A Novel LTM-based Method for Multi-partition Clustering

Tengfei Liu, Nevin L. Zhang, Kin Man Poon, Hua Liu
The Hong Kong University of Science and Technology
{liutf, lzhang, lkmpoon, aprillh}@cse.ust.hk

Yi Wang
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 (LTM). But previous methods for learning general LTM are computationally inefficient. In this paper, we propose a fast algorithm for learning LTM. Empirical results on two real world datasets are given to show that our method can produce rich and meaningful partitions.

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 partitions sequentially. One such kind of method is known as *alternative clustering* (Cui *et al.*, 2007; Gondek and Hofmann, 2007; Qi and Davidson, 2009). 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 partitions simultaneously. Both distance-based and model-based methods have been proposed. The distance-based methods (Jain *et al.*, 2008; Niu *et al.*, 2010) 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. Unlike distance-based methods, model-based methods can automatically determine the number of partitions and the number of clusters in each partition based on statistical principles.

Among the model-based methods, Galimberti and Soffritti (2007) and Guan *et al.* (2010) assume that each latent variable, which gives one view of data, is associated with a subset of attributes. 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 (2004) and Poon *et al.* (2010) use latent tree models (LTM). Different from aforementioned methods, the latent variables in LTM are connected and form an interpretable tree structure.

This paper is concerned with the use of LTM for producing multiple partitions of categorical data. Currently, there is a lack of efficient algorithms for learning general LTM. In this paper we propose a fast algorithm to learn LTM. Em-

pirical results are also given to show that the new algorithm can produce rich and meaningful clustering results.

2 Latent Tree Model

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. 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. An example of LTM is shown in Figure 1. In the example model, the Y-variables are latent variables. The number in parenthesis is called cardinality which indicates the number of states of the latent variable. The leaf nodes in this example are different words which take binary values to indicate presence or absence of the word.

Throughout the paper, we use the term 'node' interchangeably with 'variable', and term 'leaf node' interchangeably with 'attribute'. A 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*. For example, in Figure 1, attributes *austin*, *utexas* , *texas* and *ut* form one sibling cluster because they are all connected to latent node Y_1.

The numerical information of LTM includes a marginal distribution $P(Y_1)$ for the root Y_1 and one conditional distribution for each edge. For example, for the edge $Y_2 \rightarrow$ *object*, we have distribution $P(object \mid Y_2)$. The product of these distributions defines a joint distribution over all the latent and observed variables.

To learn an LTM from a dataset $\mathbf{D}$, one needs to determine: (1) the number of latent variables, (2) the cardinality of each latent variable, (3) the connections among the latent and observed variables, and (4) the probability 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 (Schwarz, 1978):

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

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

Several algorithms for learning LTM have been proposed. The latest algorithm for learning general LTM is the one called EAST (Chen *et al.*, 2012, 2008). 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 LTM. Harmeling and Williams (2010) consider only binary trees, while Choi *et al.* (2011) assume all the variables share the same domain. Neither methods are intended for cluster analysis.

3 The Bridged Islands Algorithm

We now set out to present a new algorithm for learning general LTM that is more efficient than previous algorithm EAST. The new algorithm proceeds in four 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. Refine the model.

If we imagine the sibling clusters formed in Step 1, together with 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.

3.1 Determining Sibling Clusters

The sibling cluster determination is the first step of the new algorithm. It is based on two intuitions: First, in an LTM, 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 LTM for one set of attributes, they should also be siblings in the optimal LTM for a subset of the attributes. We determine the

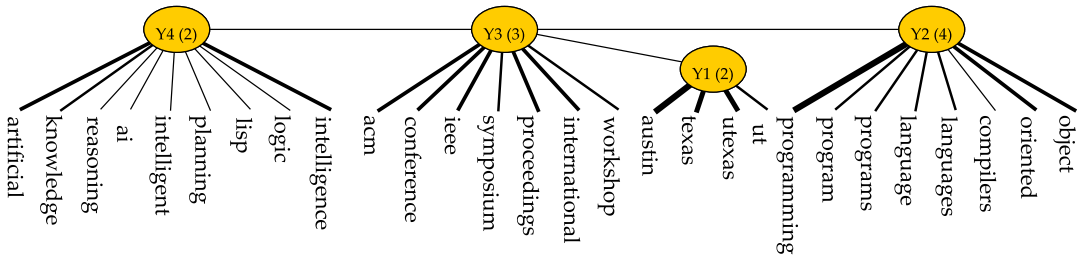

Figure 1: An example of latent tree model. The width of edges represent strength of probabilistic dependence.

