LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

Detecting Clusters in Moderate-to-high Dimensional Data:

Subspace Clustering, Pattern-based Clustering, Correlation Clustering

Tutorial ICDM 2007

Hans-Peter Kriegel, Peer Kröger, Arthur Zimek

Ludwig-Maximilians-Universität München
Munich, Germany
http://www.dbs.ifi.lmu.de
{kriegel,kroegerp,zimek}@dbs.ifi.lmu.de

General Issues

1. Please feel free to ask questions at any time during the presentation
2. Aim of the tutorial: get the big picture
   - NOT in terms of a long list of methods and algorithms
   - BUT in terms of the basic algorithmic approaches
   - Sample algorithms for these basic approaches will be sketched
     - Due to the large number of existing approaches, the selection of the presented algorithms is somewhat arbitrary
     - Please don't mind if your favorite algorithm is missing
     - Anyway you should be able to classify any other algorithm not covered here by means of which of the basic approaches is implemented
3. The revised version of tutorial notes will soon be available on our websites

Outline

1. Introduction
2. Axis-parallel Subspace Clustering

Peer — Room Platte/Missouri

LUNCH BREAK

3. Pattern-based Clustering
4. Arbitrarily-oriented Subspace Clustering

Arthur

5. Summary

Peer

Room Big Blue

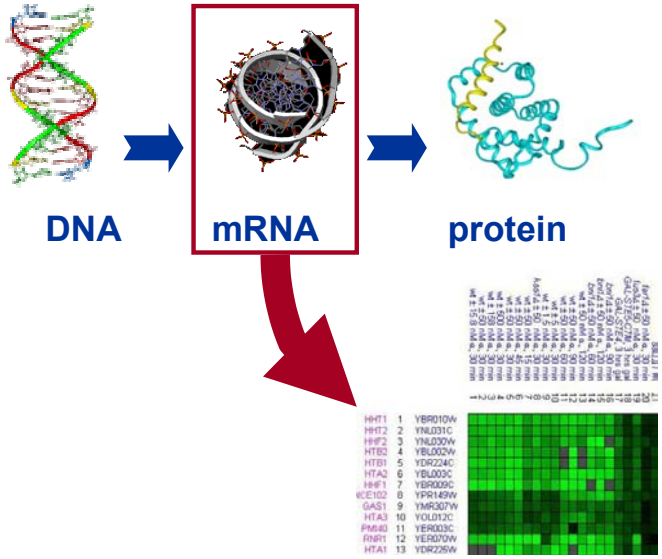
Outline

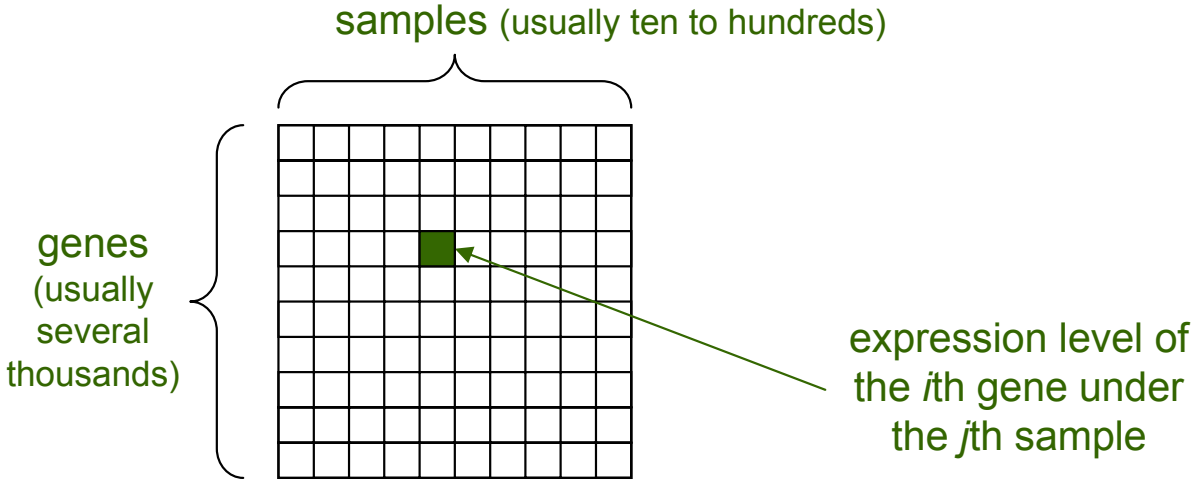
Outline: Introduction

- Sample Applications
- General Problems and Challenges
- A First Taxonomy of Approaches

Sample Applications

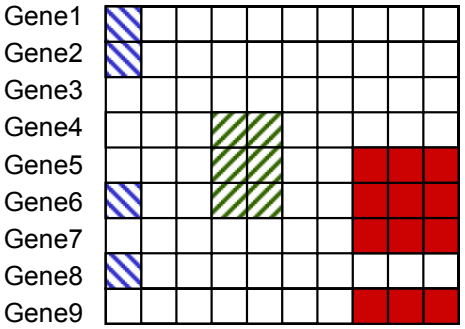
- Gene Expression Analysis
  - Data:
    - Expression level of genes under different samples such as
      - different individuals (patients)
      - different time slots after treatment
      - different tissues
      - different experimental environments
    - Data matrix:





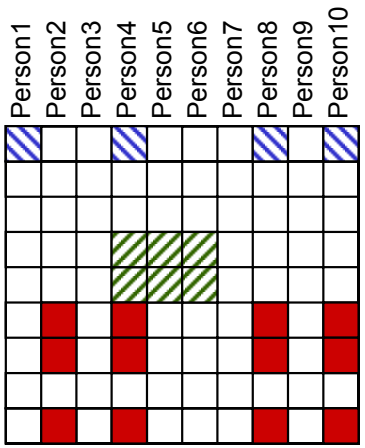
Sample Applications

- Task 1: Cluster the rows (i.e. genes) to find groups of genes with similar expression profiles indicating homogeneous functions
  - *Challenge*:
    genes usually have different functions under varying (combinations of) conditions



Cluster 1: {G1, G2, G6, G8}
Cluster 2: {G4, G5, G6}
Cluster 3: {G5, G6, G7, G9}

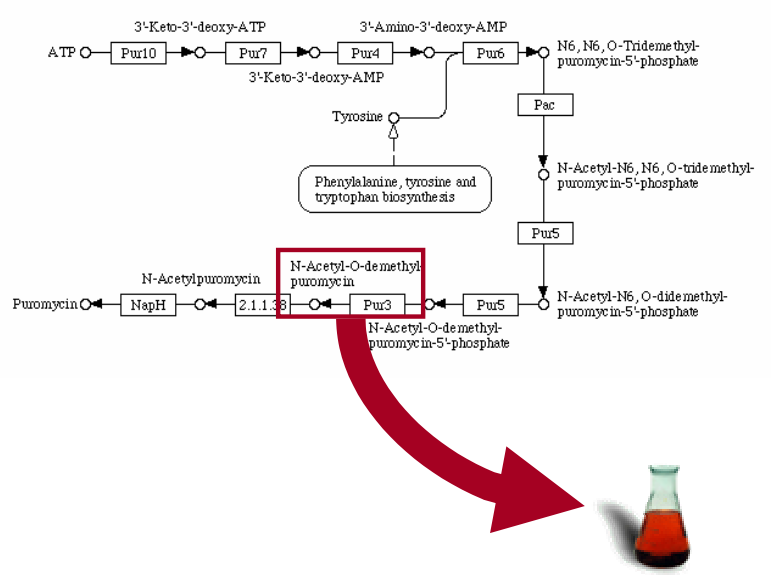
- Task 2: Cluster the columns (e.g. patients) to find groups with similar expression profiles indicating homogeneous phenotypes
  - *Challenge*:
    different phenotypes depend on different (combinations of) subsets of genes

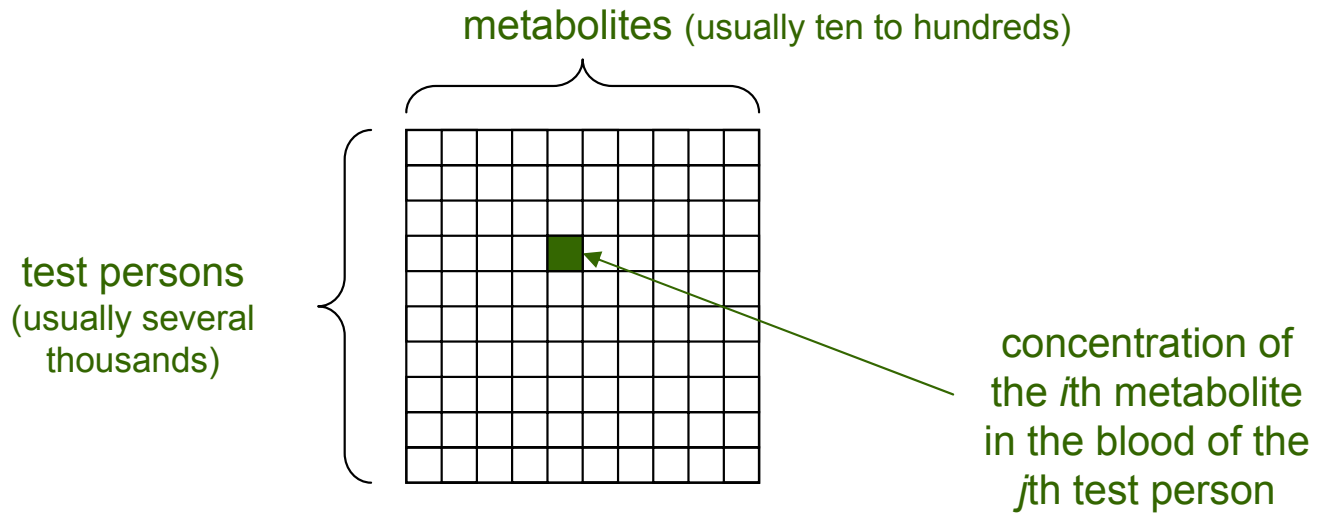


Cluster 1: {P1, P4, P8, P10}
Cluster 2: {P4, P5, P6}
Cluster 3: {P2, P4, P8, P10}

Sample Applications

- Metabolic Screening
  - Data
    - Concentration of different metabolites in the blood of different test persons
    - Example:
    - *Bavarian Newborn Screening*
    - Data matrix:

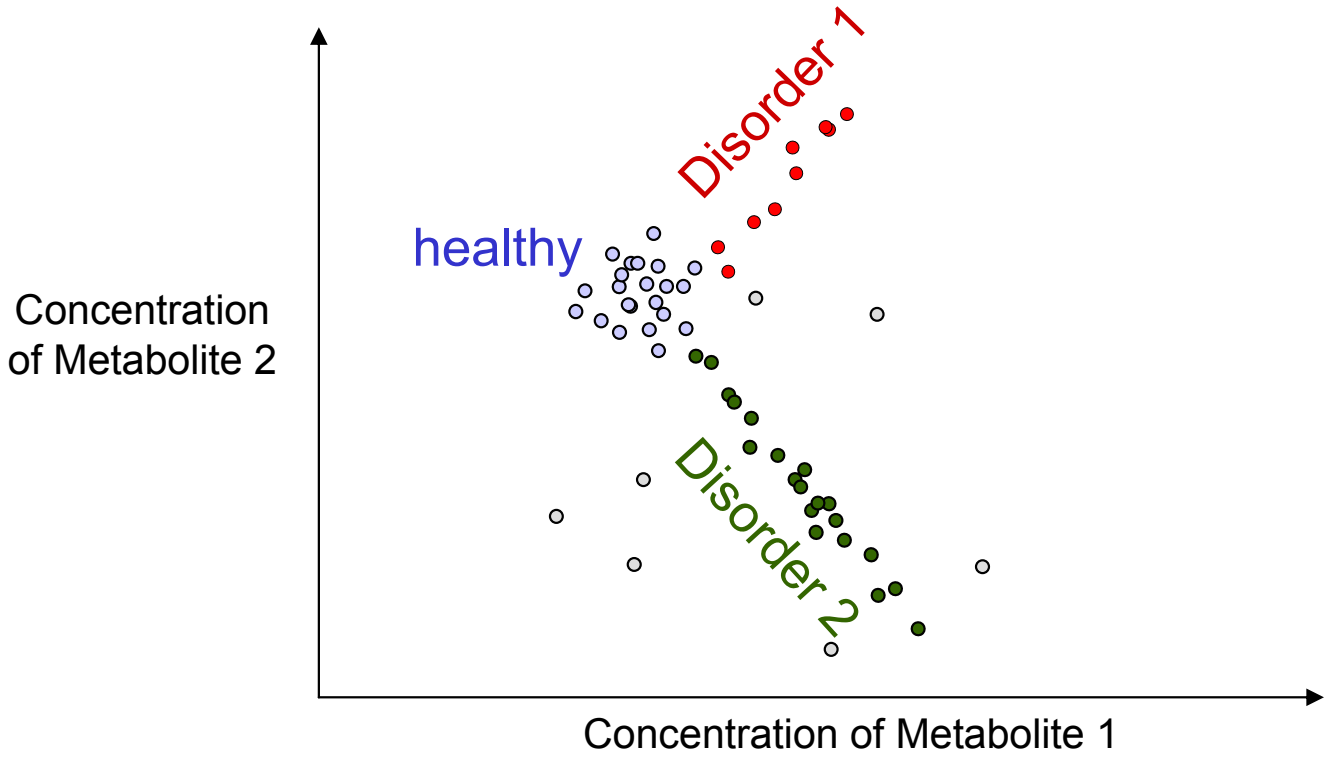




Sample Applications

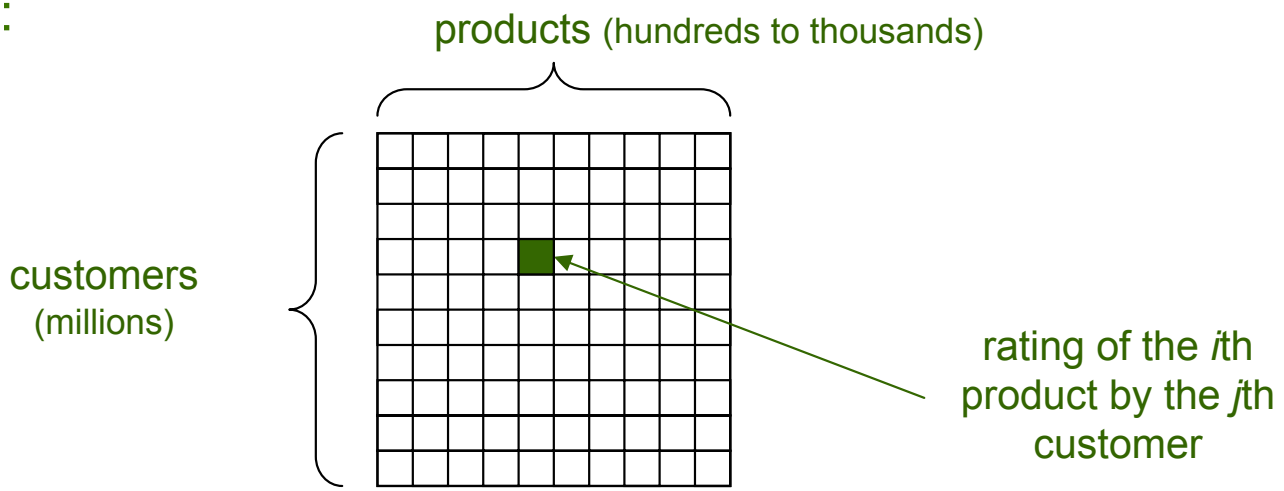
- Task: Cluster test persons to find groups of individuals with similar correlation among the concentrations of metabolites indicating homogeneous metabolic behavior (e.g. disorder)
  - *Challenge*:

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



Sample Applications

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



  - 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

Sample Applications

- And many more …
- In general, we face a steadily increasing number of applications that require the analysis of moderate-to-high dimensional data
- Moderate-to-high dimensional means from appr. 10 to hundreds or even thousands of dimensions

General Problems & Challenges

- The curse of dimensionality
  - In [BGRS99,HAK00] it is reported that the ratio of ($Dmax_d$ – $Dmin_d$) to $Dmin_d$ converges to zero with increasing dimensionality *d*
    - $Dmin_d$ = distance to the nearest neighbor in *d* dimensions
    - $Dmax_d$ = distance to the farthest neighbor in *d* dimensions

Formally:

$$\forall \varepsilon > 0 : \lim_{d \to \infty} \mathrm{P}[dist_d(\frac{\mathrm{Dmax}_d - \mathrm{Dmin}_d}{\mathrm{Dmin}_d}, 0) \leq \varepsilon] = 1$$

- This holds true for a wide range of data distributions and distance functions

General Problems & Challenges

- What does that mean for clustering high dimensional data?
  - The relative difference of distances between different points decreases with increasing dimensionality
  - The distances between points cannot be used in order to differentiate between points
  - The more the dimensionality is increasing, the more the data distribution degenerates to random noise
  - ***All points are almost equidistant from each other — there are no clusters to discover in high dimensional spaces!!!***

- Why do distances behave like they do in high dimensional spaces?
  - Usually the distance functions used give equal weight to all dimensions
  - However, all dimensions are not of equal importance
  - Adding irrelevant dimensions ruins any clustering based on a distance function that equally weights all dimensions

General Problems & Challenges

- Beyond the curse of dimensionality

  From the above sketched applications we can derive the following observations for high dimensional data

  – Subspace clusters:

    Clusters usually do not exist in the full dimensional space but are often hidden in subspaces of the data (e.g. in only a subset of experimental conditions a gene may play a certain role)

  – Local feature relevance/correlation:

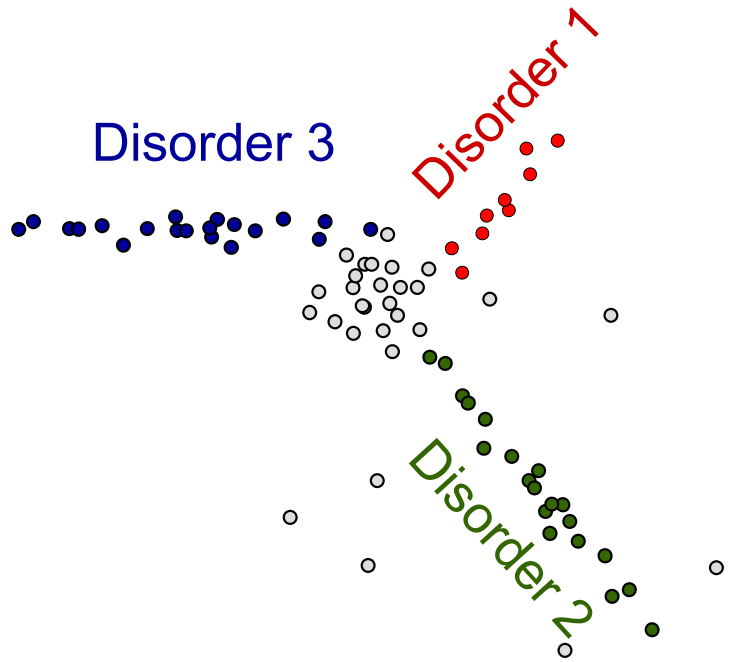
    For each cluster, a different subset of features or a different correlation of features may be relevant (e.g. different genes are responsible for different phenotypes)

  – Overlapping clusters:

    Clusters may overlap, i.e. an object may be clustered differently in varying subspaces (e.g. a gene may play different functional roles depending on the environment)

