Fast Multidimensional Clustering of Categorical Data

Tengfei Liu[1], Nevin L. Zhang[1], Kin Man Poon[1], Yi Wang[2], and Hua Liu[1]

[1] Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{liutf, lzhang, lkmpoon, aprillh}@cse.ust.hk

[2] Department of Computer Science
National University of Singapore
wangy@comp.nus.edu.sg

Abstract. Early research work on clustering usually assumed that there was one true clustering of data. However, complex data are typically multifaceted and can be meaningfully clustered in many different ways. There is a growing interest in methods that produce multiple partitions of data. One such method is based on latent tree models (LTMs). This method has a number of advantages over alternative methods, but is computationally inefficient. We propose a fast algorithm for learning LTMs and show that the algorithm can produce rich and meaningful clustering results in moderately large data sets.

Keywords: Multidimensional Clustering, Model-based Clustering, Latent Tree Models

1 Introduction

There are several clustering methods that produce multiple partitions. We refer to them as *multi-partition clustering (MPC)* methods. MPC methods, according to the way that partitions are found, can be divided into two categories: *sequential MPC* methods and *simultaneous MPC* methods.

Sequential MPC methods produce multiple clustering solutions sequentially. One such kind of method is known as *alternative clustering* [1–3]. It aims to discover a new clustering that is different from a previously known clustering. The key issue is how to ensure the novelty of the new clustering. One can repeatedly apply such methods to produce a sequence of clusterings.

Simultaneous MPC methods, on the other hand, produce multiple clustering solutions simultaneously. Both distance-based and model-based methods have been proposed for simultaneous MPC. The distance-based methods proposed by [4, 5] require as inputs the number of partitions and the number of clusters in each partition. They try to optimize the quality of each individual partition while keeping different partitions as dissimilar as possible. Model-based methods fit data with a probabilistic model that contains multiple latent variables. Each latent variable represents a soft partition and can be viewed as a hidden dimension of data. So we refer to model-based simultaneous MPC also

as *multidimensional clustering (MDC)*. Unlike distance-based methods, model-based methods can automatically determine the numbers of partitions and the number of clusters in each partition based on statistical principles.

Among the MDC methods, Galimberti et al. [6] and Guan et al. [7] use what we call *disjoint and independent view (DIV) models*. In a DIV model, each latent variable is associated with a subset of attributes, which gives one *view* of data. The subsets for different latent variables are disjoint. A latent variable is independent of all the other latent variables and all the attributes that are not associated with it. On the other hand, Zhang [8] and Poon et al. [9] use latent tree models (LTMs). An LTM can be viewed as a DIV model with the latent variables connected to form a tree structure.

This paper is concerned with the use of LTMs for producing multiple clustering solutions. Currently, there is a lack of efficient algorithms for learning LTMs. The fastest algorithm takes weeks to process data sets with around 100 attributes and a few thousands samples. We propose a more efficient algorithm that can analyze data with hundreds of attributes in a few hours. We also provide empirical results to show that the algorithm can produce much richer clustering results than alternative methods.

2 Latent Tree Models

Technically an LTM is a Markov random field over an undirected graph, where variables at leaf nodes are observed and variables at internal nodes are hidden. An example of LTM is shown in Figure 1. It is part of the latent tree model learned from WebKB data which will be described in more details in Section 4.2. The Y-variables are the latent variables. The numbers in parenthesis are the numbers of states of the latent variables. The leaf nodes here are different words which take binary values to indicates presence or absence of the word. The edges represent probabilistic dependence and their width represents dependence strength.

For technical convenience, we often root an LTM at one of its latent nodes and regard it as a directed graphical model, i.e., a Bayesian network. For the model in Figure 1, suppose we use Y_{43} as the root. Then the edges are directed as pointing away from Y_{43}. For example, the edges $Y_{43} - Y_{53}$ and $Y_{53} -$ ut should be directed as $Y_{43} \rightarrow Y_{53}$ and $Y_{53} \rightarrow$ ut. The numerical information of the model includes a marginal distribution $P(Y_{43})$ for the root and one conditional distribution for each edge. For the edge $Y_{43} \rightarrow Y_{53}$, we have distribution $P(Y_{53}|Y_{43})$; for the edge $Y_{53} \rightarrow$ ut, we have distribution $P(\mathsf{ut}|Y_{53})$; and so on. The product of those distributions defines a joint distribution over all the latent and observed variables. In this paper, we assume all variables are discrete.

