TDDD41: Datamining - Clustering and Association Analysis
Lecture Notes

Pontus Persson

VT-2016

Contents

1 Partition Methods

Partitions. No overlap, not empty, all should be in. k-mean - problem with outliers. Mitigate with metoids (PAM). CLARA, CLARANS?

2 Hierarchical Methods

Lecture 3
2016-01-27

Build clusters with sub-clusters, by merging or splitting clusters.

AGNES agglomatric

DIANA division

Complete-link clustering uses the longest distance of the objects inside a cluster. (the maximum path).
Single-link uses the shortest distance instead.
Build the tree graph of clustering and introduce a threshold to see how many clusters it gets.

2.1 AGNES

- Implemented in static analysis
- Single-link

2.2 DIANA

- Reverse order

Disadvantages - DO NOT SCALE! $O(n^2)$. Can not undo.

2.3 Birch

Balanced Iterative Reducing and Clustering using Hierarchies.

1. Scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)

2. Use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

Clustering feature: $CF = (N, LS, SS)$ $LS = \sum X_i$, $SS = \sum X_i^2$ X is coordinates.
EX: CF=(5,(16,30),(54,190)).

Centroid $\frac{LS}{N}$

Add two clusters together $CF_1 + CF_2$

A CF-tree is a height balanced tree (all leaves at the same level). All nodes include CF, a nonleaf node in a tree stores the sums of the CFs of their children. Leaves stores the actual data.

Branch factor Maximum number of children

threshold max diameter of sub-clusters stored at the leaves

When creating the CF-tree, add element to same node if it fits. Otherwise "split", take the two object with the furthest distance, put one of them in the first cluster and the other one in the second (newly created) cluster. Add the third element in the cluster where the other object is the closest. As soon as the leaves are created, create a parent with the sum of its children's CFs.
Next element: start at top of tree, check centroid of children and choose the closest one.

```
If T requirement satisfied
then if data in cluster is not too large
        Add data point
else
        split
```

After creating the tree we can use for example PAM. Scales linearly. Handles only numeric data and is sensitive to the order of the data.

2.4 ROCK

Robust, Clustering using Links. Works on categorical data.

Definition 1. *p_1 and p_2 are neighbours*
iff $sim(p_1, p_2) \geq t$ and $sim \in [0, 1]$

Definition 2. *$Link(p_1, p_2)$ is the number of common neighbours.*

Definition 3. *$Link(C_i, C_j)$ = the number of cross links between clusters C_i, C_j.*
$\sum Link(p_i, p_j)$

Definition 4. *Goodness $G(C_i, C_j)$ when to merge = $Link(C_i, C_j)$ divided by the expected number of cross links.*

- Drag random sample
- Hierarchical clustering with links using goodness
- Label data in disk, add other data to cluster with most neighbours.

Lecture 4
2016-02-02

Expected number of cross-links between C_i and C_j. = Expected number of links in $C_i \cup C_j -$ Expected number of links in $C_i -$ Expected number of links in C_j.
Expected number of links in C_i = number of data points in $C_i *$ contribution of one data point to the "links" in C_i
Assume m_i data points in C_i. Assume each point in C_i has $m_i^{f(t)}$ neighbours. T is similarity threshold. $f(t) \leq 1$. For market data $f(t) = \frac{1-t}{1+t}$.
Data point contributes to link when it is a common neighbour for two data points.