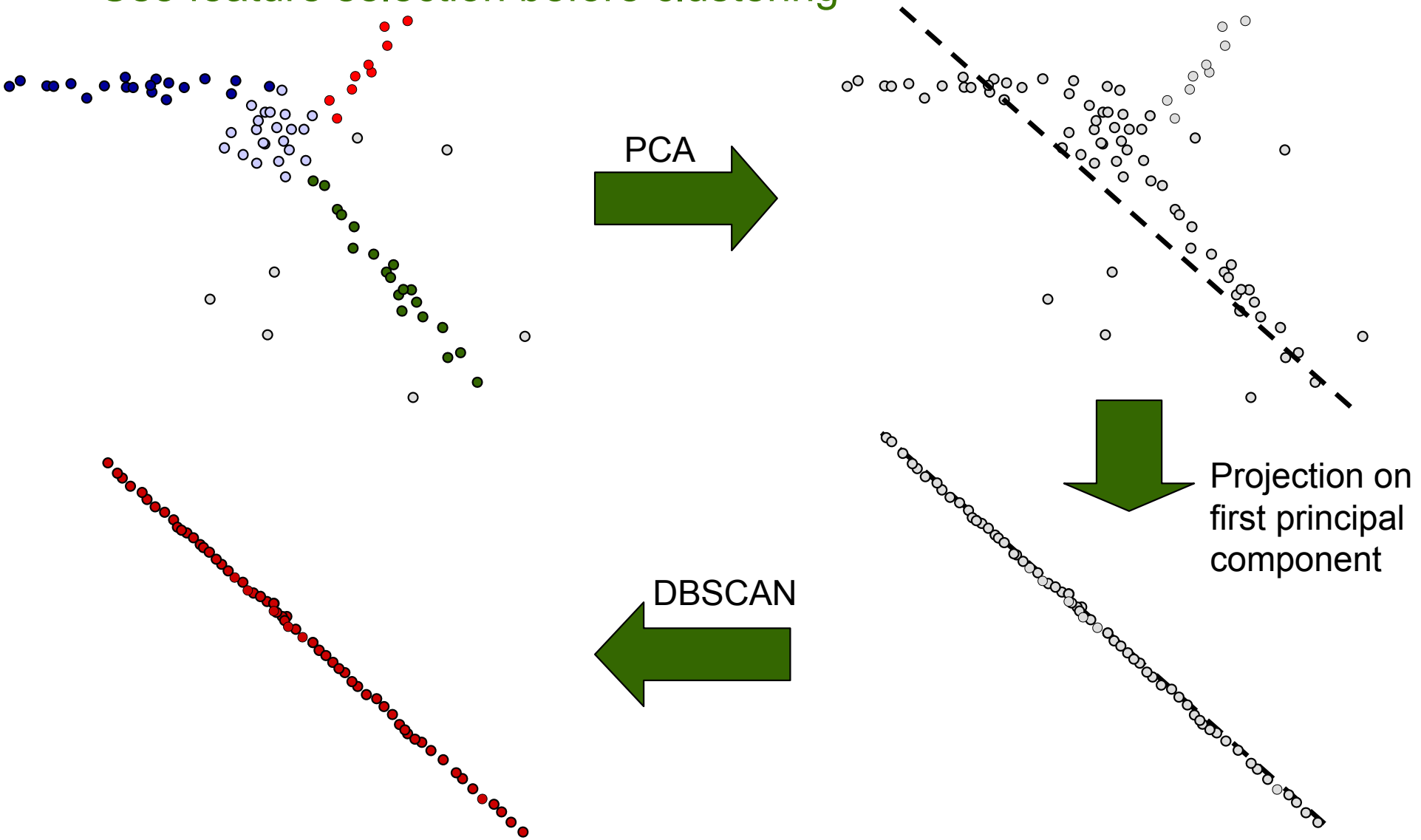
General Problems & Challenges

- 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



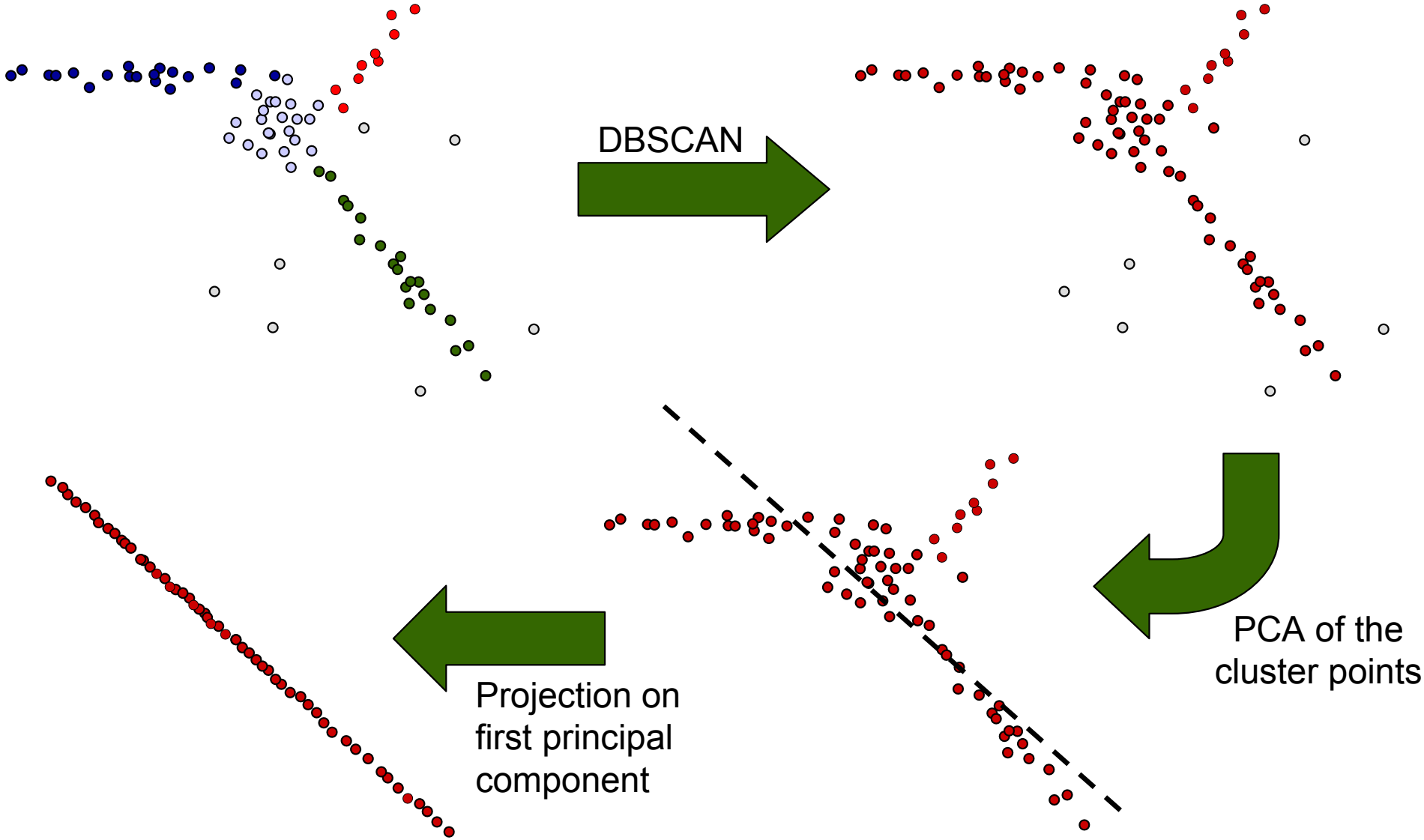
General Problems & Challenges

– Use feature selection before clustering



General Problems & Challenges

– Cluster first and then apply PCA



General Problems & Challenges

- Problem summary
  - Curse of dimensionality:
    - In high dimensional, sparse data spaces, clustering does not make sense
  - Local feature relevance and correlation:
    - Different features may be relevant for different clusters
    - Different combinations/correlations of features may be relevant for different clusters
  - Overlapping clusters:
    - Objects may be assigned to different clusters in different subspaces

General Problems & Challenges

- Solution: integrate variance / covariance analysis into the clustering process
  - Variance analysis:
    - Find clusters in axis-parallel subspaces
    - Cluster members exhibit low variance along the relevant dimensions
  - Covariance/correlation analysis:
    - Find clusters in arbitrarily oriented subspaces
    - Cluster members exhibit a low covariance w.r.t. a given combination of the relevant dimensions (i.e. a low variance along the dimensions of the arbitrarily oriented subspace corresponding to the given combination of relevant attributes)

A First Taxonomy of Approaches

- So far, we can distinguish between
  - Clusters in axis-parallel subspaces
    Approaches are usually called
    - "subspace clustering algorithms"
    - "projected clustering algorithms"
    - "bi-clustering or co-clustering algorithms"
  - Clusters in arbitrarily oriented subspaces
    Approaches are usually called
    - "bi-clustering or co-clustering algorithms"
    - "pattern-based clustering algorithms"
    - "correlation clustering algorithms"

A First Taxonomy of Approaches

- Note: other important aspects for classifying existing approaches are e.g.
  - The underlying cluster model that usually involves
    - Input parameters
    - Assumptions on number, size, and shape of clusters
    - Noise (outlier) robustness
  - Determinism
  - Independence w.r.t. the order of objects/attributes
  - Assumptions on overlap/non-overlap of clusters/subspaces
  - Efficiency

… so we should keep these issues in mind …

Outline

1. Introduction
2. Axis-parallel Subspace Clustering
3. Pattern-based Clustering
4. Arbitrarily-oriented Subspace Clustering
5. Summary

Outline: Axis-parallel Subspace Clustering

- Challenges and Approaches
- Bottom-up Algorithms
- Top-Down Algorithms
- Hybrid Algorithms
- Summary

Challenges

- What are we searching for?
  - Overlapping clusters: points may be grouped differently in different subspaces

    => "***subspace clustering***"
  - Disjoint partitioning: assign points uniquely to clusters (or noise)

    => "***projected clustering***"

  *Note: the terms **subspace** clustering and **projected** clustering are not used in a unified or consistent way in the literature*

- The naïve solution:
  - Explore each possible subspace of a *d*-dimensional dataset whether it contains a cluster
  - Runtime complexity: depends on the search space, i.e. the number of all possible subspaces of a *d*-dimensional data set

Challenges

- What is the number of all possible subspaces of a *d*-dimensional data set?
  - How many *k*-dimensional subspaces ($k \leq d$) do we have?

    The number of all *k*-tupels of a set of *d* elements is

$$\binom{d}{k}$$

  - Overall:

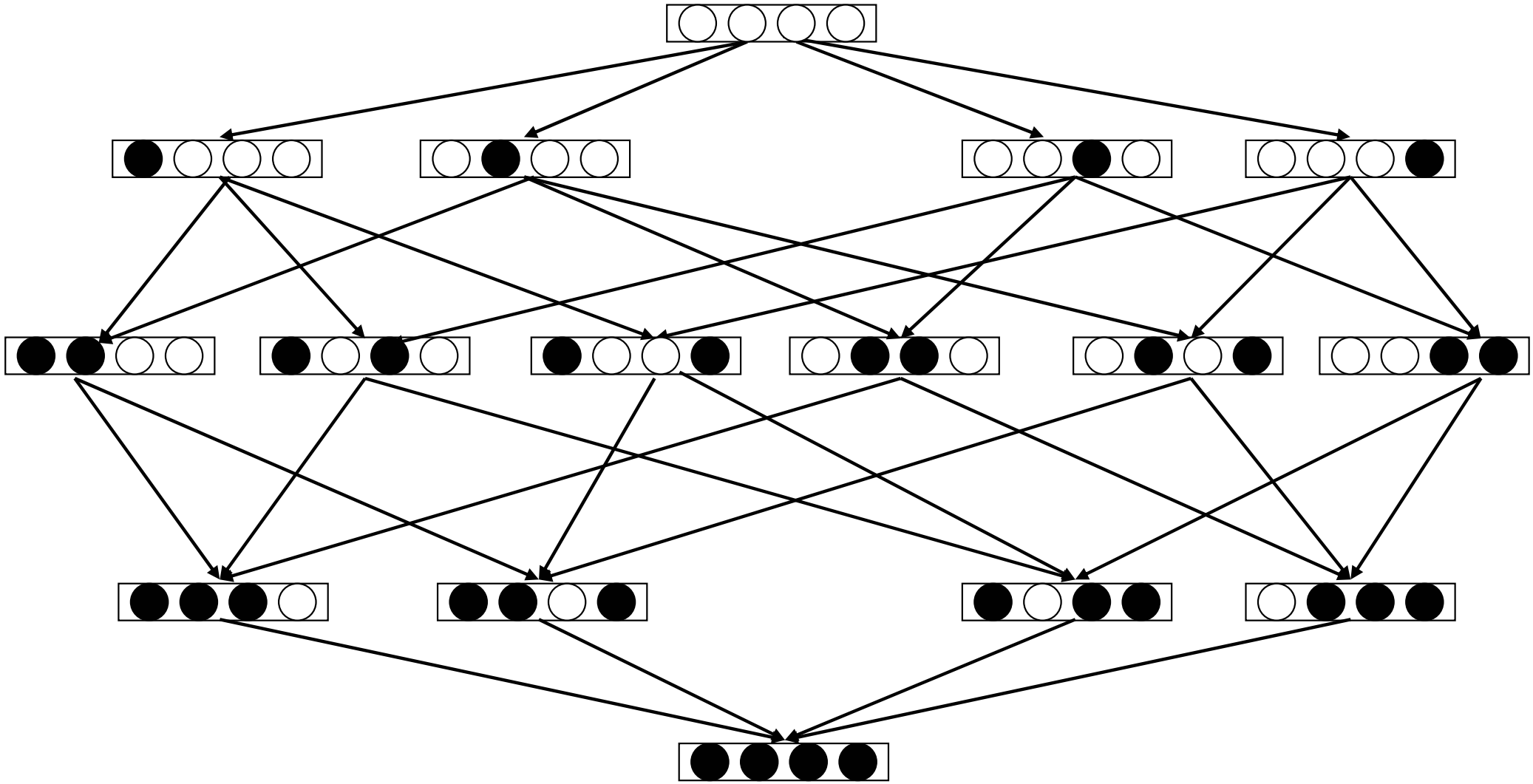
$$\sum_{k=1}^{d} \binom{d}{k} = 2^d - 1$$

  - So the naïve solution is computationally infeasible:

    We face a runtime complexity of $O(2^d)$

Challenges

- Search space for $d = 4$



Approaches

- Basically, there are two different ways to efficiently navigate through the search space
  - Bottom-up:
    - If the cluster criterion implements the downward closure property, one can use any bottom-up frequent itemset mining algorithm (e.g. APRIORI [AS94])
    - *Key*: downward-closure property
  - Top-down:
    - The search starts in the full *d*-dimensional space and iteratively learns for each point or each cluster the correct subspace
    - *Key*: procedure to learn the correct subspace
  - Some approaches actually use a hybrid approach in combination with other search heuristics

Bottom-up Algorithms

- Rational:
  - The cluster criterion must implement the downward closure property
    - If the criterion holds for any *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 APRIORI-style breadth-first search
    - Start with the 1-dimensional subspaces for which the criterion holds
    - Iteration (*k*–1)-dimensional to *k*-dimensional
      - Merge all (*k*–1)-dimensional subspaces for which the criterion holds to generate *k*-dimensional candidate subspaces
      - Prune all candidates that have at least one projection for which the criterion does not hold
      - Test the criterion for the remaining candidates

Bottom-up Algorithms

- 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

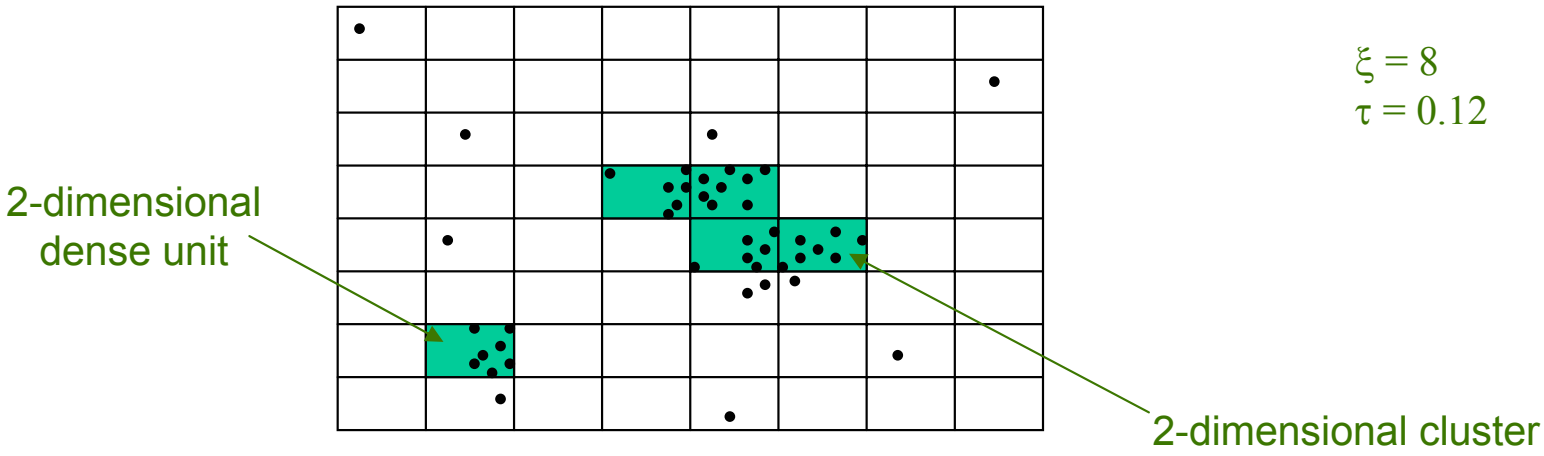
Bottom-up Algorithms

- Properties:
  - Generation of all clusters in all subspaces => overlapping clusters
  - Subspace clustering algorithms usually rely on bottom-up subspace search
  - Worst-case: complete enumeration of all subspaces, i.e. $O(2^d)$ time

- See some sample bottom-up algorithms on the following slides …

Bottom-up Algorithms

- CLIQUE [AGGR98]
  - Cluster model
    - Each dimension is partitioned into ξ equi-sized intervals called units
    - A *k*-dimensional unit is the intersection of *k* 1-dimensional units (from different dimensions)
    - A unit *u* is considered dense if the fraction of all data points in *u* exceeds the threshold τ
    - A cluster is a maximal set of connected dense units



Bottom-up Algorithms

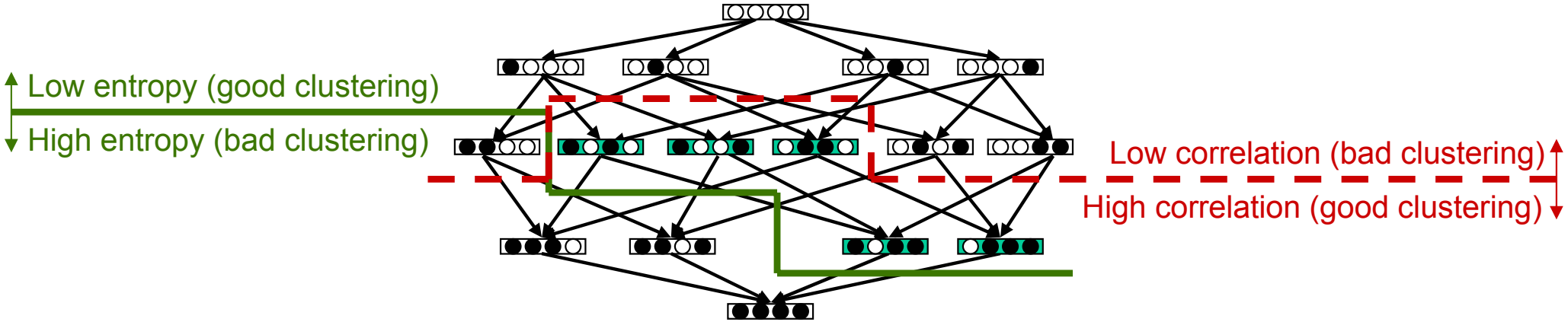
- Downward-closure property holds for dense units
- Algorithm
  - All dense cells are computed using the bottom-up approach
  - A heuristics based on the coverage of a subspace is used to further prune units that are dense but are in less interesting subspaces
    (coverage of subspace *S* = fraction of data points covered by the dense units of *S*)
  - All connected dense units in a common subspace are merged to generate the subspace clusters
  - Minimal cluster descriptions are generated
- Discussion
  - 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

Bottom-up Algorithms

- ENCLUS [CFZ99]
  - Cluster model uses a fixed grid similar to CLIQUE
  - Algorithm first searches for subspaces rather than for dense units
  - Subspaces are evaluated following three criteria
    - Coverage (see CLIQUE)
    - Entropy
      - Indicates how densely the points are packed in the corresponding subspace (the higher the density, the lower the entropy)
      - Implements the downward closure property
    - Correlation
      - Indicates how the attributes of the corresponding subspace are correlated to each other
      - Implements an upward closure property

Bottom-up Algorithms

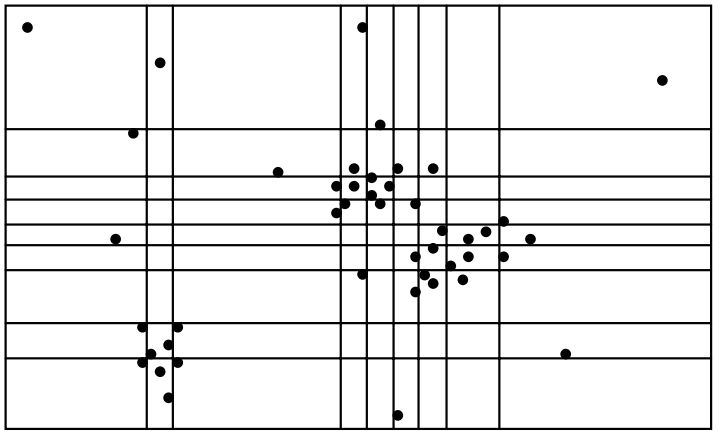
- Subspace search algorithm is bottom-up similar to CLIQUE but determines subspaces having
  - Entropy below threshold ω
  - Correlation above threshold ε



- Discussion
  - Input: thresholds ω and ε
  - Output: all subspaces that meet the above criteria (far less than CLIQUE)
  - Uses fixed thresholds for entropy and correlation for all subspaces
  - Simple but efficient cluster model

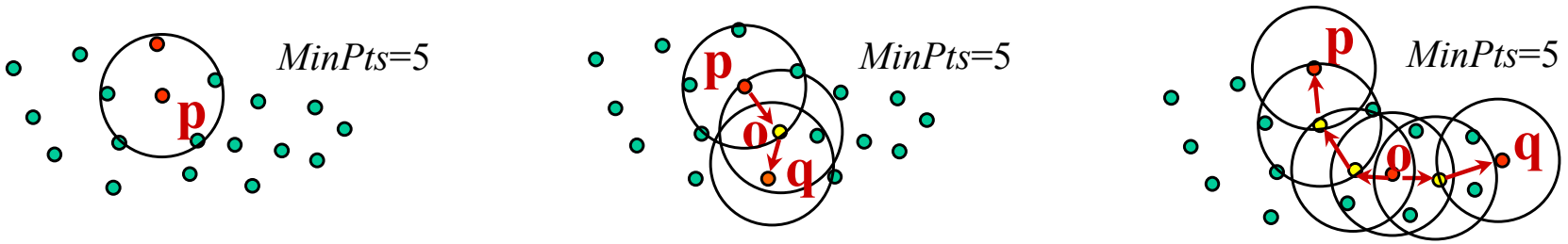
Bottom-up Algorithms

- MAFIA [NGC01]
  - Variant of CLIQUE, cluster model uses an adaptive grid:
    - each 1-dimensional unit covers a fixed number of data points
    - Density of higher dimensional units is again defined in terms of a threshold τ (see CLIQUE)
    - Using an adaptive grid instead of a fixed grid implements a more flexible cluster model – however, grid specific problems remain
  - Parallelism for speed-up
  - Discussion
    - Input: ξ and τ (density threshold)
    - Output: all clusters in all subspaces
    - Uses a fixed density threshold for all subspaces
    - Simple but efficient cluster model
    - Large speed-up over CLIQUE due to parallelism



Bottom-up Algorithms

- SUBCLU [KKK04]
  - Cluster model:
    - Density-based cluster model of DBSCAN [EKSX96]
    - 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



