SS-ZG548: ADVANCED DATA MINING

Lecture-04: Incremental Mining

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems Engineering,
BITS Pilani, Rajasthan-333031 INDIA

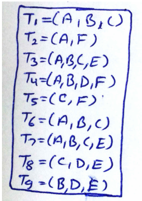
Jan 20, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

Recap

- **Association Rule Mining** involves the discovery of frequent item-sets bsed on **support** and **confidence** parameters
- Approaches involves Apriori, Hash Based (DHP), Partition Based Algorithm
- Real databases are generally dynamic $D' = D - \triangle^- + \triangle^+$
- Therefore, **Incremental** association rule mining is needed
- **F**ast **UP**date (FUP) can handle insertions
  1. An original frequent item set $X \in L$, becomes infrequent in D' iff $support(X)_{D'} < S_{min}$
  2. An item set $X \notin L$, becomes frequent in D' iff $support(X)_{\triangle^+} \geq S_{min}$
  3. If a k-item set X whose $(k-1)$-subset(s) becomes infrequent, *i.e.*, the subset is in L_{k-1} but not in L'_{k-1}, then X must be infrequent in D'.

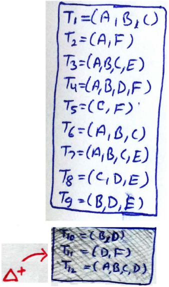
Recap: FUP at work

Consider the database D and the related frequent set discovered with Apriori

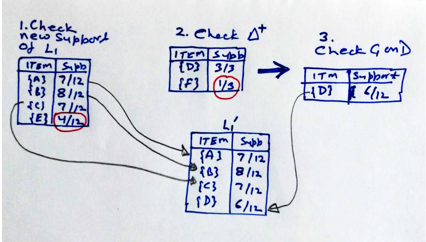


Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

Consider the arrival of $\triangle^+$ more transactions



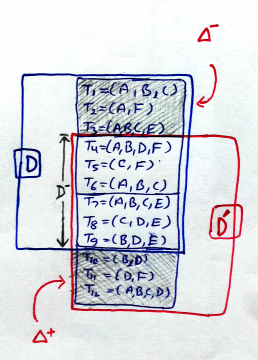
The first iteration, is as below.



Recap: FUP$_2$ at work

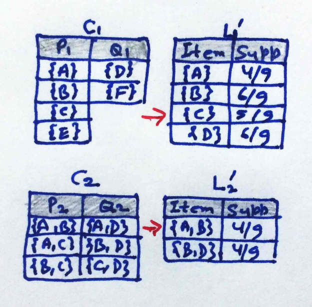
FUP$_2$ can handle insertion and deletion both

- C_k is divided into two parts. $P_k = L_k$ and $Q_k = C_k - P_k$
- Being frequent, support for all items in P_i is known. It could be updated using $\triangle^-$ and $\triangle^+$ only
- Count({A})$_{D'}$ = Count({A})$_D$ − Count({A})$_{\triangle^-}$ + Count({A})$_{\triangle^+}$ $= 6 - 3 + 1 = 4$



Item set	Support
{A}	6/9
{B}	6/9
{C}	6/9
{E}	4/9
{A B}	5/9
{A C}	4/9
{B C}	4/9
{A B C}	4/9

Frequent itemsets of D



Variations of FUP

- **Update With Early Pruning (UWEP):** Occurrence of potentially huge set of candidate itemset and multiple scans of the database is the issue
  - If a k-itemset is frequent in $_{\triangle^+}$ but infrequent in D', it is not considered when generating C_{k+1}
  - This can significantly reduce the number of candidate itemsets, with the trade-off that an additional set of unchecked itemsets has to be maintained.
- **Utilizing Negative Borders:** Negative border set consists of all itemsets that are closest to be frequent
  - Negative border consists of all itemsets that were candidates of level-vise method but did not have enough support

$$Bd^{-}(L) = C_k - L_k$$

  - Find negative border set for $L = \{\{A\}, \{B\}, \{C\}, \{E\}, \{AB\}, \{AC\}, \{BC\}, \{ABC\}\}$
  - Full scan of dataset is only required when *itemsets outside negative border set* get added to frequent itemsets or negative border set.

