Ludwig Maximilians Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Knowledge Discovery in Databases II

im Sommersemester 2008

Kapitel 3: Clustering in hochdimensionalen Räumen

Skript basiert auf Tutorial von Hans-Peter Kriegel, Peer Kröger und Arthur Zimek, ICDM 2007, PAKDD 2008

http://www.dbs.ifi.lmu.de/Lehre/KDD_II

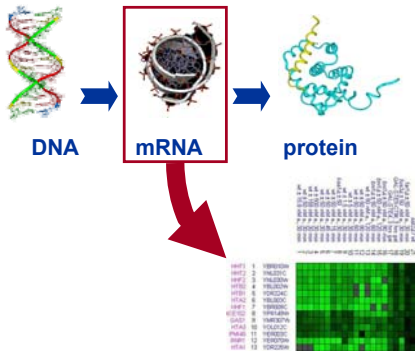
Outline

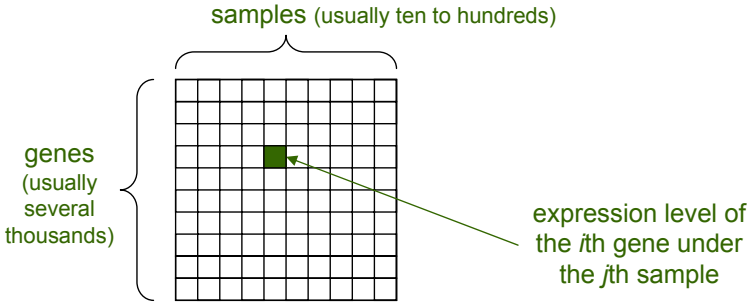
Outline: Introduction

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

Sample Applications

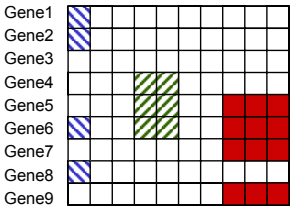
- Gene Expression Analysis
  - Data:
    - Expression level of genes under different samples such as
      - different individuals (patients)
      - different time slots after treatment
      - different tissues
      - different experimental environments
    - Data matrix:





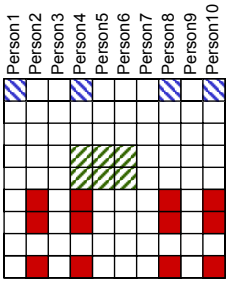
Sample Applications

- 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}

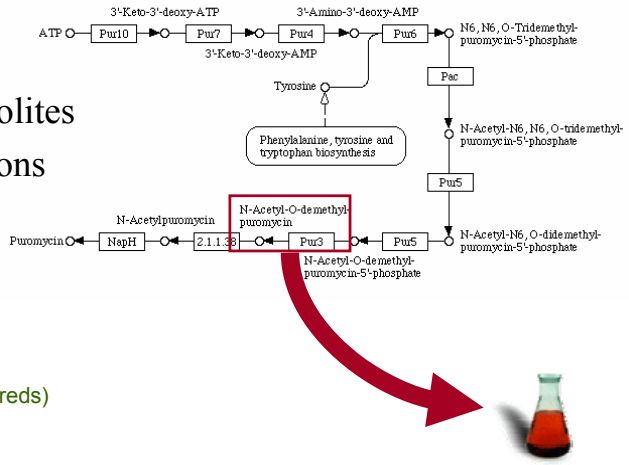
- Task 2: Cluster the columns (e.g. patients) to find groups with similar expression profiles indicating homogeneous phenotypes
  - *Challenge*:
    different phenotypes
    depend on different
    (combinations of)
    subsets of genes

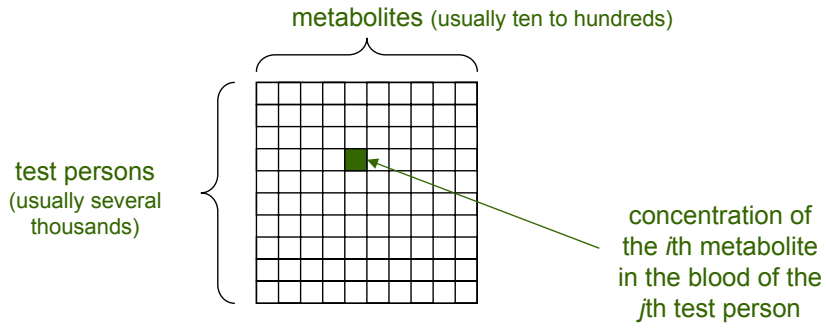


Cluster 1: {P1, P4, P8, P10}
Cluster 2: {P4, P5, P6}
Cluster 3: {P2, P4, P8, P10}

