Spanning Tree Based Attribute Clustering

Yifeng Zeng, Jorge Cordero Hernandez, Shuyuan Lin

{yfzeng,corde}@cs.aau.dk,S030602108@fzu.edu.cn
Department of Computer Science, Aalborg University, DK-9220 Aalborg, Denmark
Department of Computer Science, Fuzhou University, FuJian, P.R.China

Abstract. Attribute clustering has been previously employed to detect statistical dependence between subsets of variables. We propose a novel attribute clustering algorithm motivated by research of complex networks, called the Star Discovery algorithm. The algorithm partitions and indirectly discards inconsistent edges from a maximum spanning tree by starting appropriate initial modes, therefore generating stable clusters. It discovers sound clusters through simple graph operations and achieves significant computational savings. We compare the Star Discovery algorithm against earlier attribute clustering algorithms and evaluate the performance in several domains.

Key words: Maximum Spanning Tree, Clustering

1 Introduction

Probably one of the widest use of clustering in the past years has been the task of selecting genes (variable selection) in bioinformatics. The use of attribute clustering can be extended to any domain in the search for statistical correlation of variables. Several conventional clustering algorithms have been applied to regroup and reveal subsets of correlated attributes such as: the k-means algorithm [1], fuzzy clustering [2] and hierarchical clustering [3].

Recently, the k-modes algorithm [4] has been proved as one of the most efficient approaches for performing attribute clustering. However, it is subject to local optima due to a random selection of initial modes. In a parallel line, clustering based on tree partition receives more and more attention since it is firmly rooted in classical graph partition methods (detailed methods will be presented soon in the next section). More precisely, the clustering methods firstly build a maximum spanning tree (MAST) and then get the clusters using appropriate partition methods. For convenience, we call the methods as MAST-based clustering algorithms in this paper. Since the standard tree partition method is not directly oriented toward attribute clustering it may not produce competitive results. However, it avoids heavy computation in contrast with k-modes algorithm. Accordingly, the MAST-based clustering algorithms contribute to the growing line of research on attribute clustering.

For the effect of this investigation we focus on the MAST-based clustering method. Specifically, we introduce the Star Discovery (SD) algorithm that is inspired by the research of complex networks [5]. We adopt the assumption that

all variables can be seen as points in an Euclidean space (close points have a high correlation) because we have complete information regarding pairwise proximities. The SD algorithm sections the tree by detecting nodes which have a strong connectivity; then, it pulls neighboring nodes into clusters based on a simple heuristic. We compare our approach against both earlier tree-based clustering algorithms and the k-modes algorithm in comprehensive experiments.

2 Background

Given n domain attributes, $X = \{x_1, \cdots, x_n\}$, clustering methods aim to group a set of attributes[1] into clusters based on a similarity measure. In general, attributes in a cluster are more correlated to each other than to those ones belonging to different clusters. For this study, the data is statistically measured in terms of the interdependency redundancy measure $R(x_i, x_j) = \frac{I(x_i,x_j)}{H(x_i,x_j)}$; whereas $I(x_i, x_j) = \sum_{x_i,x_j \in X} p(x_i, x_j) \log \frac{P(x_i,x_j)}{p(x_i)p(x_j)}$ is the mutual information and $H(x_i, x_j) = \sum_{x_i,x_j \in X} p(x_i, x_j) \log p(x_i, x_j)$ is the joint entropy for the discrete random variables x_i and x_j [4]. The $R(\cdot, \cdot)$ measure discriminates a variable (containing many states) which has a weak statistical correlation with respect to another variable.

Without loss of generality, given a set of domain variables X, the objective of attribute clustering is to find a disjoint set of clusters $C = \{C_i | (i = 1, \cdots, k) \wedge (\forall_{i \neq j} C_i \cap C_j = \emptyset)\}$ that maximizes Eq. 1; where w_{o_i,x_j} denote the attached weight (measured by $R(o_i, x_j)$) from the center o_i to other variables x_j in the cluster C_i.

$$W^C = \sum_{C_i} \sum_{x_j \in (C_i - \{o_i\})} w_{o_i,x_j} \tag{1}$$

Two paradigms of clustering were taken in order to find optimal clusters of discrete random variables. The first technique is the k-modes algorithm that optimize Eq. 1 directly [4]. The k-modes can be seen as a graph partitioning algorithm. Thus, a set of discrete random variables are exhibited as nodes in a complete graph ($K = (V, E)$, where V denotes a set of nodes representing variables X, and E includes all edges that are associated with all pair-wise $R(\cdot, \cdot)$ estimates). Another clustering method is the MAST-based clustering algorithm which partitions and clusters a tree instead of the complete graph. The overhead of constructing a maximum spanning tree is in the order of $O(n \log n)$ using the Kruskal's algorithm. Typical approaches include: the standard Euclidean maximum spanning tree(SEMST) [6], the maximum cost spanning tree(CEMST) [7], and the Zahn's maximum spanning tree(ZEMST) [8].

