Artificial Intelligence 176 (2012) 2246–2269

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Model-based multidimensional clustering of categorical data

Tao Chen [a], Nevin L. Zhang [b,*], Tengfei Liu [b], Kin Man Poon [b], Yi Wang [c]

[a] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[b] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China
[c] Department of Computer Science, National University of Singapore, Singapore 117417, Singapore

ARTICLE INFO

Article history:
Received 29 May 2010
Received in revised form 2 July 2011
Accepted 29 September 2011
Available online 6 October 2011

Keywords:
Model-based clustering
Categorical data
Multidimensional clustering
Latent tree models

ABSTRACT

Existing models for cluster analysis typically consist of a number of attributes that describe the objects to be partitioned and one single latent variable that represents the clusters to be identified. When one analyzes data using such a model, one is looking for *one* way to cluster data that is jointly defined by all the attributes. In other words, one performs unidimensional clustering. This is not always appropriate. For complex data with many attributes, it is more reasonable to consider multidimensional clustering, i.e., to partition data along multiple dimensions. In this paper, we present a method for performing multidimensional clustering on categorical data and show its superiority over unidimensional clustering.

1. Introduction

Cluster analysis is about the grouping of similar objects into meaningful clusters. There are two different approaches. In the distance-based approach, one defines a distance/similarity measure between objects and assigns similar objects to the same cluster. In the model-based approach, one assumes that data are generated by a finite mixture model (FMM), estimates parameters of the model from data, and assigns objects to clusters based on posterior probability. In both approaches, one can perform either partitional clustering or hierarchical clustering. In this paper, we focus on partitional clustering.

In an FMM, there are a number of observed variables that represent attributes of objects, and there is one single latent variable that represents the clusters to be identified. The latent variable has a finite number of states, each corresponding to a cluster. Assumptions are imposed on the conditional distributions of the attributes given the latent variable. Different assumptions lead to different models. In Gaussian mixture models (GMMs) [28], the attributes are continuous and are assumed to jointly follow a Gaussian distribution. In latent class models (LCMs) [26], the attributes are discrete. They are assumed to be mutually independent given the latent variable and each follows a multinomial distribution.

GMMs and LCMs are two basic types of models for cluster analysis. A variety of restrictions, extensions, combinations, and variations have been proposed. For example, Banfield and Raftery [2] place constraints on the covariance matrices in GMMs to reduce model parameters. This technique is used in the MCLUST program [14]. In their well-known AutoClass program, Cheeseman and Stutz [4] use a version of LCM where the attributes can be either continuous or discrete. The MULTIMIX program by Hunt and Jorgensen [22] also allows both discrete and continuous attributes. Unlike AutoClass, it does not assume conditional independence for continuous attributes. Peña et al. [32] start from LCMs with continuous attributes and relax the independence assumption by adding edges between the attributes. Law et al. [25] add saliency variables to the AutoClass model to facilitate feature selection. This work is recently extended by Li et al. [27]. McLachlan

* Corresponding author.
E-mail addresses: tao.chen@siat.ac.cn (T. Chen), lzhang@cse.ust.hk (N.L. Zhang), liutf@cse.ust.hk (T. Liu), lkmpoon@cse.ust.hk (K.M. Poon), wangy@comp.nus.edu.sg (Y. Wang).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2011.09.003

and Peel [29] consider mixtures of factor models for the purpose of dimension reduction. Hoff [21] introduces a variant of GMMs that allows different clusters to be characterized by different subsets of attributes. Zhong and Ghosh [43] propose a bipartite graph framework that bridges various model-based and distance-based clustering methods.

1.1. Unidimensional clustering

In all variants of FMM, there is a single latent variable. This implies that, when one uses an FMM to analyze data, one is looking for one single way to partition the data. In other words, one performs *unidimensional clustering*. One assumption is made here, that is, all the attributes in data jointly define one single meaningful way to partition the objects. This assumption is hardly true except maybe when there are only a few attributes. When the number of attributes is large, there might be multiple meaningful ways to partition data. In such cases, if one insists on looking for one single way to cluster data, one might not find any meaningful partitions at all.

How can one perform cluster analysis when all the attributes in a data set jointly do not define a coherent partition of the objects? One might suggest to rely on experts to identify subsets of attributes that do define coherent partitions and perform cluster analysis based on each subset. This approach has two drawbacks. First, it is not purely unsupervised and cannot be applied to situations when one does not know not only how to partition the objects, but also based on what attributes to partition the objects. Second, performing cluster analysis based on a given subset of attributes is to partition the objects along a certain dimension. When the analysis on different subsets is carried out independently, relationships between the different dimensions are not revealed.

Another suggestion might be to reduce the number of attributes through feature selection, so that the remaining attributes can better reveal the 'true' clusters. Feature selection is well studied for classification. However, there has been little work on feature selection for clustering, especially for model-based clustering [35,25]. It is known to be a very difficult problem. We argue that the root of the difficulty lies in the assumption that there is a single 'true' way to cluster data. The assumption is almost never true except perhaps when there are only a few attributes. When there are more than one meaningful way to partition data, the attempt to find a single 'true' partition is misguided. So are any efforts on selecting a 'good' subset of attributes for the endeavor.

1.2. Multidimensional clustering

We advocate *multidimensional clustering*. The idea is to use statistical principles to find multiple potentially overlapping subsets of attributes, each of which defining a coherent partition, and carry out cluster analysis based on those subsets simultaneously within a single model.

As mentioned earlier, we focus on partitional clustering in this paper. A partition of a collection of objects corresponds to a variable (or a concept) about the objects. For example, the division of human beings into males and females corresponds to the variable 'gender', and the partition of the world population into countries corresponds to the variable 'nationality'. Therefore, to partition a collection of objects is to identify a variable about those objects. Such a variable is not observed and is hence called a *latent variable*. In contrast, the attributes of objects are observed and are called *manifest variables*.

Conceptually, model-based cluster analysis takes place in three steps. First, one assumes a probabilistic model about the relationships between latent variables and manifest variables. Then, one determines the details of the model from data using statistical principles. Finally, one reads off partitions from the model. Each latent variable in the model represents a partition, and each of its states represents a cluster in the partition. The states of latent variables are hence called *latent classes*. When there is one single latent variable in the model, one is talking about unidimensional clustering. When there are multiple latent variables in the model, one is talking about multidimensional clustering.

Multidimensional clustering should be distinguished from multiple unidimensional clusterings. In the latter case, one obtains multiple models, each with a single latent variable. In multidimensional clustering, on the other hand, one obtains a single model with multiple latent variables. Relationships among latent variables are determined in multidimensional clustering, but not multiple unidimensional clusterings.

1.3. Cluster analysis of categorical data

While much of recent machine learning research focuses on text, image, and biology data, this paper is targeted at traditional survey-like categorical data, with dozens to over one hundred attributes and hundreds to thousands of records. Analysis of such data is interesting to psychology, social sciences, education, and marketing research.

LCMs and their variants are commonly used to analyze categorical data. Technically, an LCM is a Bayesian network [33] with one discrete latent variable and a number of discrete manifest variables. The manifest variables are assumed to be mutually independent given the latent variable. It is the same as the Naïve Bayes model, except that the class variable is hidden. The analysis of data using LCMs is known as *latent class analysis*. Performing latent class analysis on a data set means to:

- Determine the *cardinality* (i.e., the number of states) of the unique latent variable, or the number of clusters;
- Estimate model parameters, or determine the statistical properties of the clusters.

The first item is the model selection problem. In the latent class analysis literature, it is determined using goodness-of-fit tests [40] or model selection criteria such as the BIC score [39]. Model parameters are usually estimated using the EM algorithm [12].

In this paper, we study the use of a generalization of LCMs, called *latent tree models (LTMs)*, in cluster analysis of categorical data. An LTM is a tree structured Bayesian networks where leaf nodes represent manifest variables and internal nodes represent latent variables. All variables are discrete. The analysis of data using LTMs will be referred to as *latent tree analysis*. While latent class analysis results in unidimensional clusterings, latent tree analysis can produce multidimensional clusterings. Performing latent tree analysis on a data set means to:

1. Determine the number of latent variables, or the number of ways to partition data;
2. Determine the cardinality of each latent variable, or the number of clusters in each partition;
3. Determine the connections between the latent variables and the manifest variables and among the latent variables themselves; and
4. Estimate model parameters.

The first three items make up the model selection problem. In this paper, we use the BIC score for model selection. The reasons will be explained in Section 2.1. For parameter estimation, we use the EM algorithm.

In the LTM resulted from latent tree analysis, there are usually multiple latent variables. Each latent variable might be mainly related to only a subset of attributes and different latent variables might be related to different subsets. So, the results are multiple partitions of data based on potentially overlapping subsets of attributes. The relationships among the different partitions (latent variables) are also determined.

1.4. Contributions and organization

The idea of using LTM for multidimensional clustering was first proposed by Zhang [42], where LTMs were called hierarchical latent class models. An algorithm for learning LTMs from data was also proposed there. The algorithm hill-climbs in the space of LTMs guided by a scoring function. It starts with an LCM. At each step of search, it first generates a number of candidate structures by modifying the structure of the current model. It then optimizes cardinalities of latent variables, resulting in candidate models. Finally, it evaluates the candidate models and picks the best one to seed the next step of search. Search terminates when the best candidate model is no better than the current model. To optimize the cardinalities of the latent variables in a model structure, the algorithm employs another hill-climbing routine. Hence we call it the *double hill-climbing (DHC) algorithm*.

Empirical evaluation has shown that the DHC algorithm performs well in terms of model quality when coupled with the BIC score [42]. However, it has a serious drawback, namely its high complexity. Let n and l be the numbers of observed and latent variables, r be the maximum number of neighbors a node can have, and k be the maximum cardinality for a latent variable. At each step of search, DHC needs to evaluate $O((lr^2 + ln)lk^2)$ candidate models. Due to the presence of latent variables, parameter optimization requires the EM algorithm, which is known to be computationally expensive. DHC is so inefficient that it is unable to deal with data sets with more than half a dozen attributes.

We make two contributions in this paper. First, we propose a new search-based algorithm, called EAST, for learning LTMs (Section 3). EAST adopts a heuristic search strategy known as grow-restructure-thin, which originated from the literature on learning Bayesian networks without latent variables (e.g., [7]). At each step, it examines $O(\max\{lr^2, ln\})$ candidate models and runs what we call local EM (instead of full EM) when evaluating the candidate models. Consequently, EAST is much more efficient than DHC. To be specific, it is able to deal with data sets with more than 100 attributes. This enables one to perform multidimensional cluster analysis on real-world survey-like categorical data from psychology, social sciences, education, and marketing research.

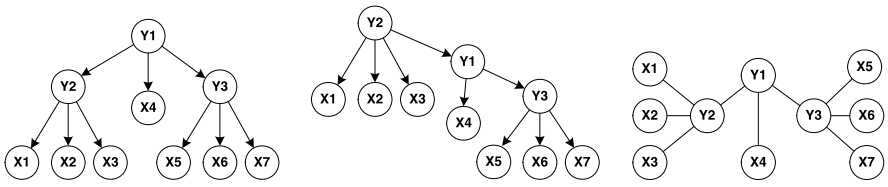
Second, we show empirically that: (1) complex data often can be clustered meaningfully in multiple ways, and (2) latent tree analysis with the EAST algorithm can obtain rich and interesting clustering results from complex data (Section 4). This second point is the most important contribution of this paper. We also compare latent tree analysis with other methods that produce multiple partitions, as well as latent class analysis, first on real-world unlabeled data (Section 5) and then on synthetic and real-world labeled data (Section 6).

We begin in Section 2 with a brief review of basic concepts and facts about LTMs. Related works are discussed in Section 7 and conclusions are given Section 8.

2. Basics of LTMs

Fig. 1(a) shows the structure of an LTM. The leaf nodes X_1–X_7 represent manifest variables, while the internal nodes Y_1–Y_3 represent latent variables. In this paper, we use the terms 'node' and 'variable' interchangeably. All the variables are categorical, each taking a finite number of values.