Sample Applications

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

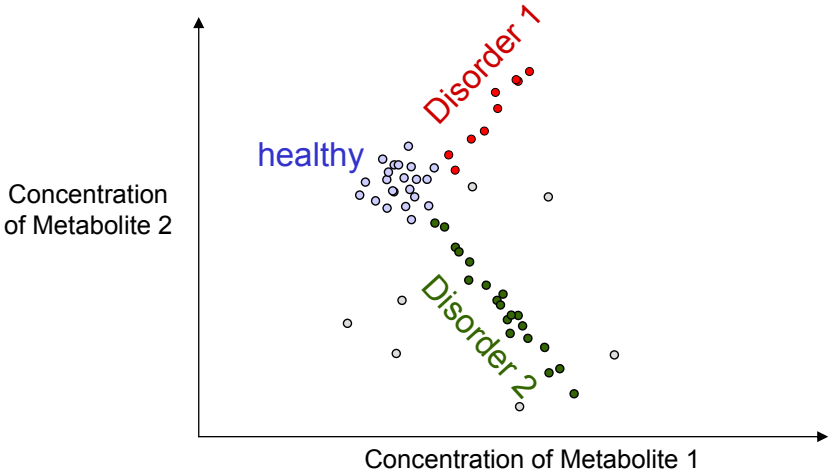




Sample Applications

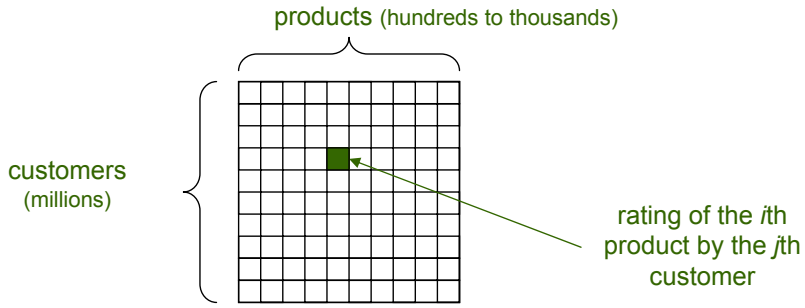
- Task: Cluster test persons to find groups of individuals with similar correlation among the concentrations of metabolites indicating homogeneous metabolic behavior (e.g. disorder)
  - *Challenge*:

    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:

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

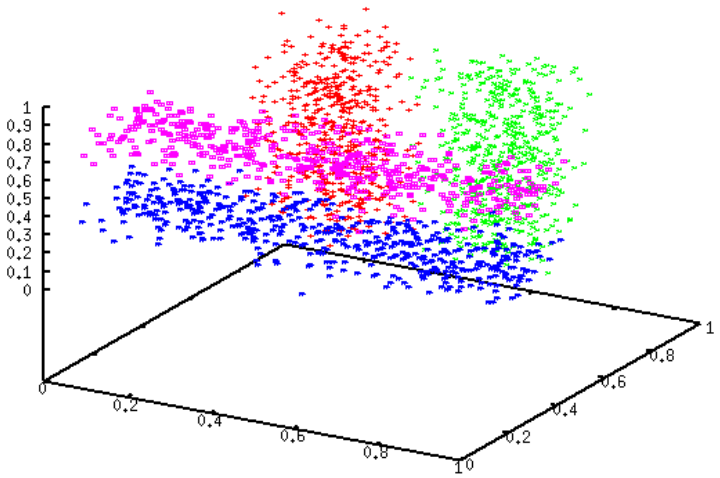
    - This holds true for a wide range of data distributions and distance functions