    - Detect clusters of arbitrary size and shape (in the corresponding subspaces)
  - Downward-closure property holds for sets of density-connected points

Bottom-up Algorithms

- 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

- P3C [MSE06]
  - Cluster model
    - Cluster cores (hyper-rectangular approximations of subspace clusters) are computed in a bottom-up fashion from 1-dimensional intervals
    - Cluster cores initialize an EM fuzzy clustering of all data points
  - Algorithm proceeds in 3 steps
    - Computing 1-dimensional cluster projections (intervals)
      - Each dimension is partitioned into $\lfloor 1+\log_2(n) \rfloor$ equi-sized bins
      - A Chi-square test is employed to discard bins containing too less points
      - Adjacent bins are merged; the remaining intervals are reported

Bottom-up Algorithms

- Aggregating the cluster projections to higher dimensional cluster cores
  - A *p*-signature (set of 1-dimensional intervals) approximates a *p*-dimensional subspace cluster by a minimum bounding *p*-dimensional box

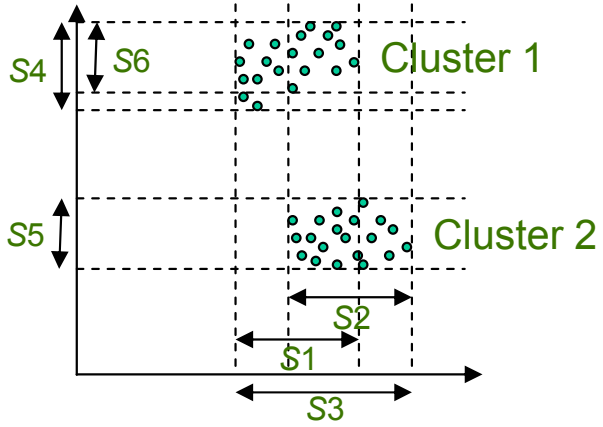
    Example:

    2-signature S4∩S1 approximates cluster 1

    2-signature S3∩S6 approximates cluster 1

    2-signature S3∩S5 approximates cluster 2

    …



  - A *p*-signature *S* is a cluster core, if
    1. the exchange of any interval in S with any interval not in S does not increase the quality of the approximation
    2. The addition of any interval to S does not increase the quality of the approximation

       (quality is measured by means of the real and the expected number of points contained in an approximation)
  - Condition 1. implements downward closure property

    => search bottom-up for cluster cores

Bottom-up Algorithms

- Computing true clusters from cluster cores
  - Let *k* be the number of cluster cores generated
  - Cluster all points with EM using *k* cluster core centers as initial clusters

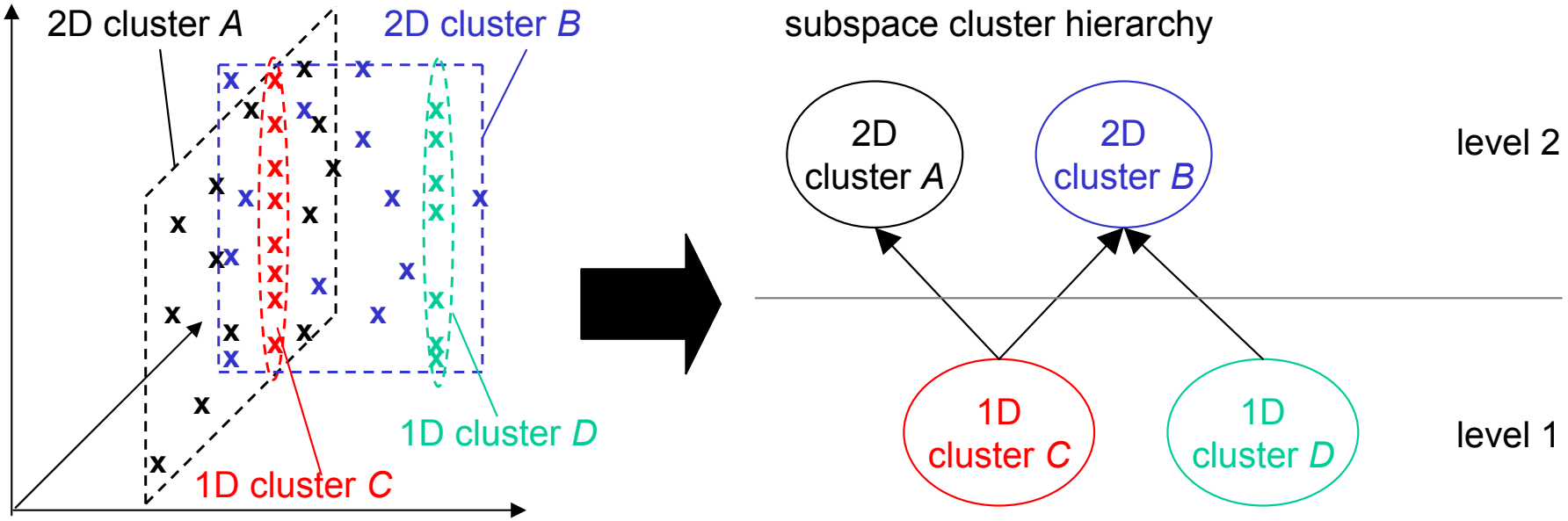
– Discussion
- Input: Poisson threshold for the Chi-square test to compute 1-dimensional cluster projections
- Output: a fuzzy clustering of points to *k* clusters (NOTE: number of clusters *k* is determined automatically), i.e. for each point *p* the probabilities that *p* belongs to each of the *k* clusters is computed

  From these probabilities
  - a disjoint partition can be derived (projected clustering)
  - also overlapping clusters can be discovered (subspace clustering)
- First subspace algorithm that does ***not*** use a global density threshold

NOTE: the method DUSC (see Session *Clustering* tomorrow in Big Blue from 13:30 to 15:30) solves the problem of global density thresholds in an elegant way!

Top-down Algorithms

- DiSH [ABK+07a]
  - Idea:
    - So far: integration of variance analysis into “flat” clustering
    - Not considered: lower dimensional clusters embedded in higher dimensional ones



    - Now: find hierarchies of subspace clusters
    - Integrate a proper distance function into hierarchical clustering

Top-down Algorithms

- Distance measure that captures subspace hierarchies assigns
  - 1 if both points share a common 1D subspace cluster
  - 2 if both points share a common 2D subspace cluster
  - …
- Sharing a common k-dimensional subspace cluster means
  - Both points are associated to the same k-dimensional subspace cluster
  - Both points are associated to different (k-1)-dimensional subspace clusters that intersect or are parallel (but not skew)
- This distance is based on the subspace dimensionality of each point *p* representing the (highest dimensional) subspace in which *p* fits best
  - Analyze the local ε-neighborhood of *p* along each attribute *a*

    => if it contains more than μ points: *a* is interesting for *p*
  - Combine all interesting attributes such that the ε-neighborhood of *p* in the subspace spanned by this combination still contains at least μ points (e.g. use APRIORI algorithm or best-first search)

Top-down Algorithms

- Discussion
  - Input: ε and μ specify the density threshold for computing the relevant subspaces of a point
  - Output: a hierarchy of subspace clusters displayed as a graph
  - Relies on a global density threshold
  - Complex but costly cluster model

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
  - 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

Top-down Algorithms

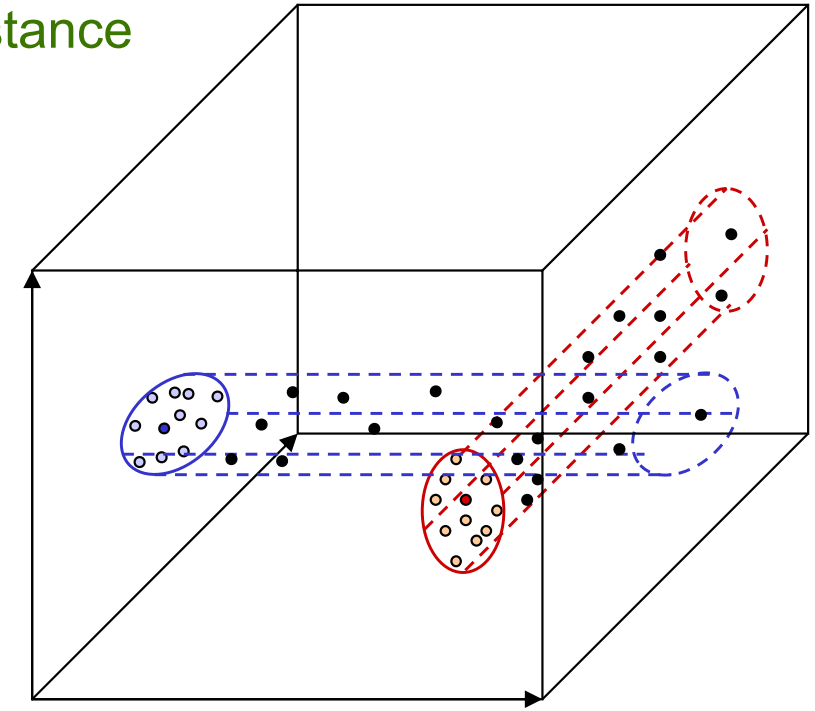
- The key limitation: ***the locality assumption***
  - How should we learn the subspace preference of a cluster or a point?
  - The subspace is usually learned from the local neighborhood of cluster representatives/cluster members in the entire feature space:
    - Cluster-based approach: the ***local 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
  - This is a rather optimistic assumption (recall the effects of the curse of dimensionality on concepts like "local neighborhood"); the learning procedure is often misled by noise points

Top-down Algorithms

- Properties:
  - Simultaneous search for the “best” partitioning of the data points and the “best” subspace for each partition => disjoint partitioning
  - Projected clustering algorithms usually rely on top-down subspace search
  - Worst-case:
    - usually complete enumeration of all subspaces is avoided
    - Worst-case costs are typically in $O(d^2)$
- See some sample top-down algorithms on the following slides …

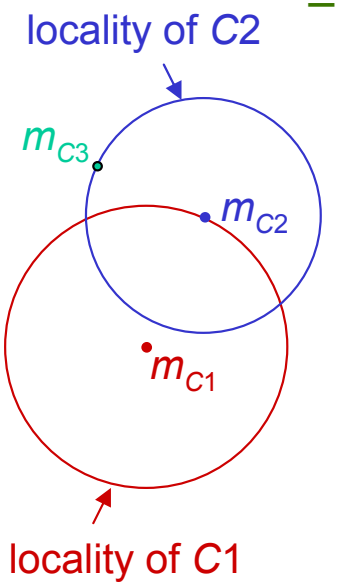
Top-down Algorithms

- PROCLUS [APW+99]
  - *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 subspaces of the medoids)
    - Points that have a large distance to its nearest medoid are classified as noise



Top-down Algorithms

- 3-Phase Algorithm
  - Initialization of cluster medoids
    - A superset M of $b \cdot k$ medoids is computed from a sample of $a \cdot k$ data points such that these medoids are well separated
    - k randomly chosen medoids from M are the initial cluster representatives
    - Input parameters a and b are introduced for performance reasons
  - Iterative phase works similar to any k-medoid clustering
    - Approximate subspaces for each cluster C
      - » The locality of C includes all points that have a distance to the medoid of C less than the distance between the medoid of C and the medoid of the neighboring cluster
      - » Compute standard deviation of distances from the medoid of C to the points in the locality of C along each dimension
      - » Add the dimensions with the smallest standard deviation to the relevant dimensions of cluster C such that
        - in summary $k \cdot l$ dimensions are assigned to all clusters
        - each cluster has at least 2 dimensions assigned



Top-down Algorithms

  - Reassign points to clusters
    - » Compute for each point the distance to each medoid taking only the relevant dimensions into account
    - » Assign points to a medoid minimizing these distances
  - Termination (criterion not really clearly specified in [APW+99])
    - » 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)

- Refinement
  - Reassign subspaces to medoids as above (but use only the points assigned to each cluster rather than the locality of each cluster)
  - Reassign points to medoids; points that are not in the locality of their corresponding medoids are classified as noise

Top-down Algorithms

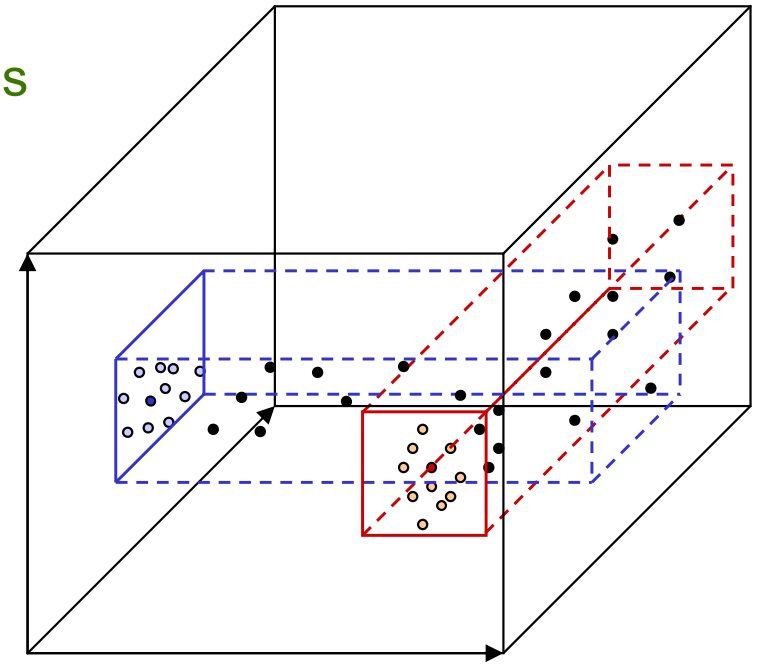
- Discussion
  - Input:
    - Number of cluster *k*
    - Average dimensionality of clusters *l*
    - Factor *a* to determine the size of the sample in the initialization step
    - Factor *b* to determine the size of the candidate set for the medoids
  - Output: partitioning of points into *k* disjoint clusters and noise, each cluster has a set of relevant attributes specifying its subspace
  - Relies on cluster-based locality assumption: subspace of each cluster is learned from local neighborhood of its medoid
  - Biased to find *l*-dimensional subspace clusters
  - Simple but efficient cluster model

Top-down Algorithms

- DOC [PJAM02]
  - Cluster model
    - A cluster is a pair (*C*,*D*) of cluster members *C* and relevant dimensions *D* such that all points in *C* are contained in a |*D*|-dimensional hyper-cube with side length *w* and $|C| \geq \alpha \cdot |DB|$
    - The quality of a cluster (C,D) is defined as

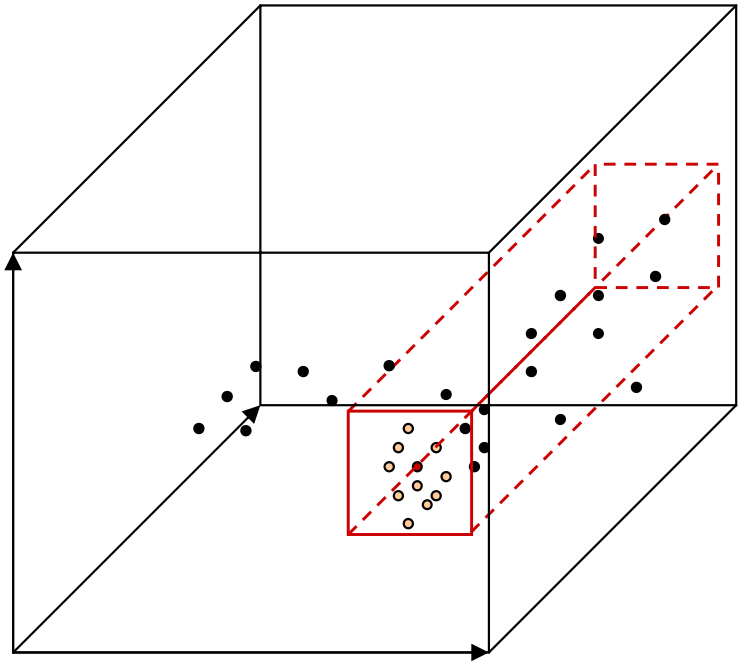
$$\mu(C,D) = |C| \cdot \left(\frac{1}{\beta}\right)^{|D|}$$

      where $\beta \in [0,1)$ specifies the trade-off between the number of points and the number of dimensions in a cluster
    - An optimal cluster maximizes μ
    - Note:
      - there may be several optimal clusters
      - μ is monotonically increasing in each argument



Top-down Algorithms

- Algorithm
  - Idea: Generate an approximation of an optimal cluster (C,D) in one run
    - Guess (via random sampling) a seed $p \in C$ and determine D *(see next slide)*
    - Let B(p,D) be the $|D|$-dimensional hyper-cube centered at p with width $2 \cdot w$ and let $C^* = DB \cap B(p,D)$
    - Then $\mu(C^*,D) \geq \mu(C,D)$ because (C^*,D) may contain additional points
    - However, (C^*,D) has side length $2 \cdot w$ instead of w



Top-down Algorithms

- Determine D from a randomly sampled seed point p and a set of sampled discriminating points X

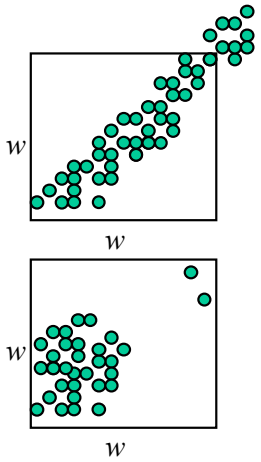
  If $|p_i - q_i| \leq w$ for all $q \in X$, then dimension $i \in D$

- Algorithm overview
  - Compute a set of $2/\alpha$ clusters (C,D) as follows
    - » Choose a seed p randomly
    - » Iterate m times (m depends non-trivially on parameters α and β):
      - Choose a discriminating set X of size r

        (r depends non-trivially on parameters α and β)
      - Determine D as described above
      - Determine C^* as described on the previous slide
      - Report (C^*,D) if $|C^*| \geq \alpha \cdot |DB|$
  - Report the cluster with the highest quality μ
- It can be shown that if $1/(4d) \leq \beta \leq ½$, then the probability that DOC returns a cluster is above 50%

Top-down Algorithms

- Discussion
  - Input:
    - w and α specifying the density threshold
    - β specifies the trade-off between the number of points and the number of dimensions in a cluster
  - Output: a $2 \cdot w$-approximation of an projected cluster that maximizes μ
  - NOTE: DOC does ***not*** rely on the locality assumption
  - But
    - it uses a global density threshold
    - the quality of the resulting cluster depends on
      - » the randomly sampled seed
      - » the randomly sampled discriminating set
      - » the position of the hyper-box

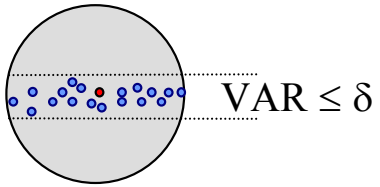


  - Needs multiple runs to improve the probability to succeed in finding a cluster; one run only finds one cluster

Top-down Algorithms

- PreDeCon [BKKK04]
  - Cluster model:
    - Density-based cluster model of DBSCAN [EKSX96] adapted to projected clustering
      - For each point *p* a subspace preference indicating the subspace in which *p* clusters best is computed
      - ε-neighborhood of a point *p* is constrained by the subspace preference of *p*
      - Core points have at least *minPts* other points in their ε-neighborhood
      - Density connectivity is defined based on core points
      - Clusters are maximal sets of density connected points
    - Subspace preference of a point p is *d*-dimensional vector w=(w_1,…,w_d), entry w_i represents dimension *i* with

$$w_i = \begin{cases} 1 & if \quad \text{VAR}_i > \delta \\ \kappa & if \quad \text{VAR}_i \leq \delta \end{cases}$$



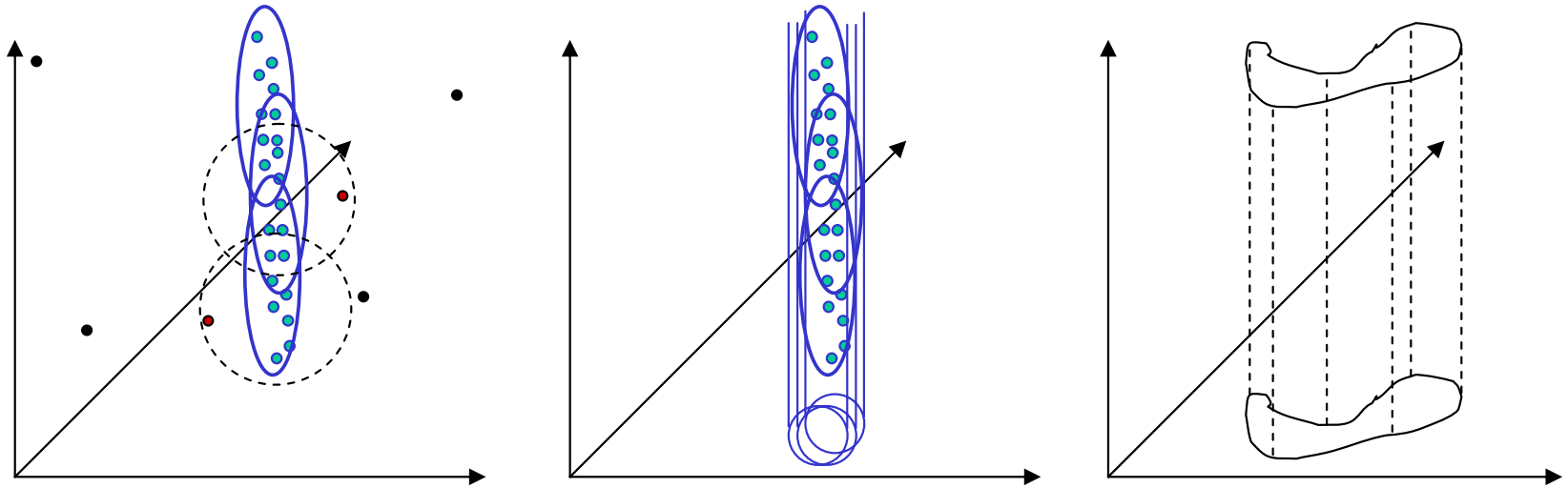
VAR$_i$ is the variance of the ε-neighborhood of *p* in the entire *d*-dimensional space, δ and κ are input parameters