In an LTM, each node Z is associated with a conditional distribution $P(Z|pa(Z))$ that characterizes how Z depends on its parent $pa(Z)$. If Z is the root, then it has no parent and is associated with a marginal distribution $P(Z)$. All the probabilistic distributions make up the parameters of the LTM. We denote the collection of all parameters by θ. The rest of the LTM is



(a) m (rooted) (b) m' (after root walking) (c) m (unrooted)

Fig. 1. Rooted latent tree model, root walking, and unrooted latent tree model. The X's are manifest variables and the Y's are latent variables.

denoted by m. It consists of the variables, the cardinalities of the variables, and the model structure. We sometimes write an LTM as a pair $M = (m, \theta)$ and refer to the first component m also as an LTM.

2.1. Learning LTMs

Suppose $\mathcal{D}$ is a collection of data over a set $\mathbf{X}$ of variables. There are infinitely many possible LTMs with $\mathbf{X}$ as manifest nodes. For this paper, to learn an LTM from data $\mathcal{D}$ means to find the LTM that is optimal according to some scoring function. We choose to use the BIC score [39]. The *BIC score of a model* m is:

$$\begin{aligned} BIC(m|\mathcal{D}) &= \max_{\theta} \log P(\mathcal{D}|m,\theta) - \frac{d(m)}{2}\log N \\ &= \log P\big(\mathcal{D}|m,\theta^*\big) - \frac{d(m)}{2}\log N \end{aligned} \tag{1}$$

where $d(m)$ is *dimension*, i.e., the number of independent parameters of the model, N is the sample size, and θ^* is the maximum likelihood estimation (MLE) of the parameters. The first term is known as the *maximized log likelihood of* m. It measures how well model m fits the data $\mathcal{D}$. The second term is a penalty term for model complexity.

There are a number of other scoring functions. As a matter of fact, BIC is one large sample approximation of the marginal likelihood [39]. There are other approximations. The Cheeseman–Stutz (CS) score [4] tries to compensate for the error in the BIC approximation using completed data. The BICe score [16] tries to do the same by replacing $d(m)$ with the effective dimension of m. Other scoring functions include holdout likelihood [10], AIC [1], MDL [36], and normalized maximum likelihood [37]. We choose to work with BIC because it is frequently used by other researchers and it has worked well in our previous research. It should be noted, however, that our method can be coupled with other scoring functions as well.

2.2. Model inclusion and equivalence

Consider two LTMs m and m' that share the same manifest variables $X_1, X_2, \ldots, X_n$. We say that m *includes* m' if for any parameter value θ' of m', there exists parameter value θ of m such that

$$P(X_1, \ldots, X_n|m,\theta) = P\big(X_1, \ldots, X_n|m',\theta'\big).$$

When this is the case, m can represent any distributions over the manifest variables that m' can. As such, the maximized log likelihood of m is larger than or equal to that of m':

$$\max_{\theta} \log P(\mathcal{D}|m,\theta) \geqslant \max_{\theta'} \log P\big(\mathcal{D}|m',\theta'\big).$$

If m includes m' and vice versa, we say that m and m' are *marginally equivalent*. Marginally equivalent models are *equivalent* if they have the same number of independent parameters. It is impossible to distinguish between equivalent models based on data if the BIC score, or any other penalized likelihood score [18], is used for model selection.

2.3. Root walking and unrooted LTMs

Let Y_1 be the root of an LTM m. Suppose Y_2 is a child of Y_1 and it is also a latent node. Define another LTM m' by reversing the arrow $Y_1 \rightarrow Y_2$. Variable Y_2 becomes the root in the new model. The operation is called *root walking*; the root has walked from Y_1 to Y_2. The model m' in Fig. 1(b) is the model obtained by walking the root from Y_1 to Y_2 in model m.

It has been shown that root walking leads to equivalent models [42]. Therefore, the root and edge orientations of an LTM cannot be determined from data. We can only learn *unrooted LTMs*, that is, LTMs with all directions on the edges dropped. An example of an unrooted LTM is given in Fig. 1(c).

An unrooted LTM represents an equivalent class of LTMs. Members of the class are obtained by rooting the model at various latent nodes. Semantically it is a Markov random field over an undirected tree. The external nodes are observed while the interior nodes are latent. Model inclusion and equivalence can be defined for unrooted LTMs in the same way as for rooted models. In the rest of this paper, LTMs always mean unrooted LTMs unless it is explicitly stated otherwise.

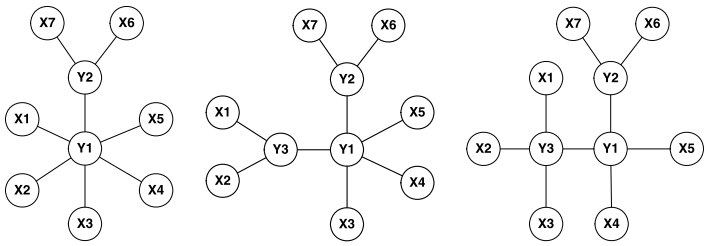


Fig. 2. The NI and NR operators. The model m_2 is obtained from m_1 by introducing a new latent node Y_3 to mediate between Y_1 and two of its neighbors X_1 and X_2. The cardinality of Y_3 is set to be the same as that of Y_1. The model m_3 is obtained from m_2 by relocating X_3 from Y_1 to Y_3.

2.4. Regular LTMs

For a latent variable Y in an LTM, enumerate its neighbors as $W_1, W_2, \ldots, W_r$. An LTM is *regular* if for any latent variable Y,

$$|Y| \leqslant \frac{\prod_{i=1}^{r} |W_i|}{\max_{i=1}^{r} |W_i|}, \tag{2}$$

and when Y has only two neighbors, strict inequality holds and one of the neighbors must be a latent node.

For any irregular model m, there always exists a regular model m' that is marginally equivalent to m and has fewer independent parameters [42]. The model m' can be obtained from m through the following *regularization* process:

1. For each latent variable Y in m,
   (a) If it violates inequality (2), reduce the cardinality of Y to $\frac{\prod_{i=1}^{r} |W_i|}{\max_{i=1}^{r} |W_i|}$.
   (b) If it has only two neighbors with one being a latent node and it violates the strict version of inequality (2), remove Y from m and connect the two neighbors of Y.
2. Repeat Step 1 until no further changes.

The regular model m' has a higher BIC score than m itself. Therefore, we can restrict our attention to the space of regular models when searching for the LTM with the highest BIC score. For a given set of manifest variables, there are only finitely many regular LTMs [42].

3. An algorithm for learning LTMs

In this section, we present a new algorithm for learning LTMs called EAST [6]. We start with the operators and the search procedure (Section 3.1). Then, we discuss two issues that are critical to the performance of EAST, namely efficient model evaluation (Section 3.2) and operation granularity (Section 3.3).

3.1. Search operators and search procedure

EAST hill-climbs in the space of regular LTMs under the guidance of the BIC score. It uses five search operators and it adopts a search strategy known as grow-restructure-thin, which originated from the literature on learning Bayesian networks without latent variables (e.g., [7]).

3.1.1. Search operators

The search operators are: state introduction (SI), node introduction (NI), node relocation (NR), state deletion (SD), and node deletion (ND). We describe them one by one in the following.

Given an LTM and a latent variable in the model, the *state introduction* (*SI*) operator creates a new model by adding a state to the domain of the variable. The *state deletion* (*SD*) operator does the opposite. Applying SI on a model m results another model that includes m. Applying SD on a model m results another model that is included by m.

Node introduction (*NI*) involves one latent node Y and two of its neighbors. It creates a new model by introducing a new latent node Z to mediate between Y and the two neighbors. The cardinality of Z is set to be the same as that of Y. In the model m_1 of Fig. 2, introducing a new latent node Y_3 to mediate Y_1 and its neighbors X_1 and X_2 results in m_2. Applying NI on a model m results another model that includes m. For the sake of computational efficiency, we do not consider introducing a new node to mediate Y and more than two of its neighbors. This restriction will be compensated in search control.

Node deletion (*ND*) is the opposite of NI. It involves two neighboring latent nodes Y and Z. It creates a new model by deleting Z and making all neighbors of Z other than Y neighbors of Y. We refer to Y as the *anchor variable* of the deletion

```
EAST(m, D):
θ* ← EM(m, D).
Repeat forever
  (m1, θ1*) ← expand(m, θ*, D).
  (m2, θ2*) ← adjust(m1, θ1*, D).
  (m3, θ3*) ← simplify(m2, θ2*, D).
  If BIC(m3, θ3*|D) ⩽ BIC(m, θ*|D),
    return (m, θ*);
  Else (m, θ*) ← (m3, θ3*).

expand(m, θ*, D):
Repeat forever
  (m1, θ1*) ← pickModel-IR(NI(m) ∪ SI(m), m, θ*).
  If BIC(m1, θ1*|D) ⩽ BIC(m, θ*|D),
    return (m, θ*).
  If m1 ∈ NI(m),
    (m, θ*) ← enhanceNI(m1, θ1*, m, D);
  Else (m, θ*) ← (m1, θ1*).

adjust(m, θ*, D):
Repeat forever
  (m1, θ1*) ← pickModel(NR(m), m, θ*).
  If BIC(m1, θ1*|D) ⩽ BIC(m, θ*|D),
    return (m, θ*);
  Else (m, θ*) ← (m1, θ1*).

simplify(m, θ*, D):
Repeat forever
  (m1, θ1*) ← pickModel(ND(m), m, θ*).
  If BIC(m1, θ1*|D) ⩽ BIC(m, θ*|D),
    break;
  Else (m, θ*) ← (m1, θ1*).
Repeat forever
  (m1, θ1*) ← pickModel(SD(m), m, θ*).
  If BIC(m1, θ1*|D) ⩽ BIC(m, θ*|D),
    return (m, θ*);
  Else (m, θ*) ← (m1, θ1*).
```

Fig. 3. The EAST algorithm. The subroutines `pickModel` and `pickModel-IR` will be given in the next two subsections, while `enhanceNI` is described in the main text. A Java implementation of the algorithm is available at: http://www.cse.ust.hk/faculty/lzhang/ltm/index.htm.

and say that Z is *deleted with respect to* Y. In the model m_2 of Fig. 2, deleting Y_3 with respect to Y_1 leads us back to the model m_1. Applying ND on a model m results another model that is included by m if the node deleted has more or the same number of states as the anchor node.

Node relocation (*NR*) involves a node W, one of its latent node neighbors Y and another latent node Z. This creates a new model by relocating W to Z, i.e., removing the link between W and Y and adding a link between W and Z. In m_2 of Fig. 2, relocating X_3 from Y_1 to Y_3 results in m_3.

There are some boundary conditions on the search operators. The SD operator cannot be applied to latent variables with only two possible states. The NI and NR operators cannot be applied if they make some latent nodes leaves. To ensure regularity, a regularization step is applied to every candidate model right after its creation.

3.1.2. Brute-force search

Let m be an LTM. In the following we use $NI(m)$, $SI(m)$, $NR(m)$, $ND(m)$, and $SD(m)$ to respectively denote the sets of candidate models that one can obtain by applying the five search operators on m. The models are sometimes referred to as *NI*, *SI*, *NR*, *ND*, *SD candidate models* respectively. The union of the five sets is denoted by $ALL(m)$.

Suppose we are given a data set $\mathcal{D}$ and an initial model m. Here is a brute-force search algorithm for learning an LTM:

```
BF(m, D):
Repeat forever
  m1 ← arg max_{m'∈ALL(m)} BIC(m'|D).
  If BIC(m1|D) ⩽ BIC(m|D),
    return m;
  Else m ← m1.
```

Brute-force search is inefficient for two reasons. First, it evaluates a large number of candidate models at each step. Let n, l, and r be the number of manifest nodes, the number of latent nodes, and the maximum number of neighbors that any latent node has in the current model respectively. The numbers of candidate models that the five operators SI, SD, NI, ND and NR generate are $O(l)$, $O(l)$, $O(lr(r-1)/2)$, $O(lr)$ and $O(l(l+n))$ respectively. So the brute-force algorithm evaluates a total number of $O(l(2+r/2+r^2/2+l+n))$ candidate models at each step. Most of the candidate models are generated by the NI and NR operators.

