1. Introduction and challenges of high dimensionality

2. Feature Selection

3. Feature Reduction and Metric Learning

4. Clustering in High-Dimensional Data
Challenges for Clustering High-Dimensional Data

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

  - 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 (See: baby shapes game)
• **Relevant and irrelevant attributes**
  – Not all features, but a subset of the features may be relevant for clustering
  – Groups of similar (e.g. “dense”) points may be identified when considering only these features
  – Different subsets of attributes may be relevant for different clusters
Effect on clustering:

- Traditional distance functions 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
agains: different attributes are relevant for different clusters
**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
• **Correlation among attributes**
  – A subset of features may be correlated
  – Groups of similar (e.g. “dense”) points may be identified when considering this correlation of features only
  – Different correlations of attributes may be relevant for different clusters
Why not feature selection/reduction?

- (Unsupervised) feature selection or feature reduction (e.g. PCA) is *global*, i.e., it transforms the original feature space into *one* new representation.
- 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, i.e., we need *multiple* representations.
Challenges for Clustering High-Dimensional Data

Example: use feature reduction (PCA, target dim = 1) before clustering

Projection on first principal component
Cluster first and then apply PCA to determine the correlations of clusters.
Problem Summary

- **Feature relevance and correlation**
  - Usually, no clusters in the full dimensional space
  - Often, clusters are hidden in subspaces of the data, i.e. only a subset of features is relevant for the clustering
  - E.g. a group of genes play a common role in a subset of experimental conditions but may behave completely different in other conditions

- **Local feature relevance/correlation**
  - For each cluster, a different subset of features or a different correlation of features may be relevant
  - E.g. different genes are responsible for different phenotypes

- **Overlapping clusters**
  - Clusters may overlap, i.e. an object may be clustered differently in varying subspaces
  - E.g. a gene plays different functional roles depending on the environment
Challenges for Clustering High-Dimensional Data

• General problem setting of clustering high dimensional data

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

• Challenges:

  • Find the correct subspace of each cluster
    - Search space:
      ▪ all possible arbitrarily oriented subspaces of a feature space
      ▪ infinite

  • Find the correct cluster in each relevant subspace
    - Search space:
      ▪ “Best” partitioning of points (e.g. minimal cut of the similarity graph)
      ▪ NP-complete [SCH75]
Challenges for Clustering High-Dimensional Data

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

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

  *simultaneously*
Challenges for Clustering High-Dimensional Data

• Keep in mind: if we found the relevant subspace, we still might have a hard time to find the correct cluster
  • Here, clusters (yellow, red, green, blue) and noise (gray) are not separable in the “correct” subspace …

• Rather:
  • We would need a new cluster model
  • Cluster = group of similar objects
  • But what is the concept of similarity that all members of a cluster share in this example (and does not include members from other clusters/noise)???
    ... an almost philosophical question …
Overview of the discussed methods

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

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

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

• And: CASH (bottom-up, arbitrarily oriented subspaces)

Find all clusters in all subspaces.
Axis-parallel subspaces

Each point is assigned to one subspace cluster or noise.
Axis-parallel subspaces

Each point is assigned to one subspace cluster or noise.
Arbitrary oriented subspaces
### Overview of the discussed methods

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

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

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

**And:** CASH (bottom-up, arbitrarily oriented subspaces)

---

Find all clusters in all subspaces.  
Axis-parallel subspaces

Each point is assigned to one subspace cluster or noise.  
Axis-parallel subspaces

Each point is assigned to one subspace cluster or noise.  
Arbitrary oriented subspaces
Bottom-up Algorithms

• Rational:
  – Similar to Branch-and-Bound feature selection: Start with 1-dimensional subspaces or subspace clusters and merge them to compute higher dimensional ones.
  – Most approaches transfer this problem into frequent item set mining.
    • The cluster criterion must implement the downward closure (monotonicity) property:
      – If the criterion holds for a $k$-dimensional subspace $S$, then it also holds for any $(k-1)$-dimensional projection of $S$
      – Use the reverse implication for pruning: If the criterion does not hold for a $(k-1)$-dimensional projection of $S$, then the criterion also does not hold for $S$
  – Some approaches use other search heuristics (especially if monotonicity does not hold) 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: example**