3 The Star Discovery Algorithm

We introduce the Star Discovery (SD) algorithm that iteratively partitions a MAST and form clusters until all nodes $x_i \in X$ are assigned to clusters. The

[1] Discrete random variables (attributes) are seen as nodes in a graph ($V = X$, where V denotes a set of nodes). We will use any of these terms indifferently throughout this paper.

SD algorithm clusters the domain in an unsupervised fashion (no initial number k of clusters is provided). Guiding the search for centers by only examining the topology or single weights is probably not a good idea since the whole domain is not taken into account. Similar to the ZEMST algorithm, we base the clustering on a simplistic search involving both topology and weights in neighborhoods. We look for subgraphs from the MAST that could reveal information about the "nature" of the domain. One abstraction of our technique is to look for spanning stars as subgraphs contained in the MAST. A spanning star [9] is a sub-tree over the MAST, $S = (V_S, E_S)$, and is composed of q nodes. It has a center $o \in V_S$ with a degree $q-1$ and all other nodes have a degree of one. The spanning star is our fundamental graph theoretical resource for expressing clusters that reside in a two dimensional Euclidean space.

Detecting the set of k-stars whose global weight is maximal(following Eq. 1) from a complete graph K requires expensive computation. Similar to the previous MAST partitioning algorithms, the SD algorithm aims to detect a set of spanning stars, $SS = \{S_1, \cdots, S_k\}$, such that the objective function in Eq. 2 is maximized.

$$W = \sum_{S_l \in SS} \Big(\sum_{x_i \in Adj_l} (w_{x_i,o_l}) + \sum_{x_j \in Adj_l, x_h \in Leaf_l} (w_{x_j,x_h}) \Big) \tag{2}$$

where o_l is the star(cluster) center, Adj_l is a set of adjacent nodes to the center node o_l, and $Leaf_l$ a set of leaf nodes that connect to either o_l or Adj_l.

Notice that we extend the notion of a star to include some leaf nodes (nodes whose degree is 1 in the graph). In the experimentation we found that leaf nodes have a higher correlation to the center of its adjacent node than to any other center in any other star. The SD algorithm optimizes the later function by ranking every variable according to its ability to serve as modes. The search heuristic will only select a star as a mode if its mode has not been used before in any other clusters. At the end we will acquire the set of clusters whose structure (modes, adjacent and leaf nodes) is maximal according to Eq. 2 and the heuristic presented in Fig. 1 [2].

The SD algorithm receives a MAST G and the set of weights W^G. At the very beginning the algorithm initializes an auxiliary set of variables V^{aux} and the counter l (line 1). After that, we build $n = |V|$ different stars, $S_r \in SS$, by specifying each variable x_r as the center o_r (line 3). For each star S_r, we include the adjacent nodes Adj_r to the center and leaf nodes $Leaf_r$ ($Deg(\cdot)$ denotes the node degree in the tree) (lines 4 and 6). Simultaneously, the edges are added (lines 5 and 7). Hence, the star S_r is a tuple having two sets: a set of nodes V_{S_r} and a set of edges E_{S_r} (line 9). In addition, we calculate the weight W^{S_r} in each star by adding all the weights attached to the star edges (line 10). Following, the auxiliary star S_r is kept in SS (line 11) as well as its corresponding weight W^{S_r} in W^{SS} (line 12).

Once the set of stars, SS, have been built from the MAST we proceed to sort them decreasingly in terms of the star weights (line 13). The sorting forms a ranking of potential modes and those ones with a higher weight W^{S_r} will be

[2] Note that $X \Leftarrow x$ indicates the addition of an element x to a given set X.

