Automatic subspace clustering of high-dimensional and streaming data
Seminar Multimedia Retrieval and Data Mining

István Sárándi

Advisor: Marwan Hassani
RWTH Aachen University
istvan.sarandi@rwth-aachen.de

January 16, 2014
What is the topic?

My paper was about the CLIQUE algorithm:


I will explain the algorithm and apply it to streams as an extra.

What follows:

- Intro to clustering
- Intro to high-dimensional data
- Intro to streaming data
- In-depth description of CLIQUE
- Description of CluStream, DenStream
- Evaluation in SubspaceMOA
- Discussion
Clustering task

Find **groups** of data objects that are **similar** to each other in **the group**, but **dissimilar** to objects in **other groups**.
Clustering

Uses

- Gain new insight about structure in the data
- Compress the data by storing clusters instead of objects
- Classification in absence of labeled data

Applications

- Marketing (e.g. customer grouping)
- Computer network analysis
- Biomedical research (e.g. computational genomics)
- Computer vision (e.g. image segmentation)
- ...
High-dimensional data

Recently, datasets in many dimensions (\(\sim 10,000\))
- Computational genomics
- Text mining, etc...

New challenges
- Traditional clustering not effective, distance measures are meaningless
- Nearest neighbor distance indistinguishable from farthest "neighbor"
- Look for clusters in subspaces
- Subspace clustering, projected clustering
On another front: Streaming data

Data streams became also widespread
Infinite stream, process as generated. Consequences:

- No random access
- Keep up with input speed (be fast on average)
- Adapt to varying input speed (flexible trade-off: accuracy vs. proc. time)
- Compress unlimited amount of past data (memory management)
- Concentrate on recent data (aging)
Overview

1. Introduction

2. CLIQUE

3. Stream clustering (CluStream, DenStream) with CLIQUE

4. Evaluation
CLIQUE is a subspace clustering algorithm (subspace = set of dimensions)

- Looks for clusters in all subspaces, efficiently
- Grid-based
- Cluster = set of (subspace) grid units having at least $\tau$ objects (dense)
- Describes clusters with disjunctive normal form formulas
  $(A \land B) \lor (C \land D \land E) \lor (\ldots)$
- Suitable for exploratory (insight) analysis
CLIQUE - Main idea
CLIQUE - Main idea
CLIQUE - Main idea
CLIQUE - Steps

- Find dense cells efficiently
- Collect connected dense cells to clusters
- Cover clusters with few hyper-rectangles
CLIQUE - Finding dense cells

- Bottom-up search (apriori-like)
- Uses monotonicity: lower-dimensional projections of a dense unit are dense
- Find 1, 2, ... dimensional dense units sequentially
- Only inspect $k$-dim. cells if all $(k - 1)$ dim. projections dense
- Candidate generation (Join procedure)
- Filter out those with a non-dense projection unit
- Filter out non-dense units: Pass over the data, build histograms
CLIQUE - Candidate generation

- Take 2 units that share their first $k - 2$ dimensions and also the projection to that subspace
- Create intersection unit
CLIQUE - Candidate generation (Join)
Red arrows = Joined to create next level candidates
Dimension (attribute): \( A = (A.name, [A.min, A.max]) \)

Data space: \( \mathcal{V} = [A_1.min, A_1.max] \times \ldots \times [A_D.min, A_D.max] \)

Set of subspaces: \( S = 2\{1, \ldots, D\} \)

\( k \)-dimensional grid: \( G_k = \{0, \ldots, \xi - 1\}^k \)

\( k \)-dimensional units: \( \mathcal{U}_k = S_k \times G_k \)

Selectivity is the number of objects in the unit:
\[
\text{sel}(u) = \left| \left\{ \mathbf{v} \in \mathcal{V} \mid \text{contains}(u, \mathbf{v}) \right\} \right|
\]
1:  function FINDDENSEUNITS
2:     make one pass over $V$ and build a histogram $hist_i$ for each dimension $A_i$
3:     $Den_1 \leftarrow \{ \langle \{ i \}, (g) \rangle \mid hist_i[g] \geq \tau \}$
4:     // dense units in 1D