2.1 The State of the Art

Suppose there is a data set $\mathbf{D}$. The attributes of the data set are the observed variables. To learn an LTM from $\mathbf{D}$, one needs to determine: (1) the number of latent variables, (2) the number of states of each latent variable, (3) the connections among the latent and observed variables, and (4) the probability

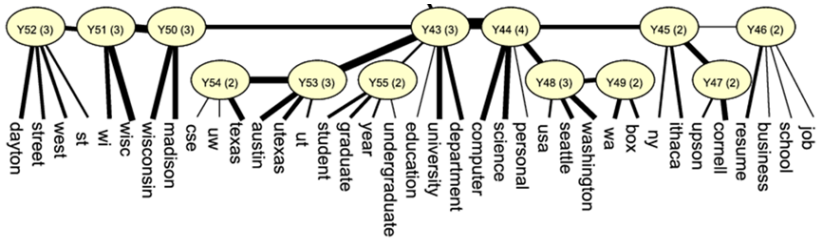

Fig. 1. Part of the latent tree model produced by new algorithm on the WebKB data.

parameters. We use m to denote the information for the first three items and θ to denote the collection of parameter values. We aim at finding the pair (m, θ^*) where θ^* is the maximum likelihood estimate of the parameters and m maximizes the BIC score [10]:

$$BIC(m \mid \mathbf{D}) = \log P(\mathbf{D}|m, \theta^*) - \frac{d(m)}{2} \log N$$

where $d(m)$ is the number of free parameters in m and N is the sample size.

Several algorithms for learning LTMs have been proposed. The state-of-the-art is an algorithm called EAST [11]. It is a search-based method and is capable of producing good models for data sets with dozens of attributes. However, it is not efficient enough for data sets with more than 100 attributes. There are two other algorithms that are more efficient than EAST, but they focus on special LTMs. Harmeling and Williams [12] consider only binary trees, while Choi et al. [13] assume all the variables share the same domain. Neither method is intended for cluster analysis.

3 The Bridged Islands Algorithm

We now set out to present a new algorithm that is drastically more efficient than EAST. In an LTM, the set of attributes that are connected to the same latent variable is called a *sibling cluster*. Attributes in the cluster are said to be *siblings*. In the LTM shown in Figure 1, attributes austin, utexas and ut form one sibling cluster because they are all connected to Y_{53}.

The new algorithm proceeds in five steps:

1. Partition the set of attributes into sibling clusters;
2. For each sibling cluster introduce a latent variable and determine the number of states of this variable;
3. Determine the connections among the latent variables so that they form a tree;
4. Determine the values of the probability parameters;
5. Refine the model.

If we imagine the sibling clusters formed in Step 1, together the latent variables added in Step 2, as islands in an ocean, then the islands are connected in Step 3. So we call the algorithm the *bridged islands (BI)* algorithm. In the following, we describe each step of BI in details.

Step 1: BI determines sibling clusters using two intuitions. First, attributes from the same sibling cluster tend to be more closely correlated than those from

different sibling clusters. Second, if two attributes are siblings in the optimal model for one set of attributes, they should also be siblings in the optimal model for a subset of the attributes.