Star Discovery (SD) Algorithm

Input: $G = (V, E)$, W^G
Output: $C = \{C_1, C_2, \ldots, C_l\}$

1: $V^{aux} = V, V^{cont} = \emptyset, l = 1$
2: FOR $r = 1$ to n
3: $o_r = x_r$
4: $Adj_r \Leftarrow x_i$ **iff** $(x_i, o_r) \in E$
5: $E_{S_r} \Leftarrow (o_r, x_i)$
6: $Leaf_r \Leftarrow x_h$ **iff** $(x_i, x_h) \in E \wedge Deg(x_h) = 1$
7: $E_{S_r} \Leftarrow (x_i, x_h)$
8: $V_{S_r} = (o_r \cup Adj_r \cup Leaf_r)$
9: $S_r = (V_{S_r}, E_{S_r})$
10: $W^{S_r} = \sum_{(x_i, x_j) \in E_{S_r}} w_{x_i, x_j}$
11: $SS \Leftarrow S_r$
12: $W^{SS} \Leftarrow W^{S_r}$
13: Sort SS decreasingly according to W^{SS}
14: WHILE $V^{aux} \neq \emptyset$
15: $C_l = V_{S_l} - V^{cont}$
16: $V^{aux} = (V^{aux} - V_{S_l})$
17: $V^{cont} \Leftarrow V_{S_l}$
18: $C \Leftarrow C_l$
19: $l = l + 1$

Fig. 1. The Star Discovery Algorithm.

selected to form clusters (this way we form only one possible arrangement of clusters). We elect the star as the cluster C_l that has the largest star weight among the remained stars (line 15). We use V^{cont} to exclude variables already contained in previous clusters (line 17). This avoids possible overlapping nodes between any pair of clusters. A set of clusters C are completed until no nodes are left.

Assuming that there are n variables and the highest cardinalities of adjacent nodes A_r and leaf nodes L_r are t and u respectively; then, the complexity in the first phase is $O(ntu)$ (lines 2-12) operations to search for all the adjacent nodes and leaves. The sorting operation takes at most $O(n \log n)$ if we use a merge-sort algorithm (line 13). The construction of clusters takes at most $O(l(t + u))$ operations (lines 14-19). Therefore the algorithm has a polynomial complexity $O((ntu) + (n \log n) + (l(t + u)))$. This polynomial complexity is better than the one in k-modes since the number of variables t and u is fairly low. Moreover, the SD algorithm is executed for a single time and not for a number of iterations as in the k-modes algorithm.

The SD algorithm always provides solutions that are deterministic. On the other hand, SD might not offer results that are better in quality than the ones

given from the k-modes algorithm. However, k-modes could obtain better solutions in some cases, but it has the risk of falling into local optima (the solution depends of the initial modes).

4 Experimental Results

We discuss the reliability of the k-modes algorithm and then compare the performance of the SD algorithm against the aforementioned algorithms. A sound estimate to evaluate the goodness of a set of clusters uses Eq. 1. In other words, we are concerned to calculate the local degree of dependency between the centers or 'modes', o_i, of each cluster, C_i, against its other elements. Then, a global weight adds up every local weight in the clusters to obtain a total weight, W^C.

For each experiment, we artificially generated datasets from some well known Bayesian networks such as: the Alarm (37 nodes), Barley (48 nodes), HeparII (70 nodes), Hailfinder (56 nodes) and Pathfinder (109 nodes) [3]. In this paper, we will show the performance of the SD algorithm against earlier algorithms; a detailed discussion of some specific application of attribute clustering is subject to future work.

Reliability of the k-modes algorithm: Indeed, the k-modes algorithm can detect the optimal clustering given our objective. However, there is a drawback by using this approach. Since the formulation of the k-modes algorithm is greedy, there is the risk of falling into local optima. In order to test the susceptibility of the k-modes algorithm to fall into local optima, we fed initial modes ($k = 2$) in each domain with all the possible $\binom{n}{2}$ combinations of variables, then we ran the experiment until it converges. For this experiment, we generated a dataset for each domain with a sample size $\Omega = 10000$. Table 1 presents the results.

Table 1. Number of local optima into which the k-modes algorithm falls.