5:     for $k \leftarrow 2; k \leq D; k \leftarrow k + 1$ do
6:         $Cand \leftarrow JOIN(Den_{k-1})$
7:         // See Alg. 2

8:         // check all projections to $(k - 1)$ dimensions
9:         for all $(S, g) \in Cand$ do
10:             $(d_1, ..., d_k) \leftarrow sorted(S)$
11:             for all $d \in \{d_1, ..., d_{k-2}\}$ do
12:                 $u_{proj} \leftarrow (S \setminus d, (g_1, ..., g_{d-1}, g_{d+1}, ..., g_k))$
13:                 if $u_{proj} \notin Den_{k-1}$ then
14:                     // $u$ has a non-dense projection
15:                     $Cand \leftarrow Cand \setminus u$
16:                     continue loop of line 9
17:                 end if
18:             end for
19:         end for
20: for all \( u \in \text{Cand} \) do
21: \hspace{1em} \text{selectivity}[u] \leftarrow 0
22: \hspace{1em} // initialize frequency counters for candidate cells
23: end for

24: for all \( v \in V \) do
25: \hspace{1em} // pass over the data
26: \hspace{2em} for all \( u \in \text{Cand} \) do
27: \hspace{3em} if \( \text{contains}(u, v) \) then
28: \hspace{4em} // See Eq. 1 for the definition of \( \text{contains} \)
29: \hspace{4em} \text{selectivity}[u] \leftarrow \text{selectivity}[u] + 1
30: \hspace{3em} end if
31: \hspace{2em} end for
32: end for

33: \( \text{Den}_k \leftarrow \{ u \mid \text{selectivity}[u] \geq \tau \} \)
34: \( \text{Den}_k \leftarrow \text{PRUNEMDL}(\text{Den}_k, \text{selectivity}) \)
35: // See Alg. 3
36: if \( \left| \text{Den}_k \right| < 2 \) then
37: \hspace{1em} break
38: end if
39: end for

40: return \( \bigcup_{k=1}^{D} \text{Den}_k \)
41: end function
1: function \texttt{JOIN}(Den_{k-1})
2: \hspace{1em} \texttt{Cand} \leftarrow \emptyset
3: \hspace{1em} \textbf{for all } ((S, g), (S', g')) \in Den_{k-1} \times Den_{k-1} \textbf{ do}
4: \hspace{2.5em} (d_1, \ldots, d_{k-1}) \leftarrow \texttt{sorted}(S)
5: \hspace{2.5em} (d'_1, \ldots, d'_{k-1}) \leftarrow \texttt{sorted}(S')
6: \hspace{1em} \textbf{if } (\forall i \in \{1, \ldots, k-2\}: d_i = d'_i \land g_i = g'_i) \land d_{k-1} < d'_{k-1} \textbf{ then}
7: \hspace{2.5em} // \text{if same projection to the subspace of their first } (k-2) \text{ dimensions}
8: \hspace{2.5em} c \leftarrow (S \cup S', (g_1, \ldots, g_{k-2}, g_{k-1}, g'_{k-1}))
9: \hspace{2.5em} \texttt{Cand} \leftarrow \texttt{Cand} \cup \{c\}
10: \hspace{1em} \textbf{end if}
11: \hspace{1em} \textbf{end for}
12: \hspace{1em} \textbf{return } \texttt{Cand}
13: \hspace{1em} \textbf{end function}
CLIQUE - Further pruning

Still too complex to be feasible (at least in 1998...)

