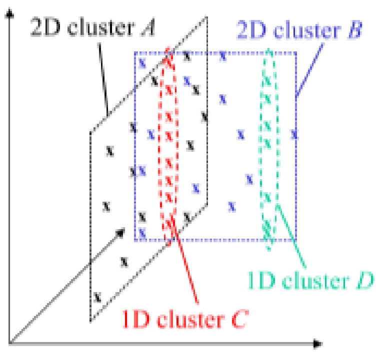
# Detection and Visualization of Subspace Cluster Hierarchies

## Introduction

2 approaches of subspace clustering algorithms

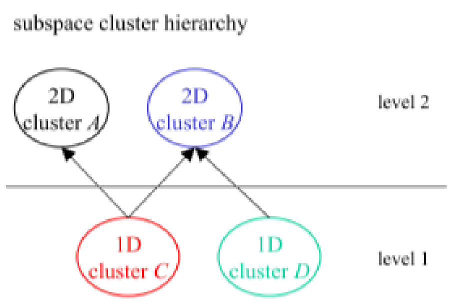
- **overlapping clusters** = points may be clustered differently in varying subspaces ➔ vast number of clusters are hard to interpret
- **Non-overlapping clusters** = points assigned uniquely to one cluster/noise
  - Problems with subspace clusters of significantly different dimensionality
  - Fail to discover clusters of different shape and densities
  - Subspace clusters may be hierarchically nested – 2 1D clusters embedded within 1 2D cluster




Solution: DiSH: approaches by considering NON-OVERLAPPING clusters

- Can detect clusters in subspaces of significantly different dimensionality
- Uncovers complex hierarchies of nested subspace clusters (i.e. clusters in lower-dimensional subspaces that are embedded within higher-dimensional subspace clusters = multiple inclusions)
- Able to detect cluster of different size, shape, and density

**Visualization** of DiSH: Subspace Clustering Graph (trees not sufficient for hierarchy-problems)




## Hierarchical Subspace Clustering

Let $D \sqsubseteq R^d$ be a data set with *n* feature vectors. $A$ is the set of attributes of $D$. For any subspace $S \sqsubseteq A, \pi_S(o)$ denotes the projection of $o \in D$ into $S$. $DIST$ shall be a distance function applicable to any $S \sqsubseteq A$.

**Idea**: to define subspace distance, which

- assigns **small values** if 2 points are in a **common low-dimensional** subspace cluster
- assigns **high values** if 2 points are in a **common high-dimensional** subspace cluster OR in **no subspace cluster** at all

→Subspace clusters with small subspace distance are **embedded** within clusters with higher subspace distance

***(1) Variance Analysis***

First, compute subspace dimensionality for each point $o \in D$, in which it fits best. “Best” = subspace with highest dimensionality (or: in tie-situations dimensionalities containing more points in neighborhood)

*How is the subspace dimensionality determined?*

Subspace dimensionality of a point $o$ is determined **by searching for dimensions of low variance** (high density) in the neighborhood of $o$.

An attribute-wise $\varepsilon$-range query ( $N_\varepsilon^{\{a_i\}}(o) = \{x \mid DIST^{\{a_i\}}(o,x) \leq \varepsilon\}\ for\ each\ a_i \in A$ ) assigns a predicate to an attribute for a certain object $o$:

- Few points within $\varepsilon$-neighborhood means high variance around $o$ in attribute $a_i$ → Predicate 0 is assigned for query point $o$ (attribute does not participate in a subspace that is relevant to any cluster)
- Otherwise, if $N_\varepsilon^{\{a_i\}}(o)$ contains at least $\mu$ objects → attribute $a_i$ will be candidate for subspace containing a cluster including object $o$

→From variance analysis the candidate attributes that might span the best subspace $S_o$ for object $o$ are determined.

