p and q are not density-connected

p and *q* density connected in {A,B}.
Thus, they are also density connected in {A} and {B}

p and *q* not density connected in {B}.
Thus, they are not density connected in{A,B}, although they are density connected in {A}.

DATABASE SYSTEMS GROUP

SUBCLU: Algorithm 4/6

SUBCLU(Set of objects *DB*, real *ε*, integer *minPts*)

Init: *// STEP 1 Generate all 1-D clusters*

- For each 1-*D* subspace *S* generate all its clusters by applying DBSCAN(*DB, S, ε, minPts*)
  - S_1: set of 1-D subspaces containing clusters ,C_1:set of all sets of clusters in 1-D subspaces
- While C_k is not empty *//STEP 2 Generate (k + 1)-D clusters from k-D clusters*

  // STEP 2.1 Generate (k + 1)-dimensional candidate subspaces
  - Build: build *(k+1)*-dimensional candidate spaces ($CandS_{k+1}$) from *k*-dimensional subspaces S_k:
    - o Combine subspaces with (k-1) dimensions in common
    - o Prune candidates having a k-dimensional subspace not in S_k (i.e., without any cluster in S_k)

  // STEP 2.2 Test candidates and generate (k + 1)-dimensional clusters
- On each candidate subspace $cand \subset CandS_{k+1}$, take one k-D subspace $T \subset Cand$ and simply call DBSCAN(cl, cand,ε,minPts) for each cluster *cl* in *T* to generate C^{cand}
    - o If any cluster is found, add candidate subspace to the k+1 subspaces and collect the clusters
    - o Else prune the candidate

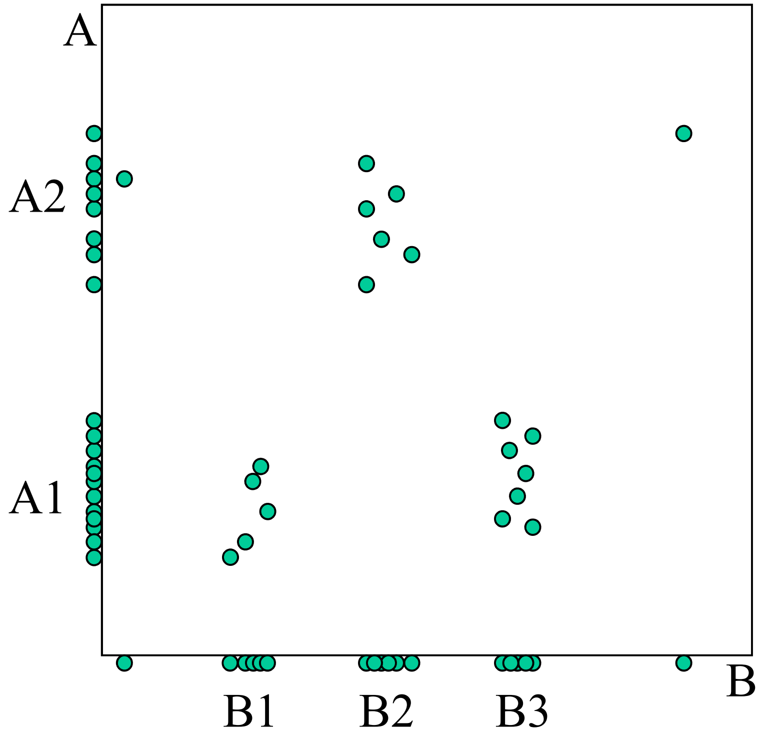
$$\mathcal{S}_{k+1} := \mathcal{S}_{k+1} \cup cand$$
$$\mathcal{C}_{k+1} := \mathcal{C}_{k+1} \cup \mathcal{C}^{cand}$$

- Terminate if no k dimensional subspace contains any cluster (i.e., C_k is empty)

Remark: Algorithmic pattern is rather close to APRIORI for frequent item set mining.

DATABASE SYSTEMS GROUP

SUBCLU: Example 5/6



DBSCAN(*DB*, S, ε, *MinPts*): computes all density-connected clusters w.r.t. ε and *MinPts* in dataset *DB* and subspace *S*

$S_1 = \{\{A\}, \{B\}\}$

$C_1 = \{A1, A2, B1, B2, B3\}$

$\text{CandS}_2 = \{\{AB\}\} \Rightarrow S_2 = \{\{AB\}\}$

- Call DBSCAN(*c*, *U*, ε, *MinPts*) for subspace *U* ⊂ *Cand* having the smallest amount of data objects in clusters (example: *U* = {B})
- Reduces the amout of range queries for each call of DBSCAN

SUBCLU: Discussion 6/6

- Algorithm
  - All subspaces that contain any density-connected set are computed using the bottom-up approach
  - Density-connected clusters are computed using a specialized DBSCAN run in the resulting subspace to generate the subspace clusters

- Discussion
  - Input: ε and *MinPts* specifying the density threshold
  - Output: all clusters in all subspaces, clusters may overlap
  - Uses a fixed density threshold for all subspaces
  - Advanced but costly cluster model

Bottom-up Algorithms: Discussion

The key limitation: *global density thresholds*

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

Overview of the discussed methods

- Bottom-Up approaches: Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]

Find all clusters in all subspaces.

Axis-parallel subspaces

- Top-Down Approaches: Projected Clustering
  - PROCLUS [APW+99]
  - PREDECON[BKKK04]

Each point is assigned to one subspace cluster or noise.

Axis-parallel subspaces

- Top-Down Approaches: Correlation Clustering
  - ORCLUS[AY00]
  - 4C [BKKZ04]

Each point is assigned to one subspace cluster or noise.

Arbitrary oriented subspaces

- Pattern based clustering

Top-down Algorithms

Rational:

- Cluster-based approach:
  - Learn the subspace of a cluster in the *entire* d-dimensional feature space
  - Start with full-dimensional clusters
  - Iteratively refine the cluster memberships of points and the subspaces of the cluster
  - PROCLUS[APW+99], ORCLUS[AY00]
- Instance-based approach:
  - Learn for each *point* its subspace preference in the entire *d*-dimensional feature space
  - The subspace preference specifies the subspace in which each point “clusters best”
  - Merge points having similar subspace preferences to generate the clusters
  - PREDECON[BKKK04] 4C[BKKZ04]

Top-down Algorithms: The key problem

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

- Most approaches rely on the so-called “locality assumption”
  - The subspace is usually learned from the local neighborhood of cluster representatives/cluster members in the entire feature space:
    - Cluster-based approach: the *local neighborhood* of each cluster representative is evaluated in the *d*-dimensional space to learn the “correct” subspace of the cluster
    - Instance-based approach: the *local neighborhood* of each point is evaluated in the *d*-dimensional space to learn the “correct” subspace preference of each point
- *The locality assumption*: the subspace preference can be learned from the *local neighborhood* in the *d*-dimensional space
  - Other approaches learn the subspace preference of a cluster or a point from *randomly sampled points*

Overview of the discussed methods

- Bottom-Up approaches: Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]

Find all clusters in all subspaces.

Axis-parallel subspaces

- Top-Down Approaches: Projected Clustering
  - PROCLUS [APW+99]
  - PREDECON[BKKK04]

Each point is assigned to one subspace cluster or noise.

Axis-parallel subspaces

- Top-Down Approaches: Correlation Clustering
  - ORCLUS[AY00]
  - 4C [BKKZ04]

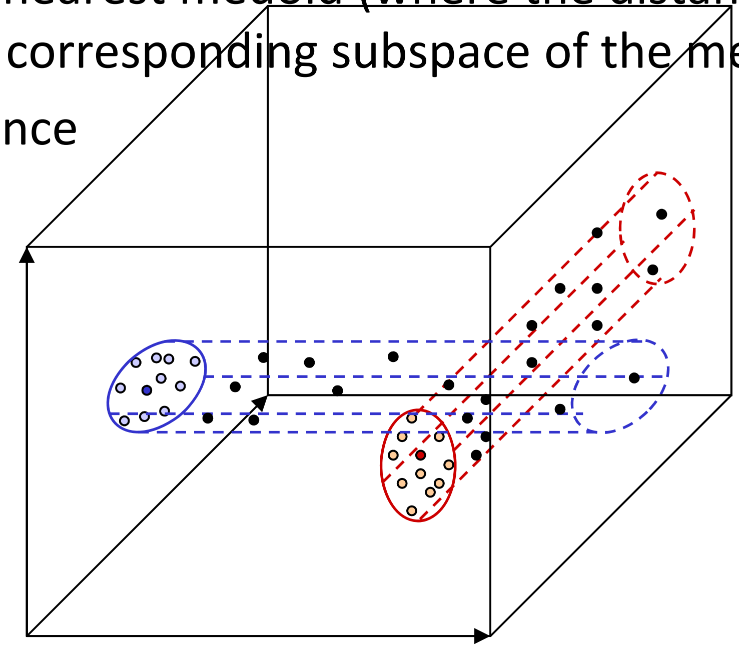
Each point is assigned to one subspace cluster or noise.

Arbitrary oriented subspaces

- Pattern based clustering

PROCLUS [APW+99] 1/6

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



PROCLUS: Algorithm –Initialization phase 2/6

- 3-phase algorithm: initialization, iterative phase, refinement
  - Input:
    - The set of data points
    - Number of clusters, denoted by k
    - Average number of dimensions for each clusters, denoted by L
  - Output:
    - The clusters found, and the their associated dimensions

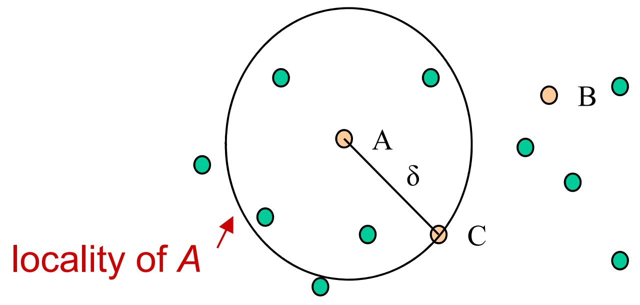
- [**Phase 1**] Initialization of cluster medoids
  - Ideally we want a set of centroids, where each centroid comes from a different cluster.
  - We don't know which are these *k* points though, so we choose a superset M of b*k medoids such that they are well separated.
    - Chose a random sample (S) of a*k data points
    - Out of S, select b*k points (M) by greedy selection : medoids are picked iteratively so that the current :medoid is well separated from the medoids that have been chosen so far.
  - Input parameters *a* and *b* are introduced for performance reasons

PROCLUS: Algorithm –Iterative phase 3/6

- [**Phase 2**] Iterative phase (works similar to any *k*-medoid clustering)
  - *k* randomly chosen medoids from *M* are the initial cluster medoids
  - Idea: replace the “bad” medoids from other points in M if necessary → we should be able to evaluate the quality of the clustering by a given set of medoids.
  - Procedure:
    - Find dimensions related to the medoids
    - Assign data points to the medoids
    - Evaluate the clusters formed
    - Find the bad medoid, and try the result of replacing bad medoid

PROCLUS: Algorithm –Iterative phase – Find cluster dimensions 4/6

- For each medoid m_i, let δ be the nearest distance to its closest medoid
- All the data points within δ will be assigned to the medoid m_i (L_i, locality of m_i)



- Intuition: to each medoid we want to associate those dimensions where the points are closed to the medoid in that dimension
- Compute the average distance along each dimension from the points in L_i to m_i.
  - Let $X_{i,j}$ be the avg distance along dimension j
- Calculate for m_i the mean Y_i and standard deviation σ_i of $X_{i,j}$
- Calculate $Z_{i,j} = (X_{i,j} - Y_i) / \sigma_i$
- Choose $k \times l$ smallest values $Z_{i,j}$ with at least 2 chosen for each medoids
- Output: A set of k medoids and their associated dimensions

PROCLUS: Algorithm –Iterative phase – Assigning data points –evaluate clusters 5/6

- *Assign each data point* to its closest medoid using Manhattan segmental distance (only relevant dimensions count)
- Manhattan segmental distance (A variance of Manhattan distance): For any two points x1,x2 and any set of dimensions D, |D|≤ d:

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

- How to *evaluate the clusters*?
  - Use average Manhattan segmental distance from the points in C_i to the *centroid* of C_i along dimension j

$$w_i = \frac{\sum_j Y_{i,j}}{|D_i|} \qquad E = \frac{\sum_{i=k}^{k} |C_i| \cdot w_i}{N}$$

- Replace bad medoids with random points from *M*
- Terminate if the clustering quality does not increase after a given number of current medoids have been exchanged with medoids from *M* (it is not clear, if there is another hidden parameter in that criterion)

PROCLUS: Algorithm –Iterative phase 6/6

- [**Phase 3**] Refinement
  - Reassign subspaces to medoids as above (but use only the points assigned to each cluster rather than the locality of each cluster, i.e., C_i not L_i)
  - Reassign points to medoids
  - Points that are not in the locality of any medoid are classified as noise