Second, one needs to compute the maximized log likelihood of a candidate model m' in order to calculate its BIC score. This requires the EM algorithm due to the presence of latent variables. EM is known to be time-consuming.

We will next describe a search procedure that generates fewer candidate models than brute-force search. In Section 3.2, we will present an efficient way to evaluate candidate models.

3.1.3. EAST search

The five operators can be classified into three groups. The NI and SI operators produce candidate models that include the current model. They are hence *expansion operators*. The ND and SD operators produce candidate models that are included by the current model. They are hence *simplification operators*. NR does not alter nodes in the current model. It only changes the connections between the nodes. Hence we call it an *adjustment operator*.

The EAST algorithm is given in Fig. 3. In the algorithm, candidate models are sometimes evaluated using sub-optimal parameter values. So, for any parameter value θ of an LTM m, we define

$$BIC(m,\theta|\mathcal{D}) = \log P(\mathcal{D}|m,\theta) - \frac{d(m)}{2}\log N.$$

Sometimes, parameters values must be optimized. This is done using the subroutine $\texttt{EM}(m, \mathcal{D})$ that computes the MLE of the parameters of m from data $\mathcal{D}$.

EAST starts by optimizing the parameters of the initial model. Then, it searches in three stages: expansion, adjustment and simplification. At each stage, it uses only the operators from the corresponding group, instead of all the operators. For example, it searches only with the expansion operators at the expansion stage. If the model score is improved in any of the three stages, the algorithm continues search by repeating the loop. This is why it is called 'EAST' — *Expansion, Adjustment, Simplification until Termination.*

At the expansion stage, EAST searches with the expansion operators until the BIC score ceases to increase. To understand the intuition, recall that the BIC score consists of a term that measures model fit and another term that penalizes for model complexity. If we start with a model that fit data poorly, which is usually the case, then improving model fit is the first priority. Model fit can be improved by searching with the expansion operators (see Section 2.2). This is exactly what EAST does at the expansion stage.

The pseudo code for the expansion stage contains two subroutines. The subroutine `pickModel-IR` selects one model from all the candidate models generated from m by the NI and SI operators. It is discussed in details in the next two subsections.

The second subroutine `enhanceNI` is called after each application of the NI operator. This is to compensate for the constraint imposed on NI. Consider the model m_1 in Fig. 2. We can introduce a new latent node Y_3 to mediate Y_1 and two of its neighbors, say X_1 and X_2, and thereby obtain the model m_2. However, we are not allowed to introduce a latent node to mediate Y_1 and more than two of its neighbors, say X_1, X_2 and X_3, and thereby obtain m_3. As a remedy we consider, after each application of the NI operator, enhancements to the operation. As an example suppose we have just applied NI to m_1 and have obtained m_2. What we do next is to consider relocating the other neighbors of Y_1 in m_1, i.e. X_3, X_4, X_5 and Y_2, to the new latent variable Y_3. If it turns out to be beneficial to relocate X_3 but not the other three nodes, then we obtain the model m_3.

In general, suppose we have just introduced a new node Z into the current model m to mediate a latent node Y and two of its neighbors, and obtained a candidate model m_1. Let L be the list of all the other neighbors of Y in m. For any $W \in L$, use $m_{1:W\to Z}$ to denote the model obtained from m_1 by relocating W to Z. What we do next is to enhance the NI operation using this subroutine:

$\texttt{enhanceNI}(m_1, \theta_1^*, m, \mathcal{D})$
while $L \neq \emptyset$:
  $(m_{1:W_1\to Z}, \theta_2^*) \leftarrow \texttt{pickModel}(\{m_{1:W\to Z} | W \in L\}, m_1, \theta_1^*)$.
  If $BIC(m_{1:W_1\to Z}, \theta_2^* | \mathcal{D}) \leqslant BIC(m_1, \theta_1^* | \mathcal{D})$, return (m_1, θ_1^*);
  Else $(m_1, \theta_1^*) \leftarrow (m_{1:W_1\to Z}, \theta_2^*)$, $L \leftarrow L \setminus \{W_1\}$.

The subroutine `pickModel` selects and returns one model from a list of candidate models. It will be given in the next subsection.

After model expansion ceases to increase the BIC score, EAST enters the adjustment stage. At this stage, EAST repeatedly relocates nodes in the current model using the NR operator until it is no longer beneficial to do so, and there is no restriction on how far away a node can be relocated. Node relocation is necessary because multiple latent nodes are usually introduced during model expansion and two nodes that should be together might end up at different parts of the model at the end of the expansion process.

The adjustment stage is followed by the simplification stage. At this stage EAST first repeatedly applies ND to the current model until the BIC score ceases to increase and then it does the same with SD. We choose not to consider ND and SD simultaneously because that would be computationally more expensive and it is not clear whether that would be helpful in avoiding the local maxima.

At each step in the expansion stage, EAST generates $O(l + lr(r-1)/2)$ candidate models. At each step in the adjustment stage, EAST generates $O(l(l+n))$ candidate models. The simplification stage consists of two substages. At the first substage, EAST searches with the ND operator and generates $O(lr)$ candidate models at each step. At the second substage, EAST searches with the SD operator and generates $O(l)$ candidate models at each step. So EAST generates fewer candidate models than the brute-force algorithm at each step of search.

3.2. Efficient model evaluation

The `pickModel` subroutine is supposed to find, from a list of candidate models, the model with the highest BIC score. A straightforward way to do so is to calculate the BIC score of each candidate model and then pick the best one. Calculating the BIC scores of a large number of models exactly is computationally prohibitive. So, we propose to use approximations of the BIC score for model selection. In this subsection, we present one approximation of the BIC score that is easy to compute. The idea is to replace the likelihood term with what we call restricted likelihood. We begin by discussing parameter sharing between a candidate model and the current model.

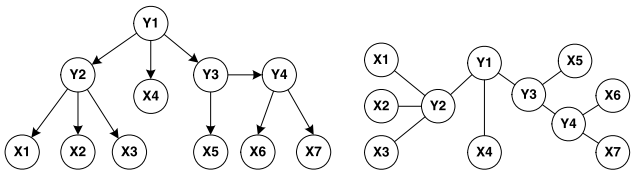


Fig. 4. A candidate model obtained by modifying the model in Fig. 1. The two models share the parameters for describing the distributions $P(Y_1)$, $P(Y_2|Y_1)$, $P(X_1|Y_2)$, $P(X_2|Y_2)$, $P(X_3|Y_2)$, $P(X_4|Y_1)$, $P(Y_3|Y_1)$ and $P(X_5|Y_3)$. On the other hand, the parameters for describing $P(Y_4|Y_3)$, $P(X_6|Y_4)$ and $P(X_7|Y_4)$ are peculiar to the candidate model.

3.2.1. Parameter sharing among models

Conceptually we work with unrooted LTMs. In implementation, however, we represent unrooted models as rooted models. Rooted LTMs are Bayesian networks and their parameters are defined without ambiguity. This makes it easy to see how the parameter composition of a candidate model is related to that of the current model.

Consider the model m in Fig. 1. Let m' be the model obtained from m by introducing a new latent node Y_4 to mediate Y_3 and two of its neighbors X_6 and X_7, as shown in Fig. 4. If both m and m' are represented as rooted models, their parameters' compositions are clear. The two models share parameters for describing the distributions $P(Y_1)$, $P(Y_2|Y_1)$, $P(X_1|Y_2)$, $P(X_2|Y_2)$, $P(X_3|Y_2)$, $P(X_4|Y_1)$, $P(Y_3|Y_1)$ and $P(X_5|Y_3)$. On the other hand, the parameters for describing $P(Y_4|Y_3)$, $P(X_6|Y_4)$ and $P(X_7|Y_4)$ are peculiar to m' while those for describing $P(X_6|Y_3)$ and $P(X_7|Y_3)$ are peculiar to m.

We write the parameters of a candidate model m' as a pair (θ_1', θ_2'), where θ_1' is the collection of parameters that m' shares with the current model m. The other parameters θ_2' are peculiar to m' and are called *new parameters* of m'. Similarly we write the parameters of the current model m as a pair (θ_1, θ_2), where θ_1 is the collection of parameters that m shares with m'. The parameters in θ_1 and θ_1' are the *common parameters*.

One unrooted LTM can be represented by multiple rooted LTMs. In the aforementioned example, if the representation of m' is rooted at Y_3 instead of Y_1, then we would have $P(Y_3)$ and $P(Y_1|Y_3)$ instead of $P(Y_1)$ and $P(Y_3|Y_1)$. The parameters describing $P(Y_3)$ and $P(Y_1|Y_3)$ would be peculiar to m'. However, this is due to the choice of representation rather than search operation. Hence those parameters are not genuinely new parameters. In implementation, one needs to coordinate the representations of the current model and the candidate models so as to avoid such fake new parameters.

3.2.2. Restricted likelihood

Suppose we have the MLE θ^* of the parameters of the current model m. We write θ^* as (θ_1^*, θ_2^*) where θ_1^* and θ_2^* are the MLE of common parameters and new parameters respectively. For a given value of θ_2', $(m', \theta_1^*, \theta_2')$ is a fully specified Bayesian network. In this network, we can compute

$$P\big(\mathcal{D}|m', \theta_1^*, \theta_2'\big) = \prod_{\mathbf{d}\in\mathcal{D}} P\big(\mathbf{d}|m', \theta_1^*, \theta_2'\big).$$

As a function of θ_2', this is referred to as the *restricted likelihood function* of m'. The *maximum restricted log likelihood*, or simply the *maximum RL*, of the candidate model m' is defined to be

$$\max_{\theta_2'} \log P\big(\mathcal{D}|m', \theta_1^*, \theta_2'\big).$$

Replacing the likelihood term in the BIC score of m' with its maximum RL, we get the following approximate score:

$$BIC_{RL}\big(m'|\mathcal{D}\big) = \max_{\theta_2'} \log P\big(\mathcal{D}|m', \theta_1^*, \theta_2'\big) - \frac{d(m')}{2}\log N. \tag{3}$$

We propose that `pickModel` uses the BIC_{RL} score for model selection instead of the BIC score. It should be noted that the idea of optimizing only some parameters of a model while freezing others is used in, among others, phylogenetic tree reconstruction [19] and learning of continuous Bayesian networks [30].

Next we describe an efficient method for approximately calculating the BIC_{RL} score. The method is called *local EM*.

3.2.3. Local EM

Local EM works in the same way as EM except with the value of θ_1' fixed at θ_1^*. It starts with an initial value $\delta_2^{(0)}$ for θ_2' and iterates. After $t-1$ iterations, it obtains $\delta_2^{(t-1)}$. At iteration t, it completes the data $\mathcal{D}$ using the Bayesian network

$(m', \theta_1^*, \delta_2^{(t-1)})$, calculates some sufficient statistics, and therefrom obtains $\delta_2^{(t)}$. Suppose the parameters θ_2' of m' describe distributions $P(Z_j|W_j)$ $(j = 1, \ldots, \rho)$.[1] The distributions $P(Z_j|W_j, \delta_2^{(t)})$ that make up $\delta_2^{(t)}$ can be obtained in two steps:

- E-Step: For each data case $\mathbf{d} \in \mathcal{D}$, make inference in the Bayesian network $(m', \theta_1^*, \delta_2^{(t-1)})$ to compute

$$P\big(Z_j, W_j|\mathbf{d}, m', \theta_1^*, \delta_2^{(t-1)}\big) \quad (j = 1, \ldots, \rho).$$

- M-Step: Obtain

$$P\big(Z_j|W_j, \delta_2^{(t)}\big) = f(Z_j, W_j)/\sum_{Z_j} f(Z_j, W_j) \quad (j = 1, \ldots, \rho)$$

where the *sufficient statistic*

$$f(Z_j, W_j) = \sum_{\mathbf{d}\in\mathcal{D}} P\big(Z_j, W_j|\mathbf{d}, m', \theta_1^*, \delta_2^{(t-1)}\big).$$

Local EM converges. That is, the series of log likelihood $\{\log P(\mathcal{D}|m', \theta_1^*, \delta_2^{(t)}) \mid t = 0, 1, \ldots\}$ increases monotonically with t and it is upper-bounded by 0.