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!!!***

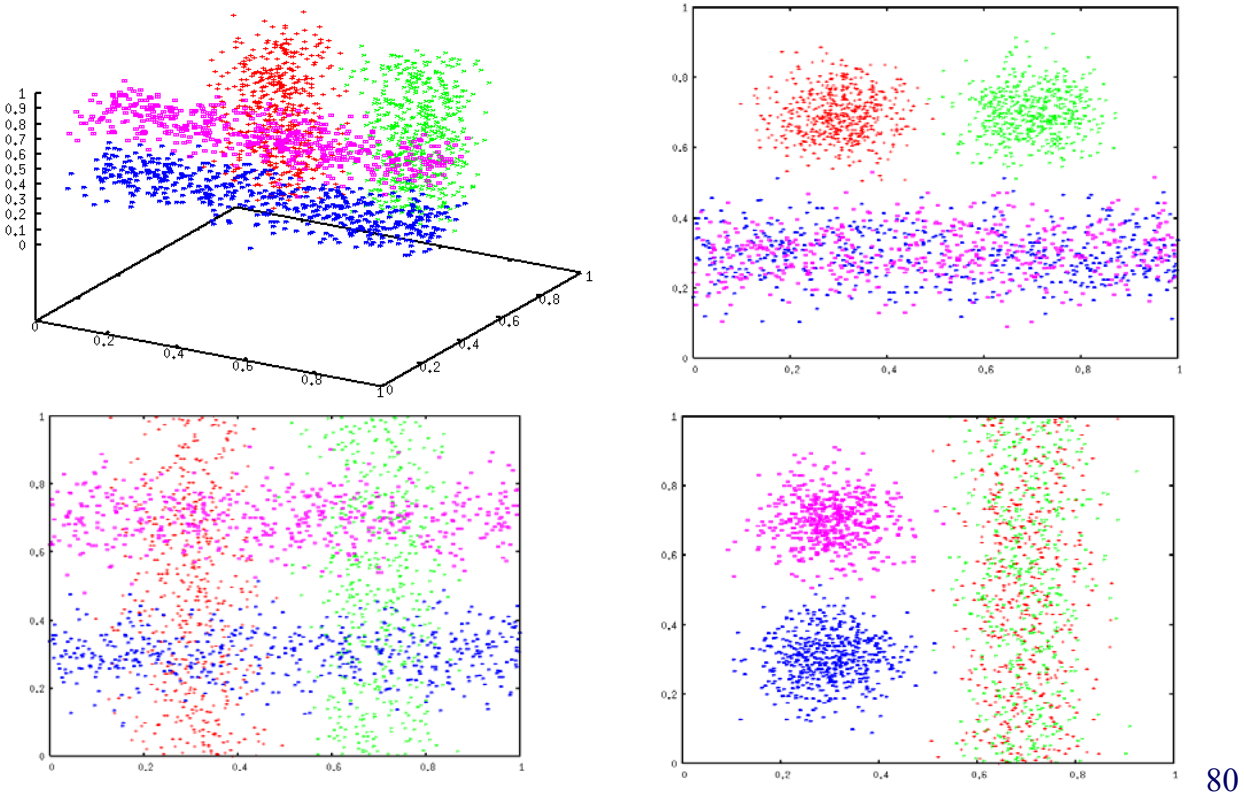
General Problems & Challenges

- Additional problem likely to occur in high dimensional spaces:
  - Usually the distance functions used give equal weight to all dimensions
  - However, not all dimensions are of equal importance
  - Adding irrelevant dimensions ruins any clustering based on a distance function that equally weights all dimensions



General Problems & Challenges

- again: different attributes are relevant for different clusters



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:

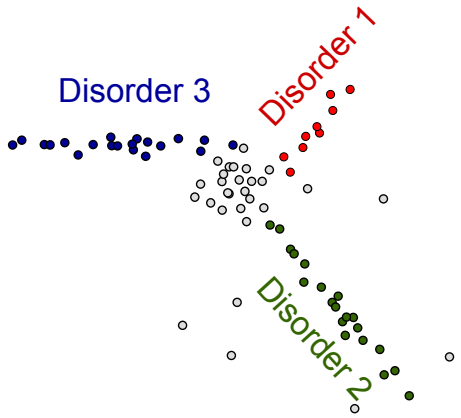
    For each cluster, a different subset of features or a different correlation of features may be relevant (e.g. different genes are responsible for different phenotypes)

  - 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)