Domains	Alarm	HeparII	Hailfinder	Pathfinder
Local Optima	17	130	91	117

We found that k-modes does fall into local optima. For example, in the Alarm domain, it was interesting to see that k-modes converges into the optimal value of 6.13 with modes VentAlv and HR. However, it falls into 17 local optima having modes (VentAlv, LVEDVolume), (VentAlv, Shunt), etc. In the optimal result, the size of the clusters is about $\frac{n}{2}$ (18 variables). In many local optima, one cluster becomes relatively small (10 variables). Clearly, a small cluster is isolated because of the sub-optimal initial mode. Whenever LVEDVolume or Shunt are selected as a mode, then no improvement is made. These modes dominate their neighborhoods. The previous analysis is a straightforward example of techniques based solely on an iterative greedy search. As shown in Table 1, the k-modes algorithm falls in more local optima values in larger domains. These findings

[3] http://genie.sis.pitt.edu/networks.html

are a strong motivation for developing an algorithm that could detect the right initial modes.

Clustering quality and sensitivity: We ran all of the algorithms SEMST, CESMT, ZEMST, k-modes and SD (using $k = 8$); then, we compared the quality of the clustering results in terms of its global weight W^C. For the effects of this experiment and to avoid local optima we fed the k-modes algorithm with the resulting modes of the SD algorithm (notice that we also fed k-modes with the final modes which were obtained by the other methods, but it fell into local optima). On the other hand, it is interesting to investigate the response of the clustering algorithms using different sample sizes (k was set to 8). As the sample size Ω decreases, the lectures of the $R(\cdot, \cdot)$ measure become less accurate. Depending on the domain in study, there is a denominated level of sufficient statistics that determines the true nature of the MAST and reveals the true structure of correlated variables. Table 2 depicts the clustering results.

Table 2. Performance (W^C) of algorithms (Alg.) in five domains over different sample sizes ($\Omega = 10000, 8000, 6000, 4000$) and k=8. The k-modes algorithm is optimal when fed with the right initial modes.

Alg.	**10000**	**8000**	**6000**	**4000**
SEMST	4.13	18.41	21.61	22.99
CESMT	5.4	18.78	22.07	23.52
ZEMST	6.11	19.10	22.85	24.66
SD	7.85	**21.30**	**23.95**	**25.38**
k-modes	**8.35**	**21.30**	**23.95**	**25.38**

(a) Alarm Data

Alg.	**10000**	**8000**	**6000**	**4000**
SEMST	2.33	14.67	19.03	22.23
CESMT	2.55	14.85	19.24	22.48
ZEMST	3.85	14.91	20.70	24.20
SD	4.88	**15.39**	**21.02**	**25.41**
k-modes	**5.61**	**15.39**	**21.02**	**25.41**

(b) Barley Data

Alg.	**10000**	**8000**	**6000**	**4000**
SEMST	50.97	50.32	51.49	52.32
CESMT	51.21	50.55	51.71	52.89
ZEMST	51.27	51.43	52.55	53.54
SD	**55.57**	**56.98**	**58.34**	**59.56**
k-modes	**55.57**	**56.98**	**58.34**	**59.56**

(c) HeparII Data

Alg.	**10000**	**8000**	**6000**	**4000**
SEMST	30.26	31.33	32.42	33.65
CESMT	31.02	32.00	33.01	34.16
ZEMST	32.41	33.28	33.81	34.97
SD	**32.48**	**33.58**	**34.69**	**35.96**
k-modes	**32.48**	**33.58**	**34.69**	**35.96**

(d) Hailfinder Data

Alg.	**10000**	**8000**	**6000**	**4000**
SEMST	85.98	87.53	88.75	89.82
CESMT	88.63	88.22	89.40	90.19
ZEMST	88.315	88.75	89.64	90.61
SD	86.61	89.31	89.71	91.03
k-modes	**90.33**	**89.41**	**91.32**	**92.72**

(e) Pathfinder Data

The SD algorithm performs better than the other tree-based clustering algorithms. Indeed, sometimes the SD algorithm is as effective as the k-modes

algorithm. The later is true because if we consider the whole MAST in the cluster identification then we easily detect the strong components in the space. A highly connected variable in a MAST is very likely to be the best center in a given region. We can also conclude that more elaborated algorithms perform a better clustering. Clearly, the search spaces of the ZEMST and SD algorithms are relatively larger than the ones in the SEMST and CEMST approaches. Nevertheless, the search space of the SD algorithm is bigger than the one of ZEMST.