- Simple cluster criterion (density of grid cells):
  - If a cell \( C \) of side length \( s \) contains more than \( m \) points, it represents a cluster

- Monotonicity:
  - if \( C \) contains more than \( m \) points in subspace \( S \) then \( C \) also contains more than \( m \) points in any subspace \( T \subset S \)
  - Example: monotonicity (left) and reverse implication (right)

---

Cell \( C \) contains more than \( m=5 \) points in subspace „AB“

=> Also in subspaces „A“⊂ „AB“ and „B“⊂ „AB“

Cell \( C \) contains less than \( m=5 \) points in subspace „A“

=> Also in subspace „AB“
CLIQUE is probably the first bottom-up algorithm; it uses a density-grid-based cluster model.

**Cluster Model**
- Clusters are “dense regions” in the feature space
- Partition the feature space into $\xi$ equal sized parts in each dimension (implicitly fixing side length $s$).
- A *unit* is the intersection of one interval from each dimension
- *Dense* unit: If unit $u$ contains more than $\tau$ objects, $\tau =$ density threshold
  => monotonicity of dense units holds (see previous slide)
- *Clusters* are maximal sets of connected dense units (e.g., $A \cup B$)

2-step Approach:
1. Find subspaces (with dense units)
2. Find subspace clusters (union of connected dense units in the same subspace)
CLIQUE:
1. Identify subspaces containing clusters

1. Task: Find subspaces with dense units
   - Greedy approach (Bottom-Up), comparable to APRIORI for finding frequent itemsets
     (Downward Closure):
     - Determine 1-dimensional dense units \( D_1 \)
     - Candidate generation procedure:
       - Based on \( D_{k-1} \), the set of \((k-1)\) dimensional dense units, generate candidate set \( C_k \) by self joining \( D_{k-1} \)
         - Join condition: units share first \( k-2 \) dimensions
       - Discard those candidates which have a \( k-1 \) projection not included in \( D_{k-1} \)
       - For the remaining candidates: check density

![Diagram showing 2-dim. dense unit (\( \in D_2 \)), 3-dim. candidate unit, and 2-dim. unit which has to be checked.](image-url)
CLIQUE: 2. Identify clusters

2. Task: Find maximal sets of connected dense units

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

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

- The problem is equivalent to finding connected components in a graph
  - Nodes: dense units
  - Edge between two nodes if the corresponding dense units have a common face (neighboring units)
  - Depth-first search algorithm: Start with a unit $u$ in $D$, assign it to a new cluster ID and find all the units it is connected to. Repeat if there are nodes not yet visited
CLIQUE: Discussion

- Input parameters: $\xi$ and $\tau$ specifying the density threshold
- Output: all clusters in all subspaces, clusters may overlap/be redundant
- Simple but efficient cluster model: Uses a fixed density threshold for all subspaces (in order to ensure the downward closure property) => to represent a cluster, a unit in 10D must contain as many points (or more) as in 2D ...
**Motivation:**

Drawbacks of a grid-based clustering model:

- Positioning of the grid influences the clustering
- Only rectangular regions
- Selection of $\xi$ and $\tau$ is very sensitive

Example:

Cluster for $\tau = 4$
(is $C_2$ a cluster?)

for $\tau > 4$: no cluster
( $C_1$ is lost)

$\Rightarrow$ define regions based on the neighborhood of data points
$\Rightarrow$ use density-based clustering
• Density-based cluster model of DBSCAN
• Clusters are maximal sets of density-connected points
• Density connectivity is defined based on core points
• Core points have at least MinPts points in their ε-neighborhood

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

If C is a density connected set in subspace S then C is a density connected set in any subspace $T \subseteq S$.
- But, if C is a cluster in S, it need not to be a cluster in $T \subseteq S$ – maximality might be violated
- All clusters in a higher-dimensional subspace will be subsets of the clusters detected in this first clustering.