Top-down Algorithms

- Algorithm
  - PreDeCon applies DBSCAN with a weighted Euclidean distance function

$$dist_p(p,q) = \sqrt{\sum_i w_i \cdot (p_i - q_i)}$$ w.r.t. *p*

  - Instead of shifting spheres (full-dimensional Euclidean ε-neighborhoods), clusters are expanded by shifting axis-parallel ellipsoids (weighted Euclidean ε-neighborhoods)
  - Note: In the subspace of the cluster (defined by the preference of its members), we shift spheres (but this intuition may be misleading)



Top-down Algorithms

- Discussion
  - Input:
    - δ and κ to determine the subspace preference
    - λ specifies the maximal dimensionality of a subspace cluster
    - ε and *minPts* specify the density threshold
  - Output: a disjoint partition of data into clusters and noise
  - Relies on instance-based locality assumption: subspace preference of each point is learned from its local neighborhood
  - Advanced but costly cluster model

Top-down Algorithms

- COSA [FM04]
  - Idea:
    - Similar to PreDeCon, a weight vector w_p for each point p is computed that represents the subspace in which each points clusters best
    - The weight vector can contain arbitrary values rather than only 1 or a fixed constant κ
    - The result of COSA is not a clustering but an $n \times n$ matrix $\boldsymbol{D}$ containing the weighted distances d_{pq}
    - A subspace clustering can be derived by applying any clustering algorithm (e.g. a hierarchical algorithm) using the distance matrix $\boldsymbol{D}$

Top-down Algorithms

- Determination of the distance matrix ***D***
  - For each point *p*, initialize the weight vector w_p with equal weights
  - Iterate until all weight vectors stabilize:
    - Compute the distance matrix ***D*** using the corresponding weight vectors
    - Compute for each point *p* the *k*-nearest neighbors w.r.t. ***D***
    - Re-compute weight vector w_p for each point *p* based on the distance distribution of the *k*NN of p in each dimension

$$w_p^i = \frac{e^{-\frac{1/k \sum_{q \in kNN(p)} \text{distance between } p \text{ and } q \text{ in attribute } i}{\lambda}}}{\sum_{k=1}^{d} e^{-\frac{1/k \sum_{q \in kNN(p)} \text{distance between } p \text{ and } q \text{ in attribute } k}{\lambda}}}$$

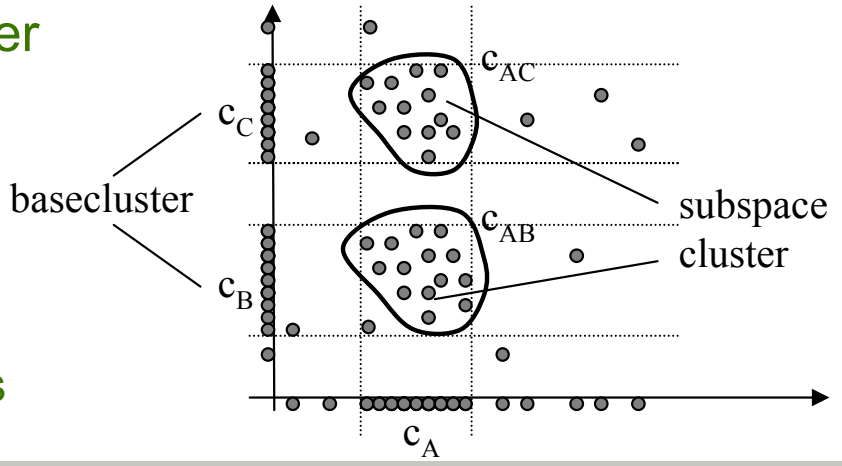
where λ is a user-defined input parameter that affects the dimensionality of the subspaces reflected by the weight vectors/distance matrix

Top-down Algorithms

- Discussion
  - Input:
    - Parameters λ and α that affect the dimensionality of the subspaces reflected by the weight vectors/distance matrix
    - The number *k* of nearest neighbors from which the weights of each point are learned
  - Output: an $n \times n$ matrix reflecting the weighted pair-wise distance between points
  - Relies on instance-based locality assumption: weight vectors of each point is learned from its *k*NN; at the beginning of the loop, the *k*NNs are computed in the entire *d*-dimensional space
  - Can be used by any distance-based clustering algorithm to compute a flat or hierarchical partitioning of the data

Hybrid Algorithms

- FIRES[KKRW05]
  - Proposes a bottom-up approach that uses different heuristics for subspace search
  - 3-Step algorithm
    - Starts with 1-dimensional clusters called *base clusters* (generated by applying any traditional clustering algorithm to each 1-dimensional subspace)
    - Merges these clusters to generate subspace cluster approximations by applying a clustering of the base clusters using a variant of DBSCAN (similarity between two clusters $C1$ and $C2$ is defined by $|C1 \cap C2|$)
    - Refines the resulting subspace cluster approximations
      - Apply any traditional clustering algorithm on the points within the approximations
      - Prune lower dimensional projections



Hybrid Algorithms

- Discussion
  - Input:
    - Three parameters for the merging procedure of base clusters
    - Parameters for the clustering algorithm to create base clusters and for refinement
  - Output: clusters in maximal dimensional subspaces, clusters may overlap
  - Allows overlapping clusters (subspace clustering) but avoids complete enumeration; runtime of the merge step is O(*d*)!!!
  - Output heavily depends on the accuracy of the merge step which is a rather simple heuristic and relies on three sensitive parameters
  - Cluster model can be chosen by the user

Summary

- The big picture
  - Subspace clustering algorithms compute all clusters in all subspaces
    - They usually implement a bottom-up search strategy
    - They usually rely on global density thresholds to ensure the downward closure property
    - They usually do not rely on the locality assumption
    - They usually have a worst case complexity of $O(2^d)$
  - Projected clustering algorithms compute a disjoint partition of the data
    - The usually implement a top-down search strategy
    - They usually rely on the locality assumption
    - They usually do not rely on global density thresholds
    - They usually scale at most quadratic in the number of dimensions
  - Hybrid approaches allow both subspace and projected clustering
    - They usually invent further assumptions and heuristics to navigate the search space

Outline

1. Introduction
2. Axis-parallel Subspace Clustering
3. Pattern-based Clustering
4. Arbitrarily-oriented Subspace Clustering
5. Summary

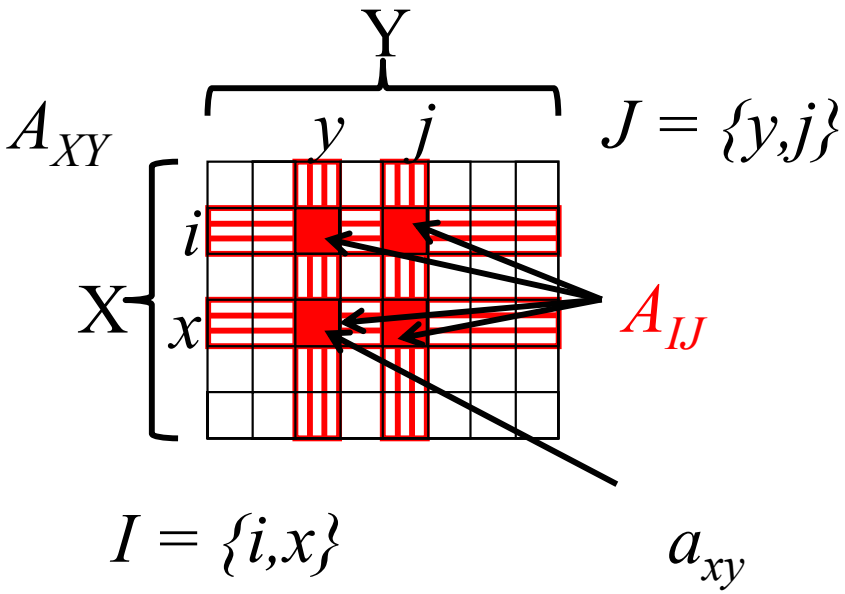
Outline: Pattern-based Clustering

- Challenges and Approaches, Basic Models
- Algorithms for
  - Constant Biclusters
  - Biclusters with Constant Values in Rows or Columns
  - Pattern-based Clustering: Biclusters with Coherent Values
  - Biclusters with Coherent Evolutions
- Summary

Challenges and Approaches, Basic Models

Pattern-based clustering relies on patterns in the data matrix.

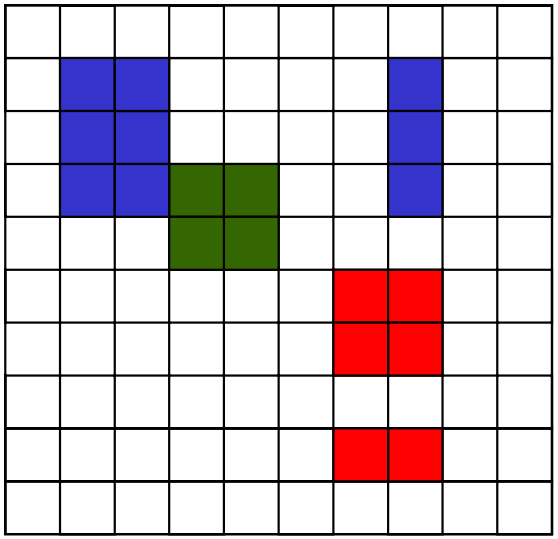
- Simultaneous clustering of rows and columns of the data matrix (hence *bi*clustering).
  - Data matrix ***A*** = (X,Y) with set of rows X and set of columns Y
  - a_{xy} is the element in row x and column y.
  - submatrix A_{IJ} = (I,J) with subset of rows I $\subseteq$ X and subset of columns J $\subseteq$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



Challenges and Approaches, Basic Models

General aim of biclustering approaches:

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.



Challenges and Approaches, Basic Models

- 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:

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

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

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

Challenges and Approaches, Basic Models

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

Challenges and Approaches, Basic Models

Constant biclusters

- all points share identical value in selected attributes.
- The constant value μ is a typical value for the cluster.
- Cluster model:

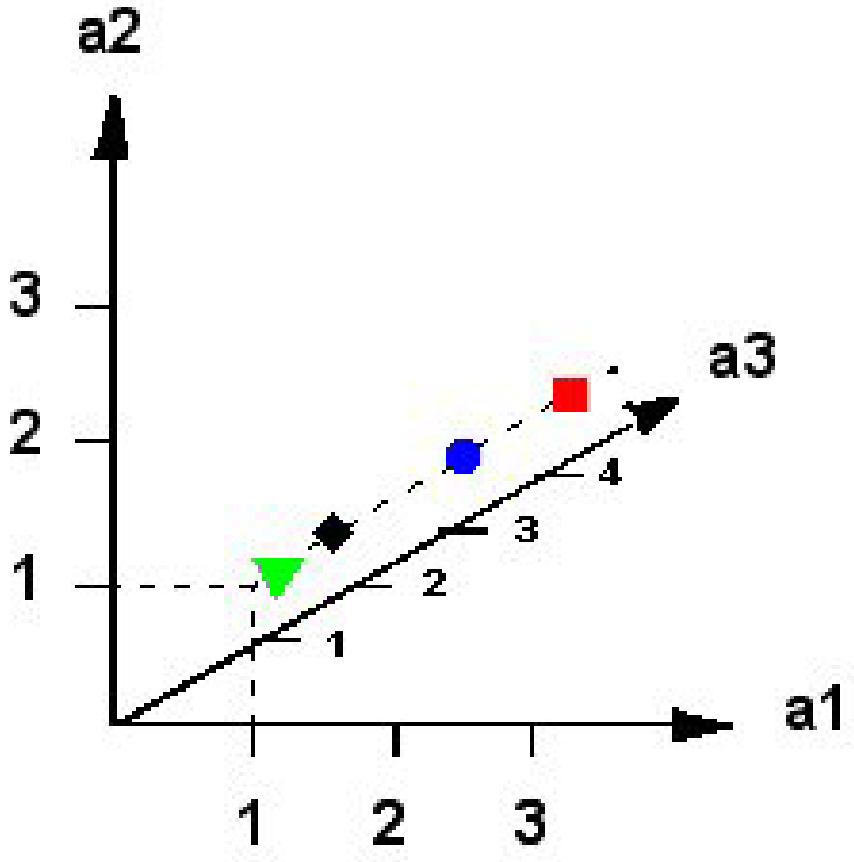
$$a_{ij} = \mu$$

- Obviously a special case of an axis-parallel subspace cluster.

Challenges and Approaches, Basic Models

- example – embedding 3-dimensional space:

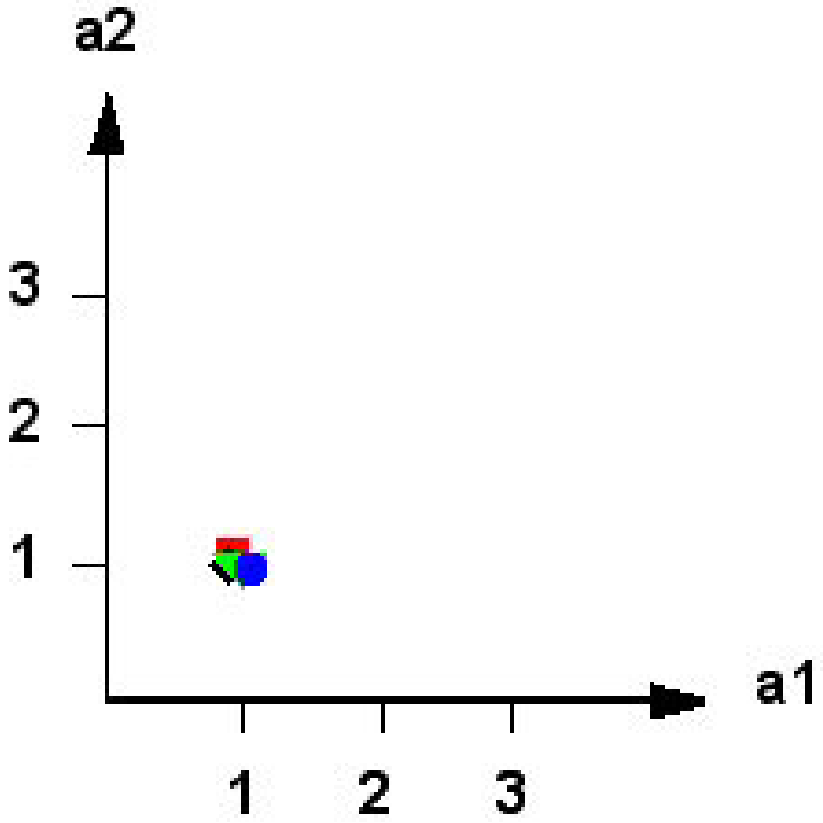
	a1	a2	a3
P1	1	1	3.5
P2	1	1	2.3
P3	1	1	0.2
P4	1	1	0.7



Challenges and Approaches, Basic Models

- example – 2-dimensional subspace:

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

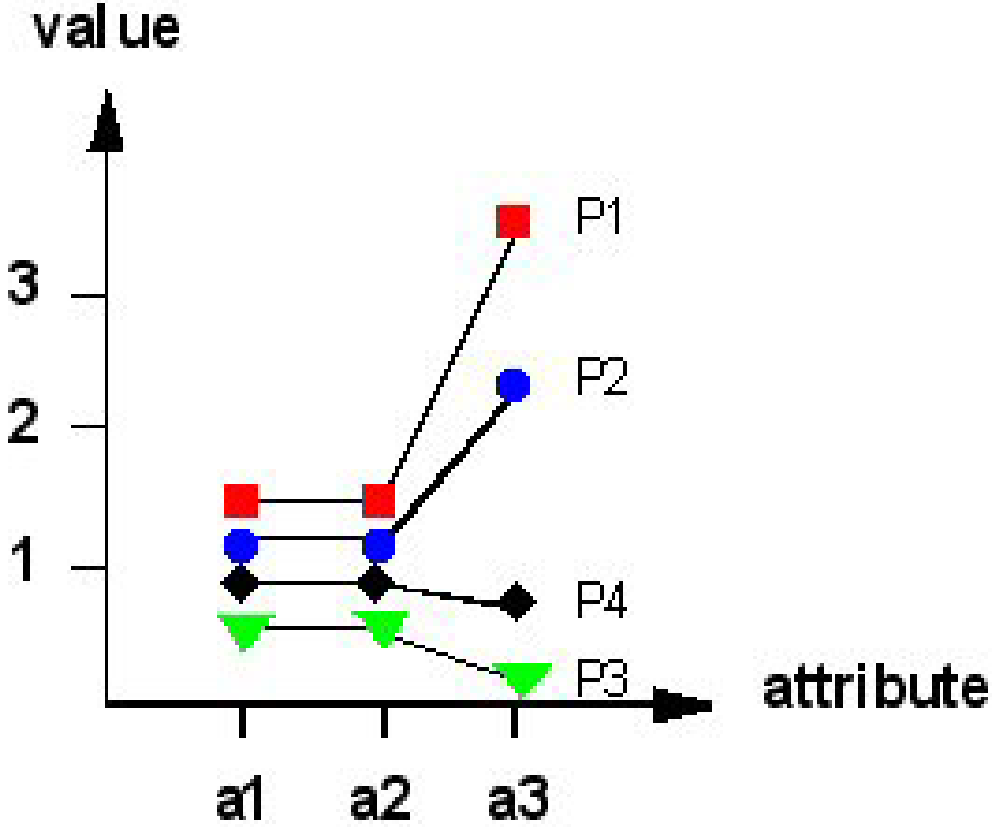


- points located on the bisecting line of participating attributes

Challenges and Approaches, Basic Models

- 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

Challenges and Approaches, Basic Models

- 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.

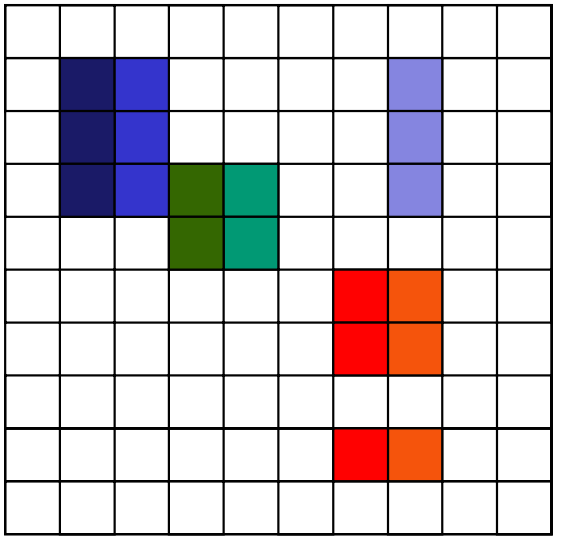
Challenges and Approaches, Basic Models

Biclusters with constant values on columns

- Cluster model for $\boldsymbol{A}_{IJ}$ = (I,J):

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

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



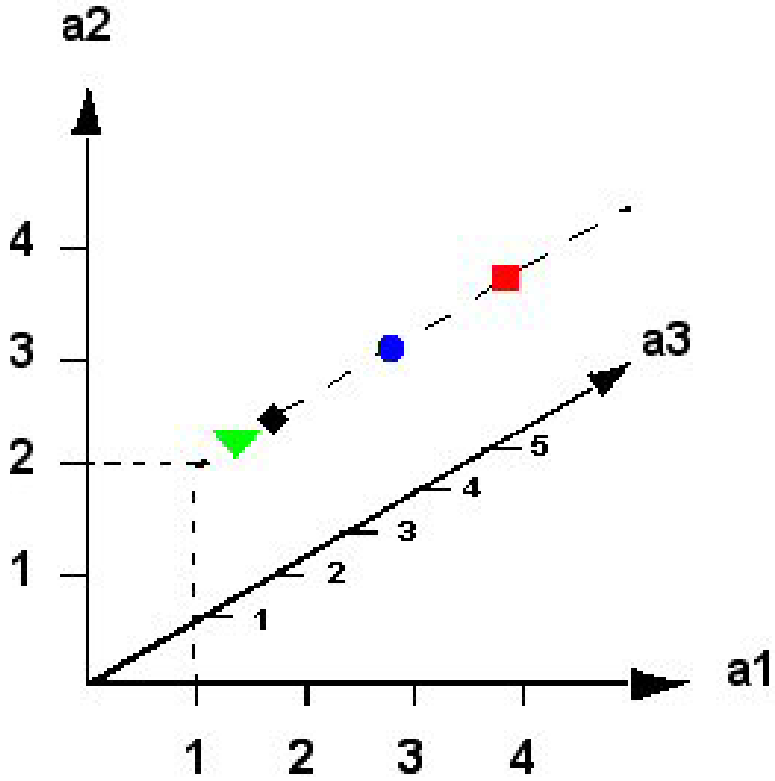
- adjustment value c_j for column $j \in J$

- results in axis-parallel subspace clusters

Challenges and Approaches, Basic Models

- example – 3-dimensional embedding space:

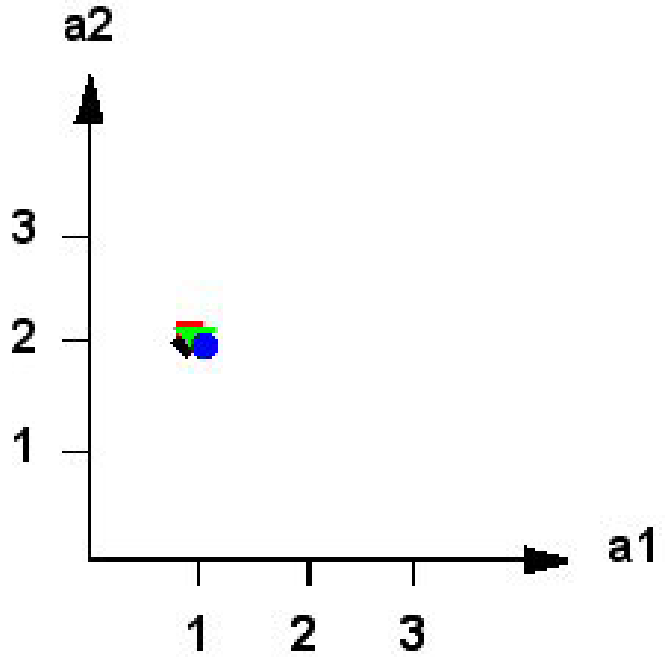
	a1	a2	a3
P1	1	2	3.5
P2	1	2	2.3
P3	1	2	0.2
P4	1	2	0.7