PREDECON[BKKK04] 1/3

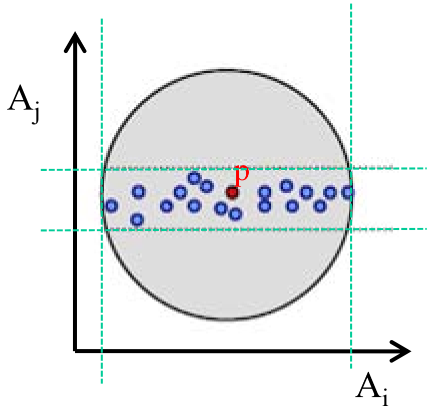
- Instance-based top-down approach: we learn the subspace for each instance
- Extends DBSCAN to high dimensional spaces by incorporating the notion of dimension preferences in the distance function

- For each point p, it defines its subspace preference vector:

$$\bar{\mathbf{w}}_p = (w_1, w_2, \ldots w_d) \qquad w_i = \begin{cases} 1 & if \quad \mathrm{VAR}_i > \delta \\ \kappa & if \quad \mathrm{VAR}_i \leq \delta \end{cases}$$

- VAR_i is the variance along dimension j in $N_\varepsilon(p)$:

$$\mathrm{VAR}_{A_i}(\mathcal{N}_\varepsilon(p)) = \frac{\sum_{q \in \mathcal{N}_\varepsilon(p)} (dist(\pi_{A_i}(p), \pi_{A_i}(q)))^2}{|\mathcal{N}_\varepsilon(p)|}$$



δ, κ (κ>>1) are input parameters

PREDECON[BKKK04] 2/3

- Preference weighted distance function:

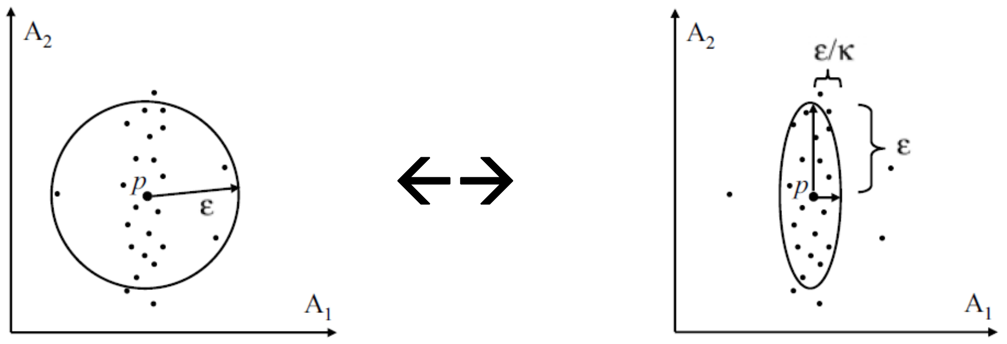
$$dist_p(p,q) = \sqrt{\sum_{i=1}^{d} \frac{1}{w_i} \cdot (\pi_{A_i}(p) - \pi_{A_i}(q))^2}$$

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

- Preference weighted ε-neighborhood:

$$\mathcal{N}_{\varepsilon}^{\bar{\mathbf{w}}_p}(p) = \{x \in \mathcal{D} \,|\, dist_{pref}(p,x) \leq \varepsilon\}$$

simple
ε-neighborhood

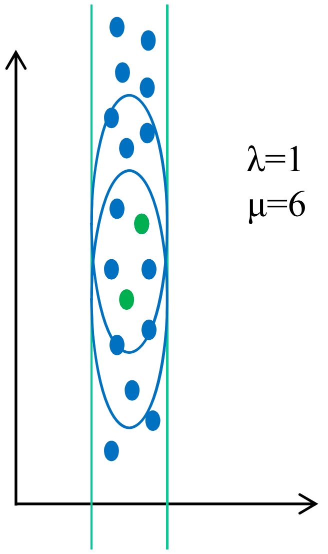


preference weighted
ε-neighborhood

PREDECON[BKKK04] 3/3

- Preference weighted core points:

$$\text{CORE}^{\text{pref}}_{\text{den}}(p) \Leftrightarrow \text{PDIM}(\mathcal{N}_\varepsilon(p)) \leq \lambda \wedge |\mathcal{N}^{\bar{\mathbf{w}}_o}_\varepsilon(p)| \geq \mu$$



- Direct density reachability, reachability and connectivity are defined based on preference weighted core points

- A *subspace preference cluster* is a maximal density connected set of points associated with a certain subspace preference vector.

Overview of the discussed methods

- Bottom-Up approaches: Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]

Find all clusters in all subspaces.

Axis-parallel subspaces

- Top-Down Approaches: Projected Clustering
  - PROCLUS [APW+99]
  - PREDECON[BKKK04]

Each point is assigned to one subspace cluster or noise.

Axis-parallel subspaces

- Top-Down Approaches: Correlation Clustering
  - ORCLUS[AY00]
  - 4C [BKKZ04]

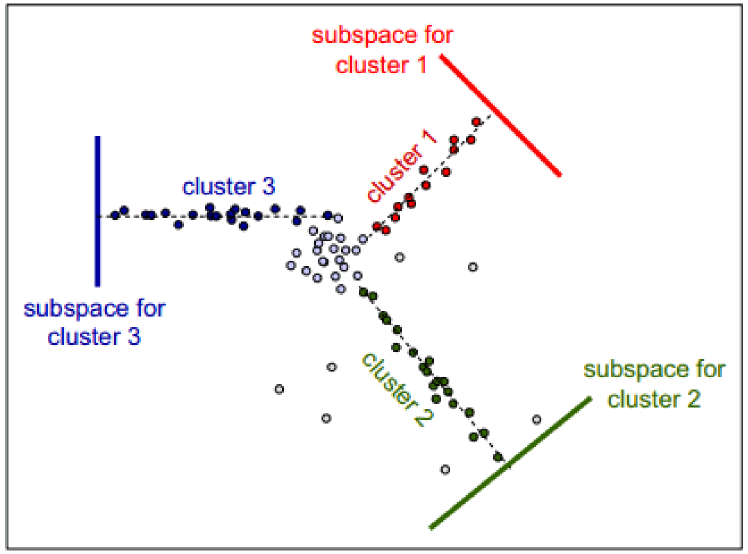
Each point is assigned to one subspace cluster or noise.

Arbitrary oriented subspaces

- Pattern based clustering

Correlation Clustering

- Motivating example:
  - Cluster 3 exists in an axis-parallel subspace
  - Clusters 1 and 2 exist in (different) arbitrarily oriented subspaces: if the cluster members are projected onto the depicted subspaces, the points are “densely packed”



- Subspace clustering and projected clustering algorithms find axis-parallel subspaces
- Correlation clustering for finding clusters in arbitrary oriented subspaces

ORCLUS[AY00] 1/3

- ORCLUS (arbitrarily ORiented projected CLUSter generation) first approach to generalized projected clustering
- A *generalized projected cluster* is a set of vectors E and a set of points C such that the points in C are closely clustered in the subspace defined by the vectors E.
  - E is a set of orthonormal vectors, $|E| \leq d$

Input:

- The number of clusters k
- The dimensionality of the subspace of the clusters, l (=|E|)

Output

- A set of k clusters and their associated subspaces of dimensionality l

Main idea

- To find the subspace of a cluster C_i, compute the dxd covariance matrix M_i for C_i and determine the eigenvectors. Pick the l_c eigenvectors with the smallest eigenvalues.

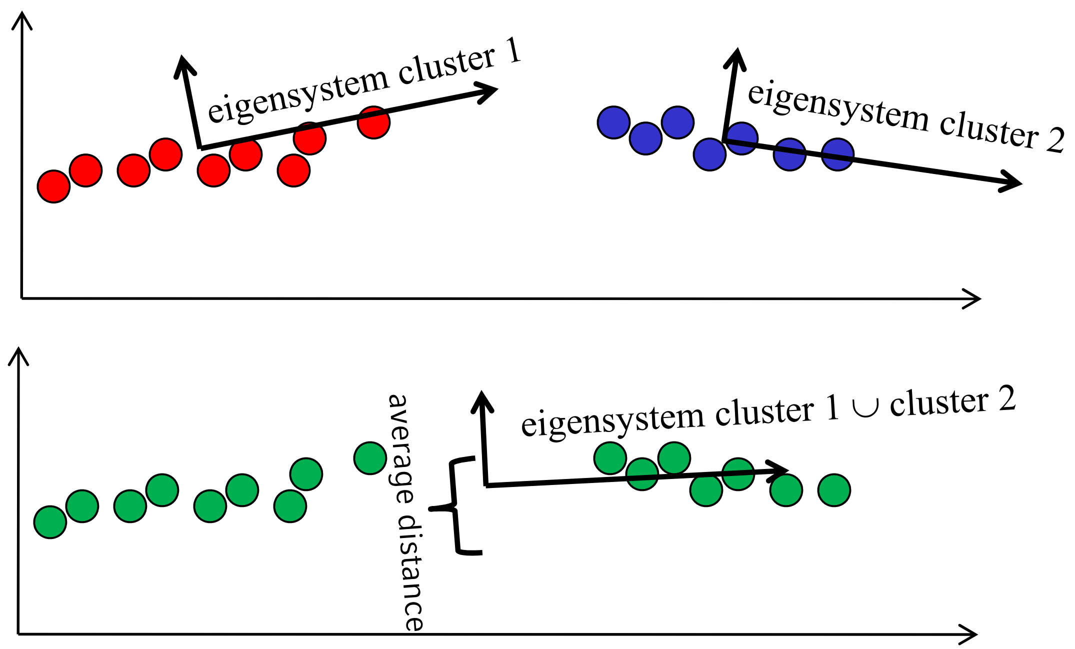
- Relies on cluster-based locality assumption: subspace of each cluster is learned from its members

ORCLUS: Algorithm 2/3

- similar ideas to PROCLUS [APW+99]
- *k*-means like approach
- start with $k_c > k$ seeds
- assign points to clusters according to distance function based on the eigensystem of the current cluster (starting with axes of data space, i.e. Euclidean distance)
- The eigensystem is iteratively adapted based on the updated cluster members
- Reduce the number of clusters k_c in each iteration by merging best-fitting cluster pairs

ORCLUS: Merging clusters 3/3

- Each cluster C_i exists in a possible different subspace S_i, how do we decide what to merge?
- Compute the subspace of their union $C_i \cup C_j$ (eigenvectors corresponding to the smallest l eigenvalues)
- Check the cluster energy of $C_i \cup C_j$ in this subspace (mean square distance of the points from the centroid in this subspace) – indicator of how well the points combine



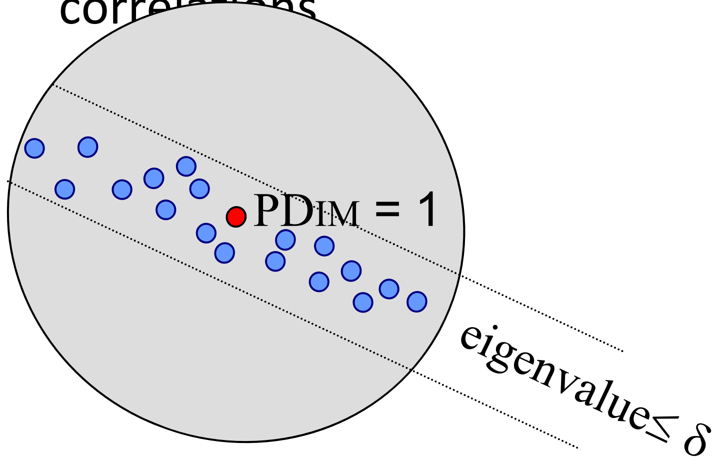
- Assess average distance in all merged pairs of clusters and finally merge the best fitting pair, that with the smallest cluster energy
- Continue until the desired number of clusters, k, is achieved.

4C [BKKZ04]

4C = Computing Correlation Connected Clusters
Idea: Integrate PCA into density-based clustering.