BI determines the first sibling cluster as follows. There is a working subset of attributes. Initially, it contains the pair of attributes with the highest mutual information(MI). Here MI is computed from the empirical distribution of the data. BI grows the working subset by adding other attributes into it one by one. At each step, it chooses the attribute that has the highest MI with the current subset. (The first intuition is used here.) The MI between a variable X and a set $\mathbf{S}$ is estimated as follows:

$$I(X;\mathbf{S}) = \max_{Z\in\mathbf{S}} I(X;Z) = \max_{Z\in\mathbf{S}} \sum_{X,Z} P(X,Z)\log\frac{P(X,Z)}{P(X)P(Z)} \tag{1}$$

BI determines when to stop expanding the working subset using the *unidimensionality test* or simply the *UD-test*. This is the most important idea of this paper. A subset of attributes is *unidimensional* if the optimal LTM (i.e., the one with the highest BIC score) for those attributes contains only one latent node. UD-test determines whether a subset of attributes is unidimensional by first projecting data onto those attributes and then running EAST on the projected data. If the resulting model contains only one latent variable, then UD-test concludes that the subset is unidimensional. Otherwise, it concludes the opposite. For computational efficiency, EAST is allowed to examine only models with 1 or 2 latent variables.

After each attribute is added to the working subset, BI runs the UD-test on the subset. If the test fails, the attributes in the subset cannot all be in one sibling cluster in the final model according to the second intuition. So, BI stops growing the subset and picks, from the local model learned during UD-test, the sibling cluster that contains (one of or both) the two initial attributes as the first sibling cluster for the whole algorithm. Attributes in the cluster are then removed from the data set and the process repeats to find other sibling clusters. Figure 2 illustrates the process. The working subset initially contains X_1 and X_2. Then X_3 is added. EAST is run to find an LTM for the expanded subset $\{X_1, X_2, X_3\}$. The resulting model, shown in (a), contains only one latent variable. So the triplet passes the UD-test. Then X_4 is added and the UD-test is again passed as shown in (b). After that X_5 is added. This time EAST yields a model, shown in (c), with more than one latent variable. So the UD-test fails and BI stops growing the subset. The sibling cluster $\{X_1, X_2, X_4\}$ of the local model is picked as the first sibling cluster for the whole algorithm.

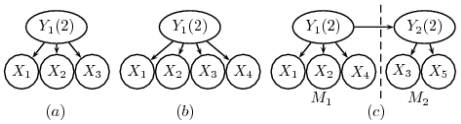

Fig. 2. An example of sibling cluster determination

Step 2: A *latent class model (LCM)* is an LTM with only one latent node. Figure 2 (a) and (b) show two examples. It is a commonly used finite mixture model

for discrete data. At Step 2, BI learns an LCM for each sibling cluster. There are two substasks: (1) to determine the cardinality of the latent variable and (2) to optimize the probability parameters. BI starts by setting the cardinality of the latent variable to 2 and optimizing the model parameters by running the EM algorithm [14]. Then it considers repeatedly increasing the cardinality. After each increase, model parameters are re-optimized. The process stops when the BIC score ceases to increase.

Step 3: After the first 2 steps, BI has obtained a collection of LCMs. We can visualize those LCMs as islands in an ocean. The next step is to link up the islands in a tree formation by adding edges between the latent variables.

Chow and Liu [15] give a well-known algorithm for learning tree-structured models among observed variables. It first estimates the MI between each pair of variables from data, then constructs a complete undirected graph with the MI values as edge weights, and finally finds the maximum spanning tree of the graph. The resulting tree model has the maximum likelihood among all tree models. Chow-Liu's algorithm can be adapted to link up the latent variables of the aforementioned LCMs. We only need to specify how the MI between two latent variables is to be estimated. Let m_1 and m_2 be two LCMs with latent variables Y_1 and Y_2. [3] Enumerate the data cases as $\mathbf{d}_1$, $\mathbf{d}_2$, ..., $\mathbf{d}_N$. Let $\mathbf{d}_{i1}$ and $\mathbf{d}_{i2}$ ($i \in \{1, 2, \ldots, N\}$) be respectively the projections of $\mathbf{d}_i$ onto the attributes of m_1 and m_2. Define the *data-conditional joint distribution* of Y_1 and Y_2 as follows:

$$P(Y_1, Y_2|\mathbf{D}, m_1, m_2) = C \sum_{i=1}^{N} P(Y_1|m_1, \mathbf{d}_{i1}) P(Y_2|m_2, \mathbf{d}_{i2}), \tag{2}$$

where C is the normalization constant. We estimate the MI between Y_1 and Y_2 from this joint distribution.