$p$ and $q$ density connected in \(\{A,B\}\).
Thus, they are also density connected in \(\{A\}\) and \(\{B\}\)

\(\varepsilon\): circles indicate

MinPts = 4

$p$ and $q$ not density connected in \(\{B\}\).
Thus, they are not density connected in \(\{A,B\}\), although they are density connected in \(\{A\}\).
SUBCLU: Discussion

– Algorithm
  • All subspaces that contain any density-connected set are computed using the bottom-up approach (similar to CLIQUE/APRIORI)
  • Density-connected clusters are computed using a specialized DBSCAN run in the resulting subspace to generate the subspace clusters

– Discussion
  • Input: \( \varepsilon \) and \( \text{MinPts} \) 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
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 an e.g. 20-dimensional subspace cluster must be as dense as in an e.g. 2-dimensional cluster
- This is a rather optimistic assumption since the data space grows exponentially with increasing dimensionality (see “curse” discussion)
- Consequences:
  - A strict threshold will most likely produce only lower dimensional clusters
  - A loose threshold will most likely produce higher dimensional clusters but also a huge amount of (potentially meaningless) low dimensional clusters
Overview of the discussed methods

- **Bottom-Up approaches:** Subspace Clustering
  - CLIQUE [AGGR98]
  - SUBCLU [KKK04]
  
  Find all clusters in all subspaces. 
  
  Axis-parallel subspaces

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

  Each point is assigned to one subspace cluster or noise. 
  
  Axis-parallel subspaces

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

  Each point is assigned to one subspace cluster or noise. 
  
  Arbitrary oriented subspaces

- And: CASH (bottom-up, arbitrarily oriented subspaces)
Rational:

- Cluster-based approach:
  - Learn the subspace of a cluster in the *entire* $d$-dimensional feature space
  - Start with full-dimensional clusters
  - Iteratively refine the cluster memberships of points and the subspaces of the cluster
  - PROCLUS[APW+99], ORCLUS[AY00]
- Instance-based approach:
  - Learn for each *point* its subspace preference in the entire $d$-dimensional feature space
  - The subspace preference specifies the subspace in which each point “clusters best”
  - Merge points having similar subspace preferences to generate the clusters
  - PREDECON[BKKK04] 4C[BKKZ04]
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 this point (i.e. the subspace in which the cluster exists that accommodates this point)
  - The locality assumption: the subspace preference can be learned from the local neighborhood in the $d$-dimensional space
  - Other approaches learn the subspace preference of a cluster or a point from randomly sampled points
Overview of the discussed methods

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

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

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

- And: CASH (bottom-up, arbitrarily oriented subspaces)
• PROjected CLUStering
  – *Cluster-based* top-down approach: we learn the subspace for each cluster
  – *K*-medoid cluster model
    - Cluster is represented by its medoid
    - To each cluster a subspace (of relevant attributes) is assigned
    - Each point is assigned to the nearest medoid (where the distance to each medoid is based on the corresponding subspace of the medoid)
    - Points that have a large distance to their nearest medoid(s) are classified as noise
• 3-phase algorithm: initialization, iterative phase, refinement
  – Input:
    o The set of data points
    o Number of clusters, denoted by k
    o Average number of dimensions for each clusters, denoted by L
  – Output:
    o The clusters found, and their associated dimensions

• [Phase 1] Initialization of cluster medoids
  – Ideally we want a set of centroids, where each centroid comes from a different cluster.
  – We don’t know which are these \( k \) points though, so we choose a superset \( M \) of \( b^*k \) medoids such that they are well separated.
    • Chose a random sample \( (S) \) of \( a^*k \) data points
    • Out of \( S \), select \( b^*k \) points \( (M) \) by greedy selection : medoids are picked iteratively so that the current medoid is well separated from the medoids that have been chosen so far.
  – Input parameters \( a \) and \( b \) are introduced for performance reasons
[Phase 2] Iterative phase (works similar to any \( k \)-medoid clustering)