Approach**:**

- Check the core point property of a point p in the complete feature space
- Perform PCA on the local neighborhood S of p to find subspace correlations



PCA factorizes M_P into $M_P = V\,E\,V^T$

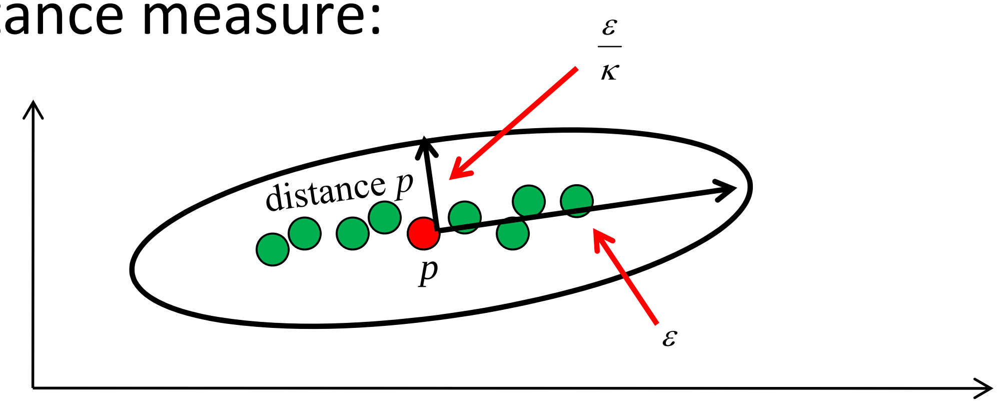
V:	eigenvectors
E:	eigenvalues

- A parameter δ discerns large from small eigenvalues.
- CorDim(S)=#eigenvalues>δ
- In the eigenvalue matrix of p, large eigenvalues are replaced by 1, small eigenvalues by a value κ >>1 → adapted eigenvalue matrix E'_p

DATABASE SYSTEMS GROUP

4C: Distance measure

- effect on distance measure:

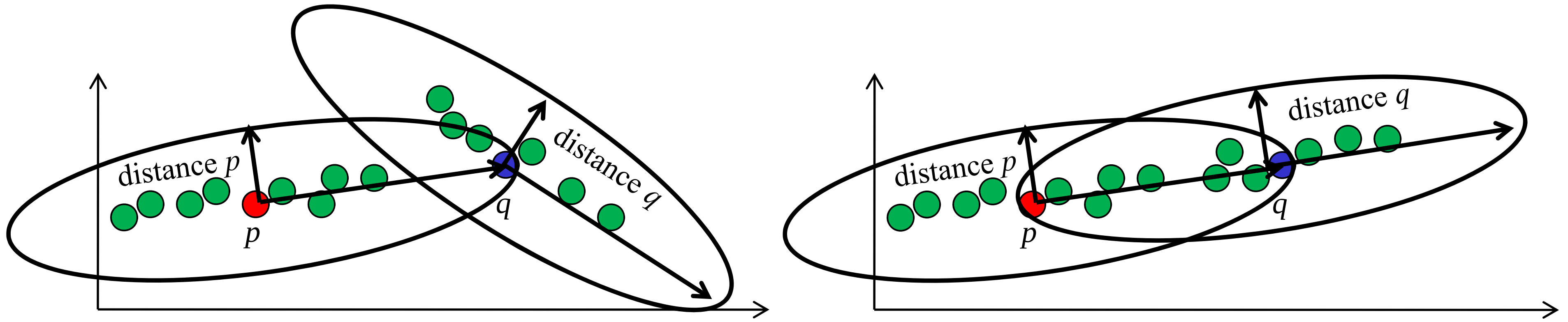


- distance of *p* and *q* w.r.t. *p*: $\sqrt{(p-q)\cdot V_p \cdot E'_p \cdot V_p^{\mathrm{T}} \cdot (p-q)^{\mathrm{T}}}$

- distance of *p* and *q* w.r.t. *q*: $\sqrt{(q-p)\cdot V_q \cdot E'_q \cdot V_q^{\mathrm{T}} \cdot (q-p)^{\mathrm{T}}}$

4C: correlation neighbors

- symmetry of distance measure by choosing the maximum:



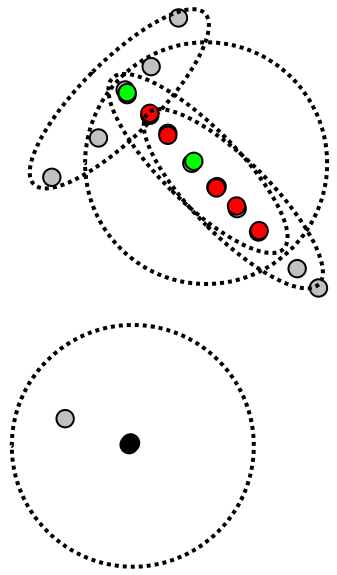
- *p* and *q* are correlation-neighbors if

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

4C: Algorithm

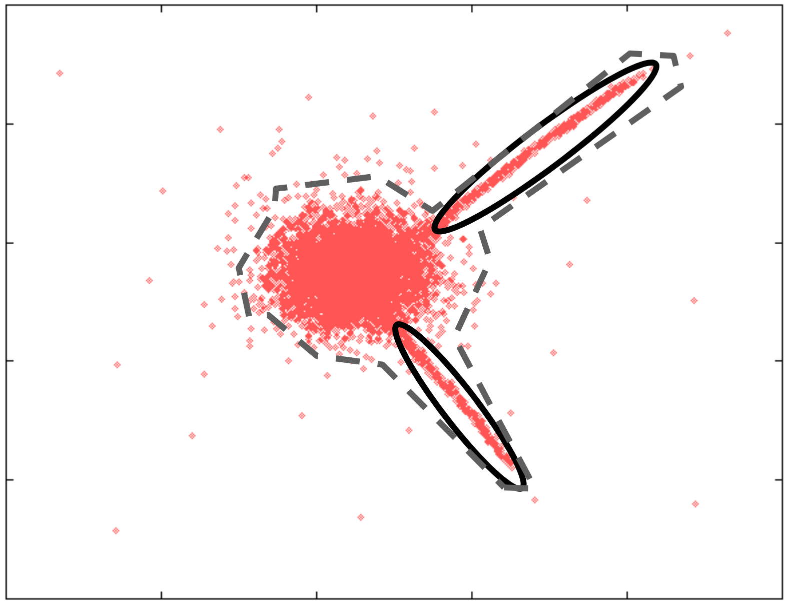
```
algorithm 4C($\mathcal{D}$, $\varepsilon$, $\mu$, $\lambda$, $\delta$)

  // assumption: each object in $\mathcal{D}$ is marked as unclassified

  for each unclassified $O \in \mathcal{D}$ do
STEP 1. test $\mathrm{Core}^{\mathrm{cor}}_{\mathrm{den}}(O)$ predicate:
      compute $\mathcal{N}_\varepsilon(O)$;
      if $|\mathcal{N}_\varepsilon(O)| \geq \mu$ then
        compute $\mathbf{M}_O$;
        if $\mathrm{CorDim}(\mathcal{N}_\varepsilon(O)) \leq \lambda$ then
          compute $\hat{\mathbf{M}}_O$ and $\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_O}(O)$;
          test $|\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_O}(O)| \geq \mu$;

STEP 2.1. if $\mathrm{Core}^{\mathrm{cor}}_{\mathrm{den}}(O)$ expand a new cluster:
      generate new clusterID;
      insert all $X \in \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_O}(O)$ into queue $\Phi$;
      while $\Phi \neq \emptyset$ do
        $Q$ = first object in $\Phi$;
        compute $\mathcal{R} = \{X \in \mathcal{D} \mid \mathrm{DirReach}^{\mathrm{cor}}_{\mathrm{den}}(Q, X)\}$;
        for each $X \in \mathcal{R}$ do
          if $X$ is unclassified or noise then
            assign current clusterID to $X$
          if $X$ is unclassified then
            insert $X$ into $\Phi$;
        remove $Q$ from $\Phi$;

STEP 2.2. if not $\mathrm{Core}^{\mathrm{cor}}_{\mathrm{den}}(O)$ $O$ is noise:
      mark $O$ as noise;

end.
```

$\mu = 3$

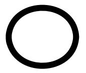


4C

4C vs. DBSCAN

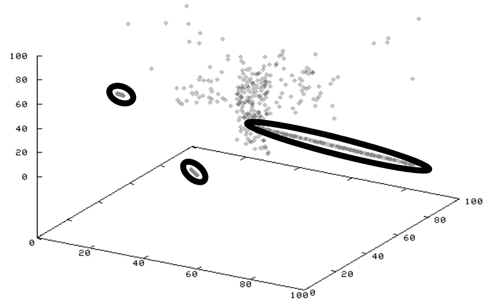


Cluster found by DBSCAN

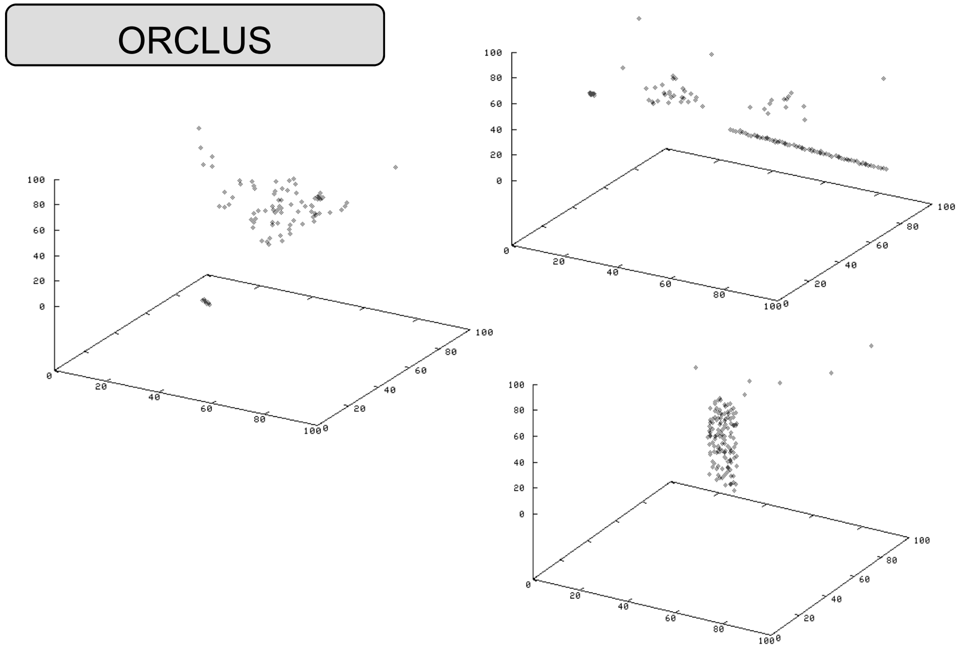
 Clusters found by 4C

4C vs. ORCLUS

4C



ORCLUS



4C: discussion

- finds arbitrary number of clusters
- requires specification of density-thresholds
  - μ (minimum number of points): rather intuitive
  - ε (radius of neighborhood): hard to guess
- biased to maximal dimensionality λ of correlation clusters (user specified)
- instance-based locality assumption: correlation distance measure specifying the subspace is learned from local neighborhood of each point in the *d*-dimensional space

enhancements also based on PCA:

- COPAC [ABK+07c] and
- ERiC [ABK+07b]

Correlation clustering: Discussion

- PCA: mature technique, allows construction of a broad range of similarity measures for local correlation of attributes
- drawback: all approaches suffer from locality assumption
- successfully employing PCA in correlation clustering in “really” high-dimensional data requires more effort henceforth

Clustering High Dimensional Data: Discussion 1/2

- Finding clusters in (arbitrarily oriented) subspaces of the original feature space.
- The subspace (where the cluster exists) is part of the cluster definition.
- The challenge is 2-fold: finding the correct subspace for each cluster and the correct cluster in each relevant subspace
  - Integrate subspace search in the clustering process
- Traditional full dimensional clustering paradigms transferred in the high dimensional space.

Clustering High Dimensional Data: Discussion 2/2

- Different types of methods
  - Bottom-Up approaches: Subspace Clustering
    - Find clusters in all subspaces
    - Restrict the search space by downward closure property
    - Axis-parallel subspaces
    - CLIQUE [AGGR98], SUBCLU [KKK04]
  - Top-Down Approaches: Projected Clustering
    - Each point is assigned to one subspace cluster or noise.
    - Subspaces are discovered based on the locality (cluster-based, instance-based)
    - Axis-parallel subspaces
    - PROCLUS [APW+99], PREDECON[BKKK04]
  - Top-Down Approaches: Correlation Clustering
    - Each point is assigned to one subspace cluster or noise.
    - Subspace are discovered based on the locality (cluster-based, instance-based)
    - Arbitrary oriented subspaces
    - ORCLUS[AY00], 4C [BKKZ04]
  - Pattern based clustering

Literature

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan.
Automatic subspace clustering of high dimensional data for data mining applications.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA, 1998.

[KKK04 K. Kailing, H.-P. Kriegel, and P. Kröger.
Density-connected subspace clustering for highdimensional data.
In Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL, 2004.

[BKKK04] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger.
Density connected clustering with local subspace preferences.
In Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K., 2004.

[APW+99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park.
Fast algorithms for projected clustering.
In Proceedings of the ACM International Conference on Management of Data
(SIGMOD), Philadelphia, PA, 1999.

[AY00] C. C. Aggarwal and P. S. Yu.
Finding generalized projected clusters in high dimensional space.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX, 2000.

[BKKZ04] C. Böhm, K. Kailing, P. Kröger, and A. Zimek.
Computing clusters of correlation connected objects.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France, 2004.

DATABASE
SYSTEMS
GROUP

Not covered material

Correlation clustering - Use Hough Transformation to determine Clusters

- Hough-Transformation

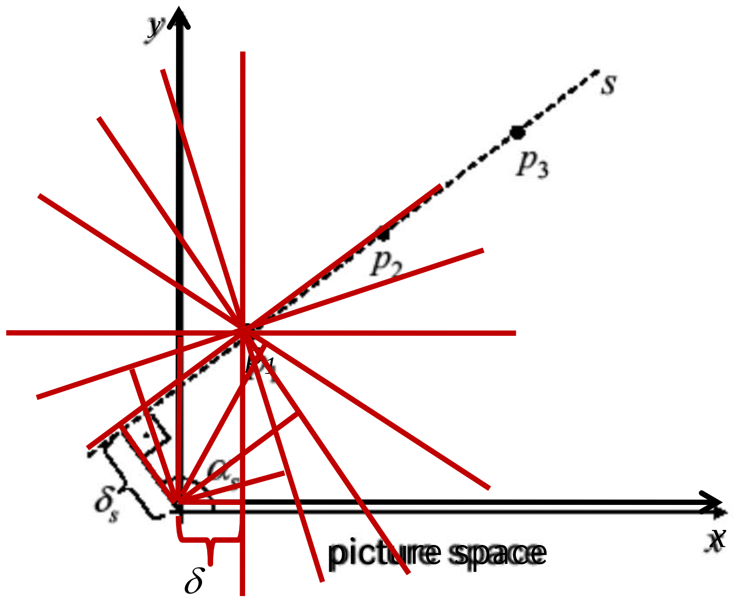
  Known from image analysis (finds geometric primitives lines, circles..)