Step 4: In this step, BI optimizes the probability parameters of the LTM resulted from Step 3 using EM.

Step 5: The sibling clusters and the cardinalities of the latent variables were determined in Steps 1 and 2. Each of those decisions was made in the context of a small number of attributes. Now that all the variables are connected in a global model, it is time to re-examine the decisions and to see whether adjustments should be made.

In Step 5, BI checks each attribute to see whether it should be relocated and each latent variable to see if its cardinality should be changed. All the potential adjustments are evaluated with respect to the model, denoted by $\hat{m}$, resulted from the previous step. The beneficial adjustments are executed in one batch after all the evaluations. Adjustment evaluations and adjustment executions are not interleaved because that would require parameter optimization after each adjustment and hence be computationally expensive.

For each attribute X and each latent variable Y, BI computes their data-conditional MI $I(X, Y|\mathbf{D}, \hat{m})$ from the following distribution:

[3] Here m_1 and m_2 include model parameter values.

$$P(X, Y|\mathbf{D}, \hat{m}) = C \sum_{i=1}^{N} P(X, Y|\hat{m}, \mathbf{d}_i), \tag{3}$$

where C is the normalization constant. Let $\hat{Y}$ be the latent variable that has the highest data-conditional MI with X. If $\hat{Y}$ is not the current parent node of X in $\hat{m}$, then it is deemed beneficial to relocate X from its parent node to $\hat{Y}$.

To determine whether a change in the cardinality of a latent variable is beneficial, BI freezes all the parameters that are not affected by the change, runs EM locally to optimize the parameters affected by the change, and recalculates the BIC score. The change is deemed beneficial if the BIC is increased. BI starts from the current cardinality of each latent variable and considers increasing it by one. If it is beneficial to do so, further increases are considered.

After model refinement, EM is run on the global model one more time to optimize the parameters.

Computational Efficiency: BI is much faster than EAST. The reason is that most steps of BI involves only a small number of variables and the EM algorithm is run on the global model only twice. We tested BI and EAST on two data sets with 81/108 attributes and 3021/2763 samples respectively. EAST took 6/24 days, while BI took 35/69 minutes respectively. Detailed running time analysis will be given in a longer version of the paper.

4 Empirical Results with BI

We now present empirical results to demonstrate BI's capability in discovering rich and meaningful multiple clusterings. Comparisons with other methods will be made in the next section.

4.1 Clustering Synthetic Data

As a test of concept, we first tried BI on synthetic data. The data set was sampled from an LTM with structure as shown in Figure 3(left). There are 3 latent variables and 15 attributes, and all variables are binary. A total number of 1000 data cases were sampled. Each data case contains values for the attributes X_1-X_{15}, but not for latent variables Y_1-Y_3. The data set has 3 'true' partitions, each given by a latent variable. To provide some intuitions on what the partitions are about, Figure 3(right) depicts the normalized mutual information(NMI)[16] between each partition and each attribute. The x-axis represents the observed variables (i.e. X_1-X_{15}). The y-axis indicates the value of NMI between each latent variable(i.e. Y_1-Y_3) and each attribute. There are three curves, labeled as Y_1-Y_3. Those are called *feature curves* of the true partitions.

The LTM obtained by BI from the synthetic data has the same structure as the generative model. In particular, it has three latent variables. Each latent variable represents a soft partition of the data. We obtained a hard partition by assigning each data case to the state with the highest posterior probability. We call those partitions the *BI partitions*. The curves labeled B_1-B_3 in Figure 3 are

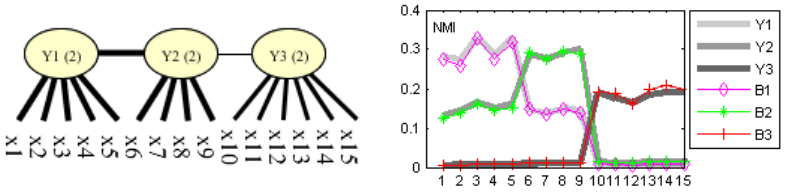

Fig. 3. Generative model and feature curves of partitions.

the feature curves of the BI partitions. They match those of the true partitions almost perfectly. Based on the feature curves, we matched up the BI partitions with the true partitions, and computed the NMI for each pair. The NMI values are 0.91(B1,Y1), 0.86(B2,Y2) and 0.98(B3,Y3) respectively. Those indicate that BI has recovered the true partitions well.