- Too many candidates
- Throw away dense units in "uninteresting" subspaces
- "Interestingness" = coverage of a subspace

\[
\text{cov}(S) = \left| \left\{ v \in V \mid \exists g : \text{sel}((S, g)) \geq \tau \land \text{contains}((S, g), v) \right\} \right|
\]

- Establish a cutting point and throw away lower coverage subspaces
- Cutting point: Minimum Description Length principle
Minimum Description Length principle

- We sort subspaces and split them to 2 groups: kept and pruned
- Imagine storing the following as rounded integers:
  - Mean coverage of kept subspaces
  - Mean coverage of pruned subspaces
  - Absolute deviation of each subspace from the group mean

- How many bits are needed?

\[
CL(i) = \log_2 \mu_{keep}(i) + \sum_{j=1}^{i} \log_2 |\text{cov}(S_j) - \mu_{keep}(i)| + \\
\log_2 \mu_{prune}(i) + \sum_{j=i+1}^{n} \log_2 |\text{cov}(S_j) - \mu_{prune}(i)|
\]

- Select \( i \) that minimizes this!
CLIQUE - Pruning

(source: Agrawal, 1998)
1: function PRUNEMDL(Den, selectivity)
2: \( \sigma \leftarrow \{ S \mid \exists g : (S, g) \in Den \} \)
3: for all \( S \in \sigma \) do
4: \( \text{cov}[S] \leftarrow \sum_{(S, g) \in Den} \text{selectivity}[(S, g)] \)
5: // Sum of selectivities of dense units
6: end for
7: \( n \leftarrow |\sigma| \)
8: \((S_1, \ldots, S_n) \leftarrow \text{sort} \ \sigma \ \text{by} \ \text{cov}[:] \ \text{to descending order} \)
9: \( i^* \leftarrow \arg \min_{1 \leq i \leq n} CL(i) \)
10: \( \text{Den}' \leftarrow \{(S, g) \in \text{Den} \mid S \in \{S_1, \ldots, S_{i^*}\}\} \)
11: return \( \text{Den}' \)
12: end function
Done with finding dense units!
Next step is to find which are connected
Connected component labeling by depth first search in each subspace separately
Recursive code in original paper
Now we have our clusters as sets of units in the same subspace
Better representation needed for intuition
Authors suggestion: cover cluster with union of hyper-rectangles
Goal: minimal number of hyper-rectangles for each cluster
Optimal solution NP-hard
Use greedy heuristic instead
CLIQUE - Covering

(source: Agrawal, 1998)
Greedy covering with maximal regions:

- Take an uncovered unit
- Expand a region around it in one dimension as far as dense units allow
- Expand the resulting region in another dimension etc.
- Order of dimensions is randomized
1: function GREEDYCOVER(C)
2:    \( (S, G') \leftarrow C \)
3:    uncovered \( \leftarrow G \)
4:    \( \mathcal{R} \leftarrow \emptyset \)
5:    \textbf{while} uncovered \( \neq \emptyset \) \textbf{do}
6:        \textbf{pick} \( g \in \text{uncovered} \)
7:        \( R \leftarrow \text{Rect}(\text{min} : g, \text{max} : g) \)
8:        \textbf{for all} \( d \in S \) \textbf{in random order} \textbf{do}
9:            \( R_{\text{min}} d \leftarrow \min \{ x \mid \forall g' \in \text{Rect}(R_{\text{min}} [d \leftarrow x], R_{\text{max}}) : g' \in G \} \)
10:           \( R_{\text{max}} d \leftarrow \max \{ x \mid \forall g' \in \text{Rect}(R_{\text{min}}, R_{\text{max}} [d \leftarrow x]) : g' \in G \} \)
11:        \textbf{end for}
12:        uncovered \( \leftarrow \text{uncovered} \setminus R \)
13:        \( \mathcal{R} \leftarrow \mathcal{R} \cup \{ R \} \)
14:    \textbf{end while}
15:    \textbf{return} \( \mathcal{R} \)
16: end function
Note that R3 is unnecessary. Idea: redundancy elimination
That is also NP-hard in itself! Use greedy heuristic again.