**Definition 1 (subspace preference vector/dimensionality of a point).** *Let $S_o$ be the best subspace determined for object $o \in \mathcal{D}$. The* subspace preference vector $w(o) = (w_1, \ldots, w_d)^{\mathbf{T}}$ *of o is defined by*

$$w_i(o) = \begin{cases} 1 & if \quad a_i \in S_o \\ 0 & if \quad a_i \notin S_o \end{cases}$$

*The* subspace dimensionality $\lambda(o)$ *of $o \in \mathcal{D}$ is the number of zero-values in the subspace preference vector $w(o)$.*

→ Overall, $o$ is assigned to the subspace containing more points

***(2) Frequent itemset mining***

For combining these attributes in a suitable way to get the best subset, frequent itemset mining is used.

- Apriori-algorithm
- Heuristics approach (outperforms Apriori): Best-first search

**best-first search**:

- determines best subspace $S_o$ for object $o$
- scales linearly in the number of dimensions
- determines $w(o)$
- assigns a d-dimensional preference vectors to each point
- Attributes with predicate “1” relevant, other remain irrelevant

1. Determine the candidate attributes of $o$: $C(o) = \{a_i \mid a_i \in \mathcal{A} \wedge |\mathcal{N}_\varepsilon^{a_i}(o)| \geq \mu\}$.
2. Add $a_i = \arg\max_{a \in C(o)} \{|\mathcal{N}_\varepsilon^{a}(o)|\}$ to $S_o$ and delete $a_i$ from $C(o)$.
3. Set current intersection $I := \mathcal{N}_\varepsilon^{a_i}(o)$.
4. Determine attribute $a_i = \arg\max_{a \in C(o)} \{|I \cap \mathcal{N}_\varepsilon^{a}(o)|\}$.
   (a) If $|I \cap \mathcal{N}_\varepsilon^{a_i}(o)| \geq \mu$ then:
   Add $a_i$ to $S_o$, delete $a_i$ from $C(o)$, and set $I := I \cap \mathcal{N}_\varepsilon^{a_i}(o)$.
   (b) Else: stop.
5. If $C \neq \emptyset$ continue with Step 4.

***(3) Similarity Measure between points***

*What is a subspace dimensionality of a pair of points?*

**Definition 2 (subspace dimensionality of a point pair).** *The* subspace preference vector $w(p, q)$ *of a pair of points* $p, q \in \mathcal{D}$ *representing the combined subspace of* $p$ *and* $q$ *is computed by an attribute-wise logical AND-conjunction of* $w(p)$ *and* $w(q)$*, i.e.* $w_i(p, q) = w_i(p) \wedge w_i(q)$ $(1 \leq i \leq d)$. *The* subspace dimensionality *between two points* $p, q \in \mathcal{D}$*, denoted by* $\lambda(p, q)$*, is the number of zero-values in* $w(p, q)$.

Note: $\lambda(p,q)$ cannot be used directly, bc**. points from parallel subspace clusters will have same subspace preference vector:**

- Check if preference verctors of two points p and q are equal or one preference vector is included in the other one:
  - Therefore, compute subspace preference vector $w(p,q)$
  - Check if $w(p,q) = w(p)$ $or$ $w(q)$
    - If so, distance between the points is determined by $w(p,q)$
    - If distance exceeds 2 $\varepsilon$, the points belong to parallel clusters
- Since $\lambda(p,q) \in$ N , many tie situations result when merging points/clusters during hierarchical clustering
  - Therefore consider distance within a subspace cluster as second criterion
    - Assign distance 1 if two points share coomon 1D subspace cluster
    - Assign distance 2 if they share a common 2D cluster
    - ➔ thus subspace clusters can exhibit arbitrary sizes, shapes and densities