Challenges and Approaches, Basic Models

- example – 2-dimensional subspace:

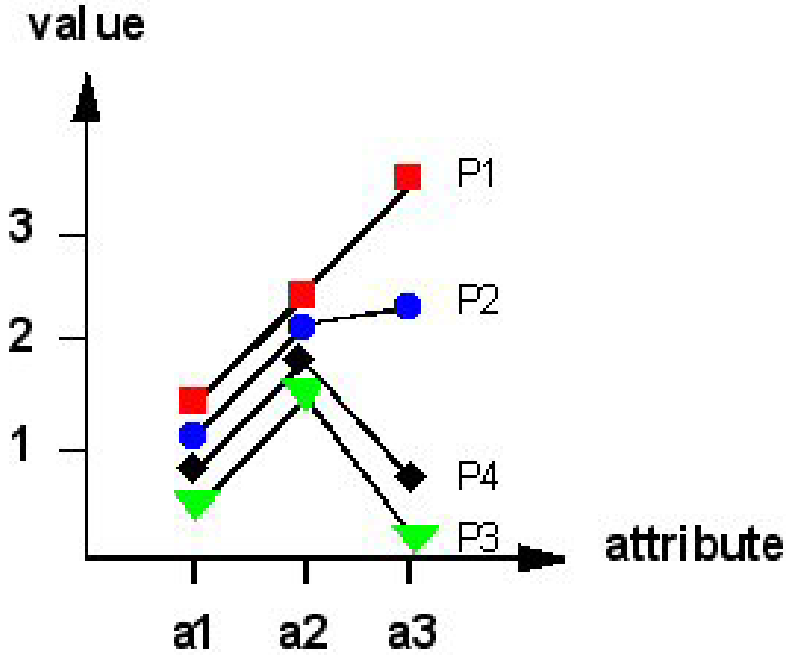
	a1	a2
P1	1	2
P2	1	2
P3	1	2
P4	1	2



Challenges and Approaches, Basic Models

- 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

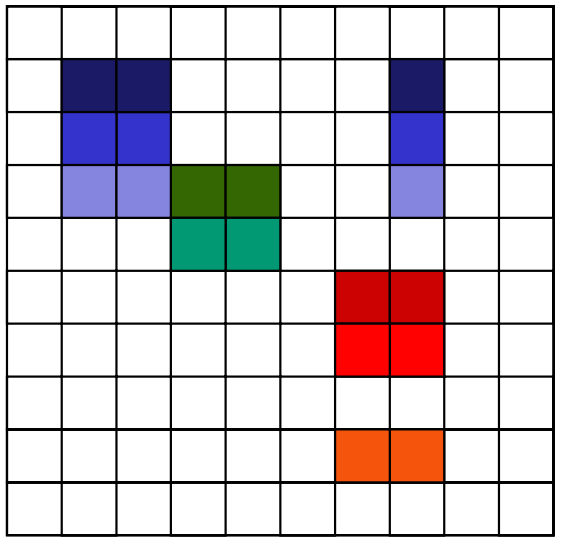
Challenges and Approaches, Basic Models

Biclusters with constant values on rows

- Cluster model for $\boldsymbol{A}_{IJ}$ = (I,J):

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

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

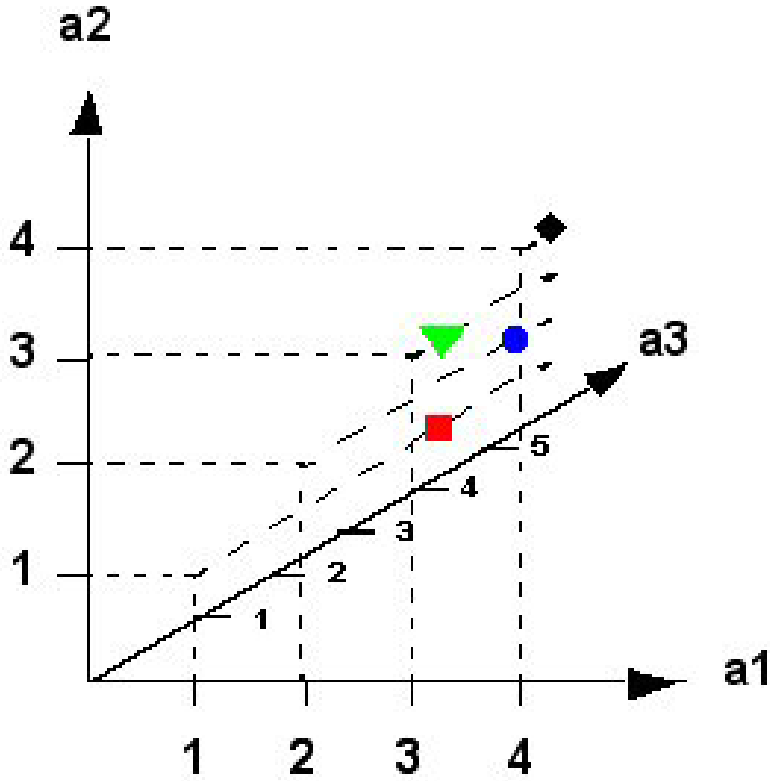


- adjustment value r_i for row $i \in I$

Challenges and Approaches, Basic Models

- 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

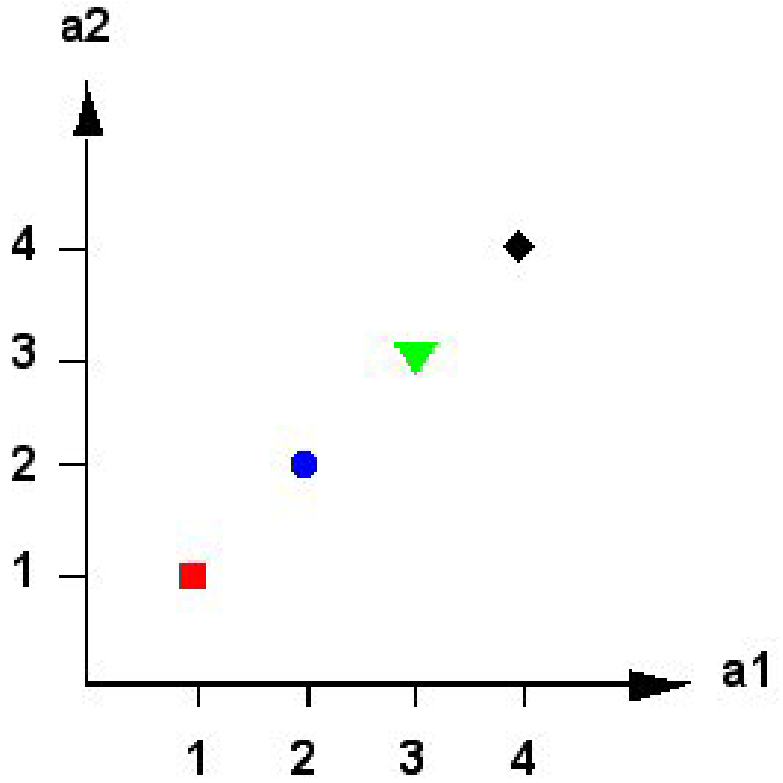


- in the embedding space, points build a sparse hyperplane parallel to irrelevant axes

Challenges and Approaches, Basic Models

- example – 2-dimensional subspace:

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

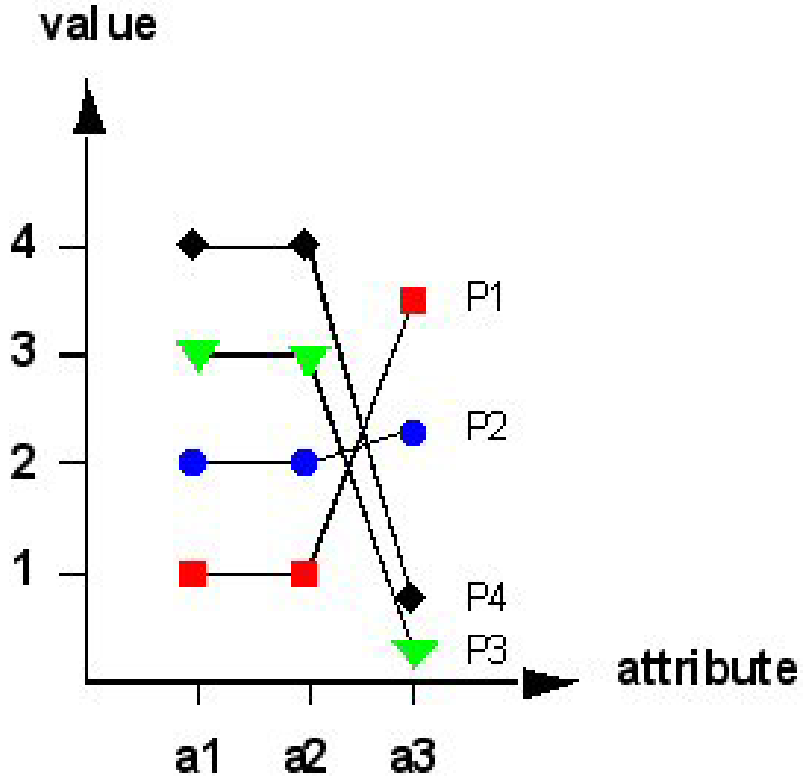


- points are accommodated on the bisecting line of participating attributes

Challenges and Approaches, Basic Models

- 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

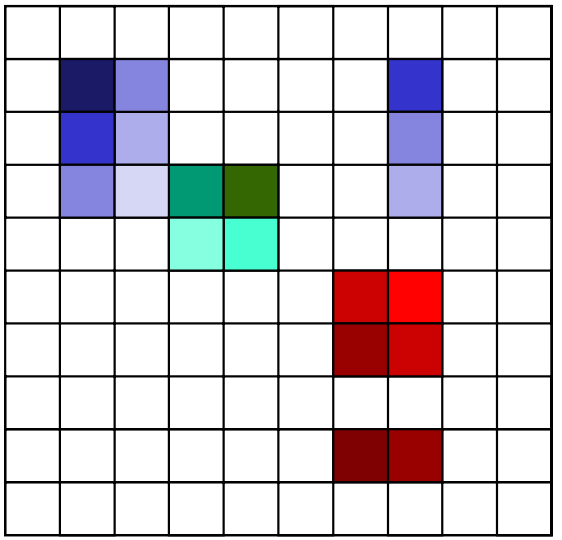
Challenges and Approaches, Basic Models

Biclusters with coherent values

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

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

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

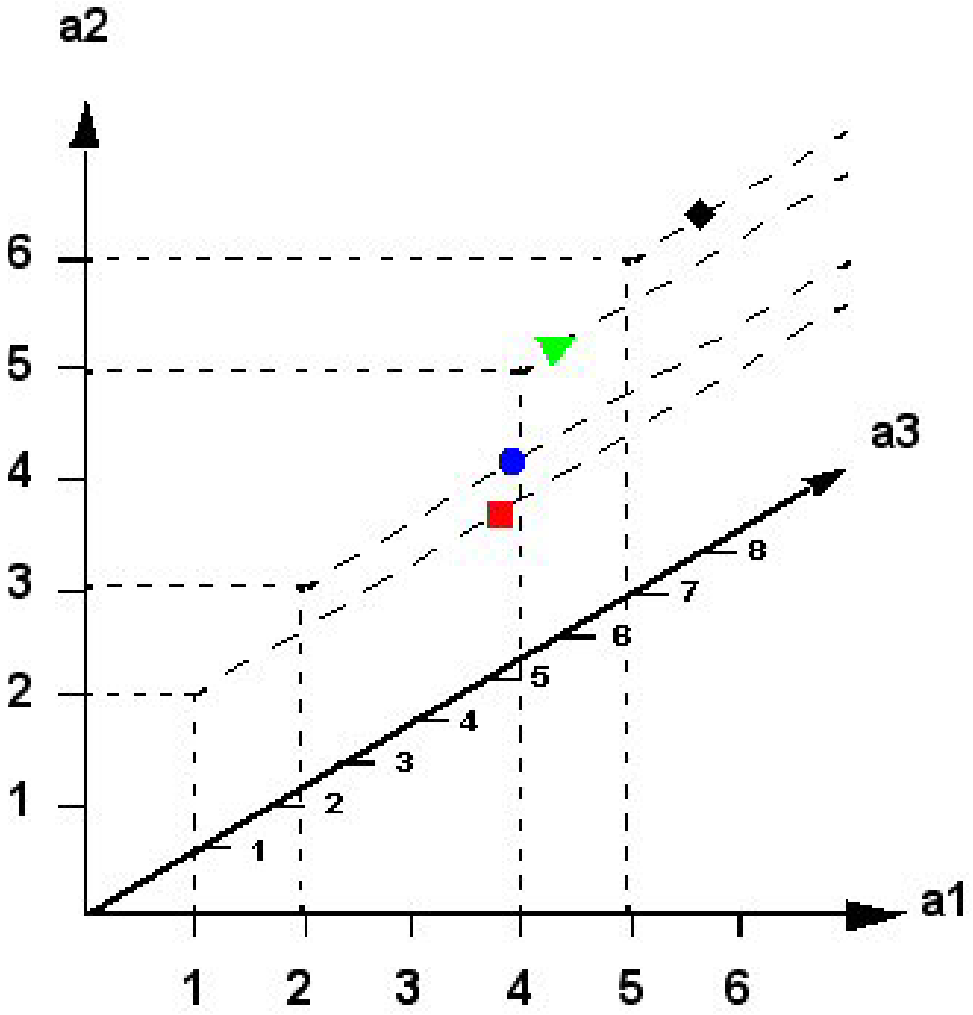


- special cases:
  - $c_j = 0$ for all j → constant values on rows
  - $r_i = 0$ for all i → constant values on columns

Challenges and Approaches, Basic Models

- embedding space: hyperplane parallel to axes of irrelevant attributes

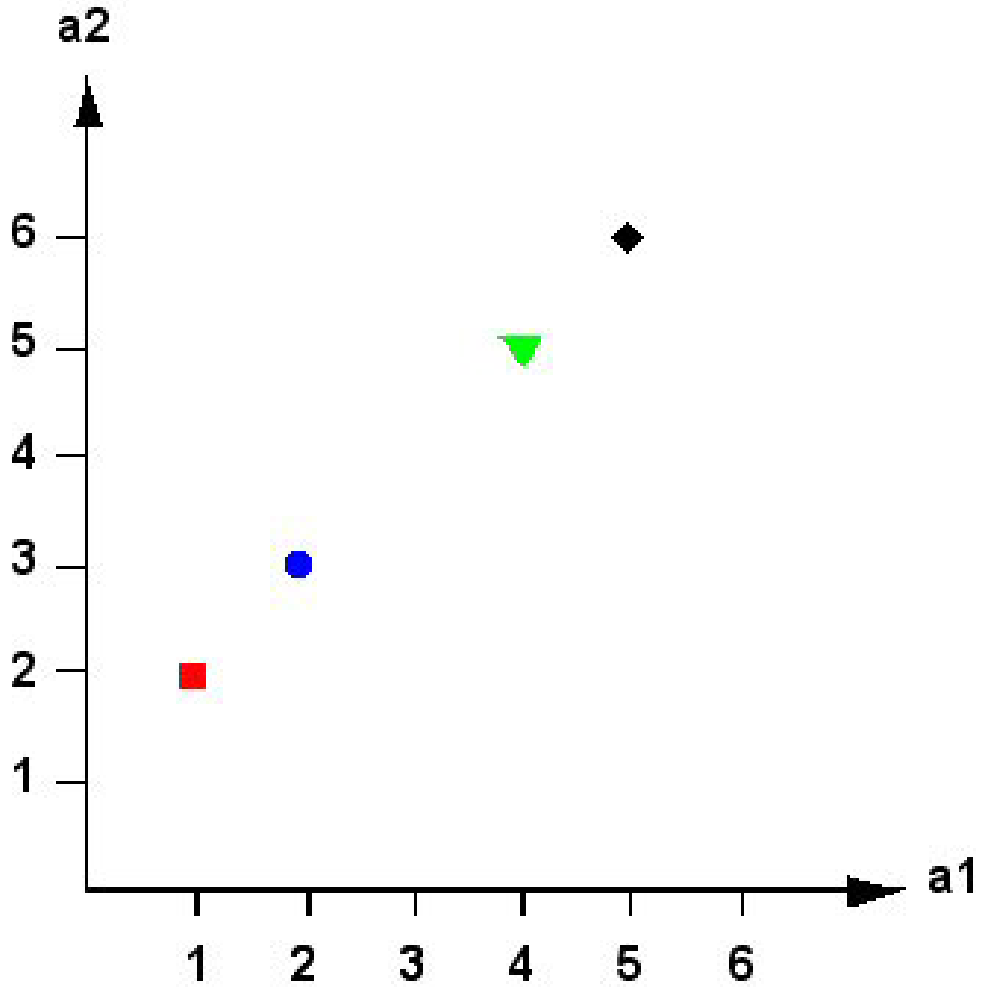
	a1	a2	a3
P1	1	2	3.5
P2	2	3	2.3
P3	4	5	0.2
P4	5	6	0.7



Challenges and Approaches, Basic Models

- subspace: increasing one-dimensional line

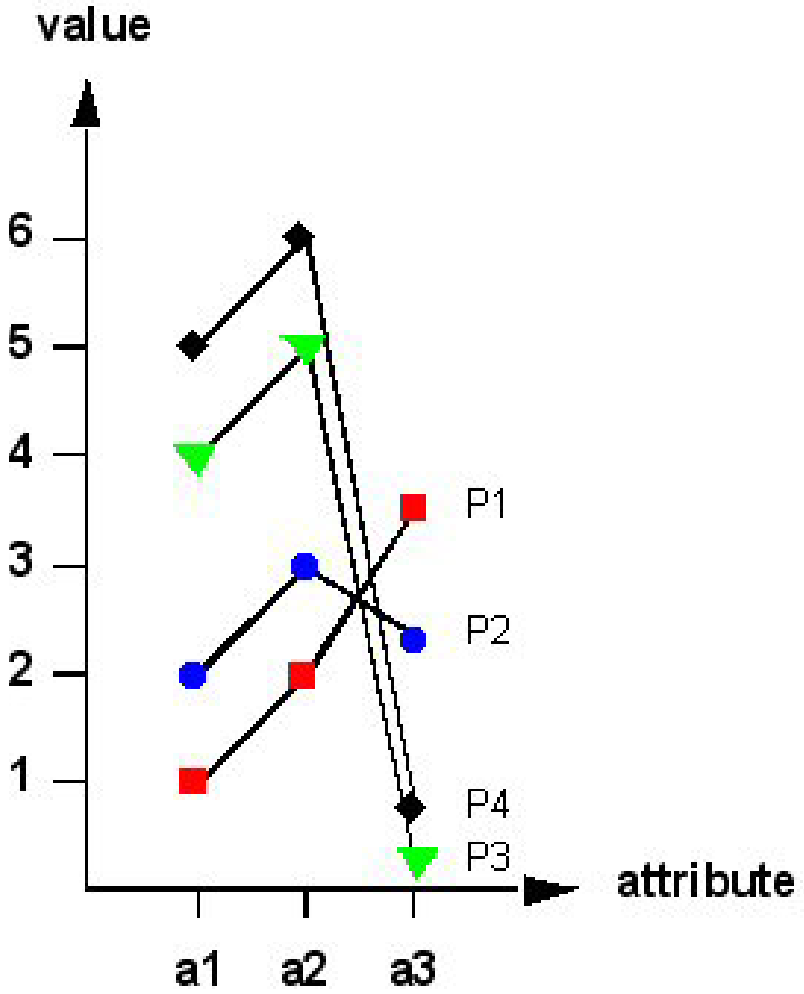
	a1	a2
P1	1	2
P2	2	3
P3	4	5
P4	5	6



Challenges and Approaches, Basic Models

- 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

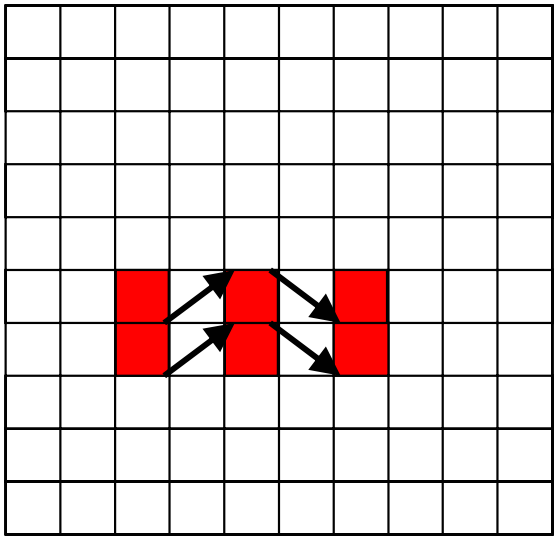


- pattern: parallel lines

Challenges and Approaches, Basic Models

Biclusters with coherent evolutions

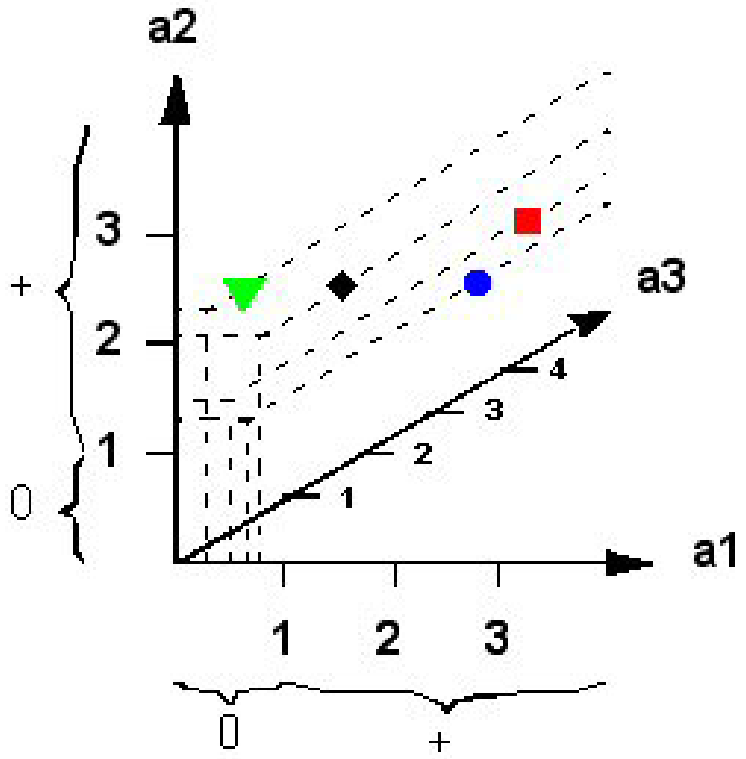
- for all rows, all pairs of attributes change simultaneously
  - discretized attribute space: coherent state-transitions
  - change in same direction irrespective of the quantity