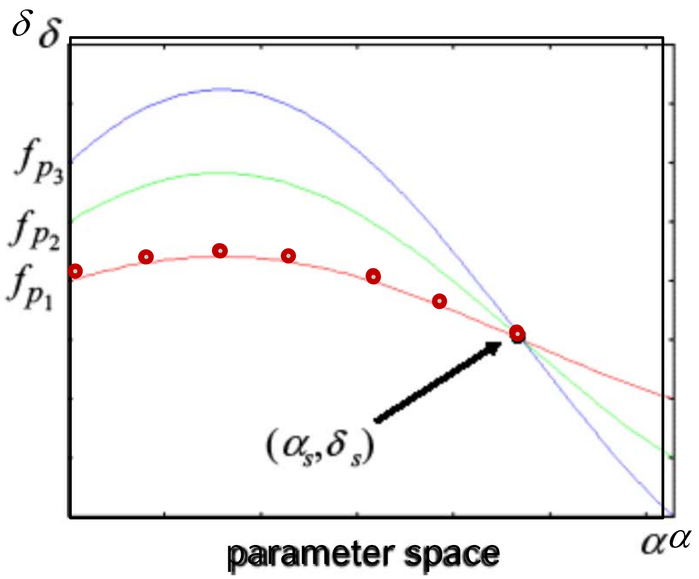
  in 2D pixel images
- Extension to arbitrary dimensions
- Transfers clustering into a new space
  (“parameter space” of the Hough transform)
- reduces the search space from not countable infinity to $O(n!)$
- Common search heuristic is full enumeration

=> *For efficient clustering a better heuristic is necessary!!*

Hough-Transformation

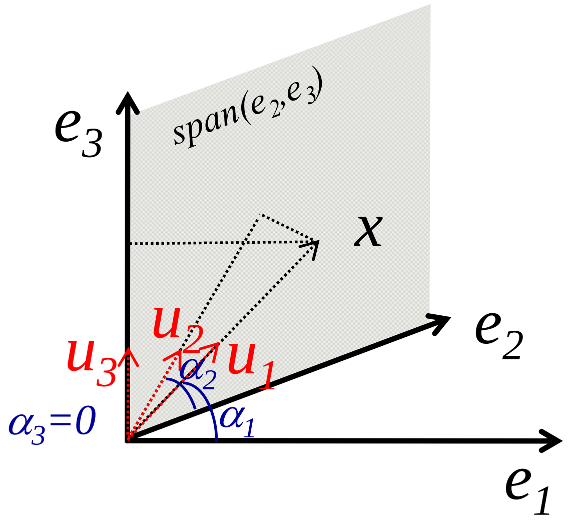
- Given: $D \subseteq IR^d$
- target: linear subspaces, containing many points x $x \in D$
- Idea: Maps points from the data space (picture space) to functions in the parameters space





d-dimensional Polar Coordinates

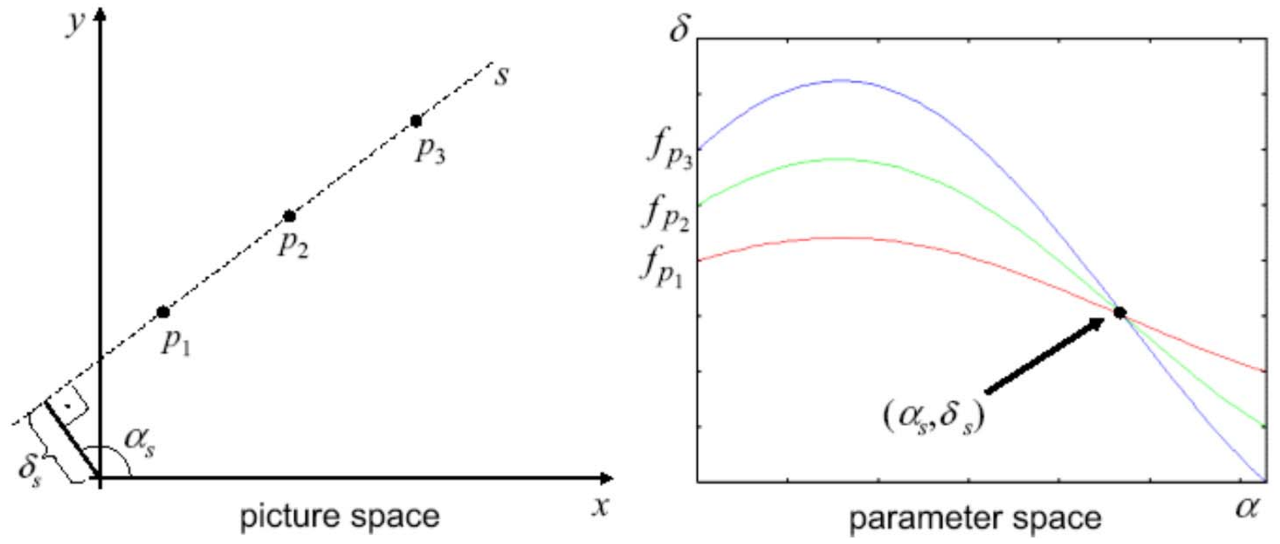
- e_i, $1 \leq i \leq d$: Orthonormal basis
- $x = (x_1,\ldots,x_d)^T$: d-dimensional Vector on the hyper sphere around the origin with radius r
- u_i: unity vector in the direction of the projection of x to the subspace $span(e_i,\ldots,e_d)$
- $\alpha_1,\ldots,\alpha_{d-1}$: α_i angle between u_i and e_i



$$x_i = r \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i)$$

Correlation Clustering Algorithms

- points in data space are mapped to functions in the parameter space



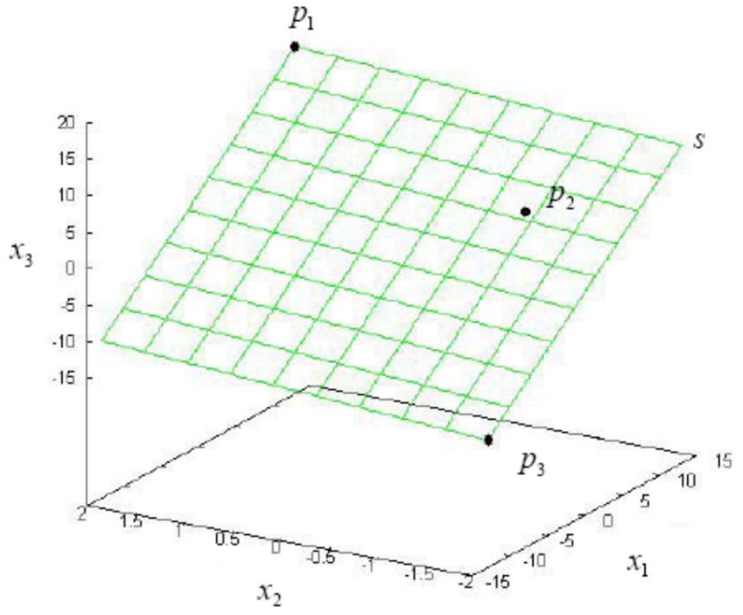
$$f_p(\alpha_1, \ldots, \alpha_{d-1}) = \langle p, n \rangle = \sum_{i=1}^{d} p_i \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i)$$

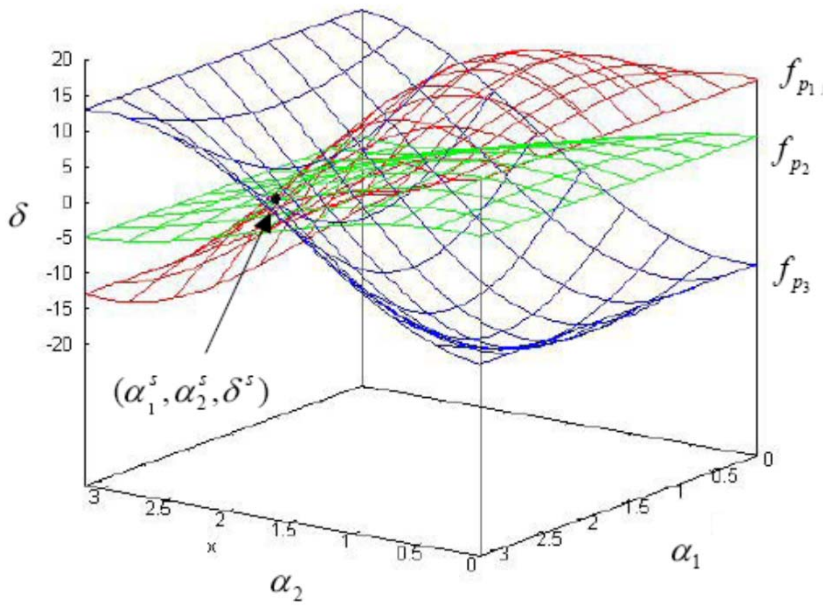
- functions in the parameter space define all lines possibly crossing the point in the data space

DATABASE SYSTEMS GROUP

Properties of the transformation

- Point in the data space = sinusoidal curve in parameter space
- Point in parameter space = hyper-plane in data space
- Points on a common hyper-plane in data space = sinusoidal curves through a common point in parameter space
- Intersections of sinusoidal curves in parameter space = hyper-plane through the corresponding points in data space



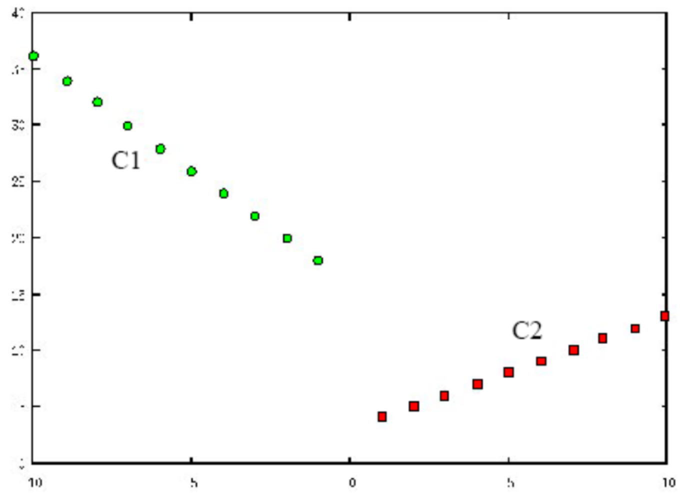


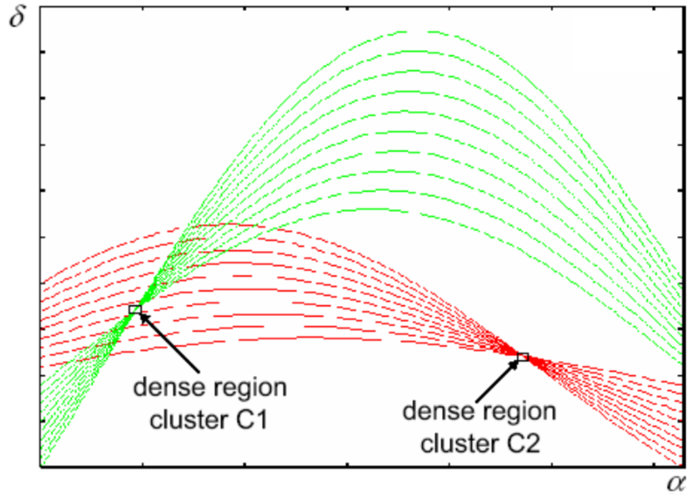
Correlation Clustering based on the Hough-Transformation

- Dense region in parameter space $\Leftrightarrow$ lineare regions in the data space

  (hyper planes wherer $\lambda \leq d\text{-}1$)
- Exact solutions: Determine all Intersections
  - Computation too expensive
  - Too exact to find linear clusters
- approximative solution: gridbased clustering in parameter spaces

  $\rightarrow$ determine grid cells intersecting at least m sinusoids
  - Search space is finite but in $O(r^d)$
  - Cluster quality depends on the resolutio r (Auflösung des Grids)





Correlation Clustering Algorithms

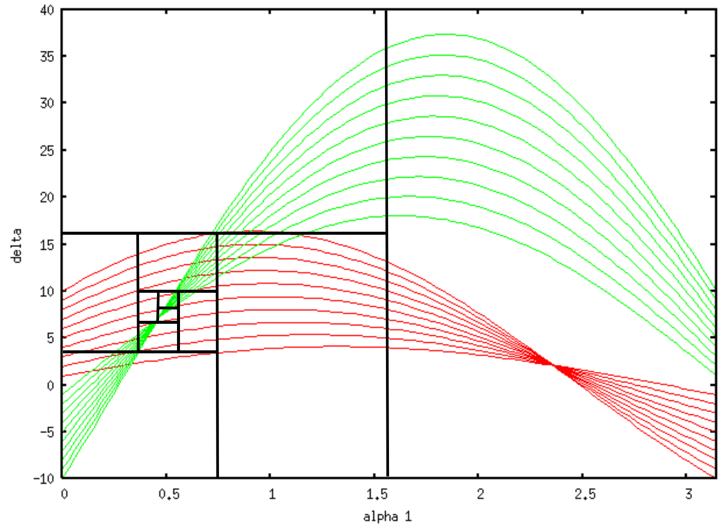
Idea: find dense regions in parameter space