Unlike local EM, standard EM optimizes all parameters. To avoid potential confusions, we call it *full EM*. The M-step of a local EM is computationally much cheaper than that of a full EM because a local EM updates fewer parameters. For the candidate model shown in Fig. 4, we need to update only the parameters that describe $P(Y_4|Y_3)$, $P(X_6|Y_4)$ and $P(X_7|Y_4)$. Besides reduction in computation, this fact also implies that a local EM takes fewer steps to converge than a full EM.

3.2.4. Avoiding local maxima

Like full EM, local EM might get stuck at local maxima. To avoid the local maxima, we adopt the scheme proposed by Chickering and Heckerman [8] and call it the *pyramid scheme*. The idea is to randomly generate a number μ of initial values for the new parameters θ_2', resulting in μ initial models. One local EM iteration is run on all the models and afterwards the bottom $\mu/2$ models with the lowest log likelihood are discarded. Then two local EM iterations are run on the remaining models and afterwards the bottom $\mu/4$ models are discarded. Then four local EM iterations are run on the remaining models, and so on. The process continues until there is only one model. After that, some more local EM iterations are run on the remaining model, until the total number of iterations reaches a predetermined number ν. Therefore there are two algorithmic parameters μ and ν.

Suppose m is the current model and θ^* is the MLE of model parameters. m' is a candidate model obtained from m. Use `localEM`$(m, \theta^*, m', \mu, \nu)$ to denote the procedure described in the previous paragraph. The output is an estimate of the new parameters θ_2'. Denote it by $\bar{\theta}_2$. The `pickModel` subroutine evaluates m' using the following quantity:

$$BIC\big(m', \theta_1^*, \bar{\theta}_2|\mathcal{D}\big) = \log P\big(\mathcal{D}|m', \theta_1^*, \bar{\theta}_2\big) - \frac{d(m')}{2}\log N. \tag{4}$$

Note that the BIC score given here is for a model m' and a set of parameter values θ_1^* and $\bar{\theta}_2$ for the model. In contrast, the BIC score given by Eq. (1) is for a model only.

3.2.5. Two stage model evaluation

Local EM is faster than full EM. To achieve further speedup, we propose to divide model evaluation into two stages, a *screening stage* and an *evaluation stage*. In the screening stage, we screen out most of the candidate models by running local EM at a low setting, while in the evaluation stage we evaluate the remaining models by running local EM at a high setting.

In local EM, the parameter μ controls the number of initial points and the parameter ν controls the number of iterations. For the screening stage, we fix the first parameter at 1 and we allow only the second parameter to vary. To distinguish it from the corresponding parameter at the evaluation stage, we denote it by ν_s.

Because local EM starting from only one initial point at the screening stage, there is no effort to avoid local maxima at all. We argue that this does not cause serious problems because there is an implicit local-maximum-avoidance mechanism built in. A particular application of a search operator is called a *search operation*. It corresponds to one candidate model. So, evaluation of candidate models can also be viewed as evaluation of search operations. Suppose local EM picks a poor initial point at one step when evaluating an operation and consequently the operation is screened out. Chances are that the same operation is also applicable at the next few steps. In that case local EM would be called to evaluate the operation again and again, each time from a different starting point. So in the end local EM is run from multiple starting points to evaluate the operation. If the operation is a good one, there is high probability for it to be picked at one of those steps.

[1] When Z_j is the root, W_j is to be regarded as a 'vacuous' variable and $P(Z_j|W_j)$ is simply $P(Z_j)$.

```
pickModel(L, m, θ*):
For each m' ∈ L,
  run localEM(m, θ*, m', 1, ν_s) to estimate the new parameters of m'.
Prune from L all models except the k models
  with highest BIC scores as given by (4).
For each m' ∈ L,
  run localEM(m, θ*, m', μ, ν) to estimate parameters of m'.
Let m_1 be the model in L with the highest BIC score as given by (4).
θ*_1 ← EM(m_1, D).
Return (m_1, θ*_1).
```

Fig. 5. The `pickModel` subroutine. There are four algorithmic parameters. The parameters ν_s and k control the screening stage, while μ and ν control the evaluation stage.

3.2.6. The `pickModel` Subroutine

Finally, the pseudo code for `pickModel` is given in Fig. 5. The inputs consist of the current model m, the MLE of its parameters θ^*, and a list of candidate models L. It first runs local EM at a low setting to screen out all but k of the candidate models. Then, it runs local EM at a higher setting to evaluate the remaining k models and picks the best one. Full EM is run on the model selected model, and the model together with the MLE of its parameters are returned. The optimal parameter values are needed when comparing the picked candidate model with m and in subsequent calls to `pickModel`. EAST calls `pickModel` (or `pickModel-IR`) once at each step of search and hence runs full EM only once at each step of search.

3.3. Operation granularity

At the expansion stage, EAST does not select models using the subroutine `pickModel`. Rather it uses another subroutine called `pickModel-IR`. This is to deal with the issue of operation granularity.

Operation granularity refers to the phenomenon where some operations might increase the complexity of the current model much more than other operations. As an example, consider the situation where there are 100 binary manifest variables. Suppose the search starts with the LCM with one binary latent node Y. Applying the SI operator to the model would introduce 101 additional model parameters, while applying the NI operator to the model would increase the number of model parameters by only 2. The latter operation is clearly of much finer-grain than the former.

Operation granularity might lead to local maxima. The reason is that, at the early stage of search, SI operations are usually of larger grain than NI operations and often have higher BIC scores. So, SI operations tend to be applied early, which sometimes leads to *fat latent variables*, i.e., latent variables with excessive numbers of states. Fat latent variables tend to attract excessive numbers of neighbors. This makes it difficult for EAST to thin fat variables despite of the SD operator. Local maxima are consequently produced. Please refer to [5] for the details.

One might suggest that we deal with fat latent variables by introducing an additional search operator that simultaneously reduces the number of states and the number of neighbors of a latent variable. However, this would complicate algorithm design and would increase the complexity of the search process. We adopt a simple and effective strategy called the cost-effectiveness principle.

Let m be the current model and m' be a candidate model. Define the *improvement ratio of* m' *over* m given data $\mathcal{D}$ to be

$$IR(m', m|\mathcal{D}) = \frac{BIC(m'|\mathcal{D}) - BIC(m|\mathcal{D})}{d(m') - d(m)}. \tag{5}$$

It is the increase in model score per unit increase in model complexity. The *cost-effectiveness* principle stipulates that one chooses, among a list of candidate models, the one with the highest improvement ratio.

The principle is applied only on candidate models generated by the SI and NI operators. The other operators do not or do not necessarily increase model complexity. Hence the term $d(m') - d(m)$ is or might be negative.

There is an interesting link between the cost-effectiveness principle and likelihood ratio test (LRT). Let θ^* and θ'^* be the MLE estimates of the parameters of m and m' respectively. Then we have,

$$IR(m', m|\mathcal{D}) = \frac{\log P(\mathcal{D}|m', \theta'^*) - \log P(\mathcal{D}|m, \theta^*)}{d(m') - d(m)} + \frac{\log N}{2}.$$

The second term is constant with respect to m'. In the first term, the nominator is the difference in log-likelihoods used in the LRT with m as the null model and m' as the alternative model. The denominator is the degrees of freedom for the test. So, loosely speaking, the cost-effectiveness principle picks the candidate model that gives the strongest evidence to reject the null model in LRT.[2]

Like `pickModel`, `pickModel-IR` does not run full EM to optimize the parameters of the candidate models. Instead, it inherits the values of the old parameters from the current model and runs local EM to optimize only the new parameters. Let m be the current model and m' be a candidate model obtained from m. Suppose the MLE (θ_1^*, θ_2^*) of the parameters m have

[2] This link was pointed out by an anonymous reviewer.

pickModel-IR(L, m):
For each $m' \in L$,
  run localEM($m, \theta^*, m', 1, \nu_s$) to estimate the new parameters of m'.
Prune from L all models except the k models
  with highest IR scores as given by (6).
For each $m' \in L$,
  run localEM($m, \theta^*, m', \mu, \nu$) to estimate parameters of m'.
Let m_1 be the model in L with the highest IR score as given by (6).
$\theta_1^* \leftarrow$ EM($m_1, \mathcal{D}$).
Return (m_1, θ_1^*).

Fig. 6. The pickModel-IR subroutine.

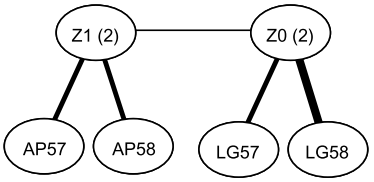


Fig. 7. The structure of the LTM obtained for the Coleman data. In the names of the manifest variables, LG is a shorthand for 'in leading group' and AP is a shorthand for 'against principles'. The numbers of states for the latent variables are shown in parentheses. Edge widths represent mutual information.

been computed. Let $\bar{\theta}_2$ be the estimate of the new parameters of m' obtained by local EM. The subroutine pickModel-IR evaluates the candidate model m' using the following IR score:

$$IR\big(m', m, \theta_1^*, \theta_2^*, \bar{\theta}_2|\mathcal{D}\big) = \frac{BIC(m', \theta_1^*, \bar{\theta}_2|\mathcal{D}) - BIC(m, \theta_1^*, \theta_2^*|\mathcal{D})}{d(m') - d(m)}. \tag{6}$$

The pseudo code of pickModel-IR is given in Fig. 6. It is the same as pickModel except IR scores, rather than BIC scores, are used to evaluate candidate models.

4. Multidimensional clustering with LTMs

In this paper, we are interested in LTMs from the perspective of cluster analysis. The idea is to analyze data using LTMs to obtain potentially multiple latent variables. Each latent variable represents one way to partition the data. Hence we get multiple partitions. In the previous section, we have developed an algorithm for performing latent tree analysis (LTA). What remains is to answer these important questions:

1. Can LTA find meaningful partitions?
2. Can LTA find more and better partitions than alternative methods?

In this section, we answer the first question by showing the results that LTA obtained on two real-world data sets. In the next two sections, we will compare LTA with alternative methods. For convenience, we do not distinguish between LTA and EAST, the algorithm used to perform LTA.

4.1. Two data sets

The first data set is known as the Coleman data. It is taken from the latent class analysis (LCA) literature [17,20]. The data set summarizes responses of 3398 schoolboys, each of whom was asked to respond to the following question and statement at two different points in time: (1) Are you a member of the leading crowd? (2) If a fellow wants to be a part of the leading crowd around here, he sometimes has to go against his principles. The survey was carried out first in October 1957 and for a second time in May 1958. So there are 4 binary manifest variables. We name them as LG57, AP57, LG58, and AP58 respectively, where LG is a shorthand for 'in leading group' and AP is a shorthand for 'against principles'.

The second data set is known as the ICAC data. ICAC stands for the Independent Commission Against Corruption, the anti-corruption agency of Hong Kong. It conducts annual telephone survey to (a) obtain an updated reading of public perception of and attitude towards the ICAC and the problem of corruption; and (b) identify any changes in public perception and attitude over time. The data set used in this paper is from the 2004 survey. After preprocessing, the data set consists of 31 manifest variables and 1200 records.

We analyzed the two data sets using the EAST algorithm. The best LTM obtained for the Coleman data is referred to as the *Coleman LTM*. The structure of the model is shown in Fig. 7. Its BIC score is -8539. The best model obtained for the ICAC data is referred to as the *ICAC LTM*. Its BIC score is $-26{,}097$. The structure is shown in Fig. 8. In both figures, the variables at the bottom are the manifest variables. They are from the data. The variables at the internal nodes, i.e., the Z and Y variables, are latent variables introduced during data analysis.