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. The method grows the working subset by adding other attributes into it one by one. At each step, we choose 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:

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

We determine when to stop expanding the working subset by using a Bayesian statistical test called *unidimensionality test* or simply the *UD-test*. When UD-test fails, the expansion stops.

We first project original data set onto the attributes in the working subset and get a data set $\mathbf{D_p}$. The UD-test is done by comparing two models learned from $\mathbf{D_p}$: an unidimensional model m_1 and a multidimensional model m_2. Model m_1 and m_2 are the best models, in terms of the BIC score, that contain 1 and 2 latent variables respectively. We say UD-test fails if the BIC score of m_2 exceeds that of m_1 by a threshold δ and passes otherwise. i.e.,

$$BIC(m_2 \mid \mathbf{D_p}) - BIC(m_1 \mid \mathbf{D_p}) \geq \delta$$

The left hand side of the inequality is an approximation to the Bayes factor (Kass and Raftery, 1995) for the two models. When it exceeds the threshold, we conclude that correlations among the attributes cannot be appropriately modeled using one single latent variable. And these attributes in the working subset cannot all be in one sibling cluster in the final model according to the second intuition.

When UD-test fails, model m_2, which has two latent variables, gives us two potential sibling clusters. If one of them contains both the two initial attributes, we pick it as our first sibling cluster. Otherwise, we pick the one with more attributes and break ties arbitrarily. Attributes in the cluster are then removed from the data set and the process repeats to find other sibling clusters. We apply EAST algorithm to learn the m_1 and m_2 models. For computational efficiency, EAST is restricted to examine only models with 1 or 2 latent variables.

To illustrate the process, suppose that we start with an initial working subset which contains X_1 and X_2. Two attributes X_3 and X_4 are successfully added. Then X_5 is added. Suppose the best models m_1 and m_2 for $\{X_1, X_2, X_3, X_4, X_5\}$ are as shown in Figure 2. And the BIC score of m_2 exceeds that of m_1 by threshold δ. Then UD-test fails and we stop growing the subset. The sibling cluster $\{X_1, X_2, X_4\}$ of model m_2 is picked as the sibling cluster for the whole algorithm.

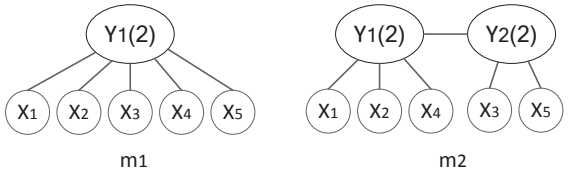

Figure 2: Model m_1 and model m_2 that are considered in UD-test.

3.2 Remaining Steps

We have described **Step 1** in Section 3.1. In the following, we describe the remaining steps. **Step 2**: A *latent class model* (LCM) is an LTM with only one latent variable. Model m_1 in Figure 2 is an example of LCM. It is a commonly used finite mixture model for discrete data. At Step 2, BI learns an LCM for each sibling cluster. It starts with an initial LCM with a binary

latent variable. The model parameters are optimized by running the EM algorithm (Koller and Friedman, 2009). 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 two steps, BI obtained a collection of LCMs. In this step, we link up these LCMs in a tree formation by adding edges between the latent variables.

Chow and Liu (1968) 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 from two disjoint LCMs is to be estimated.