4.2 Clustering Real-World Data

The real-world data set is known as WebKB data. It consists of web pages collected in 1997 from the computer science departments of 4 universities: Cornell, Texas, Washington and Wisconsin. The web pages were originally divided into 7 classes. Only 4 classes remain after preprocessing, namely student, faculty, project and course. There are 1041 pages in total and the number of words was reduced to 336. The words are used as attributes in our analysis. The attributes are binary and indicate the presence or absence of the words. Both the university labels and class labels were removed before the analysis.

On the WebKB data set, BI produced an LTM, henceforth called *WebKB-LTM*, with 98 latent variables. This means that BI has partitioned the data in 98 different ways. A big question is: Are those partitions meaningful or do they appear to be arbitrary? It turns out that many of the partitions are meaningful. We present some of them in the following.

To start with, we give an overview of the structure of WebKB-LTM. It divides itself into three parts. The model of the first part is shown in Figure 1. It involves words that usually appear in general descriptions of computer science departments. So we call it the *department subnet*. The second part contain words such as research, interest, papers, ieee, etc. We call it the *research subnet*. The third part involves words related to course teaching. So we call it the *teach subnet*.

The Department Subnet: We first examine two latent variables Y_{51} and Y_{54} in Figure 1. To inspect the meanings of partitions, we can draw the information curves as shown in Figure 4. Take information curves of Y_{51} as example, there are two curves in the figure. The lower one shows the mutual information(MI) between Y_{51} and each attribute. The attributes are sorted according to MI and only the top 5 attributes are shown here. The upper curve shows the cumulative mutual information(CMI) between Y_{51} and each attribute plus all the attributes before it. The numbers on the left vertical axis are MI values. The numbers on the right axis are the ratios of the MI values over the MI between the partition and all the attributes. The ratio is called *information coverage (IC)*. The IC of

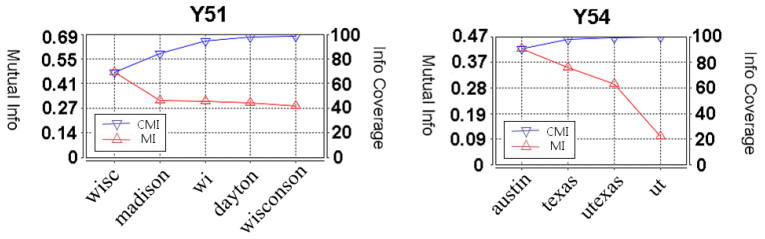

Fig. 4. Information curves of latent variables Y_{51} and Y_{54}.

the first 5 attributes is around 98%. Intuitively, this means that the differences among the different clusters on the first 5 attributes account for 98% of the total differences. So we can say that the partition is primarily based on these attributes.

The table below shows the occurrence frequencies of the words in the clusters. We see that Y_{51} represents a partition based on wisc, madison, wi, dayton and wisconsin. The clusters Y_{51}=2 and Y_{51}=3 together correspond to U Wisconsin-Madison. Y_{54} represents another partition based on austin, texas, utexas and ut. The cluster Y_{54}=2 corresponds to U Texas-Austin.