Challenges and Approaches, Basic Models

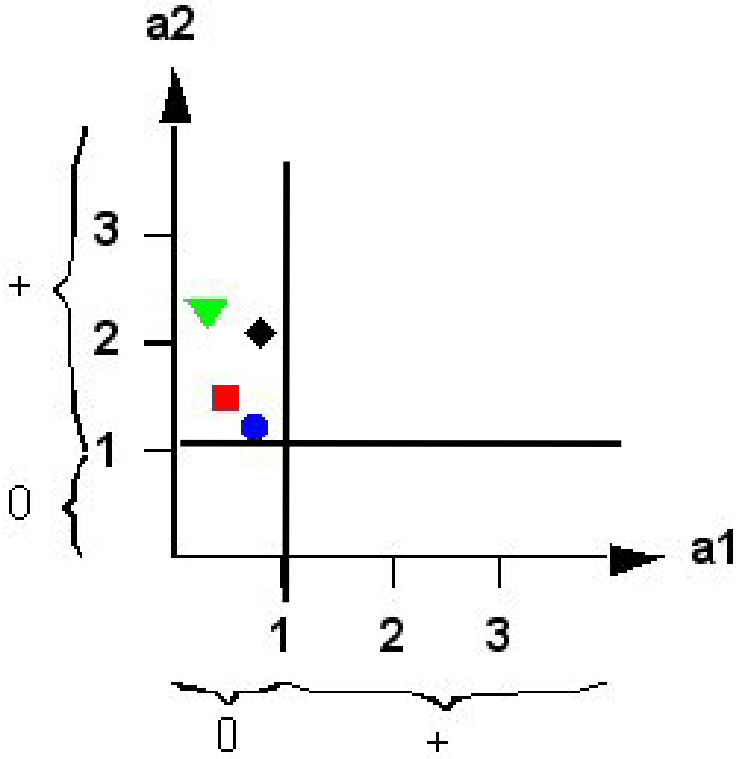
- Approaches with coherent state-transitions: [TSS02,MK03]
- reduces the problem to grid-based axis-parallel approach:

	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

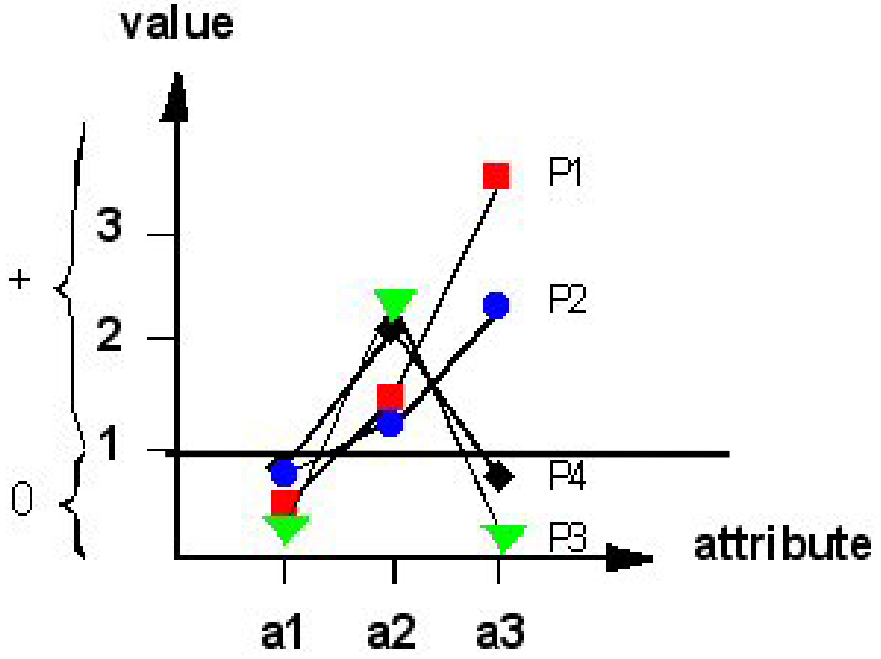


Challenges and Approaches, Basic Models

	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)

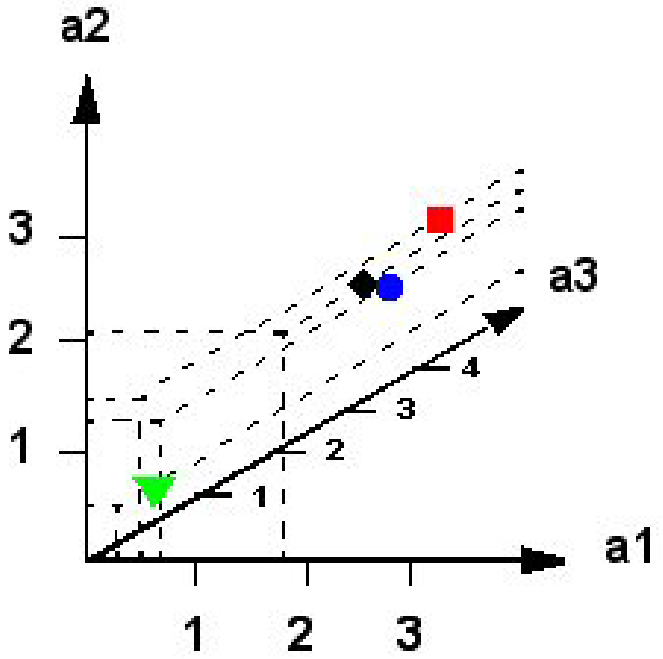
Challenges and Approaches, Basic Models

- 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

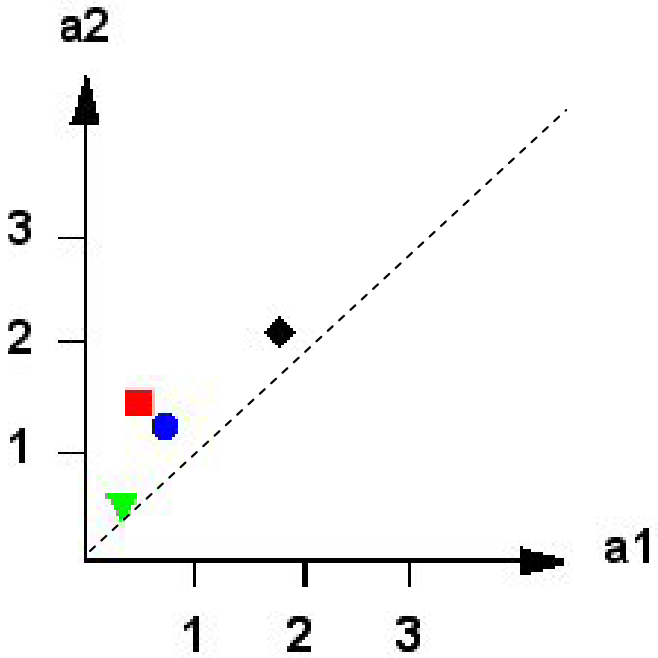
	a1	a2	a3
P1	0.5	1.5	3.5
P2	0.7	1.3	2.3
P3	0.3	0.5	0.2
P4	1.8	2.1	0.7



Challenges and Approaches, Basic Models

- example – 2-dimensional subspace

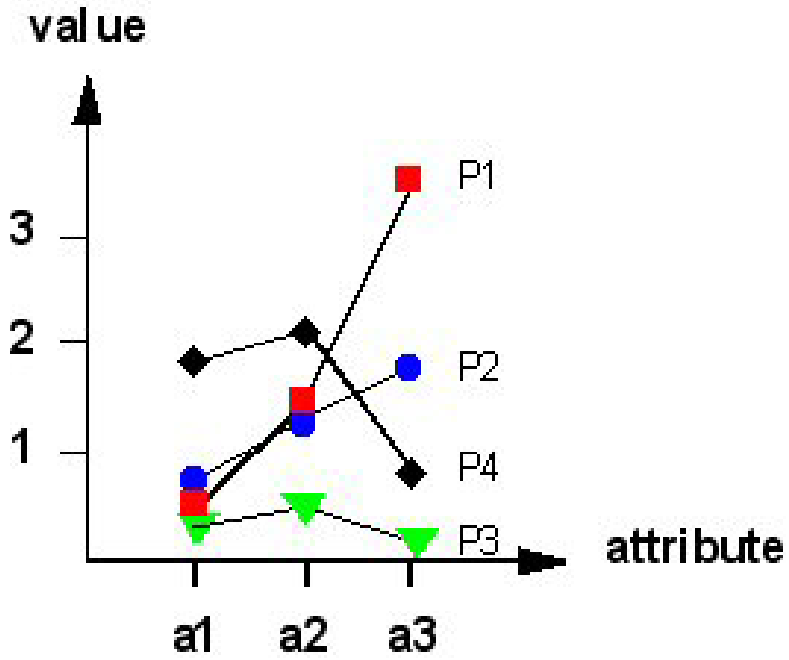
	a1	a2
P1	0.5	1.5
P2	0.7	1.3
P3	0.3	0.5
P4	1.8	2.1



Challenges and Approaches, Basic Models

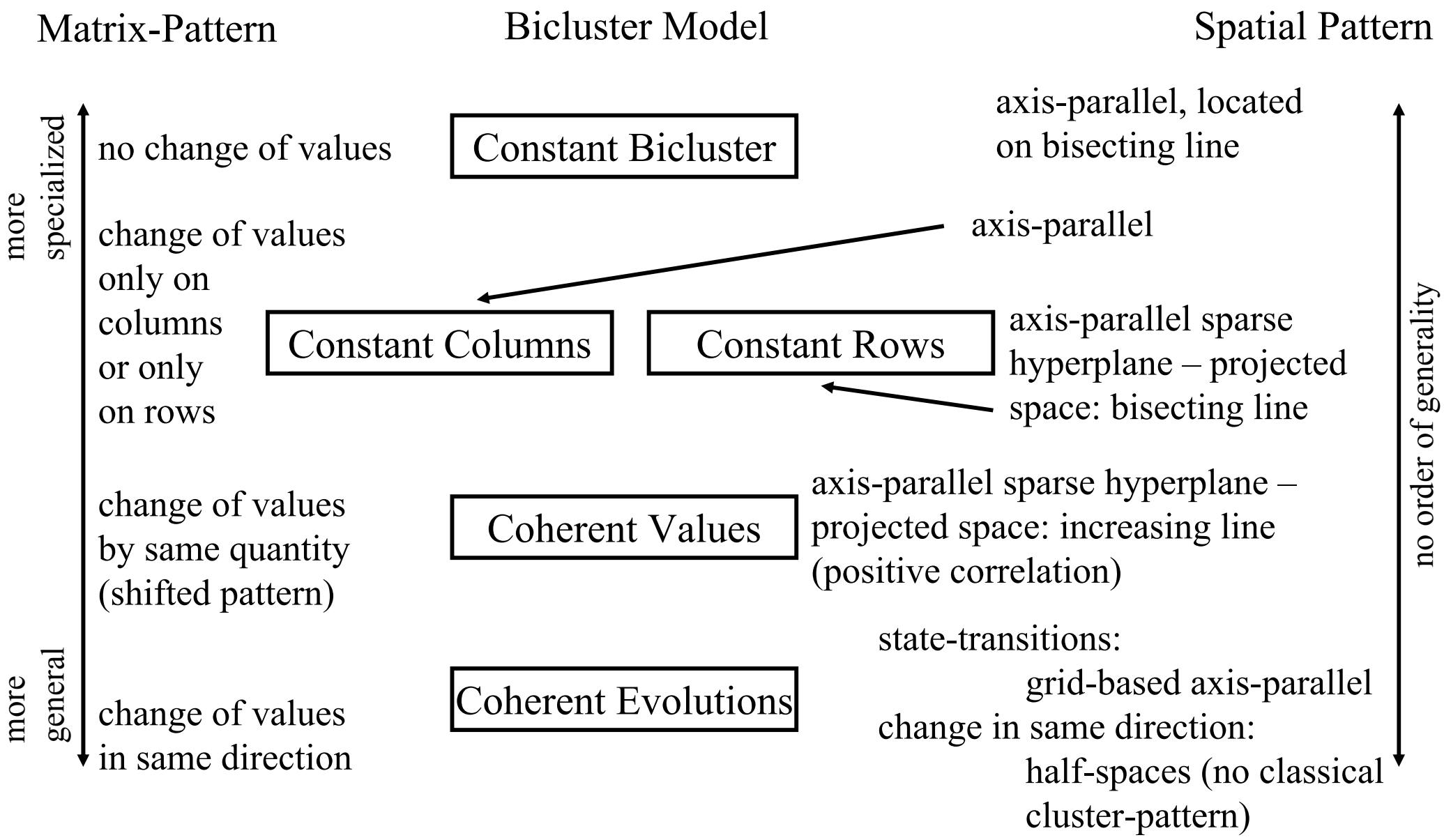
- example – transposed view of attributes

	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



Algorithms for Constant Biclusters

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

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

- avoids partitioning into |X|·|Y| singularity-biclusters (optimizing the sum of squares) by assumption of a fixed number *k* of clusters
- recursive split of data matrix into two partitions until *k* partitions exist overall
- each split chooses the maximal reduction in the overall sum of squares for all biclusters

Algorithms for 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

Pattern-based Clustering: 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

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

- δ =0 → *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:

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

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

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

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

- 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:

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

Pattern-based Clustering: 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)< δ.
  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.

Pattern-based Clustering: 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.

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

FLOC [YWWY02]

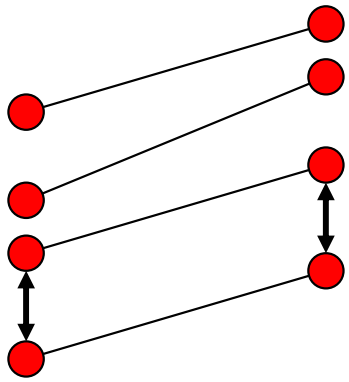
- randomized, move-based algorithm
- approximates k δ -biclusters based on minimization of the average residue
- initial seed clusters are optimized by random moves (removing or adding rows or columns)
- addressed issues:
  1. multiple clusters simultaneously
  4. allows overlapping rows and columns
  5. adapted model takes only specified values into account, allows for missing values

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

p-cluster model [WWYY02]

- FLOC pays improvements by introduction of random events.
- 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.

Pattern-based Clustering: 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: altogether, this is a complete enumeration approach

Addressed issues:

1. multiple clusters simultaneously
4. allows for overlapping rows and columns
6. allows for arbitrary number of clusters

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

related approach: MaPle [PZC+03]

- improves pruning based on the downward-closure property coming along with the δ -p-cluster model:
- for a δ -p-cluster (I,J), every submatrix (I',J') with I' $\subseteq$ I and J'$\subseteq$ J is a δ -p-cluster → allows for superset pruning
- addresses the same issues:
  1. multiple clusters simultaneously
  4. allows for overlapping rows and columns
  6. allows for arbitrary number of clusters
- Problem again: altogether, this is a complete enumeration approach

Pattern-based Clustering: Algorithms for Biclusters with Coherent Values

CoClus [CDGS04]

- marriage of a *k*-means-like approach with cluster models of Hartigan or Cheng&Church

- typical flaws of *k*-means-like approaches:
  - being caught in local minima
  - requires number of clusters beforehand
  - complete partition approach assumes data to contain no noise
  - every attribute is assumed to be relevant for exactly one cluster (contradiction to the prerequisites of high-dimensional data)

Biclusters with Coherent Evolution

OPSM (order preserving submatrix) [BDCKY02]

- submatrix (I,J) of data matrix ***A*** where a permutation π of J exists with

$$a_{i\pi(J)[m]} < a_{i\pi(J)[m+1]}$$

$$\forall i \in I,$$

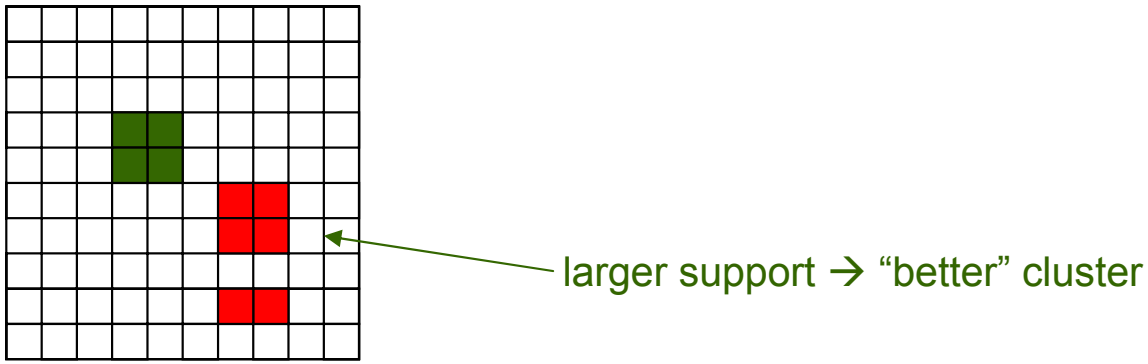
$$\forall \text{ index } m \text{ within } \pi(J),\ 1 \leq m < |J|$$

- given linear order of the columns: values in selected columns are strictly increasing
- cluster model: (J, π)
- support of the model: set I of rows fitting to the model

Biclusters with Coherent Evolution

Algorithm:

- greedy bottom-up approach to find the “best” cluster
- start with small models and iteratively extend the “best” *l* of these models
- “best” model: largest statistical significance (smallest prior probability)
- favors clusters with large support



Biclusters with Coherent Evolution

OP-cluster (order preserving cluster) [LW03]

- same general idea
- weaker conditions by introducing groups of similar attributes
- discard assessment of statistical significance, report all (maximal) submatrices (OP-clusters) covering at least a minimum number of rows and columns
- Algorithm:
  - create non-decreasing order of attributes (columns) for each row
  - group similar columns
  - mining for frequent patterns in resulting set of column-sequences

Biclusters with Coherent Evolution

general remarks concerning OPSM-model/OP-cluster

- no spatial intuition, points form half-spaces → related to quantitative association rules
- whether corresponding attributes are correlated remains unclear
- the model is not suitable to predict exact values but merely to exceed a given threshold (value of the attribute in the preceding column)
- BUT: interesting results in gene expression data reported

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].