Given an LCM L_1 with latent variable Y_1, we can first estimate probability $P(Y_1 \mid L_1, \mathbf{d}_i)$ for each datacase $\mathbf{d}_i$. We can do this for all latent variables. In this way, we completed the data by using these LCMs. Assume another LCM L_2 with latent variable Y_2. We can calculate the MI between Y_1 and Y_2 from the completed data. More specifically, the mutual information I(Y_1; Y_2) is computed from the following joint distribution:

$$\begin{aligned} &P(Y_1, Y_2 \mid \mathbf{D}, L_1, L_2) \\ &= C \sum_{i=1}^{N} P(Y_1 \mid L_1, \mathbf{d}_i) P(Y_2 \mid L_2, \mathbf{d}_i) \end{aligned}$$

where C is the normalization constant, and $\mathbf{d}_i$ ($i \in \{1, 2, \ldots, N\}$) is the *ith* datacase in dataset $\mathbf{D}$.
Step 4: 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. In Step 4, BI tries to detect the possible mistakes made in those steps. More specifically, BI checks each attribute to see whether it should be relocated and each latent variable to see if its cardinality should be changed.

In this step, BI first optimizes the probability parameters of the model resulted from the previous step using EM algorithm. The optimized model is denoted by $\hat{m}$. Then, similar to Step 3, for one latent variable Y_i in $\hat{m}$, we can estimate probability $P(Y_i \mid \hat{m}, \mathbf{d}_i)$ for each datacase $\mathbf{d}_i$. Some message passing algorithms (Koller and Friedman, 2009) can be used to facilitate this task. We can do this for all latent variables in the whole model $\hat{m}$. Then the data is re-completed by using the whole model. For each observed variable X and each latent variable Y, BI computes their mutual information $I(X; Y)$ from the completed data. Specifically, MI is computed from the following joint distribution:

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

where C' is the normalization constant. Let $\hat{Y}$ be the latent variable that has the highest 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 (Chen *et al.*, 2012) 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.

All the potential adjustments are evaluated with respect to $\hat{m}$. 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.

After model refinement, we run EM algorithm on the whole model one more time to optimize the parameters.

4 Empirical Results

We test our algorithm on two real world datasets. The first one is a text data 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. Four categories of web pages are used, namely student, faculty, project and course. There are 1041 pages and the number of words was reduced to 336. Stop words and words with a low occurrence frequency are removed. The word attributes take binary values which indicate the presence or absence of the words. The second dataset is a survey data obtained from ICAC which is the anti-corruption agency of Hong Kong. This survey aims to understand public opinion towards corruption and the work performance of ICAC. After preprocessing, the dataset consists of 31 questions and 1200 records. There are missing values since some respondents do not answer all questions.

In all our experiments, parameter δ in UD-test is set to 3 which is a suggested threshold by Kass and Raftery (1995).

4.1 Results on Text Data

On the WebKB data set, BI produced an LTM with 75 latent variables. Each latent variable represents a partition. Two big questions are: (1) Are those latent variables representing meaningful partitions? (2) How can users quickly identify the desired partitions? To answer the questions, convenient tools are needed to inspect the meaning of latent variables. One such GUI tool called Lantern[1] is developed for this task. Given an LTM, one can easily view the tree structure, draw the information curves of each latent variable and examine *class conditional probability distributions* (CCPDs) by using Lantern.

Information curves: To grasp the meaning of a partition, it suffices to ask on which attributes the classes differ significantly. The information curves can help us to find the most informative attributes. For example, we draw the information curves of Y_{28} as shown in Figure 3. There are two curves in the figure. The lower curve shows the *pairwise mutual information* (PMI) $I(Y_{28}; X_i)$ between Y_{28} and each attribute X_i $(i = 1, 2, \ldots, n)$. We sorted the attributes in decreasing order of PMI. The upper curve shows the *cumulative mutual information* (CMI) $I(Y_{28}; X_1{-}X_i)$ $(i = 2, 3, \ldots, n)$ between Y_{28} and the first i attributes. The ratio $I(Y; X_1{-}X_i)/I(Y; X_1{-}X_n)$ is the *cumulative information coverage* (IC) of the first i attributes. Only the top 8 attributes are shown here. The IC of the first 8 attributes is around 95%. Intuitively, this means that the differences among the different clusters on the first 8 attributes account for 95% of the total differences. So, according to the information curves, we can say that the partition is primarily based on these attributes. Here, it is clear that the partitions represented by Y_{28} is about programming.