- construct a grid by recursively splitting the parameter space (best-first-search)
- identify dense grid cells as intersected by many parametrization functions
- dense grid represents (*d*-*1*)-dimensional linear structure
- transform corresponding data objects in corresponding (*d*-*1*)-dimensional space and repeat the search recursively

Algorithmus CASH: effiziente Suchheuristik

CASH: Clustering in Arbitrary Subspaces based on the Hough-Transform []

- Parameter space is recursively partitioned per axis in a predefined order $[\alpha_1, \dots, \alpha_{d-1}, \delta]$
- Select the hyper rectangle representing the most points to continue (Best-First Search)



Algorithm CASH: efficient search heuristics

- Hyper rectangle representing less than m points can be pruned from the search space $\rightarrow$ early determination of the search path
- Hyper rectangles intersecting at least m sinusoids after s recursive partitionings represent correlation clusters (where $\lambda \leq d\text{-}1$)
  - Cluster points (i.e. sinusoids) are removed from any other hyper rectangle
  - To detect correlation clusters in subspaces with $\lambda \leq d\text{-}2$:
    recursive processing of the cluster after transformation into the corresponding $d\text{-}1$-dimensional subspace

Algorithmus CASH: Characateristics

- Detects an arbitrary amount of cluster
- Required input:
  - search depth (number of splits ⇔ maximal size of a cluster cell/accuracy)
  - minimal density of a cell (⇔ minimal number of point in a cluster)
- Density of a cell is not based on the "locality assumption"

  => method for global correlation clustering
- In average the search heuristic scales with $\sim d^3$
- BUT: worst case runtime degenerates to exhaustive search (exponential growth in d)

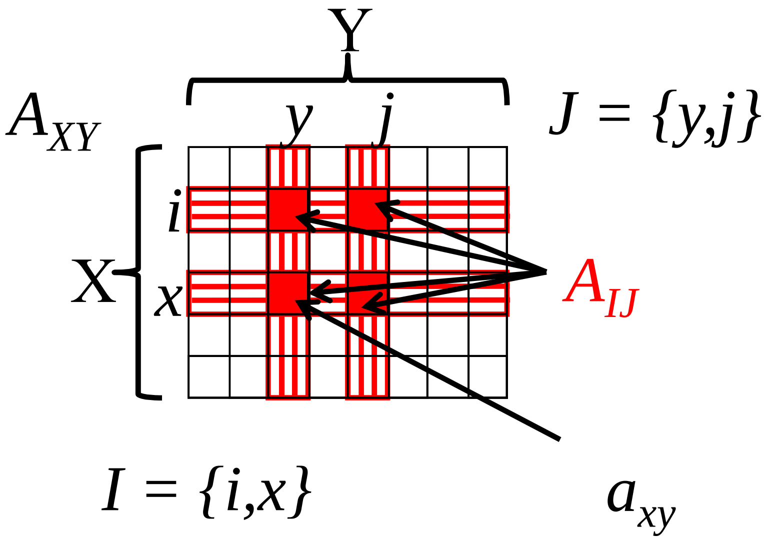
Correlation Clustering Algorithms

properties:

- finds arbitrary number of clusters
- requires specification of depth of search (number of splits per axis)
- requires minimum density threshold for a grid cell
- Note: this minimum density does not relate to the locality assumption: CASH is a global approach to correlation clustering
- search heuristic: linear in number of points, but ~ d^4
- But: complete enumeration in worst case (exponential in d)

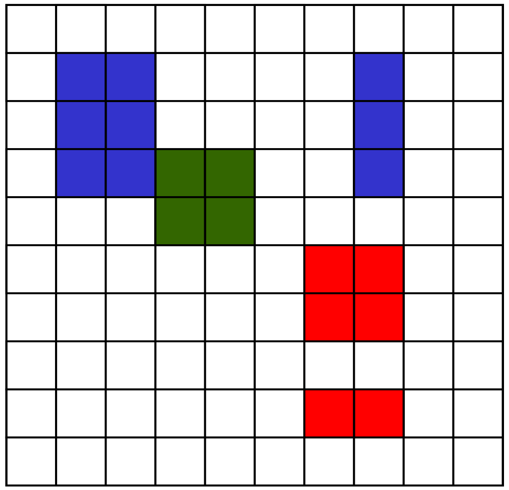
Pattern-based clustering

- Pattern-based clustering algorithms depict the data as a matrix
  - ***A*** = (X,Y) with set of rows X and set of columns Y
  - a_{xy} is the element in row *x* and column *y*.
  - submatrix A_{IJ} = (I,J) with subset of rows I $\subseteq$ X and subset of columns J $\subseteq$ Y contains those elements a_{ij} with $i \in$ I und $j \in$ J



Pattern-based clustering: Problem definition

Find a set of submatrices $\{(I_1,J_1),(I_2,J_2),...,(I_k,J_k)\}$ of the matrix $\boldsymbol{A}=(X,Y)$ (with $I_i \subseteq X$ and $J_i \subseteq Y$ for $i = 1,...,k$) where each submatrix (= bicluster) meets a given homogeneity criterion.



Pattern-based clustering: criteria

- Some values often used by bicluster models:
  - mean of row *i*:

$$a_{iJ} = \frac{1}{|J|}\sum_{j \in J} a_{ij}$$

  - mean of column *j*:

$$a_{Ij} = \frac{1}{|I|}\sum_{i \in I} a_{ij}$$

  - mean of all elements:

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

Pattern-based clustering: Biclusters types

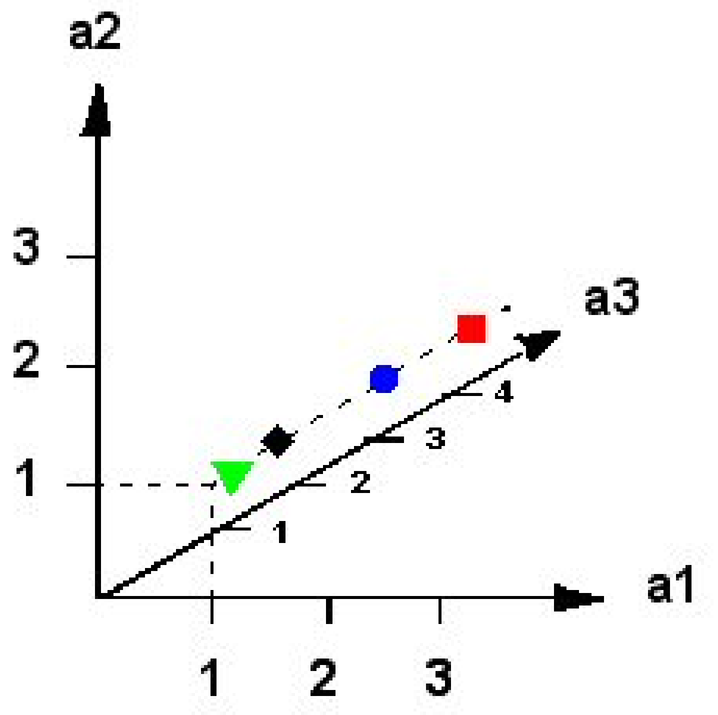
Different types of biclusters (cf. [MO04]):

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

Constant biclusters

- All points share identical value in selected attributes.
- The constant value μ is a typical value for the cluster.
- Cluster model: $a_{ij} = \mu$
- Obviously a special case of an axis-parallel subspace cluster.
- Example: embedding 3-dimensional space

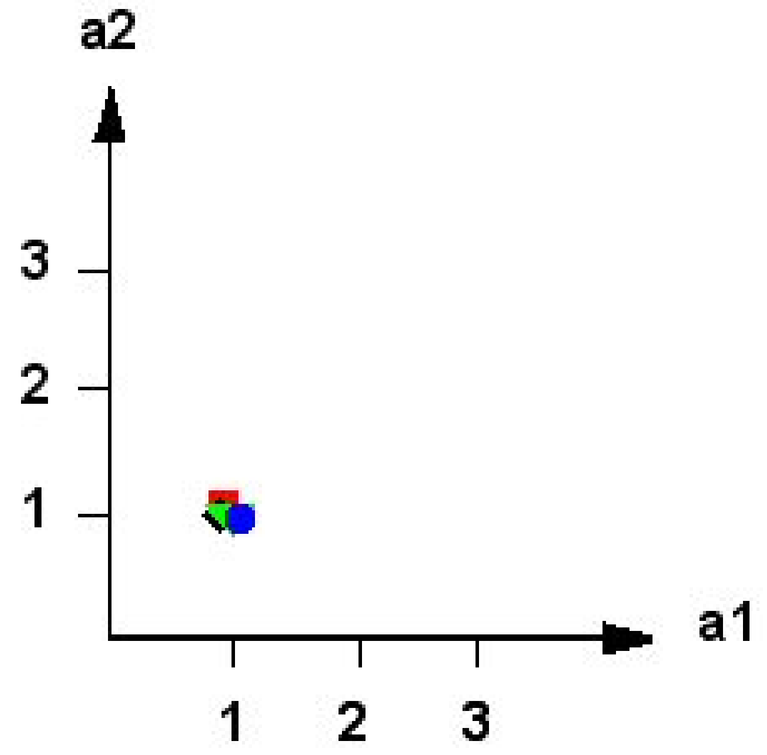
	a1	a2	a3
P1	1	1	3.5
P2	1	1	2.3
P3	1	1	0.2
P4	1	1	0.7



Constant biclusters: Examples

- Example: 2-dimensional subspace:

	a1	a2
P1	1	1
P2	1	1
P3	1	1
P4	1	1



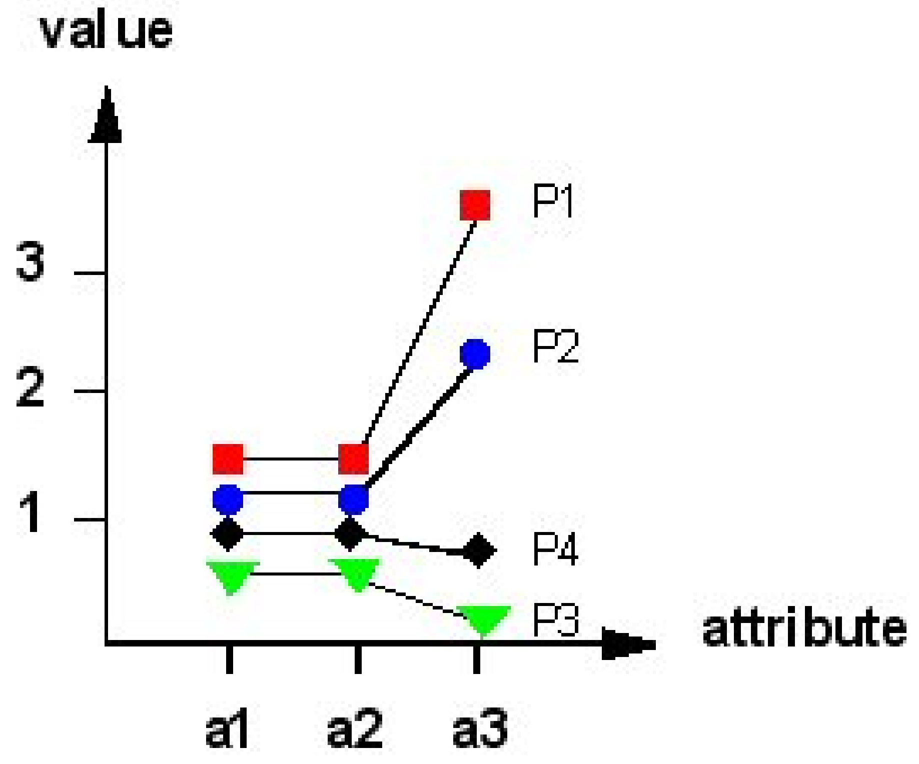
- points located on the bisecting line of participating attributes

DATABASE SYSTEMS GROUP

Constant biclusters: Examples

- Example: transposed view of attributes:

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



- pattern: identical constant lines

Constant biclusters: Discussion

- Real-world constant biclusters will not be perfect
- cluster model relaxes to: $a_{ij} \approx \mu$

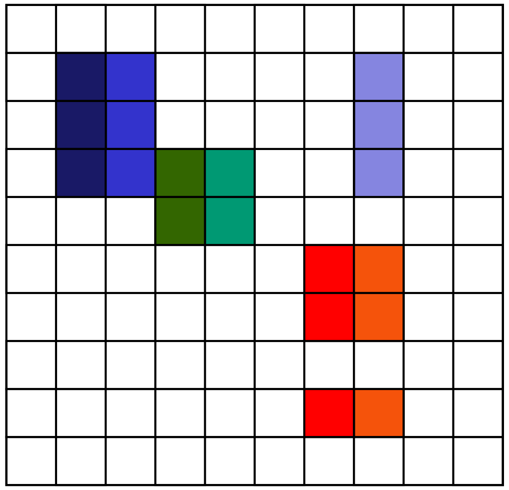
- Optimization on matrix *A* = (X,Y) may lead to |X|·|Y| singularity-biclusters each containing one entry.
- Challenge: Avoid this kind of overfitting.