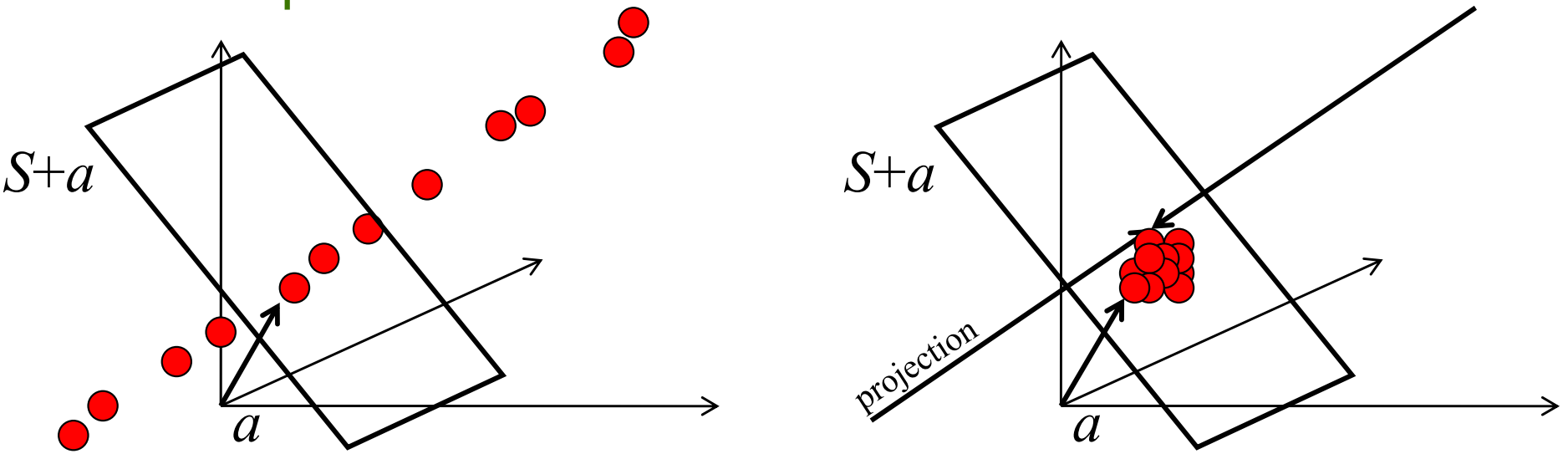
Outline

Outline: Arbitrarily-oriented Subspace Clustering

- Challenges and Approaches
- Correlation Clustering Algorithms
- Summary and Perspectives

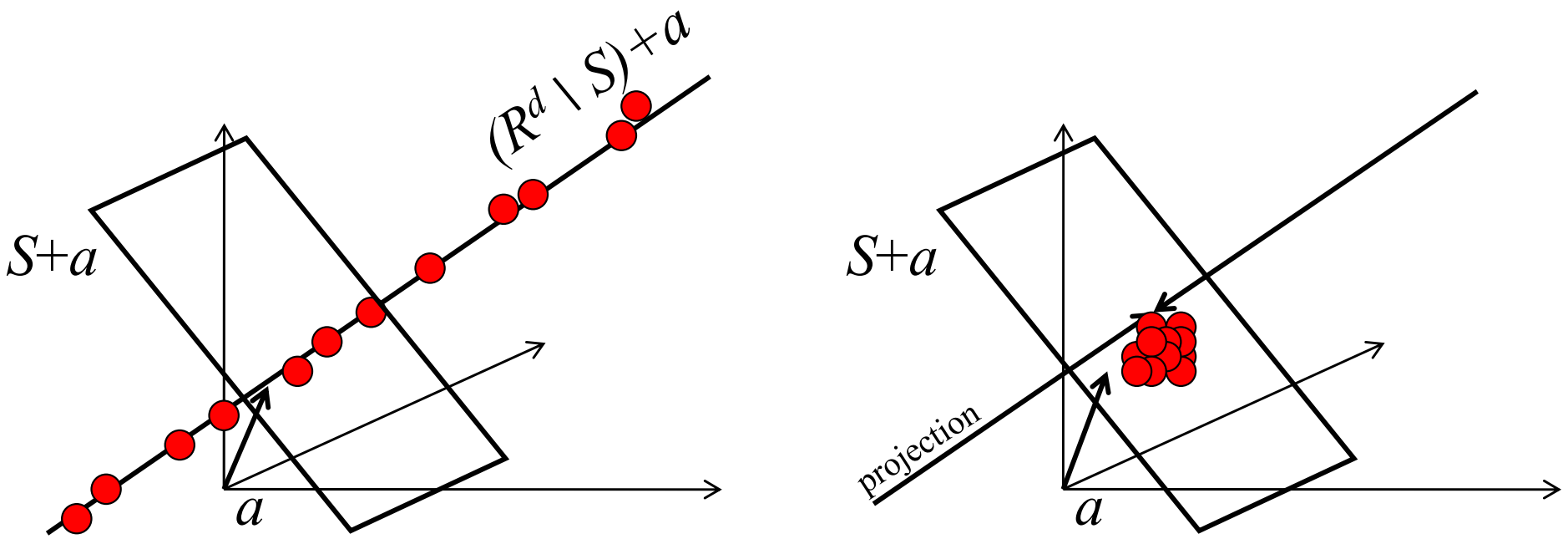
Challenges and Approaches

- Pattern-based approaches find simple positive correlations
- More general approach: oriented clustering aka. generalized subspace/projected clustering aka. correlation clustering
  - Note: different notion of “Correlation Clustering” in machine learning community, e.g. cf. [BBC04]
- Assumption: any cluster is located in an arbitrarily oriented affine subspace $S+a$ of R^d



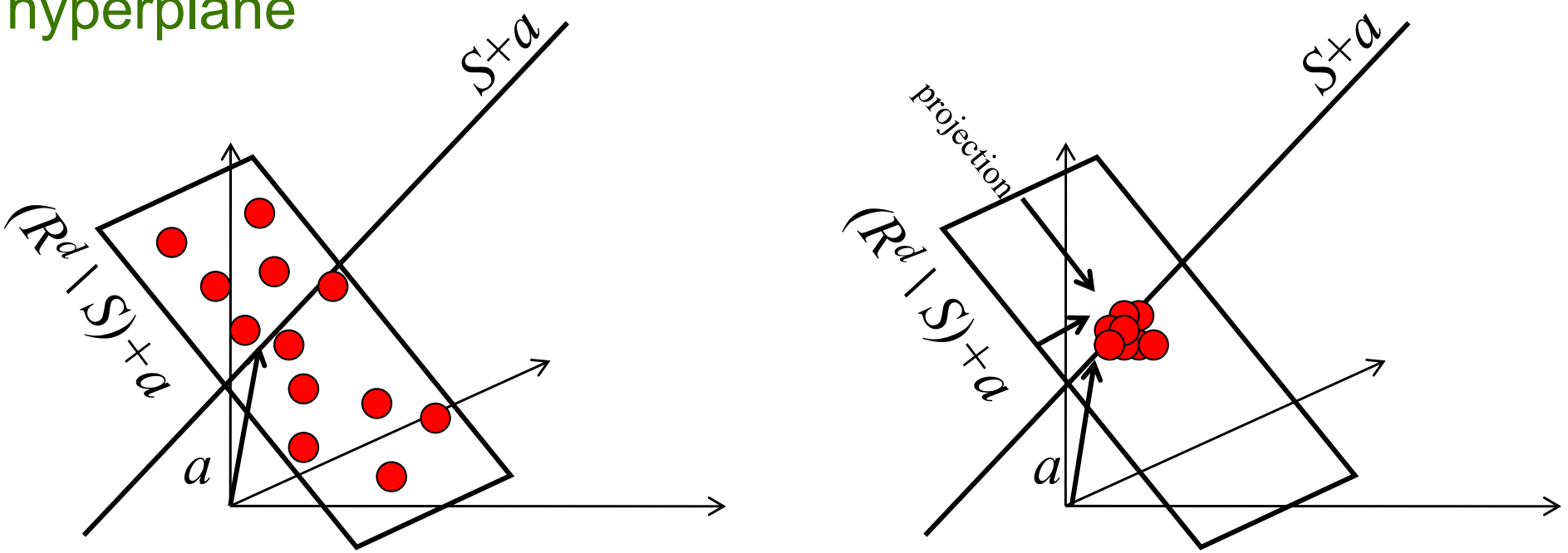
Challenges and Approaches

- Affine subspace $S+a$, $S \subset R^d$, *affinity* $a \in R^d$ is interesting if a set of points clusters within this subspace
- Points may exhibit high variance in perpendicular subspace $(R^d \setminus S)+a$



Challenges and Approaches

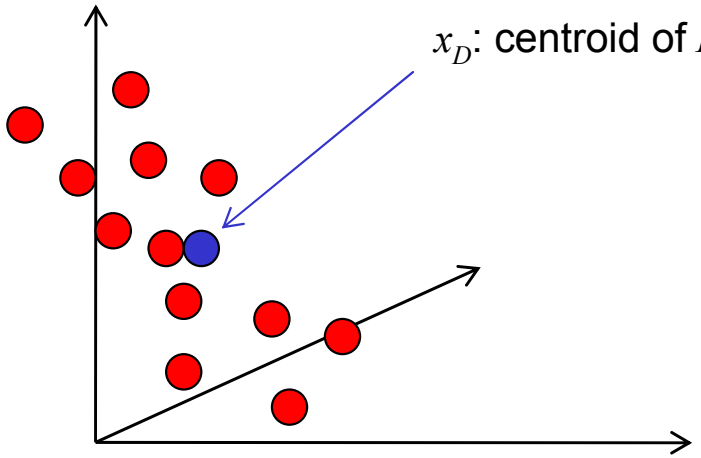
- high variance in perpendicular subspace $(R^d \setminus S)+a$ → points form a hyperplane within R^d located in this subspace $(R^d \setminus S)+a$
- Points on a hyperplane appear to follow linear dependencies among the attributes participating in the description of the hyperplane



Challenges and Approaches

- Directions of high/low variance: PCA (local application)
- locality assumption: local selection of points sufficiently reflects the hyperplane accommodating the points
- general approach: build covariance matrix Σ_D for a selection D of points (e.g. k nearest neighbors of a point)

$$\Sigma_D = \frac{1}{|D|} \sum_{x \in D} (x - x_D)(x - x_D)^T$$



properties of Σ_D:

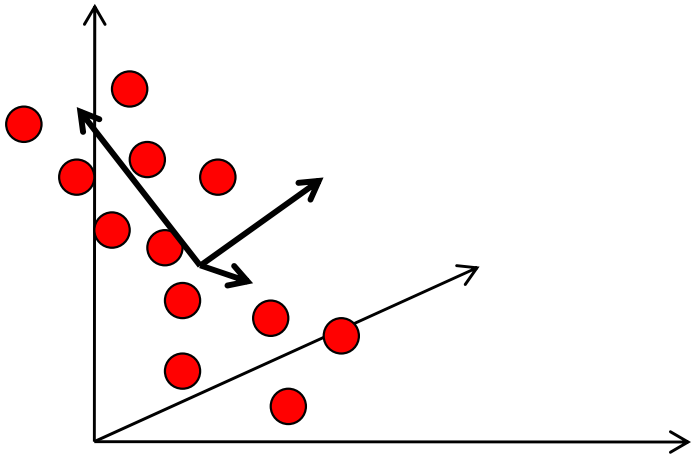
- d x d
- symmetric
- positive semidefinite
- $\sigma_{D_{ij}}$ (value at row i, column j) = covariance between dimensions i and j
- $\sigma_{D_{ii}}$ = variance in ith dimension

Challenges and Approaches

- decomposition of Σ_D to eigenvalue matrix E_D and eigenvector matrix V_D:

$$\Sigma_D = V_D E_D V_D^{\mathrm{T}}$$

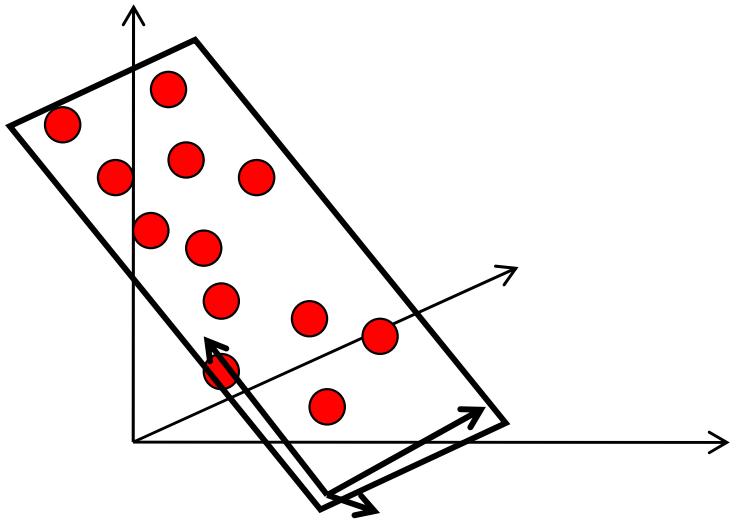
- E_D: diagonal matrix, holding eigenvalues of Σ_D in decreasing order in its diagonal elements
- V_D: orthonormal matrix with eigenvectors of Σ_D ordered correspondingly to the eigenvalues in E_D



- V_D: new basis, first eigenvector = direction of highest variance
- E_D: covariance matrix of D when represented in new axis system V_D

Challenges and Approaches

- points forming λ-dimensional hyperplane → hyperplane is spanned by the first λ eigenvectors (called "strong" eigenvectors – notation: $\breve{V}_D$)
- subspace where the points cluster densely is spanned by the remaining d-λ eigenvectors (called "weak" eigenvectors – notation: $\hat{V}_D$)



for the eigensystem, the sum of the smallest d-λ eigenvalues $\sum_{i=\lambda+1}^{d} e_{D_{ii}}$ is minimal under all possible transformations → points cluster optimally dense in this subspace

Challenges and Approaches

model for correlation clusters [ABK+06]:

- λ-dimensional hyperplane accommodating the points of a correlation cluster $C \subset R^d$ is defined by an equation system of d-λ equations for d variables and the affinity (e.g. the mean point x_C of all cluster members):

$$\hat{V}_C^{\mathrm{T}} x = \hat{V}_C^{\mathrm{T}} x_C$$

- equation system approximately fulfilled for all points $x \in C$
- quantitative model for the cluster allowing for probabilistic prediction (classification)
- Note: correlations are observable, linear dependencies are merely an assumption to explain the observations – predictive model allows for evaluation of assumptions and experimental refinements

Correlation Clustering Algorithms

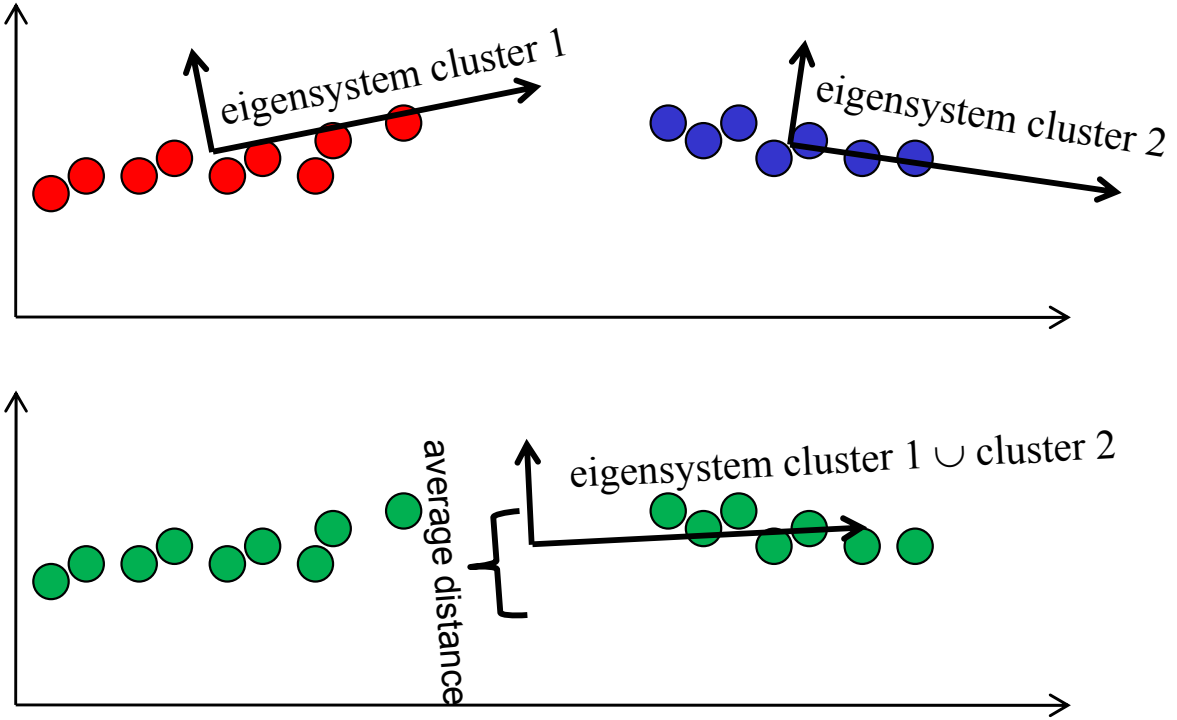
ORCLUS [AY00]:

first approach to *generalized projected clustering*

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

Correlation Clustering Algorithms

- best fitting pair of clusters: least average distance in the projected space spanned by weak eigenvectors of the merged clusters



- assess average distance in all merged pairs of clusters and finally merge the best fitting pair

Correlation Clustering Algorithms

- adapt eigensystem to the updated cluster
- new iteration: assign points according to updated eigensystems (distance along weak eigenvectors)
- dimensionality gradually reduced to a user-specified value l
- initially exclude only eigenvectors with very high variance

Correlation Clustering Algorithms

properties:

- finds k correlation clusters (user-specified)
- higher initial k_c → higher runtime, probably better results
- biased to average dimensionality l of correlation clusters (user specified)
- cluster-based locality assumption: subspace of each cluster is learned from its current members (starting in the full dimensional space)

Correlation Clustering Algorithms

4C [BKKZ04]

- density-based cluster-paradigma (cf. DBSCAN [EKSX96])
- extend a cluster from a seed as long as a density-criterion is fulfilled – otherwise pick another seed unless all data base objects are assigned to a cluster or noise
- density criterion: minimal required number of points in the neighborhood of a point
- neighborhood: distance between two points ascertained based on the eigensystems of both compared points

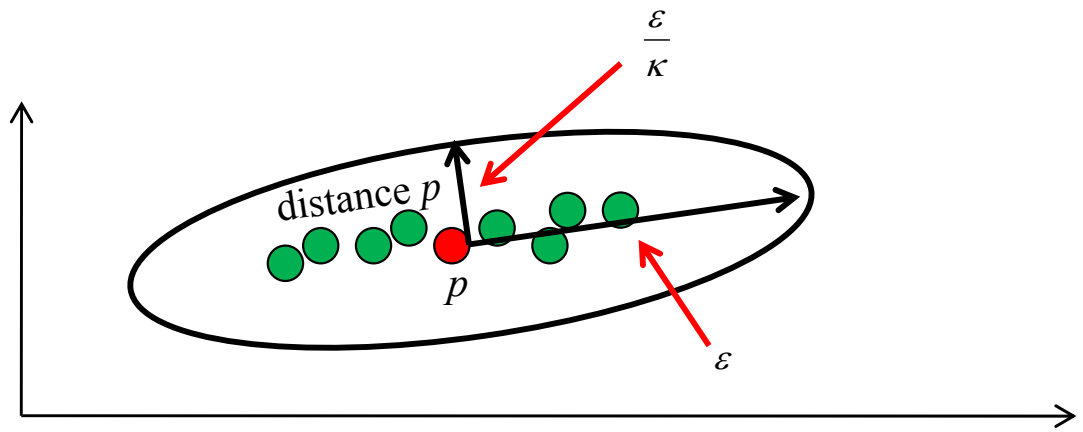
Correlation Clustering Algorithms

- eigensystem of a point p based on its ε-neighborhood in Euclidean space
- threshold δ discerns large from small eigenvalues
- in eigenvalue matrix E_p replace large eigenvalues by 1, small eigenvalues by κ>>1
- adapted eigenvalue matrix yields a correlation similarity matrix for point p:

$$V_p E'_p V_p^{\mathrm{T}}$$

Correlation Clustering Algorithms

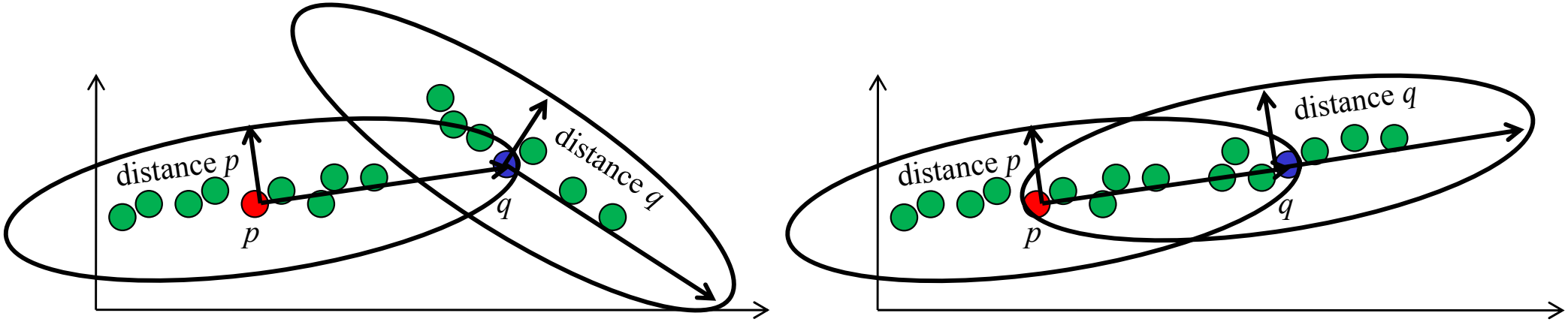
- effect on distance measure:



- 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: $\sqrt{(q-p)\cdot V_q \cdot E'_q \cdot V_q^{\mathrm{T}} \cdot (q-p)^{\mathrm{T}}}$

Correlation Clustering Algorithms

- symmetry of distance measure by choosing the maximum:



- p and q are correlation-neighbors if

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

Correlation Clustering Algorithms

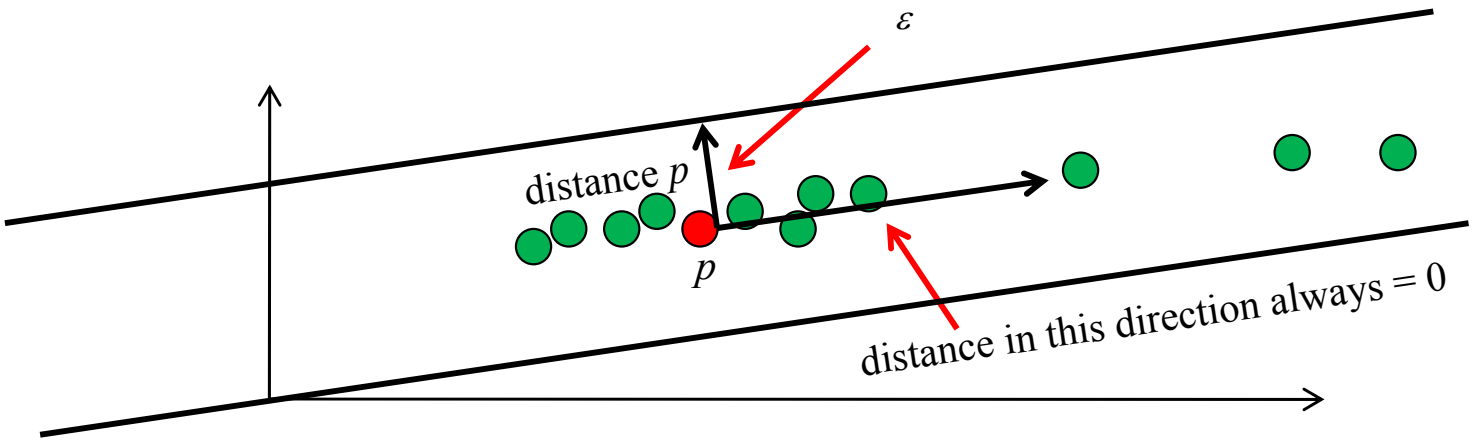
properties:

- 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

Correlation Clustering Algorithms

COPAC [ABK+07c] – similar ideas as 4C but:

- use k nearest neighbors instead of ε-neighborhood
  - $k > \lambda$ ensures meaningful definition of a λ-dimensional hyperplane
- in eigenvalue matrix E_p replace large eigenvalues by 0, small eigenvalues by 1 – effect on distance measure:



Correlation Clustering Algorithms

- main point: considerable speed up by partitioning the database according to the local correlation dimensionality
  (= number of strong eigenvectors in eigensystem based on k nearest neighbors)
- only points with common local correlation dimensionality can build a correlation cluster
- no need to check distances between points of different local correlation dimensionality
- average speed up compared to 4C: getting rid of a squared factor d^2 in a d-dimensional dataset

Correlation Clustering Algorithms

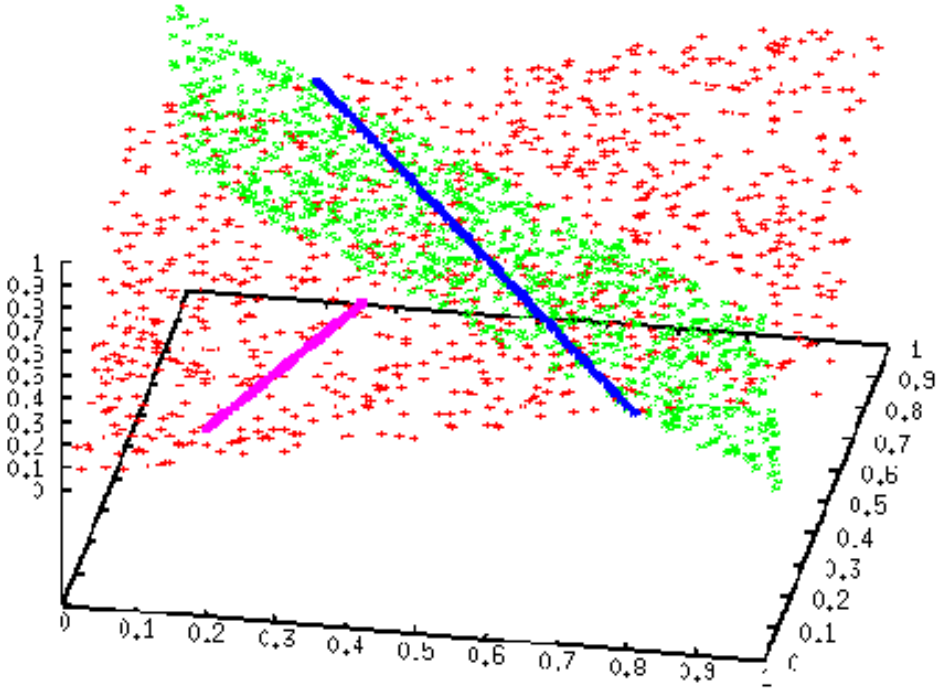
properties:

- finds arbitrary number of clusters
- requires specification of density-thresholds
  - μ (minimum number of points)
  - ε (acceptable deviation from correlation-hyperplane)
  - k (number of nearest neighbors to define local eigensystem)
- instance-based locality assumption: correlation distance measure specifying the subspace is learned from local neighborhood of each point in the d-dimensional space

Correlation Clustering Algorithms

ERiC [ABK+07b]

- hierarchical approach: correlation clusters of lower correlation dimensionality may be embedded in correlation clusters of higher correlation dimensionality



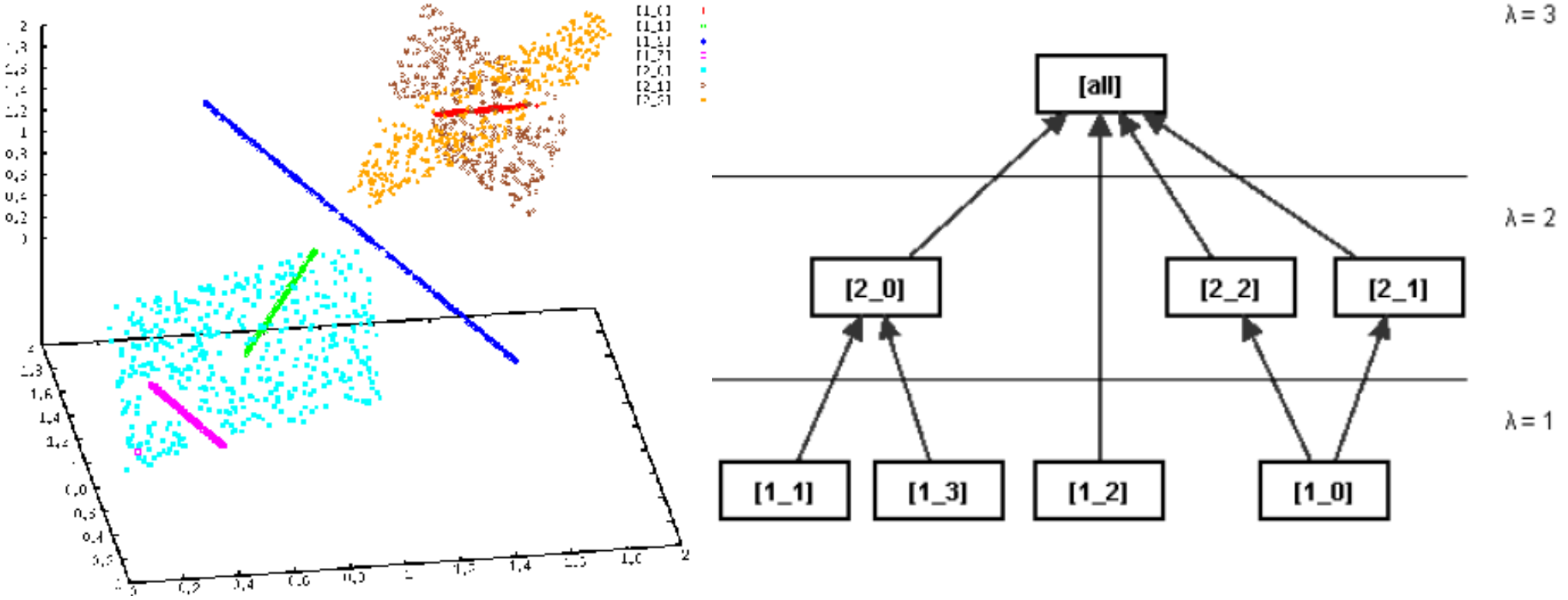
Correlation Clustering Algorithms

- derive eigensystem for each point based on k nearest neighbors
- distance between points:
  - 0 if strong eigenvectors of both eigensystems define identical affine subspace (concept of approximate linear dependency)
  - 1 otherwise
- cluster points with DBSCAN based on this distance measure with $\varepsilon = 0$
- partitioning as by COPAC

Correlation Clustering Algorithms

properties:

- provides containment hierarchy (multiple inheritance) of correlation clusters



Correlation Clustering Algorithms

properties:

- finds arbitrary number of clusters
- requires specification of density-threshold μ (minimum number of points)
- acceptable deviation δ of approximately linear dependent subspaces
- instance-based locality assumption: correlation distance measure specifying the subspace is learned from local neighborhood of each point in the d-dimensional space
- COPAC and ERiC improve over ORCLUS and 4C especially w.r.t. complex patterns of intersecting correlation clusters and w.r.t. clusters of different correlation dimensionalities

Correlation Clustering Algorithms

glance at non-linear correlation clustering: CURLER [TXO05]

- finds non-linear correlation clusters by merging micro-clusters
- not restricted to correlations of attributes but finds any narrow trajectory
- no model describing the results
- relies on the locality assumption
- first and only approach to non-linear correlation clustering

Summary and Perspectives

- 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

Summary and Perspectives

- some preliminary approaches base on concept of self-similarity (intrinsic dimensionality, fractal dimension): [BC00,PTTF02,GHPT05]
- interesting idea, provides quite a different basis to grasp correlations in addition to PCA
- drawback: self-similarity assumes locality of patterns even by definition

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

Outline

1. Introduction
2. Axis-parallel Subspace Clustering
3. Pattern-based Clustering
4. Arbitrarily-oriented Subspace Clustering
5. Summary

Summary

- Let's take a global view:
  - Traditional clustering in high dimensional spaces is most likely meaningless with increasing dimensionality (curse of dimensionality)
  - Clusters may be found in (generally arbitrarily oriented) subspaces of the data space
  - So the general problem of clustering high dimensional data is:

    "find a partitioning of the data where each cluster may exist in its own subspace"
    - The partitioning need not be unique (clusters may overlap)
    - The subspaces may be axis-parallel or arbitrarily oriented
  - Analysis of this general problem:
    - A naïve solution would examine all possible subspaces to look for clusters
    - The search space of all possible arbitrarily oriented subspaces is infinite
    - We need assumptions and heuristics to develop a feasible solution

Summary

- What assumptions did we get to know here?
  - The search space is restricted to certain subspaces
  - A clustering criterion that implements the downward closure property enables efficient search heuristics
  - The locality assumption enables efficient search heuristics
  - Assuming simple additive models (“patterns”) enables efficient search heuristics
  - …
- Remember: also the clustering model may rely on further assumptions that have nothing to do with the infinite search space
  - Number of clusters need to be specified
  - Results are not deterministic e.g. due to randomized procedures
  - …
- We can classify the existing approaches according to the assumptions they made to conquer the infinite search space

Summary

- The global view
  - Subspace clustering/projected clustering:
    - The search space is restricted to axis-parallel subspaces
    - A clustering criterion that implements the downward closure property is defined (usually based a global density threshold)
    - The locality assumption enables efficient search heuristics
  - Bi-clustering/pattern-based clustering:
    - The search space is restricted to special forms and locations of subspaces or half-spaces
    - Over-optimization (e.g. singularity clusters) is avoided by assuming a predefined number of clusters
  - Correlation clustering:
    - The locality assumption enables efficient search heuristics
- Any of the proposed methods is based on at least one assumption because otherwise, it would not be applicable

Summary

Algorithm	complex correlations	simple positive correlation	simple negative correlation	axis parallel	not relying on locality assumption	independent w.r.t. order of attributes	independent w.r.t. order of objects	deterministic	arbitrary number of clusters	overlapping clusters	overlapping subspaces	simultaneously overlapping clusters and subspaces	hierarchical structure	avoiding complete enumeration	noise robust
CLIQUE [AGGR98]				×	×	×	×	×	×	×	×	×			×
ENCLUS [CFZ99]				×	×	×	×	×	×	×	×	×			×
MAFIA [NGC01]				×	×	×	×	×	×	×	×	×			×
SUBCLU [KKK04]				×	×	×	×	×	×	×	×	×			×
P3C [MSE06]				×	×	×	×	×	×		×				×
COSA [FM04]				×		×	×	×			×			×	×
PROCLUS [APW+99]				×							×			×	
DOC [PJAM02]				×	×	×	×		×	×	×	×			
PreDeCon [BKKK04]				×		×	×	×	×		×			×	×
DiSH [ABK+07a]				×		×	×	×	×		×		×	×	×
FIRES [KKRW05]				×	×		×	×	×	×	×	×			×

Summary

Algorithm	complex correlations	simple positive correlation	simple negative correlation	axis parallel	not relying on locality assumption	independent w.r.t. order of attributes	independent w.r.t. order of objects	deterministic	arbitrary number of clusters	overlapping clusters	overlapping subspaces	simultaneously overlapping clusters and subspaces	hierarchical structure	avoiding complete enumeration	noise robust
Block clustering [Har72]					×	×	×	×					×		×
δ-bicluster [CC00]		×		×	×	×	×	×		×	×			×	×
FLOC [YWWY02]		×		×	×					×	×	×		×	×
p-Cluster [WWYY02]		×		×	×	×	×	×	×	×	×	×			×
MaPle [PZC+03]		×		×	×	×	×	×	×	×	×	×			×
CoClus [CDGS04]		×		×	×									×	
OP-Cluster [LW03]					×	×	×	×	×	×	×	×			×

Summary

Algorithm	complex correlations	simple positive correlation	simple negative correlation	axis parallel	not relying on locality assumption	independent w.r.t. order of attributes	independent w.r.t. order of objects	deterministic	arbitrary number of clusters	overlapping clusters	overlapping subspaces	simultaneously overlapping clusters and subspaces	hierarchical structure	avoiding complete enumeration	noise robust
ORCLUS [AY00]	×	×	×	×		×					×			×	
4C [BKKZ04]	×	×	×	×		×	×	×	×		×			×	×
COPAC [ABK+07c]	×	×	×	×		×	×	×	×		×			×	×
ERiC [ABK+07b]	×	×	×	×		×	×	×	×		×		×	×	×

Evaluation

- How can we evaluate which assumption is better under which conditions?
  - Basically there is no comprehensive comparison on the accuracy or efficiency of the discussed methods
  - A fair comparison on the efficiency is only possible in sight of the assumptions and heuristics used by the single methods
  - An algorithm performs bad if it has more restrictions AND needs more time
  - Being less efficient but more general should be acceptable

Evaluation

- What we find in the papers is
  - Head-to-head comparison with at most one or two competitors that do have similar assumptions
  - But that can be really misleading!!!
  - Sometimes there is even no comparison at all to other approaches
  - Sometimes the experimental evaluations are rather poor
- So how can we decide which algorithm to use for a given problem?
  - Actually, we cannot ☹
  - However, we can sketch what makes a sound evaluation

Evaluation

- How should a sound experimental evaluation of the accuracy look like – an example using gene expression data

  [Thanks to the anonymous reviewers for their suggestions even though we would have preferred an ACCEPT ;-)]

  - Good:
    - Apply your method to cluster the genes of a publicly available gene expression data set => you should get clusters of genes with similar functions
    - Do not only report that your method has found some clusters (because even e.g. the full-dimensional *k*-means would have done so)
    - Analyze your clusters: do the genes have similar functions?
      - Sure, we are computer scientists, not biologists, but …
      - In publicly available databases you can find annotations for (even most of) the genes
      - These annotations can be used as class labels, so consistency measures can be computed

Evaluation

- Even better
  - Identify competing methods (that have similar assumptions like your approach)
  - Run the same experiments (see above) with the competing approaches
  - Your method is very valuable if
    - your clusters have a higher consistency score

      [OK, you are the winner]

    OR

    - your clusters have a lower (but still reasonably high) score and represent functional groups of genes that clearly differ from that found by the competitors

      [you can obviously find other biologically relevant facts that could not be found by your competitors]
  - Open question: what is a suitable consistency score for subspace clusters?

Evaluation

- Premium
  - You have a domain expert as partner who can analyze your clustering results in order to
    - Prove and/or refine his/her existing hypothesis
    - Derive new hypotheses

  Lucky you – that's why we should make data mining ☺

List of References

Literature

[ABK+06] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek.
Deriving quantitative models for correlation clusters.
In Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA, 2006.

[ABK+07a] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek.
Detection and visualization of subspace cluster hierarchies.
In Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA), Bangkok, Thailand, 2007.

[ABK+07b] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek.
On exploring complex relationships of correlation clusters.
In Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada, 2007.

[ABK+07c] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek.
Robust, complete, and efficient correlation clustering.
In Proceedings of the 7th SIAM International Conference on Data Mining (SDM), Minneapolis, MN, 2007.

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.

[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.

[AS94] R. Agrawal and R. Srikant.
Fast algorithms for mining association rules.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Minneapolis, MN, 1994.

[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.

[BBC04] N. Bansal, A. Blum, and S. Chawla.
Correlation clustering.
Machine Learning, 56:89–113, 2004.

Literature

[BC00] D. Barbara and P. Chen.
Using the fractal dimension to cluster datasets.
In Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Boston, MA, 2000.

[BDCKY02] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini.
Discovering local structure in gene expression data: The order-preserving submatrix problem.
In Proceedings of the 6th Annual International Conference on Computational Molecular Biology (RECOMB), Washington, D.C., 2002.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft.
When is "nearest neighbor" meaningful?
In Proceedings of the 7th International Conference on Database Theory (ICDT), Jerusalem, Israel, 1999.

[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.

Literature

[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.

[CC00] Y. Cheng and G. M. Church.
Biclustering of expression data.
In Proceedings of the 8th International Conference Intelligent Systems for Molecular Biology (ISMB), San Diego, CA, 2000.

[CDGS04] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra.
Minimum sum-squared residue co-clustering of gene expression data.
In Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL, 2004.

[CFZ99] C. H. Cheng, A. W.-C. Fu, and Y. Zhang.
Entropy-based subspace clustering for mining numerical data.
In Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA, pages 84–93, 1999.

Literature

[CST00] A. Califano, G. Stolovitzky, and Y. Tu.
Analysis of gene expression microarrays for phenotype classification.
In Proceedings of the 8th International Conference Intelligent Systems for Molecular Biology (ISMB), San Diego, CA, 2000.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu.
A density-based algorithm for discovering clusters in large spatial databases with noise.
In Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, 1996.

[FM04] J. H. Friedman and J. J. Meulman.
Clustering objects on subsets of attributes.
Journal of the Royal Statistical Society: Series B (Statistical Methodology), 66(4):825–849, 2004.

[GHPT05] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas.
Dimension induced clustering.
In Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL, 2005.

Literature

[GLD00] G. Getz, E. Levine, and E. Domany.
Coupled two-way clustering analysis of gene microarray data.
Proceedings of the National Academy of Sciences of the United States of America, 97(22):12079–12084, 2000.

[GRRK05] E. Georgii, L. Richter, U. Rückert, and S. Kramer.
Analyzing microarray data using quantitative association rules.
Bioinformatics, 21(Suppl. 2):ii1–ii8, 2005.

[GW99] B. Ganter and R. Wille.
Formal Concept Analysis.
Mathematical Foundations. Springer, 1999.

[HAK00] A. Hinneburg, C. C. Aggarwal, and D. A. Keim.
What is the nearest neighbor in high dimensional spaces?
In Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt, 2000.

[Har72] J. A. Hartigan.
Direct clustering of a data matrix.
Journal of the American Statistical Association, 67(337):123–129, 1972.

Literature

[IBB04] J. Ihmels, S. Bergmann, and N. Barkai.
Defining transcription modules using large-scale gene expression data.
Bioinformatics, 20(13):1993–2003, 2004.

[Jol02] I. T. Jolliffe.
Principal Component Analysis.
Springer, 2nd edition, 2002.

[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.

[KKRW05] H.-P. Kriegel, P. Kröger, M. Renz, and S. Wurst.
A generic framework for efficient subspace clustering of high-dimensional data.
In Proceedings of the 5th International Conference on Data Mining (ICDM), Houston, TX, 2005.

[LW03] J. Liu and W. Wang.
OP-Cluster: Clustering by tendency in high dimensional spaces.
In Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL, 2003.

Literature

[MK03] T. M. Murali and S. Kasif.
Extracting conserved gene expression motifs from gene expression data.
In Proceedings of the 8th Pacific Symposium on Biocomputing (PSB), Maui, HI, 2003.

[MO04] S. C. Madeira and A. L. Oliveira.
Biclustering algorithms for biological data analysis: A survey.
IEEE Transactions on Computational Biology and Bioinformatics, 1(1):24–45, 2004.

[MSE06] G. Moise, J. Sander, and M. Ester.
P3C: A robust projected clustering algorithm.
In Proceedings of the 6th International Conference on Data Mining (ICDM), Hong Kong, China, 2006.

[NGC01] H.S. Nagesh, S. Goil, and A. Choudhary.
Adaptive grids for clustering massive data sets.
In Proceedings of the 1st SIAM International Conference on Data Mining (SDM), Chicago, IL, 2001.

[PBZ+06] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Guissem, L. Hennig, L. Thiele, and E. Zitzler.
A systematic comparison and evaluation of biclustering methods for gene expression data.
Bioinformatics, 22(9):1122–1129, 2006.

Literature

[Pfa07] J. Pfaltz.
What constitutes a scientific database?
In Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada, 2007.

[PHL04] L. Parsons, E. Haque, and H. Liu.
Subspace clustering for high dimensional data: A review.
SIGKDD Explorations, 6(1):90–105, 2004.

[PJAM02] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali.
A Monte Carlo algorithm for fast projective clustering.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI, 2002.

[PTTF02] E. Parros Machado de Sousa, C. Traina, A. Traina, and C. Faloutsos.
How to use fractal dimension to find correlations between attributes.
In Proc. KDD-Workshop on Fractals and Self-similarity in Data Mining: Issues and Approaches, 2002.

[PZC+03] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu.
MaPle: A fast algorithm for maximal pattern-based clustering.
In Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL, 2003.

Literature

[RRK04] U. Rückert, L. Richter, and S. Kramer.
Quantitative association rules based on half-spaces: an optimization approach.
In Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K., pages 507–510, 2004.

[SMD03] Q. Sheng, Y. Moreau, and B. De Moor.
Biclustering microarray data by Gibbs sampling.
Bioinformatics, 19(Suppl. 2):ii196–ii205, 2003.

[STG+01] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller.
Rich probabilistic models for gene expression.
Bioinformatics, 17(Suppl. 1):S243–S252, 2001.

[SZ05] K. Sequeira and M. J. Zaki.
SCHISM: a new approach to interesting subspace mining.
International Journal of Business Intelligence and Data Mining, 1(2):137–160, 2005.

[TSS02] A. Tanay, R. Sharan, and R. Shamir.
Discovering statistically significant biclusters in gene expression data.
Bioinformatics, 18 (Suppl. 1):S136–S144, 2002.

Literature

[TXO05] A. K. H. Tung, X. Xu, and C. B. Ooi.
CURLER: Finding and visualizing nonlinear correlated clusters.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Baltimore, ML, 2005.

[Web01] G. I. Webb.
Discovering associations with numeric variables.
In Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA, pages 383–388, 2001.

[WLKL04] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee.
FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting.
Information and Software Technology, 46(4):255–271, 2004.

[WWYY02] H. Wang, W. Wang, J. Yang, and P. S. Yu.
Clustering by pattern similarity in large data sets.
In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI, 2002.

[YWWY02] J. Yang, W. Wang, H. Wang, and P. S. Yu.
δ-clusters: Capturing subspace correlation in a large data set.
In Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA, 2002.