Variations of FUP

- **Difference Estimation for Large Itemsets (DELI):** Uses sampling technique
  - Estimate the difference between old and new frequent itemsets
  - Only if the difference is large enough, update operation using FUP_2 is performed
  - Let S be m transactions drawn from D^- with replacement, then support of itemset X in D^- is

$$\hat{\sigma_X} = \frac{T_x}{m}.|D^-|$$

where T_x is occurrence count of X in S. For large m we have 100(1-α)% confidence interval [a_x,b_x] with

$$a_x = \hat{\sigma_X} - z_{a/2}\sqrt{\frac{\hat{\sigma_X}(|D^-| - \hat{\sigma_X})}{m}}$$

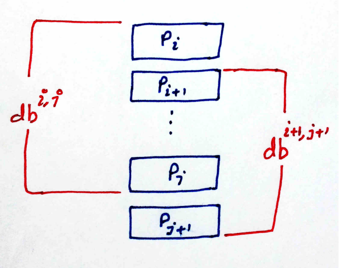
$$b_x = \hat{\sigma_X} + z_{a/2}\sqrt{\frac{\hat{\sigma_X}(|D^-| - \hat{\sigma_X})}{m}}$$

where $z_{a/2}$ is a value such that the area beyond it in standard normal curve is exactly $\alpha/2$

Sliding Window Filtering

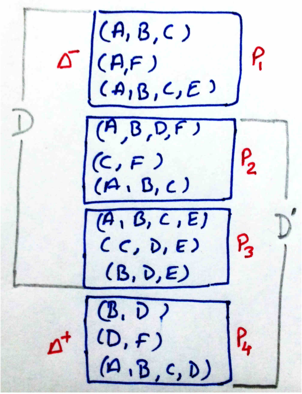
Partition-Based Algorithm for Incremental Mining: If X is a frequent itemset in a database divided into partitions p_1, p_2, ..., p_n then X must be a frequent itemset in at least one of the partitions

- Uses threshold to generate candidate itemset
- Frequent itemset remains frequent from some P_k to P_n
- A list of 2-itemsets CF is maintained to track possible frequent 2-itemsets.
- Locally frequent 2-itemsets of each partition is added (with its starting partition and supports)
- Scan reduction technique can make one database scan enough



SWF at work

With $S_{min} = 40\%$ generate frequent 2-itemsets



P_1

Item	Start	Count
{A,B}	1	2
{A,C}	1	2
{B,C}	1	2

P_2

Item	Start	Count
{A,B}	1	4
{A,C}	1	3
{B,C}	1	3

P_3

Item	Start	Count
{A,B}	1	5
{A,C}	1	4
{B,C}	1	4
{B,E}	3	2
{C,E}	3	2
{D,E}	3	2

No new 2-itemset added when processing P_2 since no extra frequent 2-itemsets. Moreover, the counts for itemsets {A,B}, {A,C} and {B,C} are all increased. Their counts are no less than 6×0.4

SWF at work

- Scan reduction technique is used to generate C_k $(k = 2, 3, ..., n)$ using C_2
- C_2 is used to generate the candidate 3-itemsets and its sequential C'_{k1} be utilized to generate C'_k
- C'_3 generated from $C_2 * C_2$ instead of $L_2 * L_2$ will have size greater but near to $|C_3|$
- Second scan would suffice for pruning

Merit of SWF lies in its incremental procedure. There are three sub-steps

- Generating C_2 in $D^- = db^{1,3} - \triangle^-$
- Generating C_2 in $db^{2,4} = D^- + \triangle^+$
- Scanning $db^{2,4}$ once

$db^{1,3} - \Delta^- = D^-$

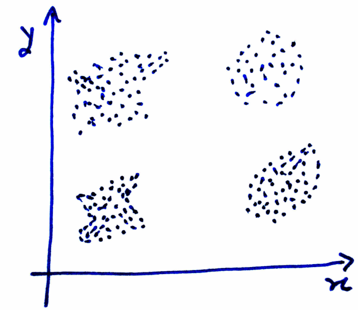
Itemset	Start	Count
{A,B}.	2	3
{A,C}	2	2
{B,C}.	2	2
{B,E}.	3	2
{C,E}.	3	2
{D,E}.	3	2

$D^- + \Delta^+ = D'$

Itemset	Start	Count
{A,B}	2	4
{B,E}	3	2
{C,E}	3	2
{D,E}	3	2
{D,B}	4	3

Clustering in dynamic databases

- Can we use *k*-Means clustering algorithm?