\[1\] T. Seidl. Data Mining Algorithms II - Subspace clustering - slide 19
1: function GREEDY_ELIMINATE_REDUNDANCY(\mathcal{R})
2: for all \( R \in \mathcal{R} \) in ascending order by size do
3: \[ \text{if} \ \forall g \in R : \exists R' \in \mathcal{R} \setminus \{ R \} : g \in R' \ \text{then} \]
4: \[ \mathcal{R} \leftarrow \mathcal{R} \setminus \{ R \} \]
5: end if
6: end for
7: return \( \mathcal{R} \)
8: end function
CLIQUE is complete now. Summarized:

- Find dense units level by level: candidate generation, histogram building, MDL pruning
- Collect connected units by DFS (clusters)
- Greedily cover clusters and eliminate redundancy
Since CLIQUE was one of the first subspace clustering algorithms. Since then:

- **MAFIA**: adaptively spaced grid, no MDL pruning[4]
- **SCHISM**: selectivity threshold depends on dimensionality of subspace[6]
- **Non-redundant subspace clustering**: Discard clusters that are sufficiently explained by other, higher dimensional clusters (INSCY, RESCU, OSCLU, STATPC...)
- **SUBCLU**: no grid, based on DBSCAN ("transitive clustering"), also Apriori-like
- **DUSC**: extension of SUBCLU, unbiased for dimensionality differences
Overview

1 Introduction

2 CLIQUE

3 Stream clustering (CluStream, DenStream) with CLIQUE

4 Evaluation
Let’s turn back to CLIQUE.

- Not suitable for streaming data (multi-pass, no compression or aging)
- How to make it suitable?
  - Derive totally new algorithm based on CLIQUE. (e.g. SOStream[7])
  - Two-phase approach (online/offline separation)
    - Online: maintain statistics about ”microclusters” (compression)
    - Offline: use non-streaming algorithm on the microclusters when requested
Streaming data - Microcluster approaches

Multiple such approaches exist. Now concentrate on those available in the SubspaceMOA Framework: CluStream[1] and DenStream[3].

- CluStream incrementally updates $q$ microclusters with the following data:
  - Number of objects
  - Linear sum of objects
  - Squared sum of objects
  - Linear sum of timestamps
  - Squared sum of timestamps
  - (list of identifiers of previous clusters merged into this)

- An incoming object is either merged into an existing microcluster (if there is one sufficiently nearby), or new mcluster is formed.

- To keep constant number of mclusters, discard a mcluster with old timestamps (if old enough) or merge the two nearest mclusters.

- Offline algorithm should run on recent data! Therefore: keep snapshots of the situation regularly (pyramidal timeframe).
DenStream incrementally updates p-microclusters and o-microclusters with the following data:

- Time-weighted number of objects (weighting by exponential decay)
- Time-weighted linear sum of objects
- Time-weighted squared sum of objects
- (Time of creation for o-microclusters)

An incoming object is either merged into an existing p or o-microcluster (if its variance would stay low enough), or new o-microcluster is formed. (When merging to o-mcluster, promote to p-mcluster if time-weighted object count high enough)

To keep a bounded number of mclusters, periodically discard p-microclusters with low time-weighted object count, and o-microclusters created long ago.
CLIQUE with microclusters

CluStream and DenStream yield microcluster statistics, how to use it in CLIQUE?

- Modify pass over data to pass over microclusters?
- Simpler: (re)generate objects from a distribution fitted to the microcluster
Overview

1. Introduction

2. CLIQUE

3. Stream clustering (CluStream, DenStream) with CLIQUE

4. Evaluation
SubspaceMOA

MOA Graphical User Interface

Clustering Algorithm Settings

Stream: RandomRBFSubspaceGeneratorEvents

- Setting 1
  - Micro: clustream.Clustream
  - Macro: CLIQUE
  - One-stop: predeconStream.PreDeConStream

- Setting 2
  - Micro: denstream.DenStream
  - Macro: CLIQUE
  - One-stop: hddstream.HDDStream