Y_{47}: IC=94%

cluster	1	2
cornell	.04	1
ithaca	0	.47
ny	.01	.39
Size	.84	.16

Y_{48}: IC=94%

cluster	1	2	3
washington	.03	1	.98
seattle	.01	.13	1
wa	0	0	.92
Size	.8	.1	.1

Y_{51}: IC=98%

cluster	1	2	3
wisc	0	.86	.96
madison	.01	.33	.98
wi	0	.03	.92
dayton	0	.04	.92
wisconsin	0	.31	.94
Size	.74	.14	.12

Y_{54}: IC=99%

cluster	1	2
austin	0	.86
texas	.01	.33
utexas	0	.03
ut	0	.04
Size	.82	.18

We can do similar analysis to Y_{47} and Y_{48}. As shown in above tables, the most informative attributes are selected based on information curves, which are omitted to save space. Information coverages (IC) are given to show how well the attributes collectively convey the meanings of the latent variables. It is clear that Y_{47} and Y_{48} are also meaningful. Y_{47}=2 seems to identify web pages from Cornell, and Y_{48}=2 and Y_{48}=3 together seem to identify web pages from U Washington-Seattle.

The Research Subnet: The following tables contain information about three latent variables from the faculty subnet.

Y_{10}: IC=92%

cluster	1	2
image	.02	.5
images	.02	.38
visual	.01	.31
pattern	0	.19
vision	.02	.3
Size	.91	.09

Y_{11}: IC=90%

cluster	1	2
management	.02	.57
database	.04	.58
storage	0	.29
databases	.01	.33
query	0	.3
Size	.88	.12

Y_{14}: IC=95%

cluster	1	2
april	.02	.42
march	.01	.35
february	0	.33
conference	.06	.44
symposium	.03	.33
proceedings	.04	.38
january	.02	.31
international	.05	.32
Size	.86	.14

Latent variable Y_{10} partitions the web pages based on words such as image, pattern and vision. Y_{10}=2 seems to identify web pages belonging to faculty members who work in the vision/image analysis area. Y_{11} partitions the web pages based on words such as database, storage and query. Y_{11}=2 seems to identify web pages belonging to faculty members who work in the database area. Other

latent variables in the faculty subset give us clusters of web pages for other research areas such as artificial intelligence, networking, etc. Besides research area-related partitions, the faculty subnet also gives us some other interesting clusterings. One example is Y_{14}. Y_{14}=2 seems to identify web pages containing papers published at conferences held in January-April.

The Teach Subnet: The teach subnet gives us partitions based on various aspects of course information. One example is Y_{66}. It is clear from the information given below, Y_{66}=4 identifies web pages on objected-oriented programming (OOP), while Y_{66}=2 identifies web pages on programming but not mentioning OOP. Those might be web pages of OOP courses and of introductory programming courses respectively. Y_{66}=3 seems to correspond to web pages of other courses that involve programming, while Y_{66}=1 seems to mean web pages not on programming.

Y_{66}: IC=94%

cluster	1	2	3	4
programming	.05	.77	1	.71
oriented	0	0	.21	1
object	.02	.06	.46	.91
language	.05	.32	.86	.59
languages	.01	.39	.22	.48
program	.09	.26	.96	.27
programs	.04	.2	.6	.16
Size	.69	.21	.05	.05

In summary, the BI algorithm has identified a variety of interesting clusters in the WebKB data. The situation is similar on several other data sets that we tried. It should be noted that not all the results obtained by BI are meaningful and interesting. After all, it is an explorative data analysis method.

5 Comparisons with Alternative Methods

In this section we compare BI with four other MPC algorithms: orthogonal projection (OP) [1], singular alternative clustering (SAC) [2], DK [4] and EAST. Those are the algorithms that we were able to obtain implementations for or implement ourselves.

The motivation for MPC is the observation that complex data can be meaningfully clustered in multiple ways. As such, we need to consider two questions when comparing different MPC methods: (1) Do they find meaningful clusterings? (2) How many meaningful clusterings do they find?

5.1 Meaningful Clustering

A common way to evaluate a single-partition clustering algorithm is to start with labeled data, remove the class labels, perform cluster analysis, and compare the partition obtained with the partition induced by the class labels. We refer to those two partitions as the *cluster partition* and the *class partition* respectively. The quality of the cluster partition is often measured using its NMI with the class partition.