Biclusters with constant values on columns

- Cluster model for A_{IJ} = (I,J):

$$a_{ij} = \mu + c_j$$
$$\forall i \in I, j \in J$$

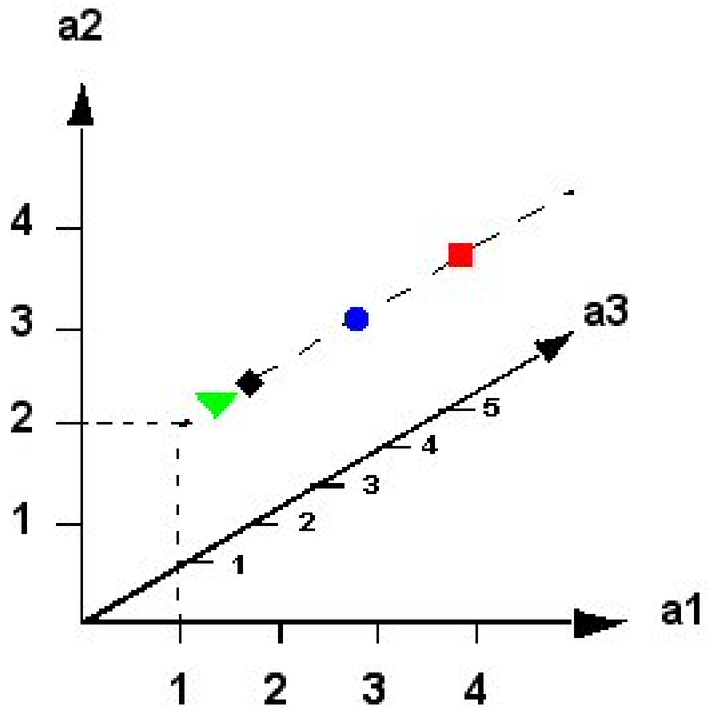
- adjustment value c_j for column $j \in$ J
- results in axis-parallel subspace clusters



Biclusters with constant values on columns: Examples

- Example: 3-dimensional embedding space

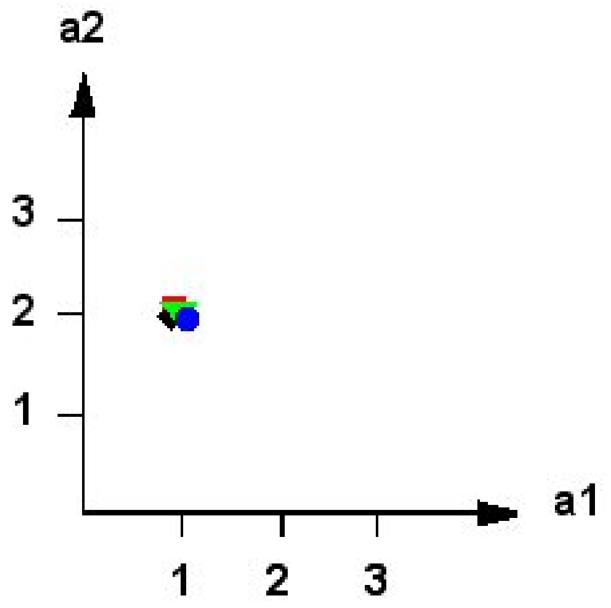
	a1	a2	a3
P1	1	2	3.5
P2	1	2	2.3
P3	1	2	0.2
P4	1	2	0.7



Biclusters with constant values on columns: Examples

- Example: 2-dimensional subspace:

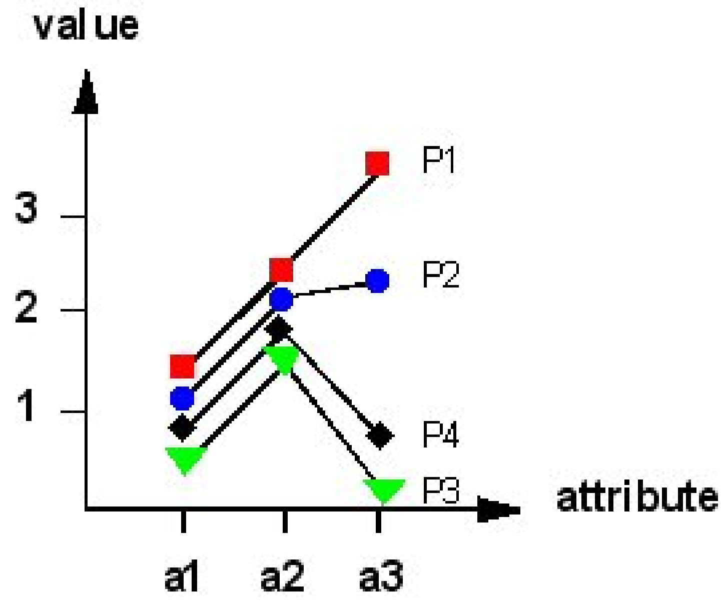
	a1	a2
P1	1	2
P2	1	2
P3	1	2
P4	1	2



Biclusters with constant values on columns: Examples

- Example: transposed view of attributes:

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



- pattern: identical lines

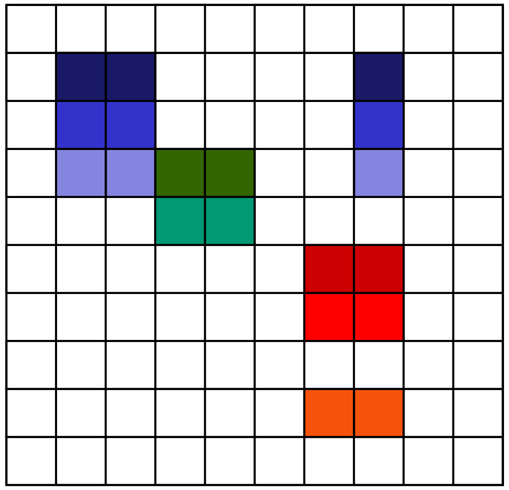
Biclusters with constant values on rows

- Cluster model for A_{IJ} = (I,J):

$$a_{ij} = \mu + r_i$$

$$\forall i \in I, j \in J$$

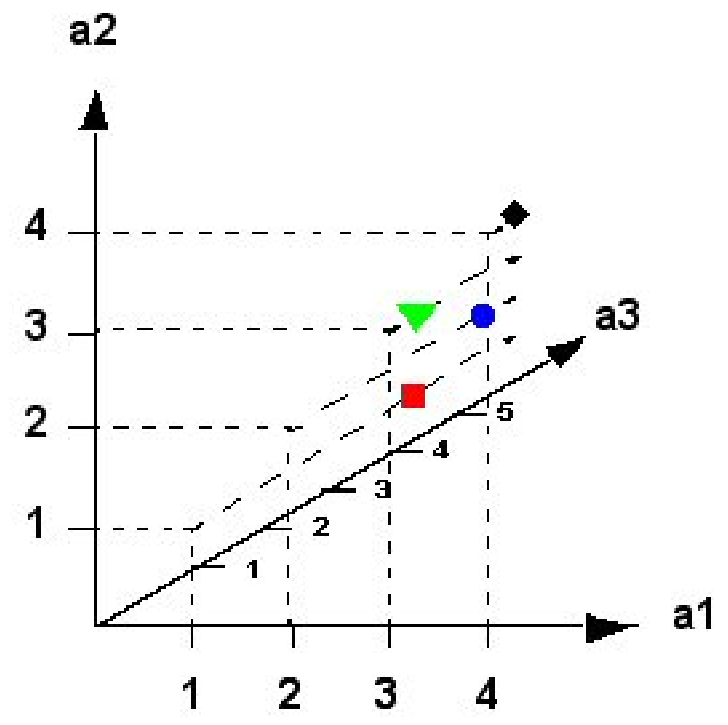
- adjustment value r_i for row $i \in$ I



Biclusters with constant values on rows: Examples

- Example: 3-dimensional embedding space:

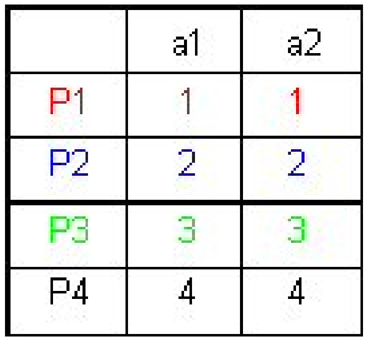
	a1	a2	a3
P1	1	1	3.5
P2	2	2	2.3
P3	3	3	0.2
P4	4	4	0.7



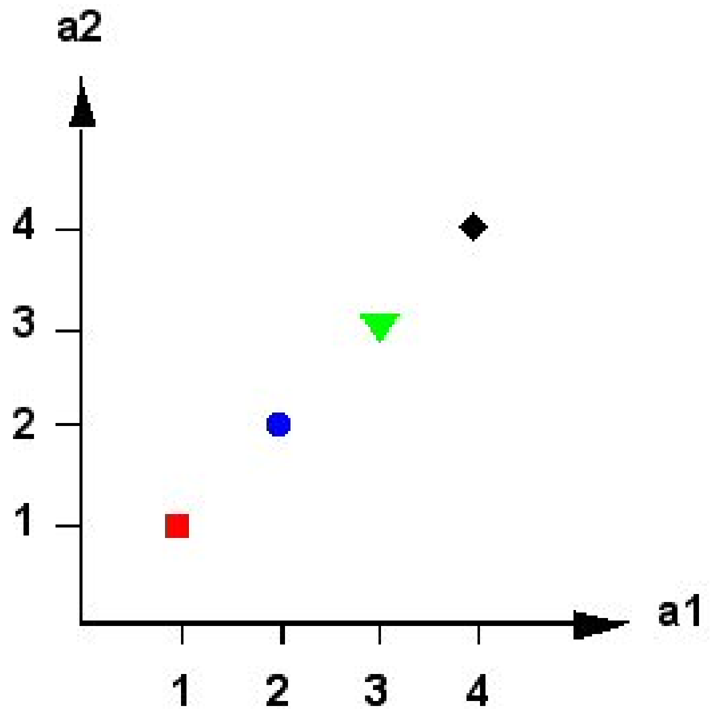
- in the embedding space, points build a sparse hyperplane parallel to irrelevant axes

Biclusters with constant values on rows: Examples

- example – 2-dimensional subspace:



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

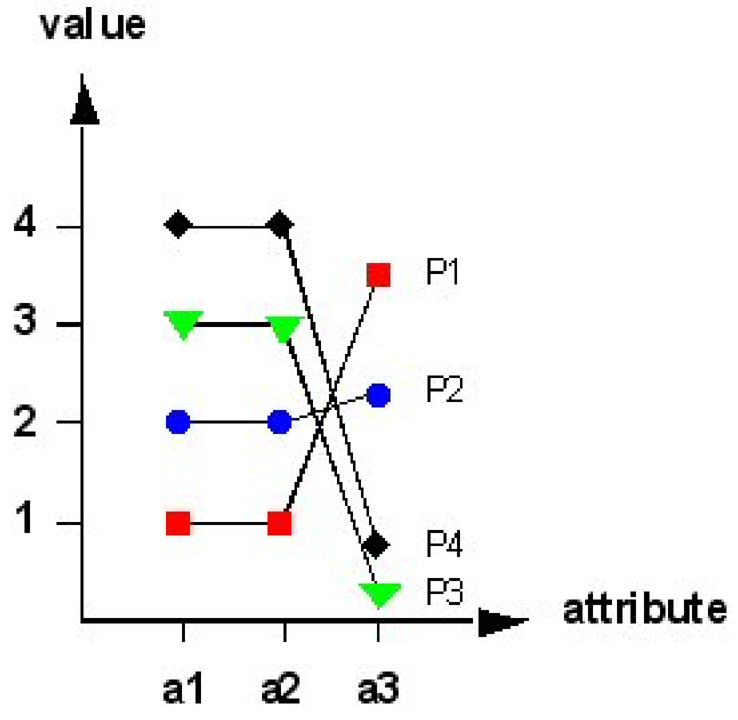


- points are accommodated on the bisecting line of participating attributes

Biclusters with constant values on rows: Examples

- example – transposed view of attributes:

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

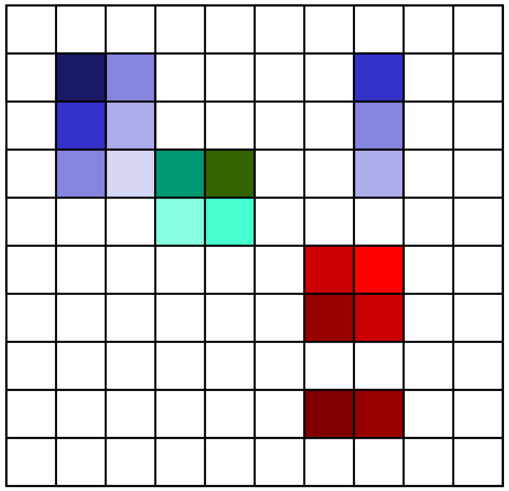


- pattern: parallel constant lines

Biclusters with coherent values

- based on a particular form of covariance between rows and columns

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

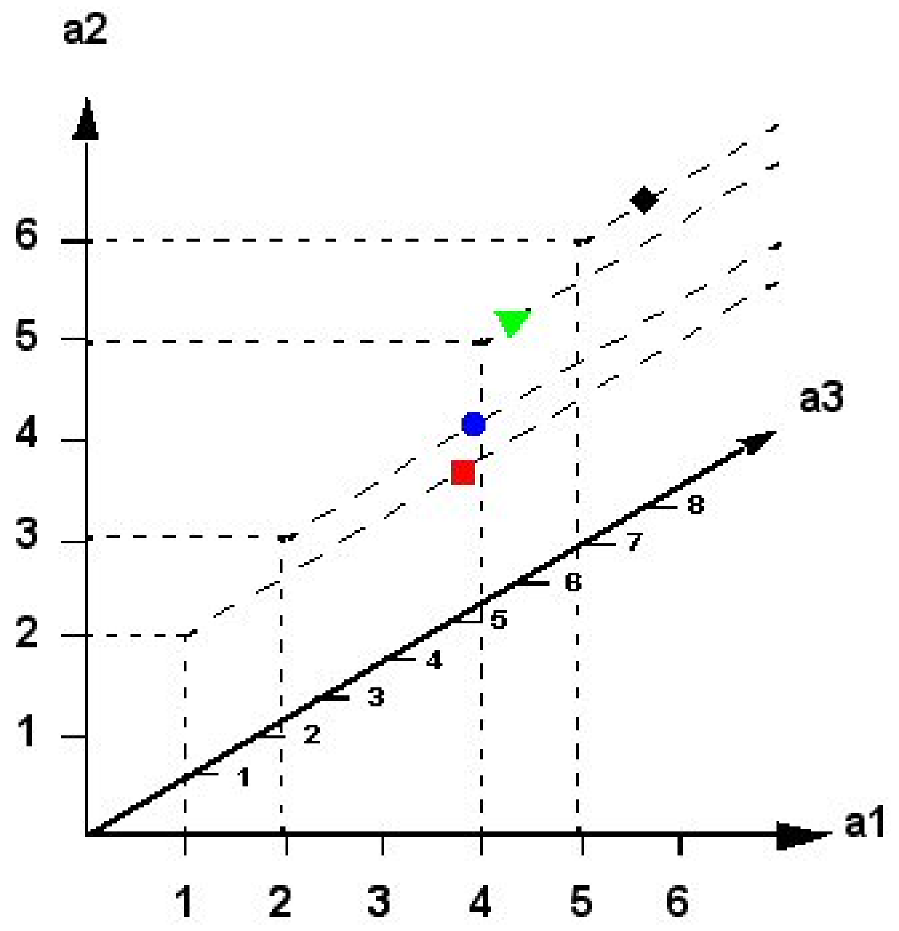


- special cases:
  - $c_j = 0$ for all j → constant values on rows
  - $r_i = 0$ for all i → constant values on columns

Biclusters with coherent values: Examples

- embedding space: sparse hyperplane parallel to axes of irrelevant attributes

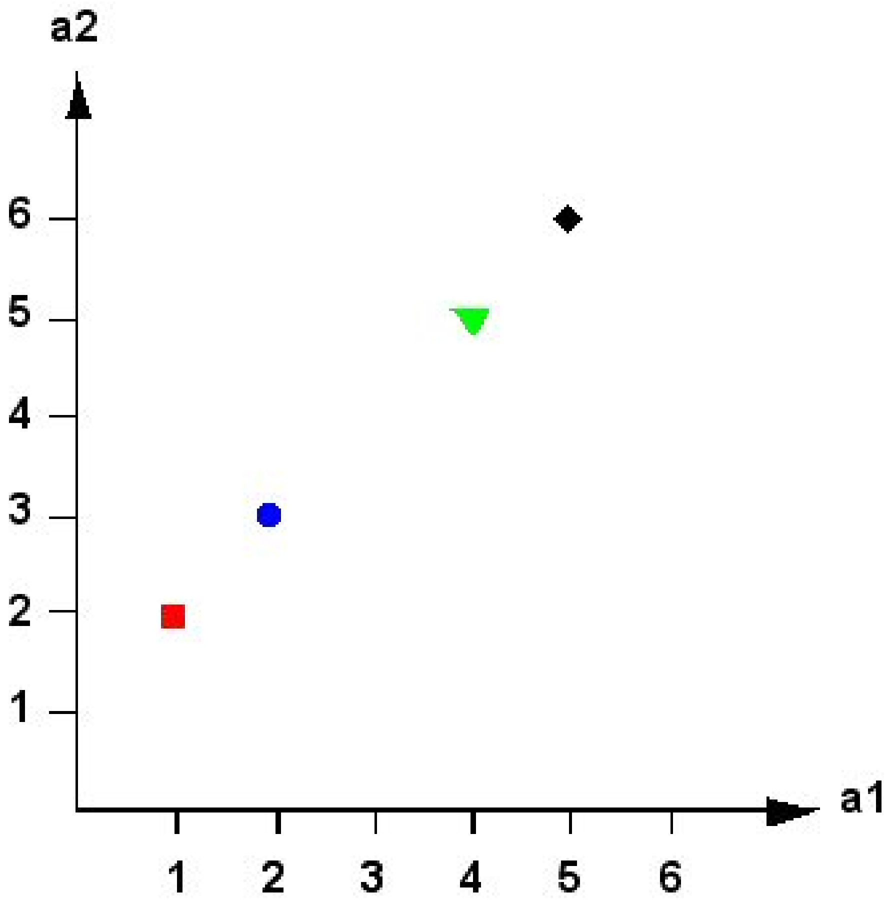
	a1	a2	a3
P1	1	2	3.5
P2	2	3	2.3
P3	4	5	0.2
P4	5	6	0.7



Biclusters with coherent values: Examples

- subspace: increasing one-dimensional line

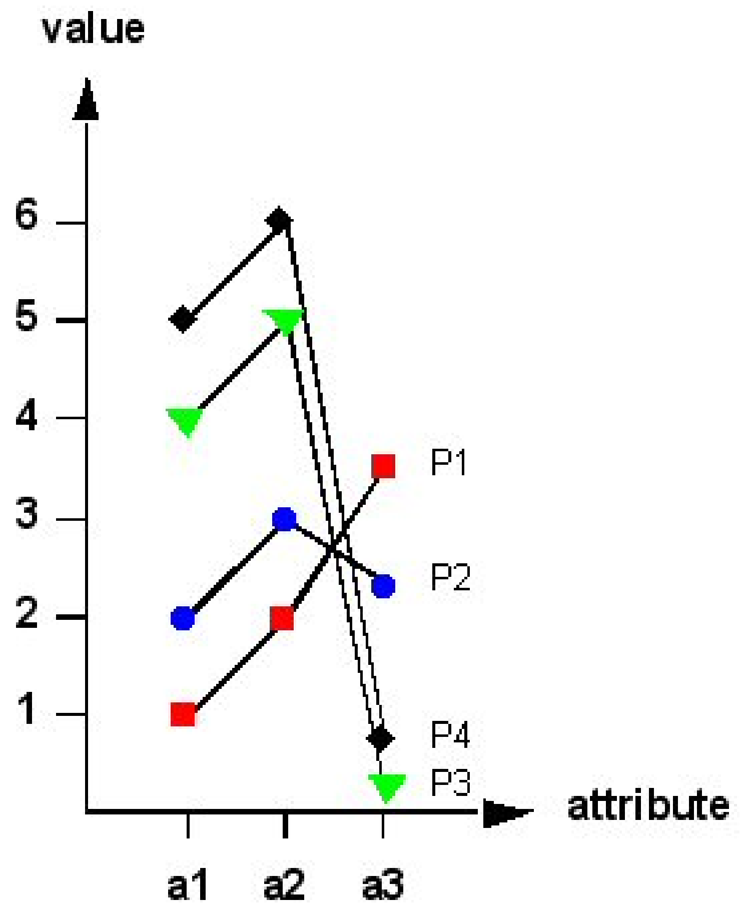
	a1	a2
P1	1	2
P2	2	3
P3	4	5
P4	5	6



Biclusters with coherent values: Examples

- transposed view of attributes:

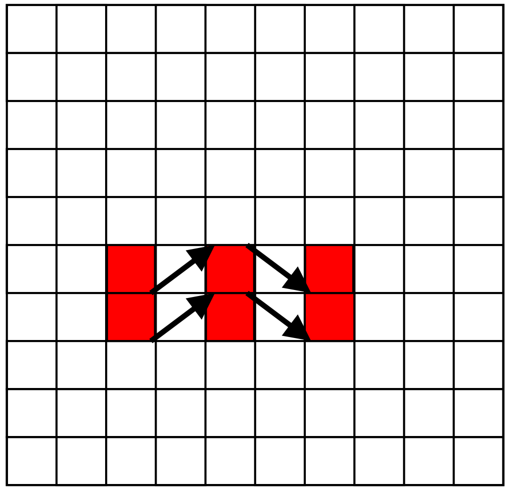
	a1	a2	a3
P1	1	2	3.5
P2	2	3	2.3
P3	4	5	0.2
P4	5	6	0.7



- pattern: parallel lines

Biclusters with coherent evolutions

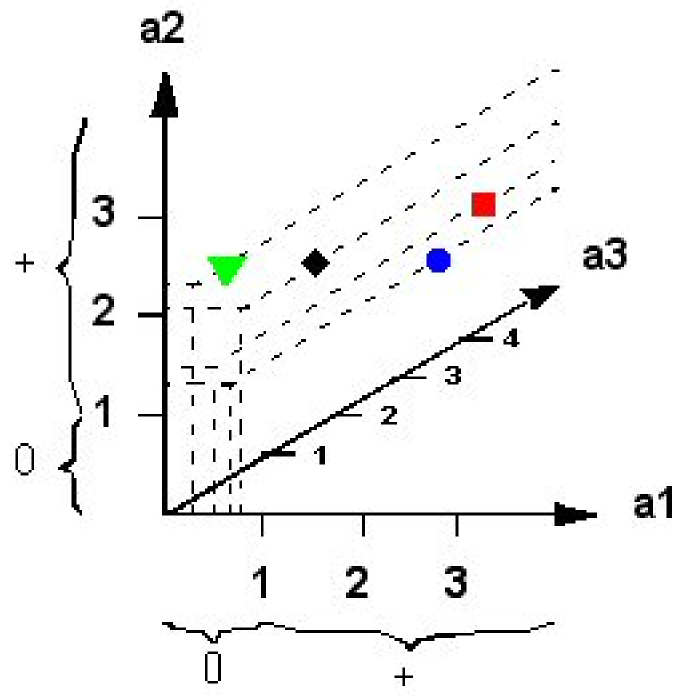
- For all rows, all pairs of attributes change simultaneously
  - discretized attribute space: coherent state-transitions
  - change in same direction irrespective of the quantity



Challenges and Approaches, Basic Models

- Approaches with coherent state-transitions: [TSS02,MK03]
- reduces the problem to grid-based axis-parallel approach:

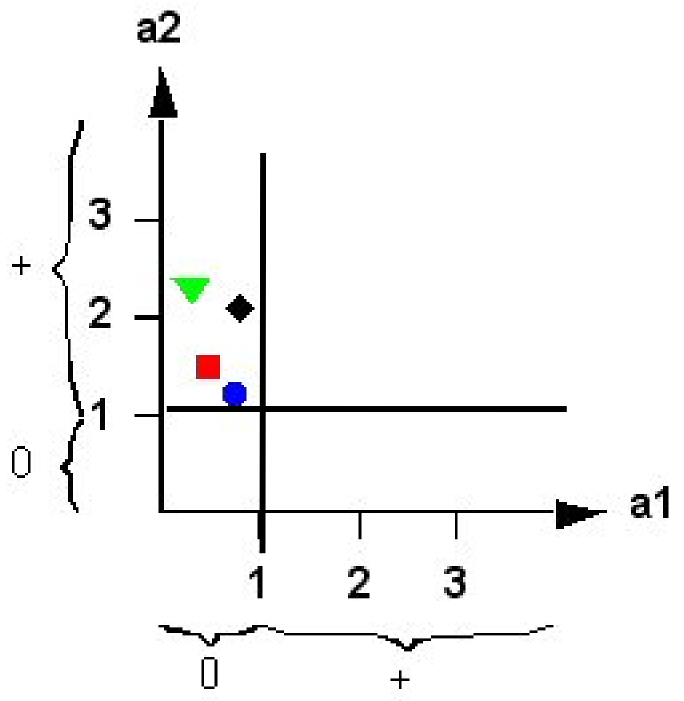
	a1	a2	a3
P1	0.5	1.5	3.5
P2	0.7	1.3	2.3
P3	0.3	2.3	0.2
P4	0.8	2.1	0.7