**Definition 3 (subspace distance).** *Let* $w$ *be an arbitrary preference vector. Then* $S(w)$ *is the subspace defined by* $w$ *and* $\bar{w}$ *denotes the inverse of* $w$*. The* subspace distance SDIST *between* p *and* q *is a pair* $\text{SDIST}(p, q) = (d_1, d_2)$*, where* $d_1 = \lambda(p, q) + \Delta(p, q)$ *and* $d_2 = \text{DIST}^{S(\bar{w}(p,q))}(p, q)$*, and* $\Delta(p, q)$ *is defined as*

$$\Delta(p,q) = \begin{cases} 1 & \text{if } (w(p,q) = w(p) \vee w(p,q) = w(q)) \wedge \text{DIST}^{S(w(p,q))}(p,q) > 2\varepsilon \\ 0 & \text{else,} \end{cases}$$

*We define* $\text{SDIST}(p, q) \leq \text{SDIST}(r, s) \iff \text{SDIST}(p, q).d_1 < \text{SDIST}(r, s).d_1$ *or* $(\text{SDIST}(p, q).d_1 = \text{SDIST}(r, s).d_1$ *and* $\text{SDIST}(p, q).d_2 \leq \text{SDIST}(r, s).d_2)$*.*

***(4) Cluster Order***

Introduction of $\mu$ : represents minimum number of points in a cluster, equals $\mu$ , which determines best subspace for a point.

Instead of using subspace distance SDIST(p, q) to measure similarity of two points $p$ and $q$, $subspacereachability\ REACHDIST_{\mu}(p,q) = max(SDIST(p,r), SDIST(p,q))$, where $r$ is the $\mu$-nearest neighbor of $p$, is used.

→Computes a "walk" through the data set, assigning to each point $o$ its smallest subspace reachibility with respect to a point visited before $o$ in the walk = **cluster order**

```
algorithm DiSH(𝒟, μ, ε)
  co ← cluster order; // initially empty
  pq ← empty priority queue ordered by REACHDIST_μ;
  foreach p ∈ 𝒟 do
    compute w(p);
    p.REACHDIST_μ ← ∞;
    insert p into pq;
  while (pq ≠ ∅) do
    o ← pq.next();
    r ← μ-nearest neighbor of o w.r.t. SDIST;
    foreach p ∈ pq do
      new_sr ← max(SDIST(o,r), SDIST(o,p));
      pq.update(p, new_sr);
    append o to co;
  return co;
```

**Fig. 3.** The DiSH algorithm.

## Visualizing Subspace Cluster Hierarchies

Subspace clustering graph

- Consists of nodes at several levels
  - Level = subspace dimension (top level: highest subspace dim.)
  - Nodes = clusters in subspace with corresponding dim.
  - Connection between nodes = relationship between clusters, multiple parents for a cluster are allowed
  - Inner node n = cluster of all points that are assigned to n and of all points assigned to its child nodes
- One root node = all points without commo subspace, i.e. noise points

Build a subspace clustering graph

(1) Extract all clusters from cluster order

```
method extractCluster(ClusterOrder co)
  cl ← empty list;    // cluster list
  foreach o ∈ co do
    p ← o.predecessor;
    if (∄c ∈ cl with w(c) = w(o,p) ∧ dist_w(o,p)(o, c.center) ≤ 2·ε) then
      create a new c;
      add c to cl;
    add o to c;
  return cl;
```

**Fig. 5.** The method to extract the clusters from the cluster order.

(2) Build subspace cluster hierarchy: check for each cluster, if it is part of one or more higher-dim. Clusters (each cluster is at least the child of the noise cluster- root)

```
method buildHierarchy(cl)
  d ← dimensionality of objects in D;
  foreach c_i ∈ cl do
    foreach c_j ∈ cl do
      if (λ_{c_j} > λ_{c_i}) then
        d ← dist_{w(c_i,c_j)}(c_i.center, c_j.center);
        if (λ_{c_j} = d ∨ (d ≤ 2·ε ∧ ∄c ∈ cl : c ∈ c_i.parents ∧ λ_c < λ_{c_j})) then
          add c_i as child to c_j;
```

**Fig. 6.** The method to build the hierarchy of subspace clusters.