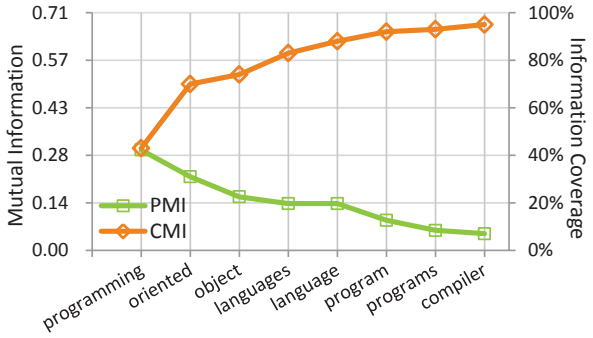

Figure 3: Information curves of latent variable Y_{28}.

Y_{28}: IC=95%

cluster	1	2	3	4
programming	1	.77	.06	.72
oriented	.16	0	0	1
object	.51	.05	.02	.91
languages	.12	.41	.01	.5
language	.81	.34	.05	.61
program	.92	.29	.09	.31
programs	.53	.23	.04	.17
compiler	.31	.18	.01	.17
Size	.04	.21	.69	.06

CCPDs: To further examine how the classes differ on the attributes, we can check its *class conditional probability distributions* (CCPDs), i.e., the distributions of attributes in the class. For latent variable Y_{28}, it has 4 states, each of which represents a cluster. The table above shows part of its CCPDs. To save space, only the occurrence frequencies of the words in each cluster are shown. For example, the value in line 2 and column 3 indicates the probability that word *programming* appears in class $Y_{28} = 2$

[1] http://www.cse.ust.hk/~lzhang/ltm/index.htm

is 0.77, i.e., $P(programming = 1 \mid Y_{28} = 2) = 0.77$. Probability for the absence of the word in this class is omitted. The CCPDs show that Y_{28}=4 identifies web pages on *objected-oriented programming* (OOP) since the probabilities of all words are high, while Y_{28}=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_{28}=1 seems to correspond to web pages of other courses that involve programming, while Y_{28}=3 seems to mean web pages not on programming.

Similar analysis can be done to other latent variables. Two more examples are given below. The most informative attributes are also selected based on information coverage.

Y_{55}: IC=96%				Y_{57}: IC=100%		
cluster	1	2	3	cluster	1	2
networks	.88	.43	.01	intelligence	.03	.59
communication	.03	.39	.01	artificial	.03	.58
neural	.77	0	.01	knowledge	.03	.6
protocols	0	.23	0	ai	.02	.49
protocol	0	.19	0	intelligent	.01	.34
intelligence	.68	.03	.05	planning	.01	.32
artificial	.66	.03	.05	reasoning	.01	.33
parallel	.2	.43	.11	learning	.05	.31
network	.14	.33	.06	logic	.05	.3
architecture	.15	.33	.07	neural	.02	.19
high	.17	.38	.09	lisp	.01	.11
performance	.17	.36	.09	networks	.09	.26
Size	.04	.14	.82	Size	.93	.07

We can see that Y_{55} seems to be related with networks. The probabilities of four words, i.e. *networks*, *neural*, *artificial* and *intelligence*, are high in state 1 and low in other two states. Y_{55}=1 seems to mean the networks in AI area (i.e. neural networks) and Y_{55}=2 is about networks in networking and high performance computing area. Y_{57}=2 seems to identify web pages belonging to faculty members who work in artificial intelligence area.

We can use Lantern to quickly identify dozens of meaningful partitions from the LTM produced by BI. To show the richness of partitions, ten other partitions are listed in Table 1. Only the words of each partition are shown. The first 4 variables Y_{10}, Y_{47}, Y_{46} and Y_{35} correspond to 4 universities. Y_6, same as Y_{28}, is a partition about course. Y_{73} and Y_{49} are partitions related to faculty homepages. Latent variables Y_{69}, Y_{71} and Y_{59}, same as Y_{55} and Y_{57}, represent partitions of different research areas.