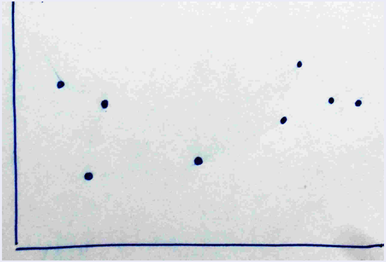


  - Randomly choose *k* data points as centroid
  - Assign each data point to closest centroid
  - Update centroid until converge
  - Typically SSD (sum of square error) is optimized
- What about PAM (partitioning around medoids)? NO
- Single/Average/Farthest link clustering?

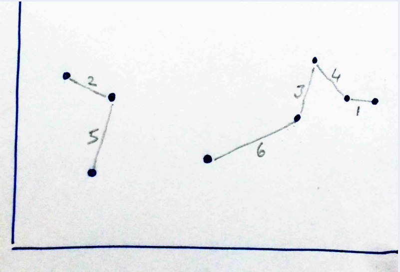
Single Link

1. Starts with each point as cluster
2. Merges two nearest clusters $n - k$ times

Consider following data points in 2D space



Finally we get

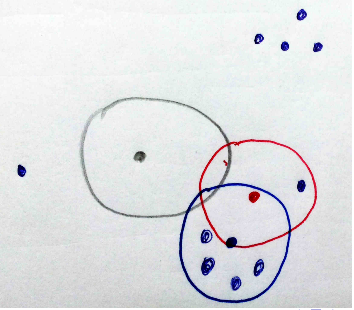


Can we make it adaptable for dynamic databases?

Incremental DBSCAN

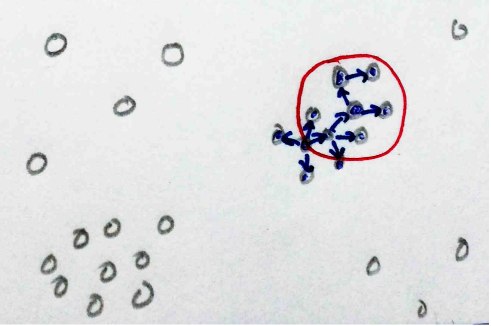
It is a density based spatial clustering algorithm

- Density-Based Spatial Clustering of Applications with Noise (KDD96)
- Can discover clusters of arbitrary shape
- Uses notion of density and 2 parameters Eps and MinPts
- Points are first classified as core (has MinPts in Eps radius), border (has a core in Eps radius), or noise (otherwise)
- Performs DFS starting on unassigned core point

