Clustering Trees with Instance Level Constraints (Extended Abstract)\(^1\)

Jan Struyf\(^a\)  
Sašo Džeroski \(^b\)

\(^a\) Dept. of Computer Science, Katholieke Universiteit Leuven  
Celestijnenlaan 200A, 3001 Leuven, Belgium  
\(^b\) Dept. of Knowledge Technologies, Jožef Stefan Institute  
Jamova 39, 1000 Ljubljana, Slovenia

Clustering methods partition a given set of instances into subsets (clusters) such that the instances in a given cluster are similar. Traditional clustering algorithms, such as \(k\)-means and hierarchical agglomerative clustering, are unsupervised, that is, they only have access to the attributes describing each instance; no direct information about the actual assignment of instances to clusters is available. This distinguishes clustering from supervised classification, where the class of each instance is given.

Over the past five years, constrained clustering methods have become popular, motivated by applications such as gene clustering, document clustering, web search result clustering, and lane finding from GPS traces. Constrained clustering investigates how domain knowledge can improve clustering performance. Domain knowledge is given as a set of constraints that must hold on the clusters. We consider two common types of instance level (IL) constraints (Fig. 1.a): must-link and cannot-link constraints [3]. A must-link constraint \(ML(a,b)\) specifies that instances \(a\) and \(b\) must belong to the same cluster, and a cannot-link constraint \(CL(a,b)\) specifies that \(a\) and \(b\) must not be placed in the same cluster. IL constraints provide additional information about the assignment of instances to clusters. Clustering with IL constraints is therefore considered to be a form of semi-supervised learning. IL constraints have been successfully incorporated into popular clustering algorithms, such as \(k\)-means [3]. This paper investigates how clustering trees can support IL constraints.

Clustering trees are decision trees that are used for clustering [1]. Each leaf of a clustering tree corresponds to a cluster and is labeled with the cluster’s centroid. Similar to regular decision trees, the internal nodes of a clustering tree contain attribute-value tests. The main advantage of clustering trees over other clustering methods is that they provide a symbolic description for each cluster (the tests in the internal nodes). For example, cluster \(C_1\) in Fig.1 is the set of instances for which \(Y > 103.5\) and \(X \leq 113.5\). Clustering trees are constructed with a recursive top-down decision tree induction algorithm, similar to C4.5, but using as heuristic weighted average variance reduction instead of information gain.

A disadvantage of clustering trees is that they only allow conjunctive cluster descriptions. This corresponds to rectangular clusters in the two-dimensional case. One of the main goals of constrained clustering is dealing with non-trivial cluster shapes. We therefore extend clustering trees to support disjunctive cluster descriptions. To this end, we introduce cluster labels in the leaves of the clustering tree. We call a clustering tree with such labels a disjunctive clustering tree. All leaves that share the same label make up one cluster. For example, the L-shaped cluster \(C_2\) in Fig.1.a is represented by two leaves in Fig.1.b and its disjunctive description is “\(Y \leq 103.5 \lor (Y > 103.5 \land X > 113.5)\)”.

\(^1\)This is the extended abstract of [2].
The paper’s central contribution is CLUSILC, an efficient algorithm that builds a disjunctive clustering tree given a set of instances and a set of IL constraints. CLUSILC performs a greedy search through the space of disjunctive clustering trees. As a result, there is no guarantee that CLUSILC will find a tree that satisfies all IL constraints (if such a tree exists), but in most cases it will find a tree that satisfies a sufficiently large number of constraints. CLUSILC’s heuristic consists of two components: the tree leaves’ weighted average variance and the number of constraints violated by the tree. The smaller the heuristic value (smaller variance or fewer constraints violated), the better the tree. CLUSILC’s heuristic is not compatible with a standard recursive top-down decision tree induction algorithm because it is global: the heuristic value of a test in a given node also depends on the other nodes of the tree because the node’s instances may be linked with IL constraints to instances in other nodes. CLUSILC therefore uses a different search strategy. Instead of adding nodes in a depth-first manner, it works best-first. In each main loop iteration, it selects the refinement that maximally reduces the heuristic value. A refinement consists of replacing a leaf of the tree by a new internal node with two new leaves (we consider binary trees). The labels are assigned to the leaves based on the IL constraints; they are chosen such that as few as possible constraints are violated.

CLUSILC contains a number of optimizations for efficiently finding “$x > a$” tests for a numeric attribute $x$ and for assigning cluster labels to the leaves that result from each such candidate split. This part of the algorithm is similar to the corresponding part in a standard decision tree learner (it relies on sorting the instances based on their value for $X$), but includes a number of modifications to deal with the IL constraints.

The experimental evaluation compares CLUSILC to COP-$k$-means [3], which is a $k$-means implementation that supports IL constraints. COP-$k$-means greedily assigns instances to their closest cluster such that no constraints are violated. That is, if it returns a clustering, then all constraints are guaranteed to be satisfied. We base the evaluation on classification data from the UCI repository and randomly generate IL constraints consistent with the classes in the data (this is often done to benchmark constrained clustering algorithms). We run experiments on 12 data sets with varying numbers of constraints and use the Rand index to evaluate clustering performance. Following Wagstaff et al. [3], we report the Rand index for all data and also the 10 fold cross-validated Rand index (computed on the held-out sets, for which no IL constraints were available to the learner).

For the all data evaluation, COP-$k$-means performs better than CLUSILC for a sufficiently large number of IL constraints (depending on the data size). The reason is that COP-$k$-means only returns a clustering if it can satisfy all constraints: given enough constraints, the Rand index will become 1.0 if COP-$k$-means can find a solution. CLUSILC, on the other hand, may also return a solution that does not satisfy all constraints. This can happen either because, even if a solution exists, the greedy search fails to find it, or because the target concept cannot be expressed as a clustering tree with the given set of features. In the cross-validated results, CLUSILC performs better than COP-$k$-means on 6/12, comparable on 5/12 and worse on 1/12 data sets. The reason for the good generalization performance of CLUSILC is that it can represent more complex clusters than COP-$k$-means, which essentially assumes spherical clusters (i.e., a strong bias).

To conclude, this paper’s chief contribution is CLUSILC, an algorithm for constructing a disjunctive clustering tree given a set of instances and IL constraints. The main advantage of using such trees over other clustering methods is that they provide a symbolic description for the clusters. While COP-$k$-means performs better on all data, CLUSILC has a better or comparable generalization performance. In future work, we plan experiments comparing CLUSILC to other constrained clustering algorithms and plan to investigate how other constraint types, such as constraints on the tree size, can be integrated in CLUSILC.

Acknowledgments: JS is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

References