4.1.1 Comparison with Alternative Methods

In this section, we compare BI with four other MPC algorithms: orthogonal projection (OP) (Cui *et al.*, 2007), singular alternative clustering (SAC) (Qi and Davidson, 2009), DK (Jain *et al.*, 2008) and EAST. For DK, OP and SAC, 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 true class partitions divides the web pages into four classes according to the four universities. BI has recovered the four university classes. However, they were given in the form of four latent variables instead of one. For comparability, we transformed the 4-class university 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 true class partition and the partitions obtained by the alternative algorithms. After the transformations, we matched up the binary class partitions with the obtained partitions and computed the *normalized mutual information* (NMI)(Strehl *et al.*, 2002) of each matched pair. The NMI between two partitions C and Y is given by $NMI(C;Y) = I(C;Y)/\sqrt{H(C)H(Y)}$, where I(.) stands for the mutual information and H(.) stands for the entropy. The results are shown in following table. The average was taken over 10 runs of the algorithms.

	DK	SAC	OP	BI
course	.43±.01	.47±.01	.47±.02	**.63±.02**
faculty	.18±.04	.17±.07	.18±.01	**.30±.01**
project	.04±.00	.04±.00	.05±.04	**.07±.00**
student	.18±.00	.20±.00	.20±.01	**.25±.01**
cornell	.22±.15	.09±.02	**.36±.24**	.34±.01
texas	.31±.18	.20±.20	.45±.23	**.61±.02**
washington	.22±.13	.41±.23	.56±.25	**.59±.12**
wisconsin	.38±.12	.16±.12	.45±.13	**.55±.11**

We see that the NMI is the highest for BI in almost all cases. We also tested EAST on WebKB data. However, it did not finish in 14 days. In contrast, BI took only 1.1 hour.

4.2 Results on Survey Data

On the survey data, BI produced an LTM with 7 latent variables. The structure of the model is shown in Figure 4. Similar as in Section 4.1,

Table 1: 10 other partitions found by BI. We cut the words when IC $\geq$ 95% and only nine words at most are shown.

Universities				**Course**	**Faculty Homepage**		**Research Areas**		
Y_{10}	Y_{47}	Y_{46}	Y_{35}	Y_6	Y_{73}	Y_{49}	Y_{69}	Y_{71}	Y_{59}
ithaca	washington	utexas	wisc	assignments	journal	ph	image	management	high
ny	cse	texas	madison	instructor	pp	fax	video	database	performance
cornell	uw	austin	wisconsin	hours	vol	research	vision	systems	hardware
upson		ut	dayton	syllabus	conference	professor	images	system	software
hall			wi	class	proceedings	university	pattern	databases	instruction
				grading	international	publications	digital	storage	parallel
				pm	symposium	interests	applications	large	network
				lecture	acm			support	architecture
				homework	workshop			applications	cache

Table 2: The CCPDs of Y_1. The information coverage of the four attributes is around 98%. The states of Income: s_0 (none), s_1 (less than 4k), s_2 (4-7k), s_3 (7-10k), s_4 (10-20k), s_5 (20-40k), s_6 (more than 40k). The states of Age: s_0 (15-24), s_1 (25-34), s_2 (35-44), s_3 (45-54), s_4 (above 55). The states of Education: s_0 (none), s_1 (primary), s_2 (Form 1-3), s_3 (Form 4-5), s_4 (Form 6-7), s_5 (diploma), s_6 (degree). The states of Sex: s_0 (male), s_1 (female).

$Y_1 = 1$ Size: 0.37

P(· | Y_1)	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	0	.04	.1	.42	.28	.14	.02
Age	.05	.35	.39	.17	.03		
Education	0	0	.04	.41	.09	.09	.37
Sex	.57	.43					

$Y_1 = 2$ Size: 0.24

P(· | Y_1)	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	.29	.24	.04	0	0	0	.43
Age	.03	.08	.41	.35	.13		
Education	.05	.29	.35	.26	.04	0	.01
Sex	0	1					

$Y_1 = 3$ Size: 0.22

P(· | Y_1)	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	.11	.17	.25	.31	.07	0	.1
Age	0	.07	.22	.4	.3		
Education	.02	.29	.43	.19	.05	.01	0
Sex	.8	.2					