The SEMST, CEMST and ZEMST algorithms perform a *local* search on the MAST for clustering. For example, in the SEMST algorithm we completely disregard the inner relevance of an arc given the MAST topology. Thus, in practice, SEMST normally selects arcs connecting to leaf nodes as inconsistent (which in turn produces unbalanced bad clusters). In the CEMST algorithm, we take into account both weights and (up to some extent) the structure of the MAST. In this case, the inconsistent arcs have a maximal cost (which biases the search towards those arcs that are likely linked to highly connected nodes). The previous search technique is not enough since the search of inconsistent arcs is limited to a path of length 1. On the other hand, the ZEMST extends the search space by comparing the impact of removing an arc given some neighboring arcs and variables. Ultimately, the SD algorithm outperforms all the other tree-based algorithms because it calculates the clusters by considering both the weight and topology in the search. From the star formulation we realize that we could avoid local optima by discriminating those nodes that have a low connectivity and weight.

Conclusively, we can learn that the MAST is in fact a useful dependence graph whenever a sound clustering method is applied to section it. The same trend holds if we supply different sample sizes or change the number k of clusters.

We can see that all algorithms have the same behavior for different sample sizes. Clearly, the SD algorithm outperforms any of other MAST-based clustering algorithms and obtains the same results as k-modes. Thus, the extensive search procedure of the SD algorithm secures competitive clustering.

Elapsed times: Finally, we investigated the running time of SD and other algorithms ($\Omega = 10000$). We used a system Centrino Duo with 2Ghz and 2 Gigabytes of memory. From Table 3 we can confirm that the algorithms calculate clusters obeying their complexity.

Table 3. Elapsed times (in seconds) for algorithms in all domains.

Alg.	Alarm	Barley	HeparII	Hailfinder	Pathfinder
SEMST	**0.031**	**0.04**	**0.044**	**0.049**	**0.047**
CESMT	0.04	0.042	0.056	0.05	0.062
ZEMST	0.078	0.057	0.065	0.07	0.094
SD	0.047	**0.04**	0.046	0.061	0.062
k-modes	0.109	0.063	0.077	0.078	0.125

Logically, the SEMST algorithm is the fastest approach since it discards edges with the simplest rules. Ultimately, the elapsed times grow as the search

space increases. The SD algorithm has a very competitive elapsed time (similar to the SEMST algorithm). We can see that in most cases, the SD clustering outperforms the k-modes algorithm in terms of elapsed times by a 50 percent ratio.

5 Conclusion

In this paper, we illustrated a comprehensive study between several clustering algorithms. We found that the SD algorithm is able to obtain clusters having a better quality than using other MAST-based clustering algorithms. Hence, the SD algorithm can compete with the k-modes algorithm in some cases; the advantage of the SD algorithm over k-modes is that we obtain a deterministic solution. The SD algorithm can also be used to select the initial modes to be fed to the k-modes algorithm for further clustering. We aid the search of clusters by revealing the nature of the domain through a MAST structure. The SD algorithm can be either used to perform the sectioning of a whole domain by itself, or to construct a hybrid algorithm (merged with the k-modes algorithm) which can find optimal clusterings (as shown in our experiments).

References

1. Smet, F.D., Mathys, J., Marchal, K., Thijs, G., DeMoor, B., Moreau, Y.: Adaptive quality-based clustering of gene expression profiles. Artificial Intelligence **18**(5) (2002) 735–746
2. Madeira, S., Oliveira, A.: Biclustering algorithms forn biological data analysis: A survey. IEEE/ACM Trans. on Computational Biology and Bioinformatics **1**(1) (2004) 24–45
3. Eisen, M.B., Spellman, P.T., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. In: In Proceedings of National Academy of Sciences of the United States of America. (1998) 14863–14868
4. Au, W.H., Chan, K., Wong, A., Wang, Y.: Attribute clustering for grouping, selection, and classification of gene expression data. IEEE Trans. on Computational Biology and Bioinformatics **2**(2) (2005) 83–101
5. Cohen, D.B., Havlin, S.: Structural Properties of Scale Free Networks. Wiley-Vch., Berlin GmbH (2004)
6. Asano, M.K.T., Bhattacharya, B., Yao, F.: Clustering algorithms based on minimum and maximum spanning trees. In: In Proceedings of the fourth annual symposium on Computational Geometry. (1998) 252–257
7. Ye, B., Chao, K.M.: Spanning Trees and Optimization Problems. Chapman and Hall (2004)
8. Zahn, C.: Graph theoretical methods for detecting and describing gestalt clusters. IEEE Trans. in Computers **20** (1971) 68–86
9. Gallian, J.: Dynamic survey of graph labeling. Electronic Journal of Combinatorics **14**(6) (2007)