In the context of MPC, we have multiple class partitions and multiple cluster partitions. How should the evaluation be carried out? Following the literature,

we match each class partition up with the cluster partition with which it has the highest NMI and report the NMI values of the matched pairs. This means that if one of the cluster partitions closely resemble a class partition, then we claim that the class partition is recovered from data. Similar partitions have similar feature curves. So, we sometimes can do the match up based on feature curves, as was done in Section 4.1. A nice thing with the use of feature curves is that they tell us what the partitions are about.

Results on Synthetic Data: We tested OP, SAC, DK and EAST on the same synthetic data that was used to test BI in Section 4.1. The published version of DK can only produce two partitions. We ran DK on the synthetic data to obtain two, instead of three, binary partitions. OP and SAC are sequential MPC methods. Following the authors, we first ran K-means to find a partition of two clusters, and then continue to run OP and SAC to find two more binary partitions.

We have run the experiments 10 times for each algorithm. Here are the NMI values between each true class partition and the matched cluster partition obtained by the algorithms:

	DK	SAC	OP	EAST	BI
Y1	.78±.00	.73±.04	.73±.02	**.91±.00**	**.91±.00**
Y2	.48±.00	.57±.04	.55±.04	**.86±.00**	**.86±.00**
Y3	.97±.00	.59±.49	.91±.20	**.98±.00**	**.98±.00**

The NMI values are high for EAST and BI, indicating that those two algorithms have recovered the three true class partitions well. On the other hand, the NMI values for the other algorithms are relatively lower, indicating that they have not been able to recover the true class partitions as well as EAST and BI.

The results by EAST and BI are identical in terms of NMI values. However, BI took much less time than EAST. The running time of BI was 22±3 seconds, while that of EAST was 485±34 seconds.

Results on Real-World Data: DK, OP and SAC were also tested on the WebKB data. They were told to find two partitions each with four clusters, because it is known that there are two true class partitions each with four classes.

One of class partition divides the web pages into four classes according to the four universities. It is clear from Section 4.2 that BI has recovered the four classes. However, they were given in the form of four partitions (Y_{47}, Y_{48}, Y_{51} and Y_{54}), instead of one. For comparability, we transformed the 4-class class partition into four logically equivalent binary class partitions. Each binary class partition divides the web pages according to whether they are from a particular university. The same transformation was applied to the other class partition and the cluster partitions obtained by the alternative algorithms. After the transformations, we matched up the binary class partitions with the cluster partitions and computed the NMI of each matched pair. The results are shown in following tables. The average was taken over 10 runs of the algorithms.

We see that the NMI is the highest for BI in almost all cases. This means that BI has recovered most of the 8 classes better than the alternative algorithms.[4]

[4] As seen in Section 4.2, some of the partitions obtained by BI contain multiple clusters that correspond to a true class. For example, both $Y_{48} = 2$ and $Y_{48} = 3$ correspond

	DK	SAC	OP	BI
course	.43±.01	.47±.01	.47±.02	**.59±.01**
faculty	.18±.04	.17±.07	.18±.01	**.31±.01**
project	.04±.00	.04±.00	.05±.04	**.07±.00**
student	.18±.00	**.20±.00**	**.20±.01**	**.20±.02**
cornell	.22±.15	.09±.02	.36±.24	**.48±.11**
texas	.31±.18	.20±.20	.45±.23	**.60±.02**
washington	.22±.13	.41±.23	**.56±.25**	.52±.12
wisconsin	.38±.12	.16±.12	.45±.13	**.48±.10**

We also tested EAST on WebKb data. However, it did not finish in two weeks. In contrast, BI took only 90 minutes.

5.2 Richness of Clusterings

The WebKB data was analyzed by a number of authors using different methods [1, 5, 7]. In all cases, only two partitions and a few clusters were obtained. In contrast, BI has discovered , as shown in Section 4.2, a rich collection of meaningful clusters. This is what sets BI far apart from other methods.