$Y_1 = 4$ Size: 0.17

P(· | Y_1)	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	.78	.08	.09	.03	0	0	.02
Age	.99	.01	0	0	0		
Education	0	0	.08	.47	.21	.1	.16
Sex	.5	.5					

we can identify the meaning of each partition by checking the information curves and CCPDs of each latent variable. We first look at latent variables Y_1 and Y_2. The most informative attributes of Y_1 are *Income*, *Age*, *Education* and *Sex*. These attributes are also selected based on information curves. It is clear that Y_1 partitions people based on *demographic information*. The CCPDs of Y_1 are given in Table 2. It has 4 states, each of which represents a cluster. We begin with cluster $Y_1 = 4$. It consists people aged between 15 and 24. The the average income is significantly low (78% of people in this cluster do not have income). So $Y_1 = 4$ represents a class of *low income youngsters*. Cluster $Y_1 = 2$ only consists of women. Between the remaining two clusters, $Y_1 = 1$ has, on average, higher education and higher income than $Y_1 = 3$. Hence $Y_1 = 1$ represents a class of people with good education and good income, while $Y_1 = 3$ represents a class of people with poor education and average income.

The most informative attributes of Y_2 are *C-NextY* (change in the level of corruption next year) and *C-PastY* (change in the level of corruption in the past year). So Y_2 is about *people's view on the change of corruption level*. The C-CPDs of Y_2 is given in Table 3. Take $Y_2 = 2$ as an example. It represents a group of people (51%) who think the corruption level has remained the same in the past year (86%) and will remain the same next year (94%).

Table 3: The CCPDs of Y_2. The states of C-NextY: s_0 (increase), s_1 (decrease), and s_2 (same). The states C-PastY: s_0 (increased), s_1 (decreased), and s_2 (same).

	$P(Y_2 = 1) = 0.21$			$P(Y_2 = 2) = 0.51$			$P(Y_2 = 3) = 0.28$		
P(· | Y_2)	s_0	s_1	s_2	s_0	s_1	s_2	s_0	s_1	s_2
C-NextY	.03	.84	.13	.01	.05	.94	.79	0	.21
C-PastY	.07	.5	.43	.06	.08	.86	.66	.04	.31

Similar analysis can be done to other latent variables, we will find that Y_3 is a partition about people's *tolerance towards corruption*, Y_4 and Y_5 are about *ICAC's performance and accountability*, Y_6 relates to different *corruption scene*, Y_7 partitions people on their *view about economy*.

As an advantage, the relationship between partitions can be also inferred from the resulting LTM. For example, the relationship between Y_1 and Y_2 is revealed by the conditional probability $P(Y_2 \mid Y_1)$ which is associated with the edge $Y_1 \rightarrow Y_2$.

$P(Y_2 \mid Y_1)$	$Y_2 = 1$	$Y_2 = 2$	$Y_2 = 3$
$Y_1 = 1$	.06	.59	.34
$Y_1 = 2$	.30	.42	.28
$Y_1 = 3$	.27	.39	.33
$Y_1 = 4$	.34	60	.06

We can see from the last line of above table, among the youngsters ($Y_1 = 4$), 34% of them