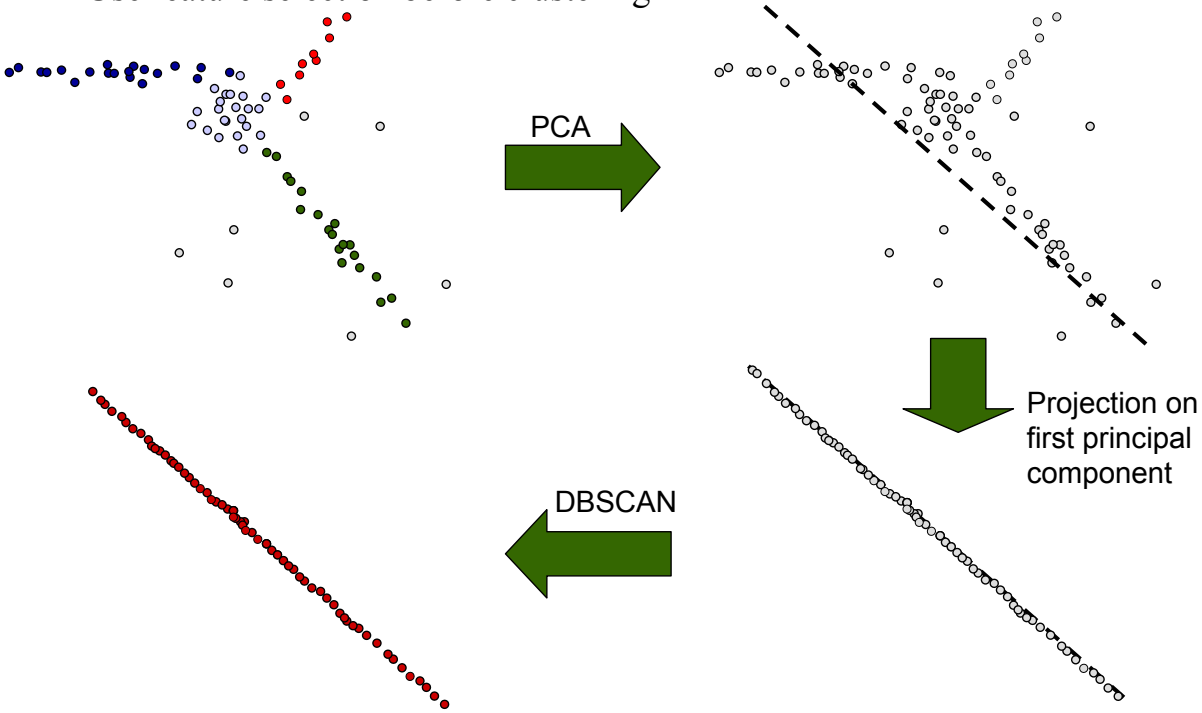
General Problems & Challenges

- 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



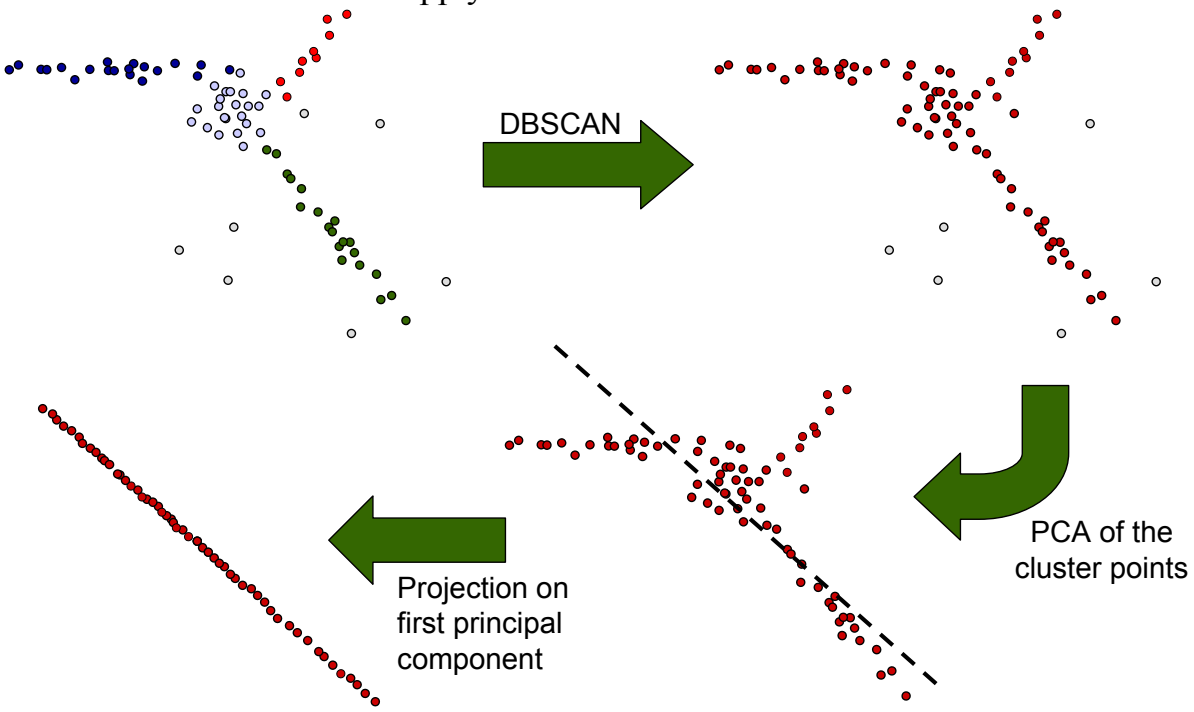
General Problems & Challenges

- 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

Outline: Axis-parallel Subspace Clustering

- Challenges and Approaches
- Bottom-up Algorithms
- Top-down 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:
  - Given a cluster criterion, 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:

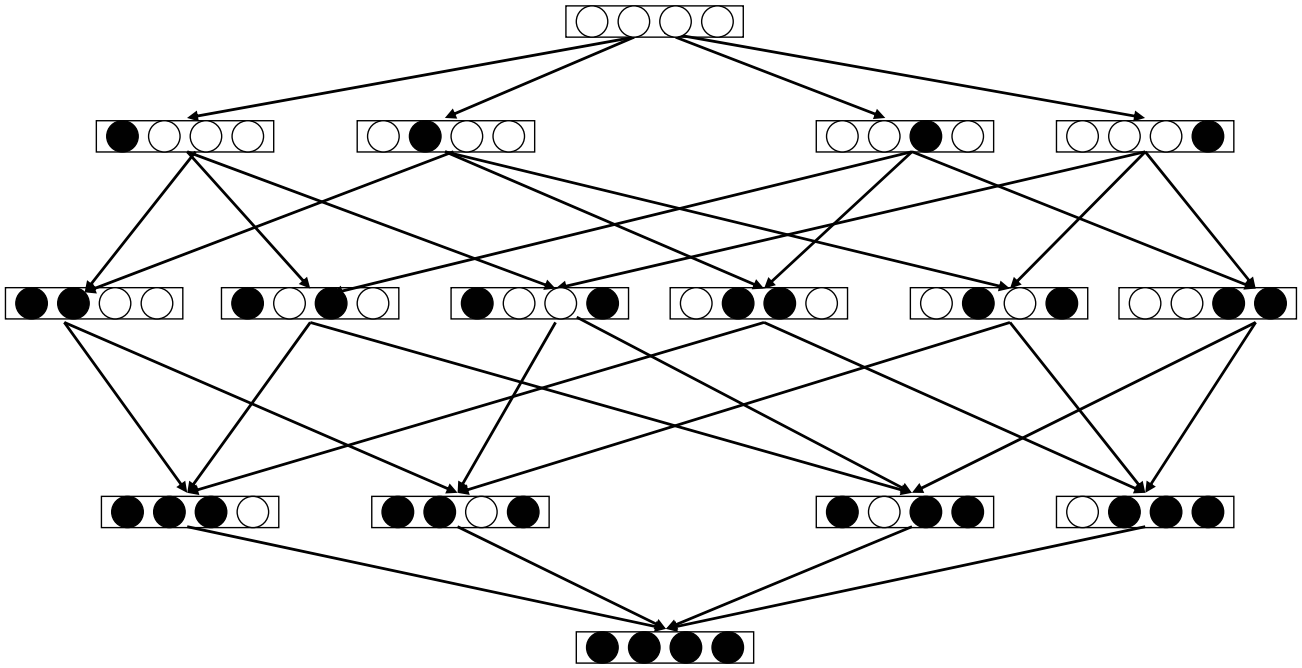
$$\sum_{k=1}^{d} \binom{d}{k} = 2^d - 1$$

  - 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 of possible subspaces
  - Bottom-up:
    - If the cluster criterion implements the downward closure, 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

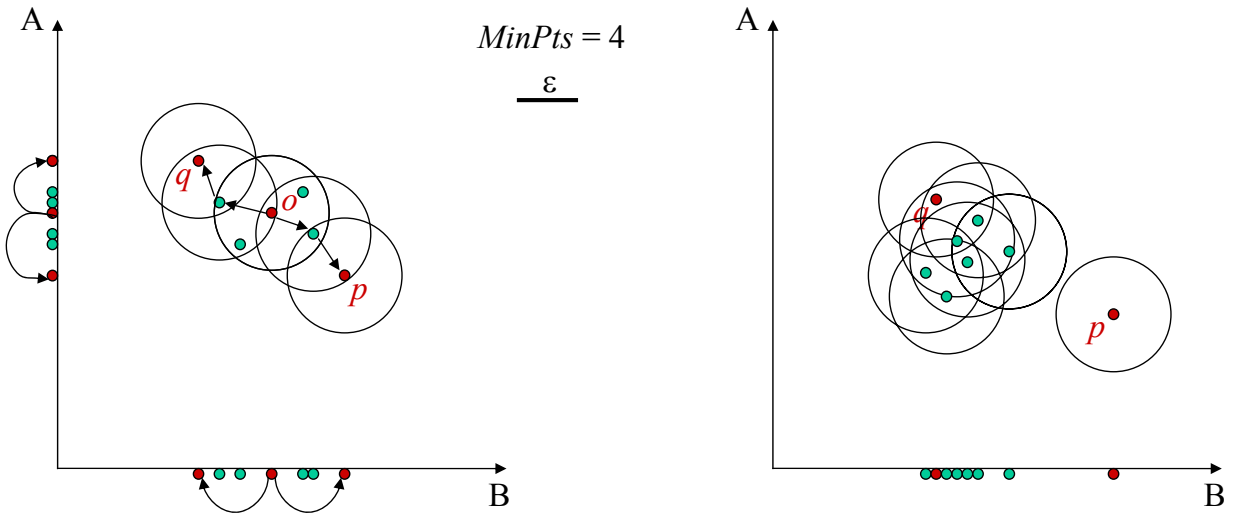
Bottom-up Algorithms

- Rational:
  - Start with 1-dimensional subspaces and merge them to compute higher dimensional ones
  - Most approaches transfer the problem of subspace search into frequent item set mining
    - The cluster criterion must implement the downward closure property
      - If the criterion holds for 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 any frequent itemset mining algorithm (e.g. APRIORI)
  - Some approaches use other search heuristics like best-first-search, greedy-search, etc.
    - Better average and worst-case performance
    - No guaranty on the completeness of results