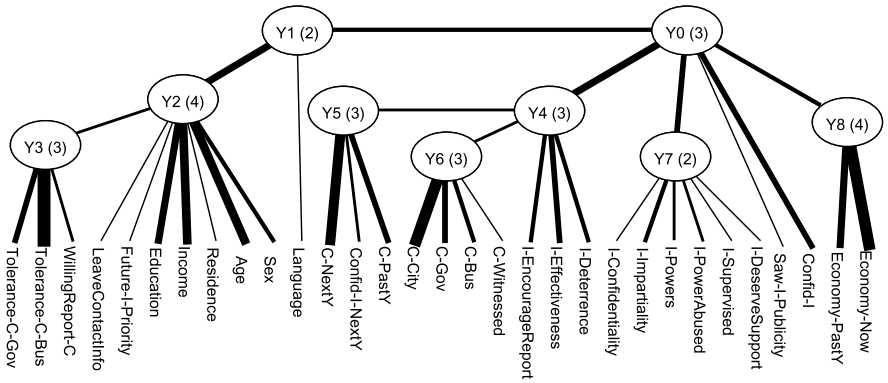


Fig. 8. The structure of the LTM obtained for the ICAC data. Abbreviations: C – Corruption, I – ICAC, Y – Year, Gov – Government, Bus – Business Sector. Meanings of manifest variables: Tolerance-C-Gov means 'tolerance towards corruption in the government'; C-City means 'level of corruption in the city'; C-NextY means 'change in the level of corruption next year'; I-Effectiveness means 'effectiveness of ICAC's work'; I-Powers means 'ICAC powers'; Confid-I means 'confidence in ICAC'; etc.

Table 1
The CCPDs for the states of the latent variables in the Coleman LTM and the probabilities of those states.

	$P(Z_0 = s_0) = .4$		$P(Z_0 = s_1) = .6$		$P(Z_1 = s_0) = .51$		$P(Z_1 = s_1) = .49$	
	y	n	y	n	y	n	y	n
AP57	.63	.37	.47	.53	.80	.20	.26	.74
AP58	.66	.36	.51	.49	.83	.17	.30	.70
LG57	.75	.25	.11	.89	.53	.47	.28	.72
LG58	.91	.09	.08	.92	.46	.54	.29	.71

4.2. Model interpretation

EAST has obtained 2 latent variables Z_1 and Z_2 for the Coleman data, and 9 latent variables Y_0–Y_8 for the ICAC data. Each latent variable represents one way to partition data. The next step is to determine the meanings of the latent variables and the partitions that they represent. In other words, we need to carry out *model interpretation*. Model interpretation is a task for domain experts. As tool developers, we need to determine what information to show to the experts so that they can appreciate model contents accurately.

4.2.1. Basics of model interpretation

A partition consists of a number of latent classes. To understand what the partition is about, one naturally would want to examine how the latent classes differ from each other. A latent class can be characterized by its class conditional probability distributions (CCPDs), i.e., the distributions of the manifest variables in the class. So, one can determine the meaning of a partition by comparing the CCPDs of different classes.

Consider the latent variable Z_0 in the Coleman LTM. It has two states, which we denote as s_0 and s_1. The CCPDs of the two latent classes $Z_0 = s_0$ and $Z_0 = s_1$ are given in Table 1 (first half). They differ mainly on the distributions of LG57 and LG58. A boy in the class $Z_0 = s_0$ has high probability of regarding himself as being in the leading group, while a boy in the class $Z_0 = s_1$ has low probability of thinking the same. The differences on AP57 and AP58 are less pronounced. To delineate those characteristics, we interpret $Z_0 = s_0$ as a class of schoolboys who incline to believe themselves to be the leading crowd, while $Z_0 = s_1$ as a class of schoolboys who incline to believe the opposite.

4.2.2. Interpretation of complex models

The number of CCPDs can be very large in complex models. Consider the latent variable Y_2 in the ICAC LTM. It has 4 states and there are 33 manifest variables. So there are totally 132 CCPDs to examine in order to make sense of the partition. This would be overwhelming for any domain expert.

Inspecting Information Curves: The key for solving the problem lies in the following observation. To grasp the meaning of a partition, it is not necessary to examine all the differences among the classes in the partition. It suffices to first ask on which manifest variables the classes differ significantly, and then examine how the classes differ on those manifest variables.

Let Y be a latent variable and X be a manifest variable. The *mutual information* $I(Y; X)$ [9] between the two variables can be used to measure how much the classes in the Y partition differ on X. It is given by:

$$I(Y; X) = \sum_{X,Y} P(X, Y) \log \frac{P(X, Y)}{P(X)P(Y)},$$

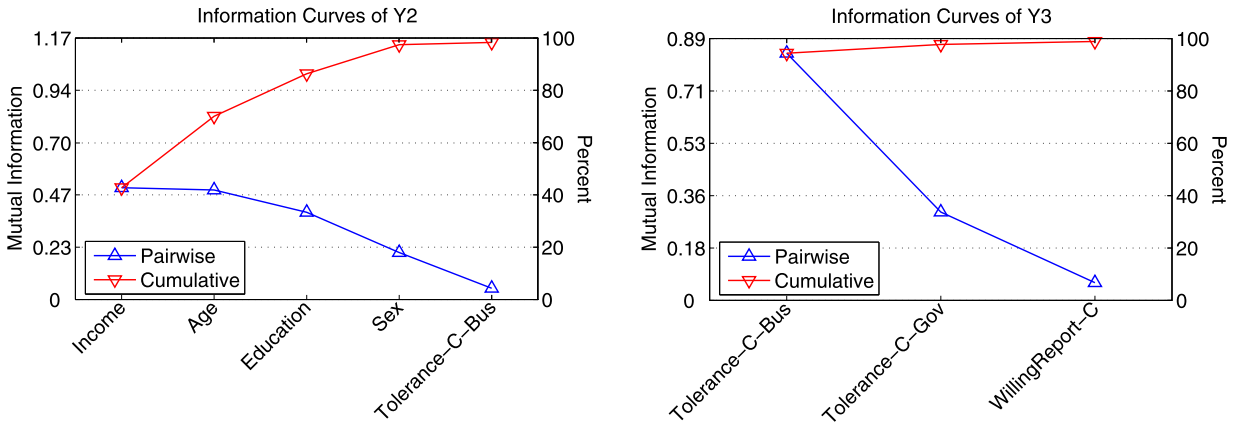


Fig. 9. The information curves of the latent variables Y_2 and Y_3 from the ICAC LTM.

where the summation is taken over all possible states of X and Y. The larger the mutual information, the more the classes in the Y partition differ on X.

Suppose that we have computed the mutual information between Y and each of the manifest variables, and we have sorted the manifest variables in decreasing order of the mutual information. Let $X_1, X_2, \ldots, X_n$ be the manifest variables in the sorted order. We depict the MI values $I(Y; X_1)$, $I(Y; X_2)$, $\ldots$, $I(Y; X_n)$ in a coordinate system and connect them using lines. This results in the *pairwise information curve* of Y.

The pairwise information curve of Y_2 is shown in Fig. 9. It shows that the classes in the Y_2 partition differ the most on the manifest variables 'Income', 'Age', 'Education' and 'Sex'. The meaning of the Y_2 partition is now clear. It is a partition based primarily on those attributes.

The *cumulative information curve* of a latent variable Y depicts $I(Y; X_1 - X_i)$ $(i = 2, 3, \ldots, n)$, the mutual information between Y and the first i manifest variables.[3] The term $I(Y; X_1 - X_i)$ increases monotonically with i. It reaches the maximum when $i = n$. The ratio $I(Y; X_1 - X_i)/I(Y; X_1 - X_n)$ is the *cumulative information coverage* of the first i manifest variables.

If the information coverage $I(Y; X_1 - X_i)/I(Y; X_1 - X_n) = 1$, then Y is conditionally independent of $X_{i+1}, \ldots, X_n$ given the first i manifest variables. In such a case, we can interpret Y based only on the first i manifest variables, while ignoring all the others. In practice, it is still reasonable to do so when the information coverage is close to 1.

The cumulative information curve of Y_2 is also shown in Fig. 9. We see that the information coverage of the first four attributes has reached 98%. Those four attributes give demographic information. Hence we can interpret Y_2 as a partition based on demographic information.

Comparing CCPDs: After inspecting the information curves of a latent variable Y, one gets an idea about what the Y partition is about. The next step is to examine the CCPDs of the latent classes in the partition and determine what they mean. We illustrate this step with an example.

Y_2 has 4 states s_0, s_1, s_2, and s_3. Their CCPDs are shown in Table 2. In comparison with the other classes, the class $Y_2 = s_0$ has two characteristics: First, it consists of only youngsters aged between 15 and 24. Second, the average income is significantly lower than those of the other classes. So $Y_2 = s_0$ represents a class of low income youngsters. The class $Y_2 = s_1$ is special in that it consists of only women. 41% of them do not have income. Those are probably housewives. The average income for the rest is low. Hence $Y_2 = s_1$ represents a class of women with no/low income. Between the remaining two classes, the class $Y_2 = s_2$ has, on average, higher education and higher income level than $Y_2 = s_3$. Hence $Y_2 = s_2$ represents a class of people with good education and good income, while $Y_2 = s_3$ represents a class of people with poor education and average income.

We now know how to do model interpretation. Next we set out to examine the results LTA obtained on the Coleman and ICAC data.

4.3. Clustering results obtained by LTA on the Coleman data

There are two latent variables Z_0 and Z_1 in the Coleman LTM model. So LTA has partitioned the data along two different dimensions. In this subsection we examine those partitions and show that they are meaningful.

[3] This is difficult to compute exactly when i is large. In such cases, we resort to the Monte Carlo method [38] and compute the MI values based on 10,000 simulated samples.

Table 2
The CCPDs for the states of Y_2 and the probabilities of those states.

	$P(Y_2 = s_0) = .18$							$P(Y_2 = s_1) = .24$						
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	.03	.76	.08	.09	.04	0	0	.41	.29	.24	.04	.02	0	0
Age	1	0	0	0	0			.01	.08	.42	.36	.13		
Education	0	0	.08	.47	.19	.10	.16	.05	.29	.35	.25	.05	0	.01
Sex	.48	.52						0	1					
	$P(Y_2 = s_2) = .33$							$P(Y_2 = s_3) = .25$						
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	.03	0	.04	.10	.39	.29	.15	.08	.10	.15	.25	.34	.08	0
Age	.04	.37	.38	.17	.04			.01	.09	.26	.38	.26		
Education	0	0	0	.40	.10	.10	.40	.02	.25	.44	.23	.05	.01	0
Sex	.57	.43						.80	.20					

	States of the manifest variables						
	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Income	none	–4k	4–7k	7–10k	10–20k	20–40k	40k–
Age	15–24	25–34	35–44	45–54	55–		
Education[a]	none	primary	f1–3	f4–5	f6–7	diploma	degree
Sex	m	f					

[a] Hong Kong adopts the British education system. High school consists of 7 years, from Form 1 to Form 7. 'f1–3' means Form 1–3, which corresponds to junior high school in the North American system. 'f4–5' means Form 4–5 and 'f6–7' means Form 6–7.

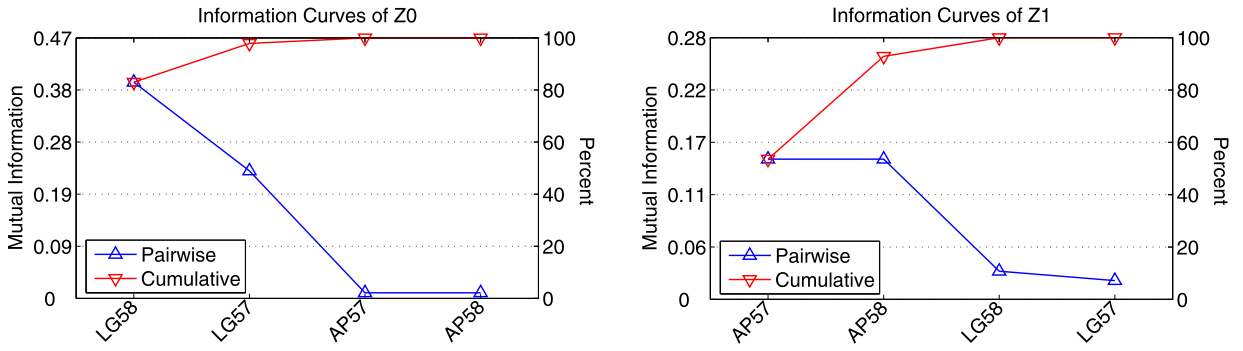


Fig. 10. The information curves of the latent variables from the Coleman LTM.