- \( k \) randomly chosen medoids from \( M \) are the initial cluster medoids
- Idea: replace the “bad” medoids with other points in \( M \) if necessary → we should be able to evaluate the quality of the clustering by a given set of medoids.
- Procedure:
  - Find dimensions related to the medoids
  - Assign data points to the medoids
  - Evaluate the clusters formed
  - Find the bad medoids, and try to improve the result by replacing these bad medoids
• For each medoid $m_i$, let $2\delta$ be the distance to its closest medoid
• All the data points within $\delta$ will be surely assigned to the medoid $m_i$ ($L_i$, locality of $m_i$)

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

– Assigning data points – evaluate clusters 5/6

- Assign each data point to its closest medoid using Manhattan segmental distance (only relevant dimensions count)

- Manhattan segmental distance (A variance of Manhattan distance): For any two points \(x_1, x_2\) and any set of dimensions \(D, |D| \leq d\):
  
  \[
d_D(x_1, x_2) = \frac{\sum_{i \in D} |x_{1,i} - x_{2,i}|}{|D|}
  \]

- How to evaluate the clusters?
  - Use average Manhattan segmental distance from the points in \(C_i\) to the centroid of \(C_i\) along dimension \(j\)
    
    \[
w_i = \frac{\sum_j Y_{i,j}}{|D_i|} \quad E = \frac{\sum_{i=k}^k |C_i| \cdot w_i}{N}
    \]

- Replace bad medoids with random points from \(M\)

- Terminate if the clustering quality does not increase after a given number of current medoids have been exchanged with medoids from \(M\) (it is not clear, if there is another hidden parameter in that criterion)
• **[Phase 3] Refinement**
  – Reassign subspaces to medoids as above (but use only the points assigned to each cluster rather than the locality of each cluster, i.e., $C_i$ not $L_i$)
  – Reassign points to medoids
  – Points that are not in the locality of any medoid are classified as noise
• Instance-based top-down approach: we learn the subspace for each instance
• Extends DBSCAN to high dimensional spaces by incorporating the notion of dimension preferences in the distance function

• For each point \( p \), it defines its subspace preference vector:

\[
\bar{w}_p = (w_1, w_2, \ldots, w_d)
\]

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

• \( \text{VAR}_i \) is the variance along dimension \( j \) in \( N_{\varepsilon}(p) \):

\[
\text{VAR}_{A_i}(N_{\varepsilon}(p)) = \frac{\sum_{q \in N_{\varepsilon}(p)}(\text{dist}(\pi_{A_i}(p), \pi_{A_i}(q)))^2}{|N_{\varepsilon}(p)|}
\]

\( \delta, \kappa (\kappa >> 1) \) are input parameters
• Preference weighted distance function:

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

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

• Preference weighted \(\varepsilon\)-neighborhood:

\[
N_{\varepsilon}^{wp}(p) = \{x \in D \mid dist_{pref}(p, x) \leq \varepsilon\}
\]
• Preference weighted core points:

\[ \text{CORE}_{\text{den}}^{\text{pref}} (p) \iff \text{PDIM}(\mathcal{N}_\varepsilon(p)) \leq \lambda \land |\mathcal{N}_\varepsilon^\text{w}_o(p)| \geq \mu \]

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

• A \textit{subspace preference cluster} is a maximal density connected set of points associated with a certain subspace preference vector.
Overview of the discussed methods

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

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

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

- And: CASH (bottom-up, arbitrarily oriented subspaces)
Correlation Clustering

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

- Subspace clustering and projected clustering algorithms find axis-parallel subspaces
- Correlation clustering for finding clusters in arbitrary oriented subspaces
• ORCLUS (arbitrarily ORiented projected CLUSter generation) first approach to generalized projected clustering

• A generalized projected cluster is a set of vectors \( E \) and a set of points \( C \) such that the points in \( C \) are closely clustered in the subspace defined by the vectors \( E \).
  