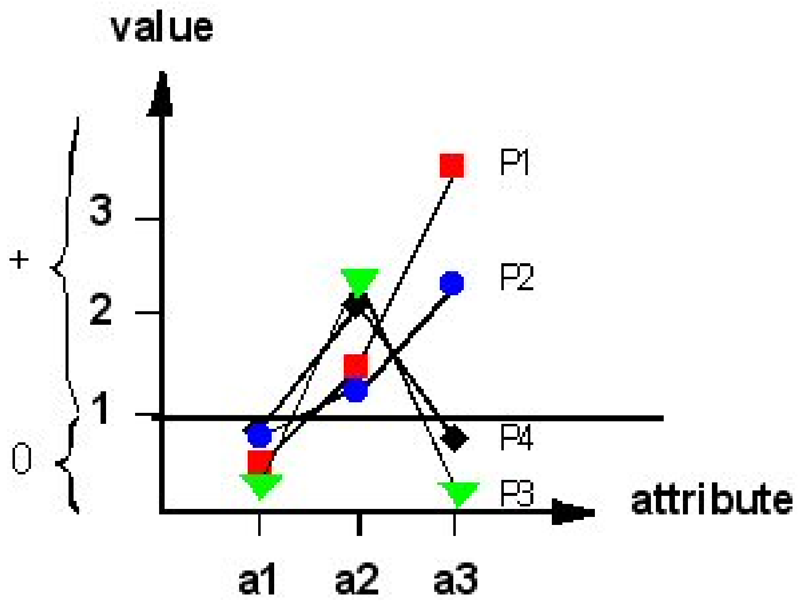


Challenges and Approaches, Basic Models

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

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





pattern: all lines cross border between states (in the same direction)

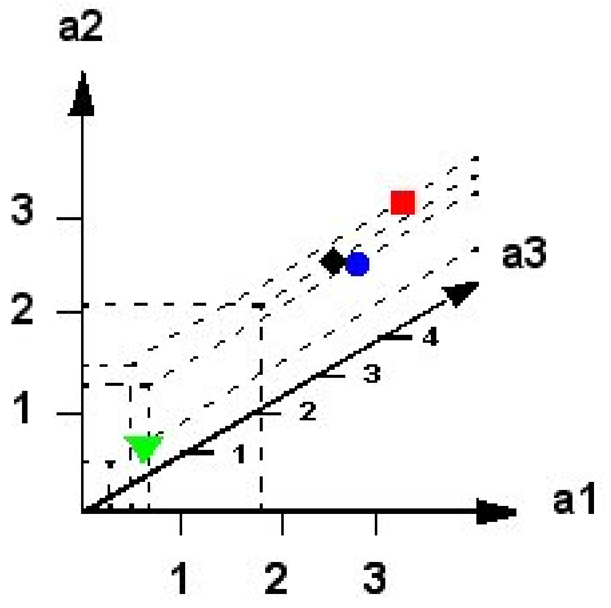
Challenges and Approaches, Basic Models

- change in same direction – general idea: find a subset of rows and columns, where a permutation of the set of columns exists such that the values in every row are increasing
- clusters do not form a subspace but rather half-spaces
- related approaches:
  - quantitative association rule mining [Web01,RRK04,GRRK05]
  - adaptation of formal concept analysis [GW99] to numeric data [Pfa07]

Challenges and Approaches, Basic Models

- example – 3-dimensional embedding space

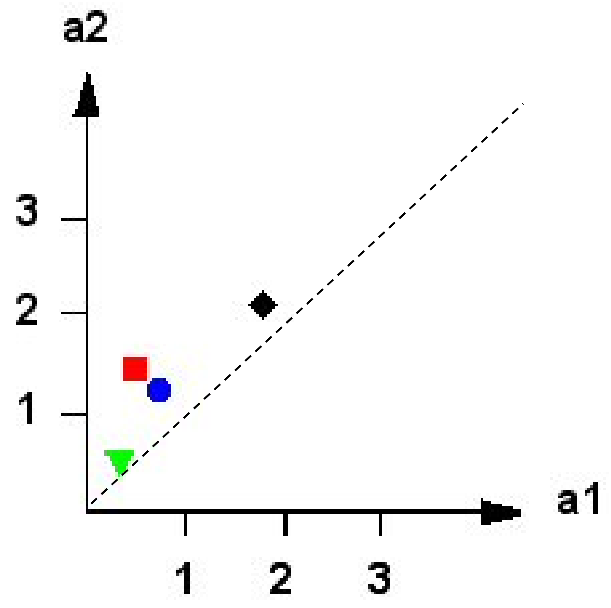
	a1	a2	a3
P1	0.5	1.5	3.5
P2	0.7	1.3	2.3
P3	0.3	0.5	0.2
P4	1.8	2.1	0.7



Challenges and Approaches, Basic Models

- example – 2-dimensional subspace

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

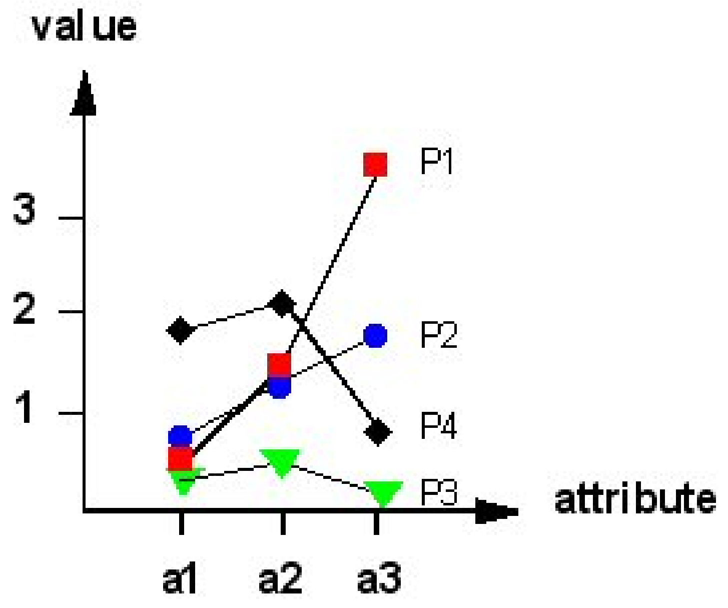


DATABASE SYSTEMS GROUP

Challenges and Approaches, Basic Models

- example – transposed view of attributes

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



- pattern: all lines increasing

Challenges and Approaches, Basic Models

more specialized ↑

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

↓ more general

↕ no order of generality

Algorithms for Constant Biclusters

- classical problem statement by Hartigan [Har72]
- quality measure for a bicluster: variance of the submatrix ***A***$_{IJ}$:

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

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

Biclusters with Constant Values in Rows or Columns

- simple approach: normalization to transform the biclusters into constant biclusters and follow the first approach (e.g. [GLD00])
- some application-driven approaches with special assumptions in the bioinformatics community (e.g. [CST00,SMD03,STG+01])
- constant values on columns: general axis-parallel subspace/projected clustering
- constant values on rows: special case of general correlation clustering
- both cases special case of approaches to biclusters with coherent values

Algorithms for Biclusters with Coherent Values

classical approach: Cheng&Church [CC00]

- introduced the term biclustering to analysis of gene expression data
- quality of a bicluster: *mean squared residue* value *H*

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

- submatrix (I,J) is considered a bicluster, if H(I,J) < δ

Algorithms for Biclusters with Coherent Values

- $\delta = 0 \rightarrow$ *perfect* bicluster:
  - each row and column exhibits absolutely consistent bias
  - bias of row *i* w.r.t. other rows: $a_{iJ} - a_{IJ}$

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

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ}$$

$$\Updownarrow \; \mu = a_{IJ}, r_i = a_{iJ} - a_{IJ}, c_j = a_{Ij} - a_{IJ}$$

$$a_{ij} = \mu + r_i + c_j$$

DATABASE SYSTEMS GROUP

Algorithms for Biclusters with Coherent Values

LMU

- for a non-perfect bicluster, the prediction of the model deviates from the true value by a residue:

$$a_{ij} = \text{res}(a_{ij}) + a_{iJ} + a_{Ij} - a_{IJ}$$

$$\Updownarrow$$

$$\text{res}(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}$$

- This residue is the optimization criterion:

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

Algorithms for Biclusters with Coherent Values

- The optimization is also possible for the row-residue of row *i* or the column-residue of column *j*.

- Algorithm:
  1. find a δ -bicluster: greedy search by removing the row or column (or the set of rows/columns) with maximal mean squared residue until the remaining submatrix (I,J) satisfies $H(I,J)< \delta$.
  2. find a maximal δ -bicluster by adding rows and columns to (I,J) unless this would increase *H*.
  3. replace the values of the found bicluster by random numbers and repeat the procedure until k δ -biclusters are found.

Algorithms for Biclusters with Coherent Values

Weak points in the approach of Cheng&Church:

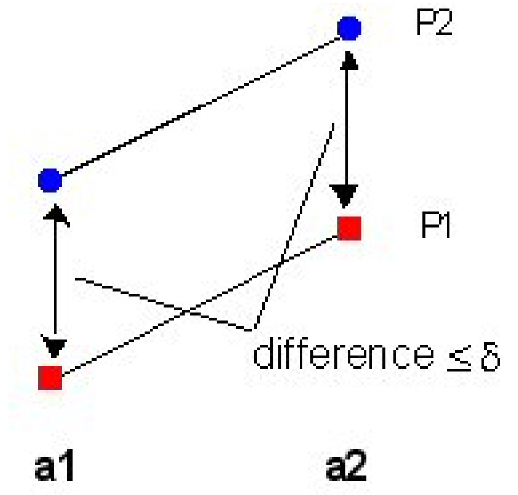
1. One cluster at a time is found, the cluster needs to be masked in order to find a second cluster.
2. This procedure bears an inefficient performance.
3. The masking may lead to less accurate results.
4. The masking inhibits simultaneous overlapping of rows and columns.
5. Missing values cannot be dealt with.
6. The user must specify the number of clusters beforehand.

Algorithms for Biclusters with Coherent Values

p-cluster model [WWYY02]

- p-cluster model: deterministic approach
- specializes δ -bicluster-property to a pairwise property of two objects in two attributes:

$$\left|\left(a_{i_1 j_1} - a_{i_1 j_2}\right) - \left(a_{i_2 j_1} - a_{i_2 j_2}\right)\right| \leq \delta$$

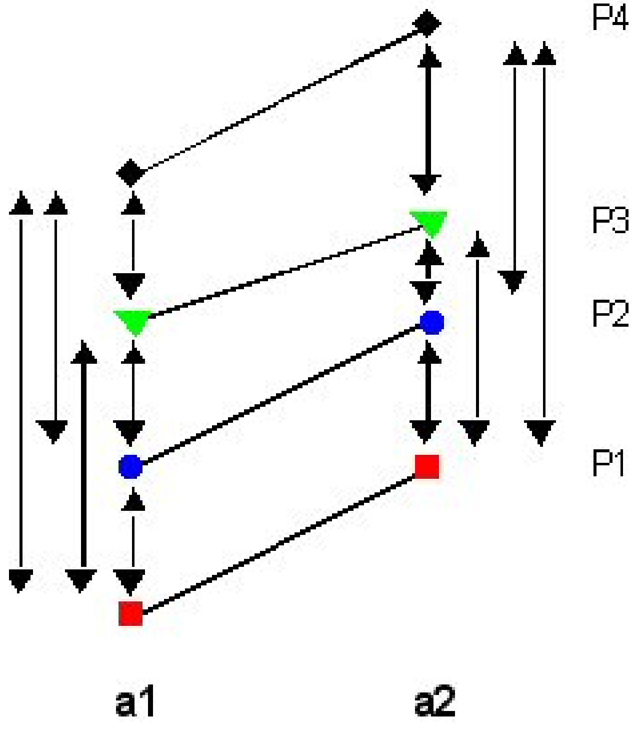


- submatrix (I,J) is a δ -p-cluster if this property is fulfilled for any 2x2 submatrix ($\{i_1, i_2\}$, $\{j_1, j_2\}$) where $\{i_1, i_2\} \in$ I and $\{j_1, j_2\} \in$J.

Algorithms for Biclusters with Coherent Values

Algorithm:

1. create maximal set of attributes for each pair of objects forming a δ -p-cluster
2. create maximal set of objects for each pair of attributes forming a δ -p-cluster
3. pruning-step
4. search in the set of submatrices



Problem: complete enumeration approach

Addressed issues:

1. multiple clusters simultaneously

4. allows for overlapping rows and columns

6. allows for arbitrary number of clusters

Related approaches:

FLOC [YWWY02],MaPle [PZC+03]

Summary

- Biclustering models do not fit exactly into the spatial intuition behind subspace, projected, or correlation clustering.
- Models make sense in view of a data matrix.
- Strong point: the models generally do not rely on the locality assumption.
- Models differ substantially → fair comparison is a non-trivial task.
- Comparison of five methods: [PBZ+06]
- Rather specialized task – comparison in a broad context (subspace/projected/correlation clustering) is desirable.
- Biclustering performs generally well on microarray data – for a wealth of approaches see [MO04].

Summary and Perspectives

comparison: correlation clustering – biclustering:

- model for correlation clusters more general and meaningful
- models for biclusters rather specialized
- in general, biclustering approaches do not rely on locality assumption
- non-local approach and specialization of models may make biclustering successful in many applications
- correlation clustering is the more general approach but the approaches proposed so far are rather a first draft to tackle the complex problem