Bottom-up Algorithms

- Downward-closure property

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



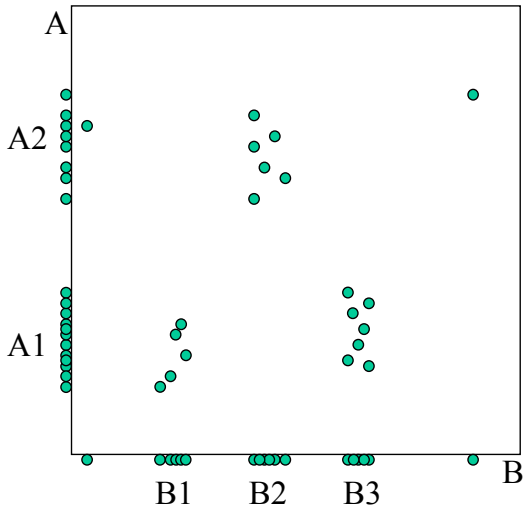
p and q density-connected in {A,B}, {A} and {B}

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

Bottom-up Algorithms

- Downward-closure property

the reverse implication does not hold necessarily



Bottom-up Algorithms

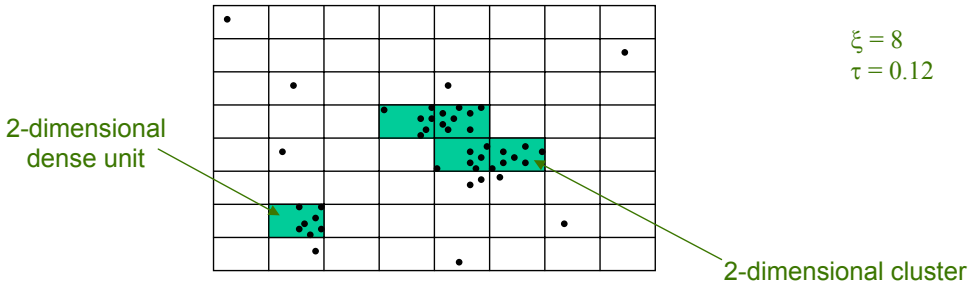
- The key limitation: ***global density thresholds***
  - Usually, the cluster criterion relies on density
  - In order to ensure the downward closure property, the density threshold must be fixed
  - Consequence: the points in a 20-dimensional subspace cluster must be as dense as in a 2-dimensional 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 (APRIORI-style algorithms):
  - 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
  - Complete results

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



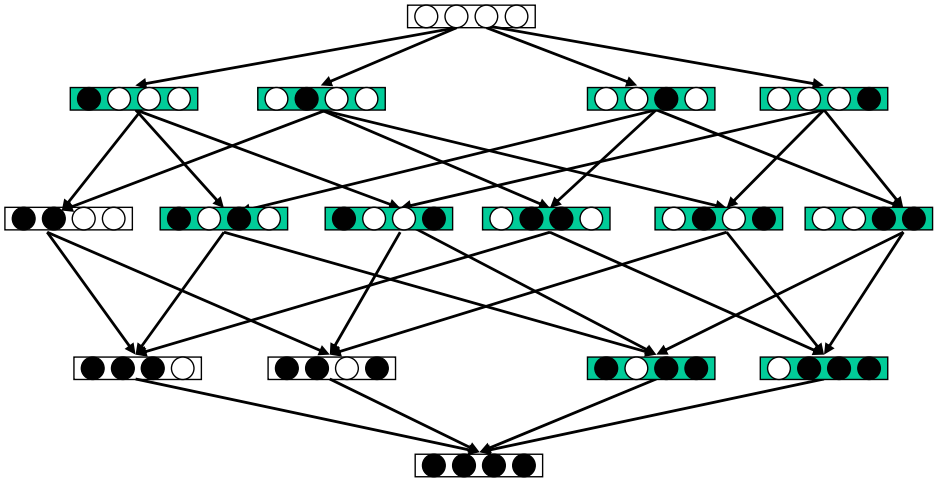
Bottom-up Algorithms

- Downward-closure property holds for dense units
- Algorithm
  - All dense cells are computed using APRIORI-style search
  - A heuristic 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

Bottom-up Algorithms

- 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



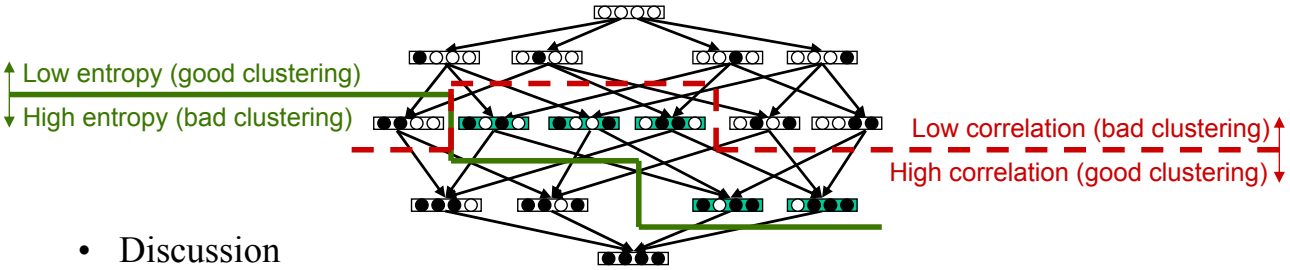
Bottom-up Algorithms

- 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

$$\textit{Entropy} < \omega \quad \text{and} \quad \textit{Correlation} > \varepsilon$$