- It is a common neighbour for:
  - Its own neighbours
- How many times is it a common neighbour?
  - For each pair of its neighbours. $= m_i^{2f(t)}$

Definition 5. *Expected number of links in* $C_i = m_i^{1+2f(t)}$

Definition 6. *Expected number of links in* $C_i \cup C_j = (m_i + m_j)^{1-2f(t)}$

2.5 CHAMELEON: Hierarchical Clustering Using Dynamic Modeling

1. Use a graph partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
   - Based on k-nearest neighbour graph (connect two if one is among k nearest neighbours)
   - Similarity on edges
   - Edge weight is density of region
2. Use an agglomerative hierarchical clustering algorithm:
   - Edge cut: cut the lowest edges to create separated graphs
   - Cut either until minimum number of edges in a cluster. Maximum $1 - 2\%$ of data points in a cluster
   - Merge if
     - Closeness of clusters: avg similarity between points in the clusters that are connected to both clusters.
     - Interconnectivity of clusters: normalized sum of the weights of the edges that connect nodes int the clusters. $\frac{2 * EC_{C_i, C_j}}{EC_{C_i} - EC_{C_j}}$. $EC_{C_i} =$ When dividing C_i in two approx. equal parts.
     - Merge if both measurements are above user=defined thresholds

3 Density-Based Clustering

- Eps: Max radius of the neighbourhood
- MinPts: Min. number of points in an Eps-neighbourhood of that point.

Definition 7. *A point is density reachable from a point q if there is a chain of points such that each are density reachable.*

Definition 8. *A pont p is density-connected to a point q if there is a point o that is density-reachable to both p and q.*

Lemma 1. *p density-reachable to q* $\Rightarrow$ *p density-connected to q*

Lemma 2. *If p DDR for q* $\Rightarrow$ *p DR for q*

3.1 DBSCAN

- Put all density-connected points in a cluster.
- Core points
- Border points, connected to a core point, but not enough neighbours to be a core point
- Outliers, not connected to a core point and not enough to be a core point

Algorithm:

1. Select point p
2. Retrieve all density-reachable points
3. If p is core point, form cluster
4. If p is a border point, no point are density-reachable from p and DBSCAN visits next point from the database
5. Continue untill all of the points have been processed

3.2 OPTICS

Order the data in such a way that you can look at different epsilons at the same time (preprocessing).

Definition 9. *Core distance is the smallest Eps needed between p and an object in its eps-neighbourhood such that p would be a core object.*

Definition 10. *Reachability distance of p and o*
max(core-distance(o),d(o,p)) if o is a core object

Algorithm:

1. Select non-processed object o
2. Find neighbours (eps-neighbourhood)

3. Compute core distance for o
4. Mark o as processed
5. If o is not a core, restart at (1)
6. Put neighbours of o in Seedlist and order
    - If neighbour n is not yet in Seedlist then add (n, reachability from o) else if reachability from o ¡ current reachability, then update reachability + order Seedlist wrt reachability
7. Take new objec from SeedList with smallest reachability and go to (2)

You can then plot the new order (x) with eps (y) to see how many clusters and outliers you get for different eps.

Lecture 6
2016-02-16

4 Association rules

Look for patterns on form *antecedent* $\rightarrow$ *consequent*. "If something happens something else with happen with a certain probability". Can be used for revenue maximization. Market basket analysis.

Definition 11. *Frequent itemset. A set of items bought together frequently*

Association rules can be produced from the frequent itemsets.

Definition 12. *Support* $= p(X, Y) =$ *probability that a transaction contains* $X \cup Y$

Definition 13. *Confidence* $= p(Y|X) = \frac{p(X,Y)}{p(X)}$

High support, many cases. High confidence, rule is likely to be "true". **Goal:** Find all rules $X \rightarrow Y$ with at least minimum support and confidence.
Solution:

- Find all itemsets with minimum support (Apriori, FP grow algorithms)
- Generate all the rules with minimum confidence from the frequent itemsets

Theorem 1. *Any subset of a frequent itemset is frequent. Or, any superset of an infrequent itemset set is infrequent*

4.1 Apriori algorithm

1. Scan db once to get frequent 1-itemsets
2. Generate candidates to frequent (k+1)-itemsets from frequent k-itemsets
3. Test the candidates against database
4. Terminate when no frequent or candidate itemsets can be generated

Create new candidates from candidates with the same prefix (all elements but the last one).