  – \( E \) is a set of orthonormal vectors, \(|E| \leq d\)

Input:

• The number of clusters \( k \)

• The dimensionality of the subspace of the clusters, \( l (=|E|) \)

Output

• A set of \( k \) clusters and their associated subspaces of dimensionality \( l \)

Main idea

• To find the subspace of a cluster \( C_i \), compute the \( d \times d \) covariance matrix \( M_i \) for \( C_i \) and determine the eigenvectors. Pick the \( l_c \) eigenvectors with the smallest eigenvalues.

• Relies on cluster-based locality assumption: subspace of each cluster is learned from its members
ORCLUS: Algorithm 2/3

• similar ideas to PROCLUS [APW+99]
• $k$-means like approach
• start with $k_c > k$ seeds
• assign points to clusters according to distance function based on the eigensystem of the current cluster (starting with axes of data space, i.e. Euclidean distance)
• The eigensystem is iteratively adapted based on the updated cluster members
• Reduce the number of clusters $k_c$ in each iteration by merging best-fitting cluster pairs
• Each cluster $C_i$ exists in a possible different subspace $S_i$, how do we decide what to merge?
• Compute the subspace of their union $C_i \cup C_j$ (eigenvectors corresponding to the smallest $l$ eigenvalues)
• Check the cluster energy of $C_i \cup C_j$ in this subspace (mean square distance of the points from the centroid in this subspace) – indicator of how well the points combine
• Assess average distance in all merged pairs of clusters and finally merge the best fitting pair, that with the smallest cluster energy
• Continue until the desired number of clusters, $k$, is achieved.
4C = Computing Correlation Connected Clusters
Idea: Integrate PCA into density-based clustering.

Approach:
• Check the core point property of a point \( p \) in the complete feature space
• Perform PCA on the local neighborhood \( S \) of \( p \) to find subspace correlations

PCA factorizes \( M_p \) into \( M_p = V E V^T \)

- \( V \): eigenvectors
- \( E \): eigenvalues

- A parameter \( \delta \) discerns large from small eigenvalues.
- \( \text{CorDim}(S) = \#\text{eigenvalues} > \delta \)
- In the eigenvalue matrix of \( p \), large eigenvalues are replaced by 1, small eigenvalues by a value \( \kappa \gg 1 \) \( \rightarrow \) adapted eigenvalue matrix \( E'_p \)
4C: Distance measure

- effect on distance measure:

\[ d(p, q) \]

- distance of \( p \) and \( q \) w.r.t. \( p \):
  \[ \sqrt{(p - q) \cdot V_p \cdot E_p^\top \cdot (p - q)^\top} \]

- distance of \( p \) and \( q \) w.r.t. \( q \):
  \[ \sqrt{(q - p) \cdot V_q \cdot E_q^\top \cdot (q - p)^\top} \]
4C: correlation neighbors

- symmetry of distance measure by choosing the maximum:

\[ \max \left\{ \frac{\sqrt{(p - q) \cdot V_p' \cdot E_p' \cdot V_p^T \cdot (p - q)^T}}{\sqrt{(q - p) \cdot V_q' \cdot E_q' \cdot V_q^T \cdot (q - p)^T}} \right\} \leq \varepsilon \]

- \( p \) and \( q \) are correlation-neighbors if
4C vs. DBSCAN

Cluster found by DBSCAN
Clusters found by 4C

4C vs. ORCLUS

4C

ORCLUS
4C: discussion

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

Enhancements also based on PCA:
- COPAC [ABK+07c] and
- ERiC [ABK+07b]
Correlation clustering with PCA: Discussion

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

- So how to overcome the locality assumption???
  => different method to determine correlation?
  => Hough transform (computer graphics)
    find structures (e.g. lines, circles) in images  => CASH coming up
Overview of the discussed methods