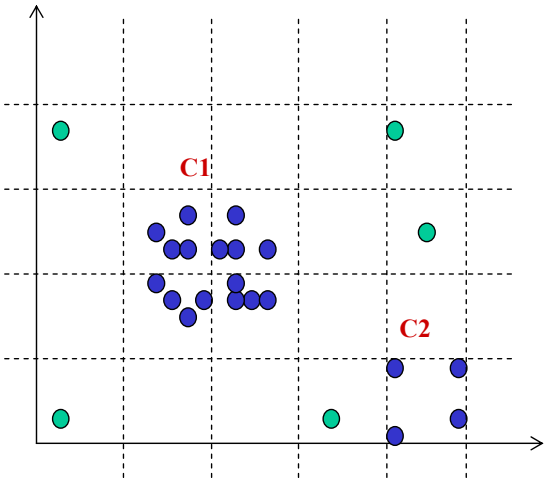


- Discussion
  - Input: thresholds ω and ε
  - Output: all subspaces that meet the above criteria (far less than CLIQUE), clusters may overlap
  - Uses fixed thresholds for entropy and correlation for all subspaces
  - Simple but efficient cluster model

Bottom-up Algorithms

- drawback of grid-based approaches:
  choice of ξ and τ

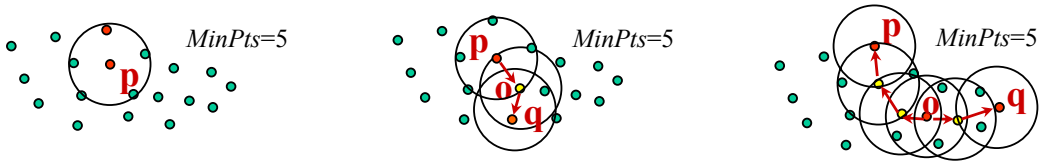
  cluster for τ = 4
  (is C2 a cluster?)
  for τ > 4: no cluster found
  (esp. C1 is lost)



- motivation for density-based approaches

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



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

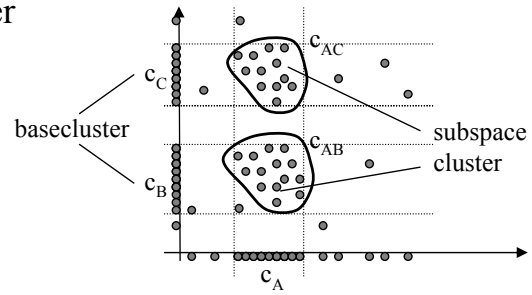
Bottom-up Algorithms

- 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

- FIRES[KKRW05]
  - Proposes a bottom-up approach that uses different heuristic 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

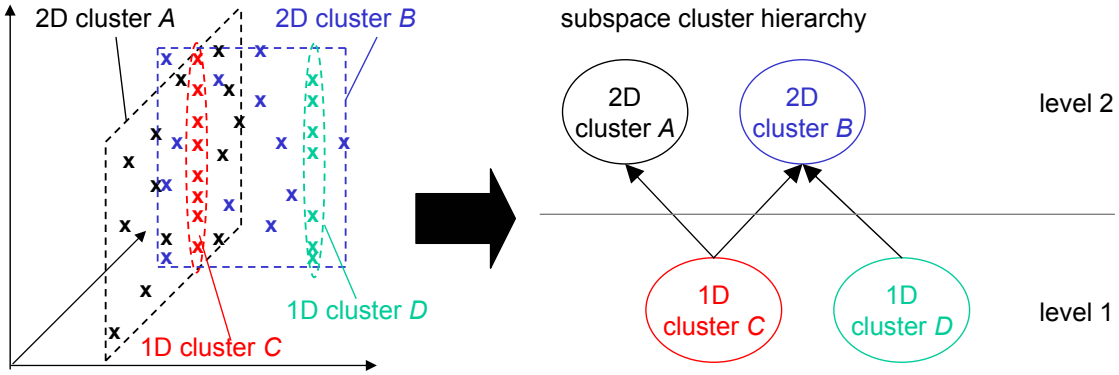


Bottom-up Algorithms

- 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

Bottom-up Algorithms

- DiSH [ABK+07a]
  - Idea:
    - Not considered so far: lower dimensional clusters embedded in higher dimensional ones