4.3.1. Z_0 and its states

The information curves of Z_0 are shown in Fig. 10. We see that the mutual information values between Z_0 and AP57 and AP58 are almost 0, while those between Z_0 and LG57 and LG58 are relatively much larger. Moreover the cumulative information coverage of LG57 and LG58 has reached 98%. Those mean that the two latent classes $Z_0 = s_0$ and $Z_0 = s_1$ differ almost totally on LG57 and LG58. Hence we can interpret Z_0 as a partition of the schoolboy population based on their views on their membership in the leading group.

We have already examined the CCPDs of the two classes $Z_0 = s_0$ and $Z_0 = s_1$ in Section 4.2. The class $Z_0 = s_0$ was interpreted as a class of schoolboys who incline to believe themselves to be the leading crowd, while $Z_0 = s_1$ as a class of schoolboys who incline to believe the opposite.

4.3.2. Z_1 and its states

We next turn to the latent variable Z_1. It has two states s_0 and s_1. The information curves of Z_1 are shown in Fig. 10. We see that the mutual information values between Z_1 and AP57 and AP58 are much higher than those between Z_1 and LG57 and LG58. Moreover the cumulative information coverage of AP57 and AP58 has reached 93%. Those mean that LG57 and LG58 exhibit little additional differences between $Z_1 = s_0$ and $Z_1 = s_1$ beyond what are already exhibited by AP57 and AP58. Hence we can interpret Z_1 as a partition of the schoolboy population primarily based on their views on the implications of membership in the leading group.

The distributions of AP57 and AP58 in the two classes $Z_1 = s_0$ and $Z_1 = s_1$ are shown in the second half of Table 1 (Section 4.2.1). We see that a boy in the class $Z_1 = s_0$ has high probability of thinking that being in the leading crowd implies compromising one's principles, while a boy in the class $Z_1 = s_1$ has low probability of thinking the same. To delineate those characteristics, we interpret $Z_1 = s_0$ as a class of schoolboys who incline to believe that membership in the leading crowd implies compromising one's principles, while $Z_1 = s_1$ as a class of schoolboys who incline to believe the opposite.

Table 3
The CCPDs for the states of Y_3 and the probabilities of those states. The states of two manifest variables are: s_0 (totally intolerable), s_1 (intolerable), s_2 (tolerable), and s_3 (totally tolerable).

	$P(Y_3 = s_0) = .57$				$P(Y_3 = s_1) = .27$				$P(Y_3 = s_2) = .15$			
	s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3
Tolerance-C-Bus	1	0	0	0	.02	.89	.09	0	0	.05	.85	.10
Tolerance-C-Gov	.97	.03	0	0	.54	.46	0	0	.29	.19	.48	.04

4.3.3. Relationship between Z_0 and Z_1

An interesting relationship between the two latent variables Z_0 and Z_1 has also been revealed. Consider the conditional probability distribution $P(Z_1|Z_0)$:

	$Z_1 = s_0$	$Z_1 = s_1$
$Z_0 = s_0$	.68	.32
$Z_0 = s_1$	.39	.61

We see that a boy's view on his membership in the leading crowd (Z_0) greatly influences his view on the implications of membership in the leading crowd (Z_1). The probability of believing that membership in the leading crowd implies compromising principles ($Z_1 = s_0$) is 68% for a boy believing himself to be in the leading crowd ($Z_0 = s_0$), while it is only 39% for a school boy believing the opposite ($Z_0 = s_1$). This is intuitively appealing.

4.4. Clustering results obtained by LTA on the ICAC data

There are 9 latent variables in the ICAC LTM. In the following we look at Y_2, Y_3, Y_4, Y_5, Y_6 and Y_7. We show that those latent variables represent meaningful partitions of the data and that interesting relationships among them have be revealed. Although several latent variables are skipped to save space, the discussion is still quite long. The reader can jump to the next section at any time.

4.4.1. Y_3 and its states

We have examined Y_2 and its states in Section 4.2.2. We now turn to Y_3. The information curves of Y_3 are shown on the right of Fig. 9 (Section 4.2.1). The first two variables on the curves are about two aspects of people's tolerance towards corruption. Their cumulative information coverage has reached 99%. So we interpret Y_3 as representing a partition of the interviewee population based on their tolerance towards corruption.

Y_3 has three states s_0, s_1, and s_2. Their CCPDs are given in Table 3. We see that almost all the people in the class $Y_3 = s_0$ find corruption in the government and in the business sector totally intolerable. All the people in the class $Y_3 = s_1$ find corruption in the government totally intolerable or intolerable, while 9% of them find corruption in the business sector tolerable. Most people ($85\% + 10\%$) in the class $Y_3 = s_2$ find corruption in the business sector tolerable, while they split roughly in the middle when it comes to corruption in the government. So the three latent classes $Y_3 = s_0$, $Y_3 = s_1$, and $Y_3 = s_2$ can be respectively interpreted as classes of people who find corruption totally intolerable, intolerable, and tolerable. The classes consist of 57%, 27%, and 15% of the population respectively.

The clusters of Y_3 present an interesting picture about people's tolerance towards corruption. We see that among all the people who find corruption intolerable ($Y_3 = s_1$), 54% find corruption in the government totally intolerable while that number drops to merely 2% when it comes to corruption in the business sector. Among all the people who find corruption tolerable ($Y_3 = s_2$), 48% ($29\% + 19\%$) find corruption in the government totally intolerable or intolerable while that number drops to merely 5% when it comes to corruption in the business sector. However, people who find corruption totally intolerable ($Y_3 = s_0$) have more or less the same attitude towards both corruption in the government and corruption in the business sector. Those suggest that people who are tough on corruption are equally tough towards corruption in the government and corruption in the business sector, and that people who are lenient towards corruption are more lenient towards corruption in the business sector than corruption in the government.

4.4.2. Relationship between Y_2 and Y_3

LTA has obtained some interesting findings about how demographics influence people's attitude towards corruption. Consider the conditional probability distribution $P(Y_3|Y_2)$:

	$Y_3 = s_0$	$Y_3 = s_1$	$Y_3 = s_2$
$Y_2 = s_0$	.43	.44	.13
$Y_2 = s_1$	.69	.15	.16
$Y_2 = s_2$	.60	.36	.04
$Y_2 = s_3$	.52	.16	.32

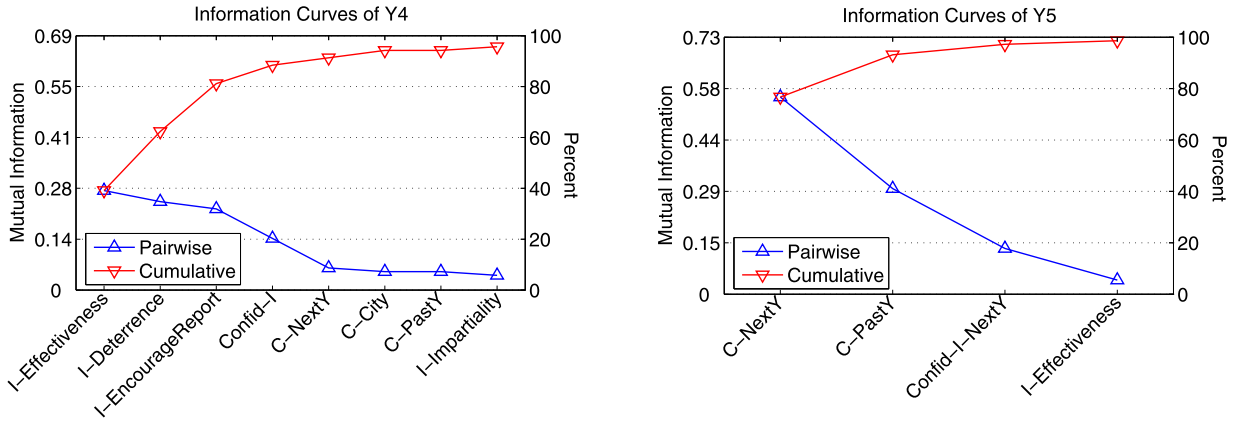


Fig. 11. The information curves of the latent variables Y_4 and Y_5.

Table 4
The CCPDs for the latent states of Y_4 and the probabilities of those classes. The states of I-Effectiveness are: s_0 (very effective), s_1 (effective), s_2 (average), s_3 (ineffective), and s_4 (very ineffective). The states of I-Deterrence and I-EncourageReport are: s_0 (very sufficient), s_1 (sufficient), s_2 (average), s_3 (insufficient), and s_4 (very insufficient).

	$P(Y_4 = s_0) = .21$					$P(Y_4 = s_1) = .53$				
	s_0	s_1	s_2	s_3	s_4	s_0	s_1	s_2	s_3	s_4
I-Effectiveness	.37	.63	0	0	0	.03	.76	.21	0	0
I-Deterrence	.17	.61	.15	.07	0	.01	.60	.37	.02	0
I-EncourageReport	.23	.70	.02	.05	0	0	.49	.43	.08	0

$P(Y_4 = s_2) = .27$				
s_0	s_1	s_2	s_3	s_4
0	.25	.68	.06	.01
0	.07	.58	.31	.04
.03	.16	.48	.29	.04

We see that $Y_2 = s_2$ (people with good education and good income) is the class with the least tolerance towards corruption: 96% (60% + 36%) of the people in the class find corruption totally intolerable or intolerable ($Y_3 = s_0$ or s_1). On the other hand, $Y_2 = s_3$ (people with poor education and average income) is the class with the most tolerance towards corruption: 32% of the people in the class find corruption tolerable ($Y_3 = s_2$). The other two classes, $Y_2 = s_0$ (youngsters with low income) and $Y_2 = s_1$ (women with no/low income) are in the middle, with the latter being slightly more tolerant.

So, the results indicate the people with good education and good income are intolerant towards corruption, while people with poor education and average income sometimes find corruption acceptable. This is intuitively appealing and can be a hypothesis for social scientist to verify further.

4.4.3. Y_4 and its states

The information curves of Y_4 are shown in Fig. 11. We see that the manifest variables that are most informative of Y_4 are I-Effectiveness, I-Deterrence, and I-EncourageReport in that order. These three variables reflect different aspects of people's view on ICAC's performance. Their cumulative information coverage is around 81%. So Y_4 can be viewed as a partition primarily based on people's view on ICAC's performance.

Y_4 has three states s_0, s_1, and s_2. Their CCPDs are shown in Table 4. We see that, first, 100% (37% + 63%) of the people in $Y_4 = s_0$ think that ICAC's anti-corruption work is very effective or effective; 21% of the people in $Y_4 = s_1$ disagree; and that number increases to 75% (68% + 6% + 1%) for $Y_4 = s_2$, where 7% of the people think that ICAC's anti-corruption is ineffective or very ineffectively. Second, 78% (17% + 61%) of the people in $Y_4 = s_0$ think that the deterrence that ICAC has against corruption is very sufficient or sufficient; that number drops to 61% (1% + 60%) in $Y_4 = s_1$, and further drops to 7% in $Y_4 = s_2$, where 35% (31% + 4%) of the people think that the deterrence is insufficient or very insufficient. Third, 93% (23% + 70%) of the people in $Y_4 = s_0$ think that ICAC's encouragement for reporting corruption is very sufficient or sufficient; that number drops to 49% in $Y_4 = s_1$, and further drops to 19% (3% + 16%) in $Y_4 = s_2$, where 33% (29% + 4%) of the people think that the encouragement is insufficient or very insufficient. So the three latent classes $Y_4 = s_0$, $Y_4 = s_1$ and $Y_4 = s_2$ can be respectively interpreted as classes of people who find ICAC's performance very good, good or average.