1. Self joining
2. Pruning (Prune candidates whose subsets are not frequent)

Lemma 3. *All frequent (k+1)-itemsets are in C_{k+1}*

Proof. $k = 0, L_1 \subseteq C_1$ Trivial
Hypothesis: $L_s \subseteq C_s \forall s \leq k$
Show that it wont be removed in self-join and pruning. □

4.1.1 Produce rules

Generate all rules on the form $a \rightarrow l - a$ with minimum confidence from a large itemset l.
If a subset $a \subseteq l$ does not generate a rule, then neither does any subset of a.
Start with rules with a large antecedent and try to make new rules with smaller antecedents of subsets of "good rules".
Problems with Apriori:

- Too many candidates to generate
- Each candidate implies expensive operations

Lecture 7
2016-02-17

4.2 Frequent pattern growth algorithm

1. Scan db once, record items as the frequent 1-itemsets.
2. Sort itemset in frequency descending order. (f-list)
3. Create tree
   (a) First transaction is put as left children
   (b) Put next transaction in tree and create minimum new branches. Mark nodes with number of transactions it represents
4. Also create header table linking all items to their place is the tree. If there are multiple instances of an item in the tree, the nodes should keep track of each other.
5. For each frequent item in the header table
   - Traverse the tree by following the corresponding link
   - Record all prefix paths leading to the item. This is the item's conditional pattern base
6. For each conditional pattern base, Start the process again recursively

Definition 14. $lift(X,Y) = \frac{support(X,Y)}{support(X)support(Y)} = \frac{p(Y|X)}{p(Y)}$

- $lift > 1$ *positive correlation*
- $list < 1$ *negative correlation*
- $lift = 1$ *independent*

Lecture 8
2016-02-23

4.3 Constrained frequent itemset mining

Only find rules that apply to a specified constraint.
A Constraint C(.) is:

Monotone If $C(A)$ then $C(B), \forall A \subseteq B$

Antomonotone •

- If $C(A)$ then $C(B), \forall B \subseteq A$
- If not $C(B)$ then not $C(A)$

Example:

$$sum(S.price) \geq v \text{ Is monotone}$$
$$sum(S.price \leq v \text{ Is anti monotone}$$

Constraints that are not monotone or anti monotone can sometimes be converted.

4.3.1 Apriori algorithm with constraints

You could remove frequent itemsets not matching the constraints with a simple brute force approach.

- If anti monotone, cross off items early to not do any extra work. (Prune search space)
- If monotone, does not prune, but avoids constrains checking. Superset of constraint OK is always OK.

4.3.2 FP grow algorithm and constraints

With anti monotone constraints:

- Remove items that do not satisfy the constraint
- If the conditioning itemset α does not satisfy the constraint, then do not generate α nor its conditional database.
- Let β denote the frequent items in the conditional database database of α. If $\alpha \cup \beta$ satisfies the constraint, then do not check the constraint in the conditional database of α.

With monotone constraints:

- If the conditioning itemset α satisfies the constraint, then do not check the constraint in its conditional database.

4.4 Constraints

- Average is neither monotone nor anti monotone
- Convertible monotone
  - If there exists an item order R such that:

    - If $C(A) \Rightarrow C(B), \forall A, B$ respecting R such that A is a suffix of B.
  - E.g. $avg(S.Price) \geq v$ with respect to decreasing price order

- Convertible anti monotone
  - It there exists an item order R such that:
    - If $C(A) \Rightarrow C(B), \forall A, B$ respecting R such that B is a suffix of A
    - Or if $\neg C(B) \Rightarrow \neg C(A), \forall A, B$ respecting R such that B is a suffix of A
  - E.g. $avg(S.Price) \geq v$ with respect to increasing price order

Definition 15. *If a constraint is* ***both*** *convertible monotone and convertible anti monotone, it is* ***strongly convertible****.*

Lecture 9
2016-02-24