Evaluation Measures

- 1.0-CE
- CMM
- Entropy
- F1
- Purity
- 1.0-RNIA
- Rand statistic
- SubCMM

Start | Stop | Export stream
Evaluation datasets

A real and a synthetic dataset was used

- KDDCup’99: network intrusion data
- Synthetic dataset with two 2D clusters evolving in 3D space

Two algorithms tested

- CluStream+CLIQUE
- DenStream+CLIQUE
- Dataset with \( \sim 5 \) million objects in 41-dimensional space
- I used a corrected, newer version called NSL-KDD
- Symbolic attributes and ones that are constant over a long horizon had to be removed
- Still 15 dimensions left
### Results on synthetic data

**Table**: Results of CluStream+CLIQUE on the synthetic dataset

<table>
<thead>
<tr>
<th>1.0-CE</th>
<th>CMM</th>
<th>Entropy</th>
<th>F1</th>
<th>Purity</th>
<th>1.0-RNI</th>
<th>Rand st</th>
<th>SubCMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2656</td>
<td>0.71489</td>
<td>0.35838</td>
<td>0.75783</td>
<td>0.65952</td>
<td>0.31056</td>
<td>0.72356</td>
<td>0.65854</td>
</tr>
</tbody>
</table>

**Table**: Results of DenStream+CLIQUE on the synthetic dataset

<table>
<thead>
<tr>
<th>1.0-CE</th>
<th>CMM</th>
<th>Entropy</th>
<th>F1</th>
<th>Purity</th>
<th>1.0-RNI</th>
<th>Rand st</th>
<th>SubCMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.58571</td>
<td>0.3152</td>
<td>0.60149</td>
<td>0.60331</td>
<td>0</td>
<td>0.72356</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table**: Results of CluStream+CLIQUE on the real dataset

<table>
<thead>
<tr>
<th>1.0-CE</th>
<th>CMM</th>
<th>Entropy</th>
<th>F1</th>
<th>Purity</th>
<th>1.0-RNI</th>
<th>Rand st</th>
<th>SubCMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3E-4</td>
<td>0.86318</td>
<td>0.7008</td>
<td>0.66651</td>
<td>0.78043</td>
<td>0.00125</td>
<td>0.99635</td>
<td>0.92804</td>
</tr>
</tbody>
</table>

**Table**: Results of DenStream+CLIQUE on the real dataset

<table>
<thead>
<tr>
<th>1.0-CE</th>
<th>CMM</th>
<th>Entropy</th>
<th>F1</th>
<th>Purity</th>
<th>1.0-RNI</th>
<th>Rand st</th>
<th>SubCMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.89291</td>
<td>0.62465</td>
<td>0.66651</td>
<td>0.7168</td>
<td>0</td>
<td>0.99635</td>
<td>0</td>
</tr>
</tbody>
</table>
Discussion

- Overall CluStream had slightly better results.
- However, some metrics were not calculated correctly by SubspaceMOA (1-CE, 1-RNI, SubCMM).
- Better calibration of parameters may be necessary.
- Other concern: CluStream and DenStream were not designed for high-dimensional data. There already exist such algorithms:
  - HPStream (projected stream clustering, not only online) [2]
  - GCHDS (grid-based with its own offline component) [5]
We discussed

- relevance of clustering high-dimensional and streaming data
- details of the CLIQUE algorithm
- main ideas of microcluster approaches like CluStream and DenStream
- connecting microclusters with CLIQUE
- evaluation in SubspaceMOA
Thank you for your attention!
C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu.
A framework for clustering evolving data streams.

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu.
A framework for projected clustering of high dimensional data streams.

F. Cao, M. Ester, W. Qian, and A. Zhou.
Density-based clustering over an evolving data stream with noise.
S. Goil, H. Nagesh, and A. Choudhary.
Mafia: Efficient and scalable subspace clustering for very large data sets.
In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–452, 1999.

Y. Lu, Y. Sun, G. Xu, and G. Liu.
A grid-based clustering algorithm for high-dimensional data streams.

K. Sequeira and M. Zaki.
Schism: A new approach for interesting subspace mining.