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

Bottom-up 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)

Bottom-up 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, clusters may overlap (but only w.r.t. the hierarchical structure!)
  - 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 problem: How should we learn the subspace preference of a cluster or a point?
  - Most approaches rely on the so-called "locality assumption"
    - The subspace is usually learned from the local neighborhood of cluster representatives/cluster members in the entire feature space:
      - Cluster-based approach: the ***local neighborhood*** of each cluster representative is evaluated in the *d*-dimensional space to learn the "correct" subspace of the cluster
      - Instance-based approach: the ***local neighborhood*** of each point is evaluated in the *d*-dimensional space to learn the "correct" subspace preference of each point
    - ***The locality assumption***: the subspace preference can be learned from the ***local neighborhood*** in the *d*-dimensional space
  - Other approaches learn the subspace preference of a cluster or a point from randomly sampled points

Top-down Algorithms

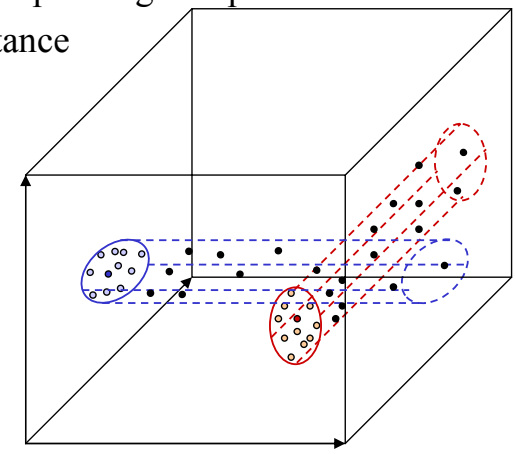
- Discussion:
  - Locality assumption
    - Recall the effects of the curse of dimensionality on concepts like "local neighborhood"
    - The neighborhood will most likely contain a lot of noise points
  - Random sampling
    - The larger the number of total points compared to the number of cluster points is, the lower the probability that cluster members are sampled
  - Consequence for both approaches
    - The learning procedure is often misled by these 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 $O(d^2)$

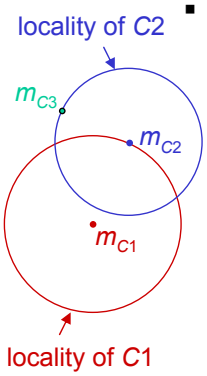
Top-down Algorithms

- 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

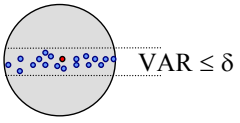
Top-down Algorithms

- Discussion
  - Input:
    - Number of clusters 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

- 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

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



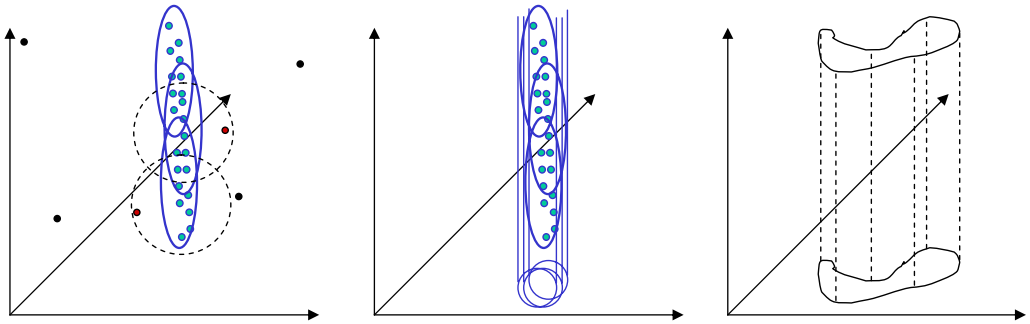
VAR_i is the variance of the ε-neighborhood of p in the entire d-dimensional space, δ and κ >> 1 are input parameters

Top-down Algorithms

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

    $dist_p(p,q) = \sqrt{\sum_i w_i \cdot (p_i - q_i)^2}$ w.r.t. p

    $dist(p,q) = \max_i \{dist_p(p,q), dist_q(q,p)\}$
  - 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

Summary

- The big picture
  - Subspace clustering algorithms compute overlapping clusters
    - Many approaches compute all clusters in all subspaces
      - These methods usually implement a bottom-up search strategy á la itemset mining
      - These methods usually rely on global density thresholds to ensure the downward closure property
      - These methods usually do not rely on the locality assumption
      - These methods usually have a worst case complexity of $O(2^d)$
    - Other focus on maximal dimensional subspace clusters
      - These methods usually implement a bottom-up search strategy based on simple but efficient heuristics
      - These methods usually do not rely on the locality assumption
      - These methods usually have a worst case complexity of at most $O(d^2)$

Summary

- The big picture
  - Projected clustering algorithms compute a disjoint partition of the data
    - They 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