Clustering aggregation

Clustering aggregation

- Given a set of clusterings or a set of categorical variables
- How to combine them to a single clustering?
-

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}$
v_1	1	1	1	1
v_2	1	2	2	2
v_3	2	1	1	1
v_4	2	2	2	2
v_5	3	3	3	3
v_6	3	4	3	3

- Gionis, Mannila, Tsaparas: Clustering aggregation, ACM Transactions on Knowledge Discovery from Data 1, 1 (2007)

Why?

- Different clustering algorithms produce different answers; which one to believe?
- Clustering categorical data: no natural similarity measures

Definitions

- n objects $V = \{v_1, \ldots, v_n\}$.
- A clustering $\mathcal{C}$ of V is a *partition* of V into k disjoint sets $C_1, \ldots, C_k$
- The k sets $C_1, \ldots, C_k$ are the clusters of $\mathcal{C}$.
- $\mathcal{C}(v)$ the label of the cluster to which the object v belongs, i.e., $\mathcal{C}(v) = j$ if and only if $v \in C_j$
- m clusterings: $\mathcal{C}_i$ to denote the ith clustering
- k_i for the number of clusters of $\mathcal{C}_i$

How to compare two clusterings?

- u and v in V
- $$d_{u,v}(\mathcal{C}_1, \mathcal{C}_2) = \begin{cases} 1 & \text{if } \mathcal{C}_1(u) = \mathcal{C}_1(v) \text{ and } \mathcal{C}_2(u) \neq \mathcal{C}_2(v), \\ & \text{or } \mathcal{C}_1(u) \neq \mathcal{C}_1(v) \text{ and } \mathcal{C}_2(u) = \mathcal{C}_2(v), \\ 0 & \text{otherwise.} \end{cases}$$
- $$d_V(\mathcal{C}_1, \mathcal{C}_2) = \sum_{(u,v) \in V \times V} d_{u,v}(\mathcal{C}_1, \mathcal{C}_2).$$

Problem definition

Problem 1 (Clustering aggregation) *Given a set of objects V and m clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_m$ on V, compute a new clustering $\mathcal{C}$ that minimizes the total number of disagreements with all the given clusterings, i.e., it minimizes*

$$D(\mathcal{C}) = \sum_{i=1}^{m} d_V(\mathcal{C}_i, \mathcal{C}).$$

Example

Equivalent to finding the "center" of the clusterings $\mathcal{C}_i$ with respect to the measure d_V

The distance is a metric

Observation 1

$$d_V(\mathcal{C}_1, \mathcal{C}_3) \leq d_V(\mathcal{C}_1, \mathcal{C}_2) + d_V(\mathcal{C}_2, \mathcal{C}_3)$$

Why?

Show that for each pair (u, v) we have
$d_{u,v}(\mathcal{C}_1, \mathcal{C}_3) \leq d_{u,v}(\mathcal{C}_1, \mathcal{C}_2) + d_{u,v}(\mathcal{C}_2, \mathcal{C}_3)$.

Correlation clustering

- A slightly more general problem
- **Problem 2 (Correlation clustering)** *Given a set of objects V, and distances $X_{uv} \in [0, 1]$ for all pairs $u, v \in V$, find a partition $\mathcal{C}$ for the objects in V that minimizes the score function*

$$d(\mathcal{C}) = \sum_{\substack{(u,v) \\ \mathcal{C}(u)=\mathcal{C}(v)}} X_{uv} + \sum_{\substack{(u,v) \\ \mathcal{C}(u)\neq\mathcal{C}(v)}} (1 - X_{uv}). \qquad (1)$$

Clustering aggregation is a special case

- $X_{uv} = \frac{1}{m} \cdot |\{i \mid 1 \leq i \leq m \text{ and } \mathcal{C}_i(u) \neq \mathcal{C}_i(v)\}|$
- The *fraction* of clusterings that assign the pair (u, v) into *different* clusters
- For any clustering $\mathcal{C}$, if $\mathcal{C}$ places u, v in the same cluster it will disagree with mX_{uv} of the original clusterings
- If $\mathcal{C}$ places u, v in different clusters it will disagree with the remaining $m(1 - X_{uv})$ clusterings.
- For any clustering $\mathcal{C}$ $m \cdot d(\mathcal{C}) = \sum_{i=1}^{m} d_V(\mathcal{C}, \mathcal{C}_i) = D(\mathcal{C})$,

Triangle inequality

Observation 2 *For all u, v and w in V, we have that* $X_{uw} \leq X_{uv} + X_{vw}$.