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

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

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

• And: CASH (bottom-up, arbitrarily oriented subspaces)
• Basic idea of CASH (= Clustering in Arbitrary Subspaces based on the Hough transform)
  – Transform each object into a so-called parameter space representing all possible subspaces accommodating this object (i.e. all hyper-planes through this object)
  – This parameter space is a continuum of all these subspaces
  – The subspaces are represented by a considerably small number of parameters
  – This transform is a generalization of the Hough Transform (which is designed to detect linear structures in 2D images) for arbitrary dimensions
• Transform
  – For each \(d\)-dimensional point \(p\) there is an infinite number of \((d-1)\)-dimensional hyper-planes through \(p\)
  – Each of these hyper-planes \(s\) is defined by \((p, \alpha_1, ..., \alpha_{d-1})\), where \(\alpha_1, ..., \alpha_{d-1}\) is the normal vector \(n_s\) of the hyper-plane \(s\)
  – The function \(f_p(\alpha_1, ..., \alpha_{d-1}) = \delta_s = <p, n_s>\) maps \(p\) and \(\alpha_1, ..., \alpha_{d-1}\) onto the distance \(\delta_s\) of the hyper-plane \(s\) to the origin
  – The parameter space plots the graph of this function
Properties of this transform

- A point in the data space is a sinusoide curve in the parameter space.
- A point in the parameter space is a hyper-plane in the data space.
- Points on a common hyper-plane in the data space (cluster) are sinusoide curves intersecting at one point in the parameter space.
- Intersection of sinusoide curves in the parameter space is a hyper-plane accommodating the corresponding points in data space.
• Detecting clusters
  – determine all intersection points of at least $m$ curves in the parameter space
    => $(d-1)$-dimensional cluster
  – Exact solution (check all pair-wise intersections) is too costly
  – Heuristics are employed

• Grid-based bisecting search
  => Find cells with at least $m$ curves
  😊 determining the curves that are within a given cell is in $O(d^3)$
  😞 Number of cells $O(r^d)$, where $r$ is the resolution of the grid
  😞 high value for $r$ necessary
• Complexity  \((c = \text{number of cluster found} – \text{not an input parameter}!!!)\)
  - Bisecting search \(O(s \cdot c)\)
  - Determination of curves in a cell \(O(n \cdot d^3)\)
  - Over all \(O(s \cdot c \cdot n \cdot d^3)\)
    (algorithms for PCA are also in \(O(d^3)\))

• Robustness against noise
(a) Data set DS2.

(b) CASH – Cluster 1 - 5.

(c) 4C – Cluster 1 - 8.

(d) ORCLUS – Cluster 1 - 5.
• Finding clusters in (arbitrarily oriented) subspaces of the original feature space.

• The subspace (where the cluster exists) is part of the cluster definition.

• The challenge is 2-fold: finding the correct subspace for each cluster and the correct cluster in each relevant subspace
  – Integrate subspace search in the clustering process

• Traditional full dimensional clustering paradigms transferred in the high dimensional space.
Different types of methods

- **Bottom-Up approaches: Subspace Clustering**
  - Find clusters in all subspaces
  - Restrict the search space by downward closure property
  - Axis-parallel subspaces
  - CLIQUE [AGGR98], SUBCLU [KKK04]

- **Top-Down Approaches: Projected Clustering**
  - Each point is assigned to one subspace cluster or noise.
  - Subspaces are discovered based on the locality (cluster-based, instance-based)
  - Axis-parallel subspaces
  - PROCLUS [APW+99], PREDECON[BKKK04]

- **Top-Down Approaches: Correlation Clustering**
  - Each point is assigned to one subspace cluster or noise.
  - Subspace are discovered based on the locality (cluster-based, instance-based)
  - Arbitrary oriented subspaces
  - ORCLUS[AY00], 4C [BKKZ04], CASH [ABKKZ07]
*Automatic subspace clustering of high dimensional data for data mining applications.*  

*Density-connected subspace clustering for high dimensional data.*  

*Density connected clustering with local subspace preferences.*  

*Fast algorithms for projected clustering.*  

[AY00] C. C. Aggarwal and P. S. Yu.  
*Finding generalized projected clusters in high dimensional space.*  

*Computing clusters of correlation connected objects.*  

[ABKKZ07] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Arthur Zimek:  
Robust, Complete, and Efficient Correlation Clustering. SDM 2007: 413-418