Why is it difficult for other methods to produce rich clustering results on complex data sets? Well, the sequential MPC methods and the distance-based simultaneous MPC methods cannot determine the numbers of partitions and clusters. The user has to provide such information as input. In single partitioning, not being able to determine the number of clusters is not a big problem. One can manually try a few possibilities and pick the one that yields the best results. In the case of MPC, however, not being able to determine the numbers of partitions and clusters is a severe drawback. There are too many possibilities to consider. As a simplification of the problem, suppose it is known that there are 10 partitions. Determining the number of clusters for the 10 partitions manually would be very difficult as there are too many possible combinations to try.

Other model-based simultaneous MPC methods can determine the numbers of partitions and clusters. However, they are still unable to produce rich clustering results on complex data sets. The algorithm by Galimberti and Soffiritti [6] is inefficient and has so far been tested only on data sets with fewer than 10 attributes. The EAST algorithm is not efficient enough to handle data sets with 100 or more attributes. The method by Guan et al. [7] is efficient enough to deal with the WebKb data, but it produces only two partitions.

5.3 A Remark

The evaluation method used in Section 5.1 has one drawback. A naïve method that generates a huge number of random partitions might get good evaluation. So, it is important to test MPC methods on real-world data set and see whether they can help find meaningful clusters. On the WebKB data, BI found 98 partitions. By inspecting their information curves, we were able to quickly identify dozens of meaningful partitions. With the random method, however, one would have to examine a huge number of partitions before finding a meaningful one. Hence it is not useful.

to U Washington-Seattle. If such clusters are manually aggregated, the NMI values for BI would look better.

6 Conclusions

This paper is concerned with the use of latent tree models (LTMs) for multiple partition clustering. We propose a new algorithm, called BI, for learning LTM that is much faster than the best previous algorithm and can handle data with hundreds of attributes. The key idea behind the algorithm is UD-test, which determines whether the interactions among a set of observed variables can be modeled using one latent variable. Empirical results are presented to show that BI is able to produce much richer clustering results than alternative methods.

7 Acknowledgements

Research on this work was supported by Hong Kong Research Grants Council GRF Grant #622408, The National Basic Research Program of China (aka the 973 Program) under project No. 2011CB505101, and HKUST Fok Ying Tung Graduate School.

References

1. Cui, Y., Fern, X.Z., Dy, J.G.: Non-reduntant multi-view clustering via orthogonalization. In: ICDM-07. (2007)
2. Qi, Z., Davidson, I.: A principled and flexible framework for finding alternative clusterings. In: KDD-09. (2009)
3. Gondek, D., Hofmann, T.: Non-redundant data clustering. KAIS-07 (1) (2007)
4. Jain, P., Meka, R., Dhillon, I.S.: Simultaneous unsupervised learning of disparate clusterings. In: SDM-08. (2008) 858–869
5. Niu, D., Dy, J.G., Jordan, M.I.: Multiple non-redundant spectral clustering views. In: ICML-10. (2010)
6. Galimberti, G., Soffritti, G.: Model-based methods to identify multiple cluster structures in a data set. CSDA-07 **52** (2007) 520–536
7. Guan, Y., Dy, J.G., Niu, D., Ghahramani, Z.: Variational inference for nonparametric multiple clustering. In: MultiClust Workshop, KDD-2010. (2010)
8. Zhang, N.L.: Hierarchical latent class models for cluster analysis. JMLR-04 **5** (2004) 697–723
9. Poon, L.K.M., Zhang, N.L., Chen, T., Yi, W.: Variable selection in model-based clustering: To do or to facilitate. In: ICML-10. (2010)
10. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics **6**(2) (1978) 461–464
11. Chen, T., Zhang, N.L., Wang, Y.: Efficient model evaluation in the search-based approach to latent structure discovery. In: PGM-08, 57-64. (2008)
12. Harmeling, S., Williams, C.K.I.: Greedy learning of binary latent trees. TPAMI-10 (2010)
13. Choi, M.J., Tan, V.Y.F., Anandkumar, A., Willsky, A.S.: Learning latent tree graphical models. Computing Research Repository (2010)
14. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
15. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory (1968)
16. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. JMLR **3** (2002) 583–617