Table 5
The CCPDs for the states of Y_5 and the probabilities of those states. The states of C-NextY are: s_0 (increase), s_1 (decrease), and s_2 (same). The states C-PastY are: s_0 (increased), s_1 (decreased), and s_2 (same).

	$P(Y_5 = s_0) = .19$			$P(Y_5 = s_1) = .24$			$P(Y_5 = s_2) = .57$		
	s_0	s_1	s_2	s_0	s_1	s_2	s_0	s_1	s_2
C-NextY	.89	0	.11	.05	.81	.14	.09	.01	.90
C-PastY	.83	0	.17	.06	.48	.46	.10	.07	.83

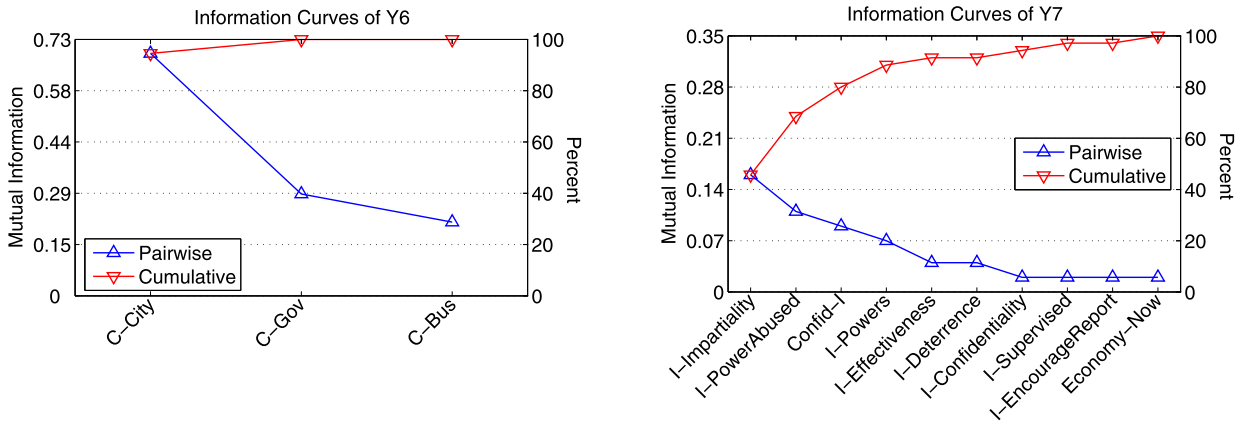


Fig. 12. The information curves of the latent variables Y_6 and Y_7.

4.4.4. Y_5 and its states

The information curves of Y_5 are shown on the right of Fig. 11. The first two variables on the curves, C-NextY and C-PastY, are about different aspects of people's view on the change in corruption scene. We choose to interpret Y_5 and its states based on those two manifest variables. The information coverage is 93%.

Y_5 has three states s_0, s_1, and s_2. Their CCPDs are given in Table 5. The differences among the three latent classes are clear. All the people in the class $Y_5 = s_0$ think that the level of corruption has increased (83%) or remained the same (17%) in the past year, and the level will increase (89%) or remain the same (11%) next year. On the other hand, most people in the class $Y_5 = s_1$ think that the level of corruption has decreased (48%) or remained the same (46%) in the past year, and the level will decrease (81%) or remain the same (14%) next year. The majority of people in $Y_5 = s_2$ think that the level of corruption has remained the same in the past year (83%) and will remain the same in the next year (90%). So we interpret Y_5 as a partition of the population based on their views on the change in corruption scene. The three states $Y_5 = s_0$, $Y_5 = s_1$, and $Y_5 = s_2$ are respectively interpreted as classes of people who believe the change in corruption scene is negative, positive, and flat. The classes consist of 19%, 24%, and 57% of the population, respectively.

4.4.5. Relationship between Y_4 and Y_5

It is interesting to examine the conditional probability distribution $P(Y_5|Y_4)$:

	$Y_5 = s_0$	$Y_5 = s_1$	$Y_5 = s_2$
$Y_4 = s_0$	.09	.42	.49
$Y_4 = s_1$	.09	.24	.67
$Y_4 = s_2$	.47	.09	.44

It shows how people's view on ICAC's performance influences their view on the change in corruption scene. We see that people who think ICAC's performance is very good ($Y_4 = s_0$) have 91% (42% + 49%) chance of believing that the change in corruption is positive ($Y_5 = s_1$) or flat ($Y_5 = s_2$). The same is true for people who think ICAC's performance is good ($Y_4 = s_1$), except that people in this class have lower probability (24%) of believing in a positive change. On the other hand, people who think ICAC's performance is average ($Y_4 = s_2$) have 47% chance of believing that the change in corruption is negative ($Y_5 = s_0$).

4.4.6. Y_6 and its states

The information curves of Y_6 are shown on the left of Fig. 12. The three manifest variables that appear on the curves are about different aspects of corruption scene. We choose to interpret Y_6 and its states based on those manifest variables. The information coverage is 100%.

Y_6 has three states s_0, s_1, and s_2. Their CCPDs are shown in Table 6. We see that 99% (8% + 91%) of the people in the class $Y_6 = s_0$ think that corruption in the city is very common or common, and 90% (10% + 80%) of the people feel the

Table 6
The CCPDs for the states of Y_6 and the probabilities of those states. The states of C-City, C-Gov, and C-Bus are: s_0 (very common), s_1 (common), s_2 (uncommon), and s_3 (very uncommon).

	$P(Y_6 = s_0) = .31$				$P(Y_6 = s_1) = .65$				$P(Y_6 = s_2) = .04$			
	s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3
C-City	.08	.91	.01	0	0	0	.99	.01	0	0	.23	.77
C-Gov	.06	.49	.44	.01	0	.03	.93	.04	0	0	0	1
C-Bus	.10	.80	.10	.01	.01	.28	.70	.01	0	.12	.56	.32

Table 7
The CCPDs for the states of Y_7 and the probabilities of those states. The states of I-Impartiality, I-PowerAbused, I-Confidentiality, and I-Supervised are: s_0 (yes) and s_1 (no). The states of I-Powers are: s_0 (too large), s_1 (too small), and s_2 (appropriate).

	$P(Y_7 = s_0) = .77$			$P(Y_7 = s_1) = .23$		
	s_0	s_1	s_2	s_0	s_1	s_2
I-Impartiality	.97	.03		.45	.55	
I-PowerAbused	.11	.89		.62	.38	
I-Powers	.02	.06	.92	.21	.21	.58
I-Confidentiality	.99	.01		.89	.11	
I-Supervised	.73	.27		.48	.52	

same way about corruption in the business sector; the percentage drops to 55% (6% + 49%) when it comes to corruption in the government, with the other 45% think the opposite. More than 90% of the people in the class $Y_6 = s_1$ think that corruption in the city and the government is uncommon; that percentage drops to 70% when it comes to corruption in the business sector, and 29% (1% + 28%) of the people in the class actually think that corruption in the business sector is very common or common. All the people in the class $Y_6 = s_2$ think that corruption in the government is very uncommon; 77% of the people in the class feel the same way about corruption in the city, while the other 23% is less extreme and think that corruption in the city is uncommon; the class spreads wide when it comes to corruption in the business sector: 32% of the people think that it is very uncommon, 56% think that it is uncommon, while 12% think that it is common. So we interpret Y_6 as a partition of the population based on their view on corruption scene. The three states $Y_6 = s_0$, $Y_6 = s_1$, and $Y_6 = s_2$ are respectively interpreted as classes of people who think corruption is common, uncommon, and very uncommon. The classes consist of 31%, 65%, and 4% of the population, respectively.

It is clear from the above discussions that people are the most positive about the level of corruption in the government and are the most negative about the level of corruption in the business sector.

4.4.7. *Relationship between Y_4 and Y_6*

It is also interesting to examine the conditional probability distribution $P(Y_4|Y_6)$:

	$Y_4 = s_0$	$Y_4 = s_1$	$Y_4 = s_2$
$Y_6 = s_0$	.17	.36	.47
$Y_6 = s_1$	.21	.61	.18
$Y_6 = s_2$	.39	.61	0

The distribution shows how the view on ICAC's performance (Y_4) is related to the view on corruption scene (Y_6). We see that 47% of the people who think corruption is common ($Y_6 = s_0$) do not think highly of ICAC's performance ($Y_4 = s_2$). One the other hand, all the people who think corruption is very uncommon ($Y_6 = s_2$) think that ICAC's performance is good or very good ($Y_4 = s_1$ or s_0).

4.4.8. *Y_7 and its states*

The information curves of Y_7 are shown on the right of Fig. 12. The leading variables on the curve I-Impartiality, I-PowerAbused, I-Powers, I-Confidentiality, and I-Supervised are about different aspects of ICAC's accountability. We choose to interpret Y_7 and its states based on those manifest variables. The information coverage is 82%.

Y_7 has two states s_0 and s_1. Their CCPDs are given in Table 7. We see that almost all the people in the class $Y_7 = s_0$ think that ICAC is impartial in its investigations (97%) and keeps reports received confidential (99%); 73% of the people in the class believe that ICAC is externally supervised; 89% of them think that ICAC has not abused its powers and 92% of them think that ICAC's powers are appropriate. As for $Y_7 = s_1$, 89% of the people in the class also think that ICAC keeps reports confidential. However, the class splits more or less in the middle on the other four questions. So we interpret Y_7 as a partition of the population based on the view on the accountability of ICAC. The states $Y_7 = s_0$ and $Y_7 = s_1$ are interpreted respectively as classes of people who think ICAC is accountable and who think ICAC is more or less accountable. The classes consist of 77% and 23% of the population, respectively.

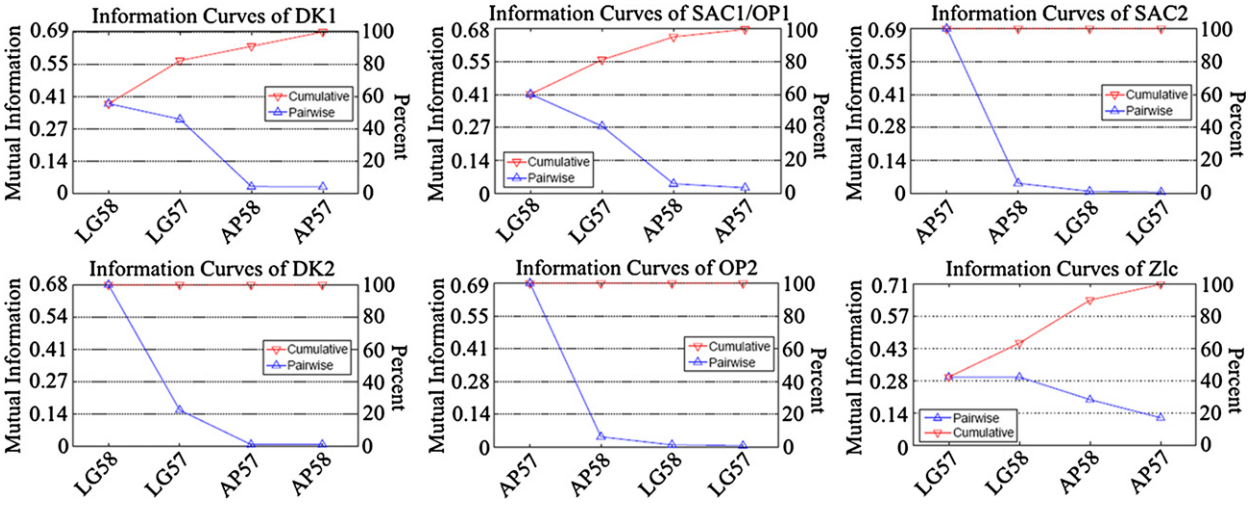


Fig. 13. The information curves of the partitions obtained by DK, OP, SAC and LCA on the Coleman data.

5. Comparisons with related methods on unlabeled data

The problem to cluster data in multiple ways was first raised by [42]. Latent tree analysis (LTA) is one approach to solve the problem. Other methods have recently been proposed. In this section we compare LTA with those other methods on the Coleman and ICAC data. In the next section we will compare them on a collection of labeled data. Latent class analysis (LCA) does not produce multiple clusterings. We include it in the comparisons nonetheless because LTA is a generalization of LCA.