Algorithms for clustering aggregation

- What types of algorithms would we like to find?
- Simple
- Good in practice
- Methods about which we can say something

BestClustering algorithm

- Given m clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_m$
- BestClustering finds the input clustering $\mathcal{C}_i$ that minimizes the total number of disagreements $D(\mathcal{C}_i)$
- Can be implemented to work in time $O(m^2 n)$

What can be said about the algorithm?

- Does not really aggregate
- Just selects the existing clustering that is most central
- Score is no more than 2 time the optimum
- Why?
- Not very good in practice

BALLS algorithm

- Works for the correlation clustering problem (and thus also for the clustering aggregation problem)
- Try to find a set of vertices that are close to each other and far from other vertices
- Such a set is a cluster
- How to find such a set? Lots of possible sets
- Triangle inequality, again

The algorithm

- Pick the first unclustered node u in some ordering
- Find the set of nodes B that are at a distance of at most $1/2$ from u
- Compute the average distance $d(u, B)$ of the nodes in B to node u
- If $d(u, B) \leq \alpha$ then the nodes in $B \cup \{u\}$ are considered to form a cluster;
- Otherwise, node u forms a singleton cluster.

Properties of the algorithm

- If$\alpha = \frac{1}{4}$, the cost of solution produced by the BALLS algorithm is guaranteed to be at most 3 times the cost of the optimal clustering.
- Not very good in practice
- $\alpha = 2/5$ seems better
- Complexity: $O(mn^2)$ for computing the distances X_{uv}, $O(n^2)$ for running the algorithm

The AGGLOMERATIVE algorithm

- Correlation clustering problem
- Agglomerative clustering algorithm
- Start with all nodes in singleton clusters
- Merge the two clusters with the smallest cost
- Cost: the average weight of edges between clusters
- If this is less than 1/2, merge; otherwise, stop

The FURTHEST algorithm

- Correlation clustering
- In the beginning all nodes are in a single cluster
- Find the pair of clusters that are furthest apart
- Make them new cluster centers
- Reassign points to the closest cluster center
- Find the node that is furthest away from existing centers; repeat
- If the cost is lower that in the previous step, continue, otherwise, the result from the previous step is the answer

Properties of the algorithm

- $O(mn^2)$ for creating the weights
- $O(k^2n)$ for running the algorithm; k is the number of clusters created

The LocalSearch algorithm

- A heuristic that can be applied on top of any correlation clustering method
- Start with some clustering
- For each node: find if the cost would improve if the node were moved to another cluster or made into a singleton cluster
- Iterate

Efficient implementation

- Cost $d(v, C_i)$ of assigning a node v to a cluster C_i

$$d(v, C_i) = \sum_{u \in C_i} X_{vu} + \sum_{u \in \overline{C_i}} (1 - X_{vu}).$$

- The first term is the cost of merging v in C_i
- The second term is the cost of *not* merging node v with the nodes not in C_i

- More efficient formulation: for every C_i we compute and store $M(v, C_i) = \sum_{u \in C_i} X_{vu}$ and the size of the cluster $|C_i|$.
-

$$d(v, C_i) = M(v, C_i) + \sum_{j \neq i} (|C_j| - M(v, C_j))$$

- The cost of assigning node v to a singleton cluster is $\sum_j (|C_j| - M(v, C_j))$.
- Given the distance matrix X_{uv} (takes time $O(mn^2)$)
- The running time of the LocalSearch algorithm is $O(Tn^2)$
- T is the number of local search iterations

Algorithm	k	$I(\%)$	E_D
Class labels	2	0	34,184
Lower bound			28,805
BestClustering	3	15.1	31,211
Agglomerative	2	14.7	30,408
Furthest	2	13.3	30,259
Balls$_{\alpha=0.4}$	2	13.3	30,181
LocalSearch	2	11.9	29,967
ROCK$_{k=2,\theta=0.73}$	2	11	32,486
LIMBO$_{k=2,\phi=0.0}$	2	11	30,147

Results on **Votes** dataset. k is the number of clusters, I is the impurity index, and E_D is the disagreement error. The lower bound on E_D is computed by considering an algorithm that merges all edges with weight less than $\frac{1}{2}$, and splits all edges with weight greater than $\frac{1}{2}$.

Large datasets

- The algorithms typically have a running time of $O(mn^2)$
- For large values of n this is too much
- Sampling algorithm: take a sample, do clustering aggregation
- Postprocessing: for each node (row), see which cluster it fits best or whether it should be a cluster of its own
- In the same way as in the LocalSearch algorithm