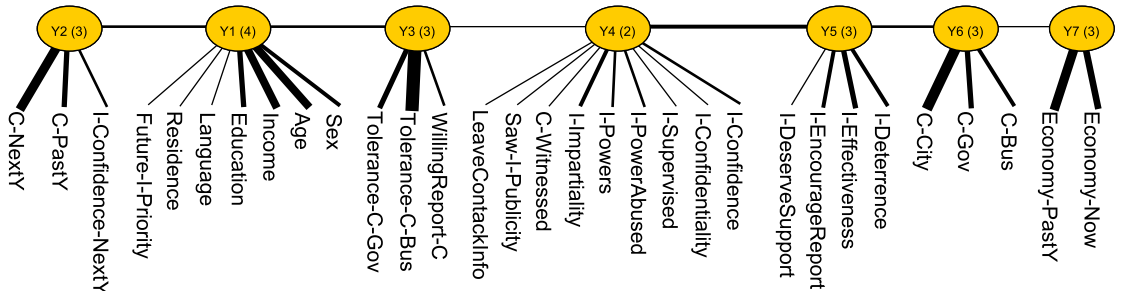

Figure 4: The structure of LTM learned by BI from the ICAC data. The width of edges represent strength of probabilistic dependence. Abbreviations: C-Corruption, I-ICAC, Y-Year, Gov-Government, Bus-Business Sector. Meanings of attributes: Tolerance-C-Gov means 'tolerance towards corruption in the government'; C-City means 'level of corruption in the city'; C-NextY means 'change in the level of corruption next year'; I-Effectiveness means 'effectiveness of ICAC's work'; I-Powers means 'ICAC powers'; etc.

have the positive view on the change of corruption level ($Y_2 = 1$), while 6% of them have the negative view ($Y_2 = 3$) on the same issue. This kind of information may be useful for users to understand the data.

For this unlabeled data, we also run EAST on it and compare the quality of models produced by both methods in terms of BIC score. EAST takes 11312±965 seconds to learn the model. The average is take over 10 runs. The average BIC score of the learned model is -26042±28. BI takes 562±39 seconds on average. The average BIC score for the learned model is -26096±13. BI archived comparative performance in much less time. For other alternative methods DK, OP and SAC, their results are not included since they can not handle datasets with missing values.

5 Conclutions

In this paper, we propose a greedy method for learning LTM. The new method is faster than previous algorithms. Empirical results are presented to show that our method is able to produce rich and meaningful partitions. An GUI tool is also provided to facilitate the analysis of these partitions.

Acknowledgments

Research on this paper was supported by China National Basic Research 973 Program project No. 2011CB505101 and Guangzhou HKUST Fok Ying Tung Research Institute.

References

Chen T, Zhang N.L, Wang Y (2008) Efficient model evaluation in the search-based approach to latent structure discovery. In: PGM-08, 57-64

Chen T, Zhang N.L, Liu T.F, Poon K, Wang Y (2012) Model-based multidimensional clustering of categorical data. Artif Intell 176:2246–2269

Choi M.J, Tan V.Y.F, Anandkumar A, Willsky A.S (2011) Learning latent tree graphical models. Journal of Machine Learning Research 12:1771–1812

Chow C.K, Liu C.N (1968) Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3):462–467

Cui Y, Fern X.Z, Dy J.G (2007) Non-reduntant multi-view clustering via orthogonalization. In: ICDM-07

Galimberti G, Soffritti G (2007) Model-based methods to identify multiple cluster structures in a data set. CSDA-07 52:520–536

Gondek D, Hofmann T (2007) Non-redundant data clustering. KAIS-07 12(1):1–24

Guan Y, Dy J.G, Niu D, Ghahramani Z (2010) Variational inference for nonparametric multiple clustering. In: MultiClust Workshop, KDD-2010

Harmeling S, Williams C.K.I (2010) Greedy learning of binary latent trees. TPAMI-10

Jain P, Meka R, Dhillon I.S (2008) Simultaneous unsupervised learning of disparate clusterings. SDM-08

Kass R.E, Raftery A.E (1995) Bayes Factors. Journal of the American Statistical Association 90(430):773–795

Koller D, Friedman N (2009) Probabilistic Graphical Models: Principles and Techniques. MIT Press

Niu D, Dy J.G, Jordan M.I (2010) Multiple non-redundant spectral clustering views. In: ICML-10

Poon K.M, Zhang N.L, Chen T, Yi W (2010) Variable selection in model-based clustering: To do or to facilitate. In: ICML-10

Qi Z, Davidson I (2009) A principled and flexible framework for finding alternative clusterings. In: KDD-09

Schwarz G (1978) Estimating the dimension of a model. The Annals of Statistics 6(2):461–464

Strehl A, Ghosh J, Cardie C (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. JMLR 3:583–617

Zhang N.L (2004) Hierarchical latent class models for cluster analysis. JMLR-04 5:697–723