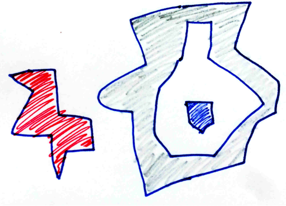


DBSCAN at Work

How It works

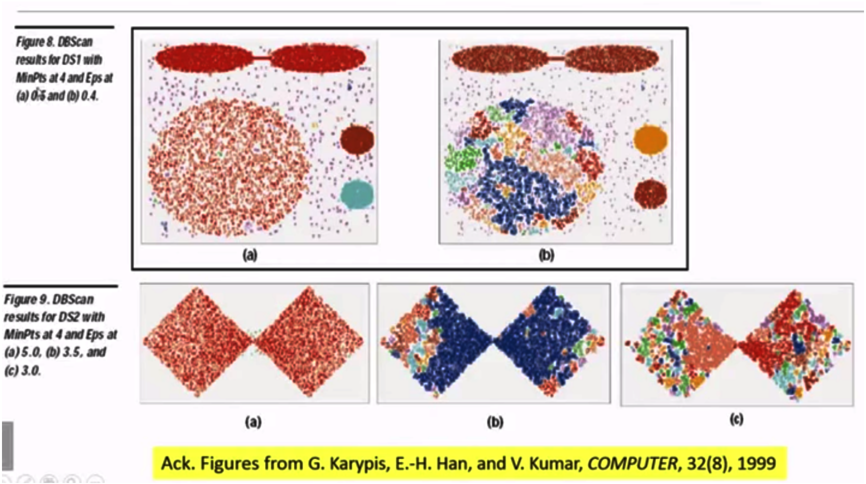


Advantages

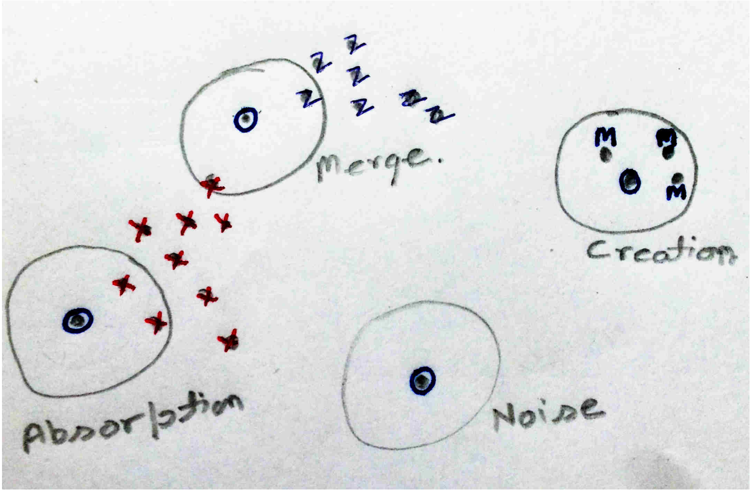


Drawback

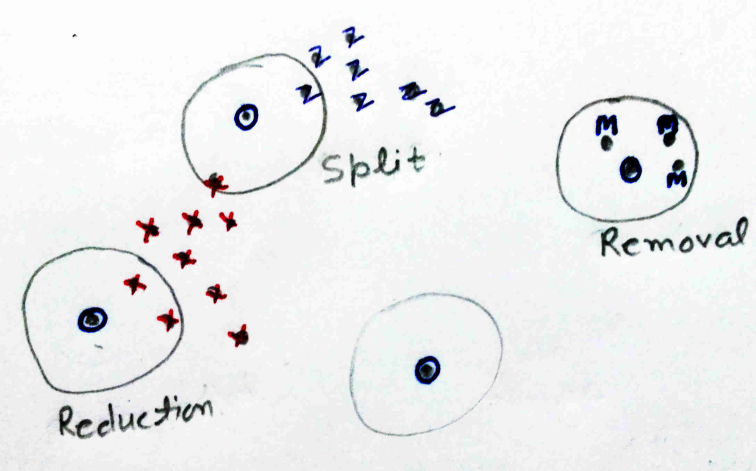
Sensitive to setting parameters



Incremental DBSCAN (Addition)



Incremental DBSCAN (Deletion)



Incremental DBSCAN

- Insertion and deletion are treated separately
- Based on change in density in affected region, clusters are updated.
- Update cost is proportional to number of points in affected region that is high
- You may be doing redundant operations

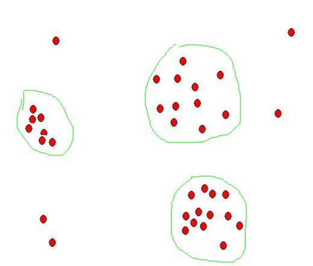
Differ update for some time

Assume periodic arrival of updates.

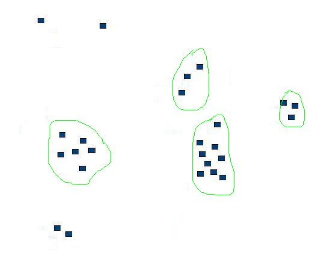
- Cluster new data
- Merge it with previous clusters (it is easy to see the density change)

Incremental DBSCAN

Initial Database

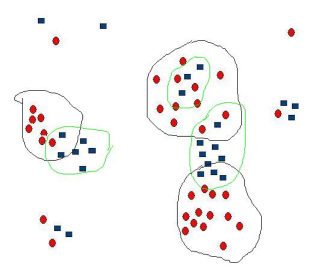


New Date



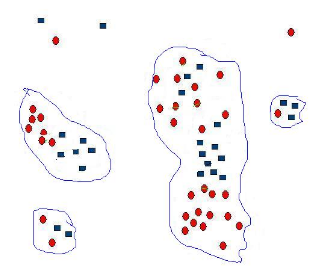
- Region based merging is applied

Incremental DBSCAN



Overlapping of clusters made from original and the new data points looks as

- Point p is in set of intersection I' if $\exists p' \in D$ such that p and p' are neighbor
- It is necessary an sufficient to process all $p \in I'$
- Efficiently compute I'. How?



Thank You!

Thank you very much for your attention!

Queries ?