5.1. The other methods

Among the other methods that produce multiple clusterings, we choose the following three to compare with LTA: orthogonal projection (OP) [11], de-correlated K-means (DK) [23] and singular alternative clustering (SAC) [34]. Those are the methods that we were able to obtain implementations for or implement ourselves. Unlike LTA, they are all distance-based. OP and SAC start by obtaining a first partition of data using, e.g., the K-means algorithm. They then try to discover a new partition that is different from the first partition. The key issue is how to ensure the novelty of the new partition. One can repeatedly apply the methods to obtain multiple partitions. DK produces multiple partitions simultaneously [23]. It requires as inputs the number of partitions and the number of clusters in each partition. It tries to optimize the quality of each individual partition while keeping different partitions as dissimilar as possible. The published version of the algorithm can obtain only two different partitions, although generalization to multiple partitions is possible. In our experiments, it was used to produce only two partitions.

5.2. Comparisons on the Coleman data

In view of the results presented in Section 4.3, we have run OP, SAC and DK on the Coleman data to obtain two partitions, each with two clusters. The algorithms have some parameters to set. We used the settings suggested by the authors. The first step of OP and SAC is to run K-means. It is well known that K-means might yield different results in different runs. So, we ran it for 10 times and picked the one with the smallest sum-squared error as the first partition. Then we applied OP and SAC to obtain the second partition. DK was also ran 10 times. The solution that optimizes the objective function of DK was picked as the final solution.

The information curves of the resulting partitions are shown in Fig. 13. We see that the partitions DK1, OP1 and SAC1 are based on the two LG variables. Their information curves are similar to those of the partition Z_0 found by LTA (Section 4.3). The CCPDs are also similar. However, the partitions DK2, OP2 and SAC2 are quite different from the partition Z_1 found by LTA. While Z_1 is based on the two AP variables, DK2 is based on the two LG variables, and OP2 and SAC2 rely almost solely on AP57. According to the discussions in Section 4.3, the partition Z_0 is natural and meaningful. Hence so are DK1, OP1 and SAC1. On the other hand, the partitions DK2, OP2 and SAC2 are not so meaningful.

We also performed LCA on the Coleman data. As a result, we obtained a latent variable Z_{lc} with 4 states, or one partition with 4 clusters. From its information curves shown in Fig. 13, we see that Z_{lc} is closely related to both the two LG variables and the two AP variables. An examination of their CCPDs reveals that the partition Z_{lc} closely resembles the joint partition one can obtain by combining Z_0 and Z_1. So, the partition Z_{lc} is meaningful. However, we would say that LTA has been more revealing than LCA. This is because LTA has explicitly identified two aspects of the data and explicated the relationship between those two aspects. Further effort would be required if one is to uncover such information from Z_{lc}.

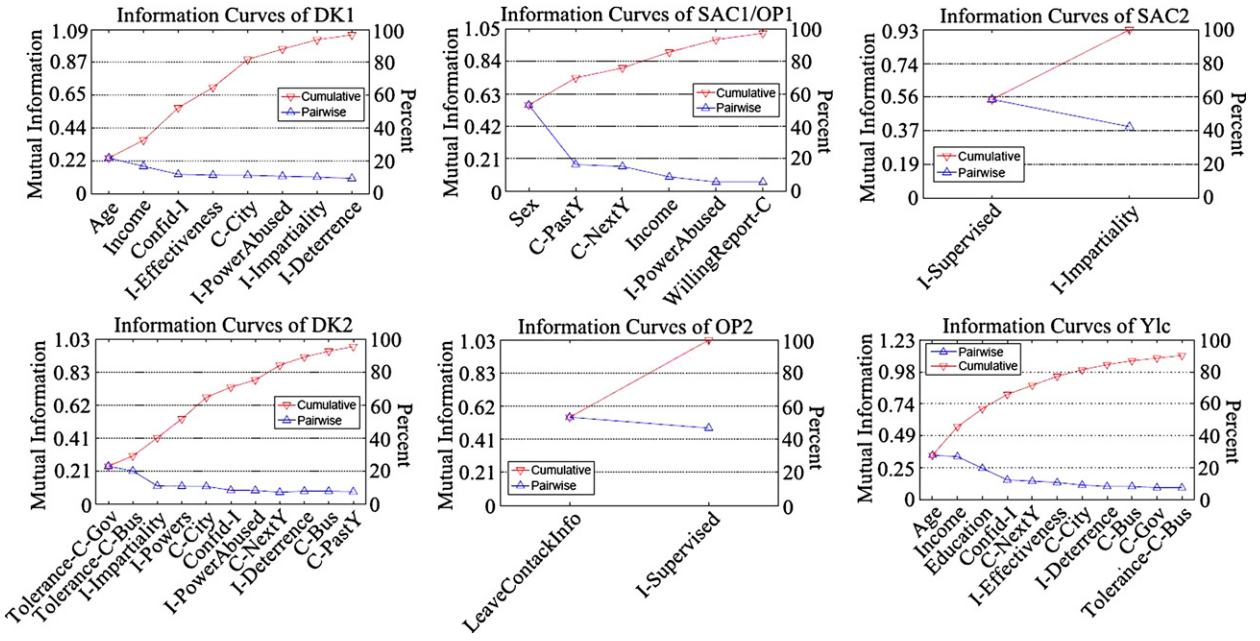


Fig. 14. The information curves of the partitions obtained by DK, OP, SAC and LCA on the ICAC data.

5.3. Comparisons on the ICAC data

As shown in Section 4.4, LTA has produced rich and meaningful clustering results on the ICAC data. We have not been able to obtain similar results with DK, OP and SAC. As a matter of fact, those three methods have a severe drawback in comparison with LTA: They require the user to specify the number of partitions and the number of clusters in each partition. This is very difficult to do for complex data sets such as the ICAC data.

We used the three alternative methods to find two partitions from the ICAC data, and we tried a few options for the number of clusters. The information curves of some of the partitions are given in Fig. 14. None of them seem meaningful. Take DK1 as an example. The first two variables on the curves are demographic information. But their information coverage is very low (around 30%). As such, DK1 is not a partition based on demographic information. The other variables on the curves are about other things. So, we do not find DK1 meaningful. The situation is similar for the other partitions shown in the figure and those not shown in the figure. For OP and SAC, we obtained several more partitions starting from the second one. None of them turned out to be meaningful either.

LCA was also run on the ICAC data. The information curves of the only partition Y_{lc} obtained are shown at the lower-right corner of Fig. 14. The first three variables on the curves are demographic information. However their information coverage is only 57%. So we cannot interpret Y_{lc} as representing a partition based on demographic information. The subsequent variables are about different things. So we do not find the partition particularly meaningful.

6. Comparisons with related methods on labeled data

In this section, we compare LTA with related methods on synthetic and real-world labeled data.

6.1. The evaluation criterion

In the past, researchers have mostly been concerned with clustering algorithms that produce a single partition. A common way to evaluate such an 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, and denote them by Y and C. The quality of the cluster partition is often measured using the *normalized mutual information* $NMI(Y; C)$ between the two partitions [43]. It is given by

$$NMI(C; Y) = \frac{I(C; Y)}{\sqrt{H(C)H(Y)}},$$

where $I(C; Y)$ is the mutual information between C and Y and $H(.)$ stands for entropy [9]. These quantities can be computed from the empirical joint distribution $P(C, Y)$ of Y and C. For distance-based methods, Y is a hard partition and $P(Y, C)$ can be obtained in a straightforward fashion. For model-based methods, Y is a latent variable or a soft partition.

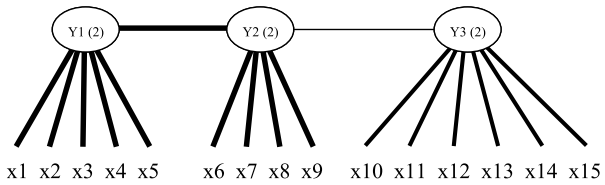


Fig. 15. The generative model for synthetic data.

$P(Y, C)$ is calculated using $P(C, Y) = \frac{1}{N}\sum_{k=1}^{N} P(C|\mathbf{d}_k)P(Y|\mathbf{d}_k)$, where $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N$ are the data cases. NMI ranges from 0 to 1, with a higher value meaning a closer match between Y and C.

This paper is concerned with clustering algorithms that produce multiple partitions. In this context, we generally have multiple class partitions and multiple cluster partitions. How should the evaluation be carried out? Following the literature [23], 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 resembles a class partition, then we claim that the class partition has been recovered from unlabeled data.

6.2. Results on synthetic data

The synthetic data in our experiments were generated from the model shown in Fig. 15, where X_1–X_{15} are manifest variables and Y_1–Y_3 are latent variables. All variables are binary. A total number of 1000 data cases were sampled. The values of the three latent variables were used as the class labels. So there are 3 class partitions. In the experiments, the class labels were first removed and the resulting unlabeled data were analyzed using various methods.

Different methods produced different numbers of cluster partitions. For LTA, the number of cluster partitions was not specified. The method automatically produced 3 cluster partitions. For OP and SAC, we first ran K-means to obtain a first cluster partition and then applied them twice to obtain two other cluster partitions. So, 3 cluster partitions were obtained in each case. For DK only 2 cluster partitions were obtained, and for LCA only 1 cluster partition was obtained.

To see how well the different methods have recovered the class partitions, we match each class partition with the cluster partition with which it has the highest NMI. The NMI values of the matched pairs are given in the following table. The results were obtained over 10 runs.

	DK	SAC	OP	LCA	LTA
Y_1	$.78 \pm .00$	$.73 \pm .04$	$.73 \pm .02$	$.65 \pm .00$	$\mathbf{.91 \pm .00}$
Y_2	$.48 \pm .00$	$.57 \pm .04$	$.55 \pm .04$	$.61 \pm .00$	$\mathbf{.86 \pm .00}$
Y_3	$.97 \pm .00$	$.59 \pm .49$	$.91 \pm .20$	$.48 \pm .00$	$\mathbf{.98 \pm .00}$

The NMI values are the highest for LTA, indicating that the method has recovered the three true class partitions well. On the other hand, the NMI values for the other methods are relatively lower, indicating that they have not been able to recover the true class partitions as well as LTA.

6.3. Results on real-world data

To get real-world labeled data for our experiments, we started with all the data sets used by Friedman et al. [15] and recommended by Weka [41]. A few other data sets were added. Continuous data were discretized using the method by Fayyad and Irani [13]. The methods DK, OP and SAC cannot handle missing values. So, all data cases with missing values were removed from the data sets. A number of data sets became very small after this step and were consequently discarded. At the end, we got 36 data sets. The data are from various domains such as medical diagnosis, handwriting recognition, Biology, Chemistry, etc. The number of attributes ranges from 4 to 69; the number of classes ranges from 2 to 26; and the sample size ranges from 57 to 20,000.

The 36 data sets are all singly labeled. Each data set has only one class partition. The class labels were first removed, and then different methods were used to recover the class partition from the resulting unlabeled data. Here, DK, OP and SAC were instructed to find two partitions, and the number of clusters was set to be the number of classes in the class partition. For LTA, the numbers of partitions and clusters were determined automatically. As always, LCA yielded only one partition. The number of clusters was determined automatically.

The NMI between the class partitions and the cluster partitions obtained by various methods are given in Table 8. The values were computed from the results of 10 runs. For each data set, the best results is highlighted in bold face. It is clear that LTA beats SAC, OP and LCA on most of the data sets. LTA beats DK on 4/7 of the data sets, but loses on the other 3/7 data sets.

Table 8
The performances of LTA and related methods on 36 real-world data sets. The numbers shown are the NMI values between the original class partitions and the cluster partitions obtained by the various methods.

	DK	SAC	OP	LTA	LCA
australian	.35 ± .00	.30 ± .01	.30 ± .00	**.39 ± .00**	.16 ± .00
breast-cancer	.08 ± .00	.04 ± .04	.08 ± .00	**.09 ± .00**	**.09 ± .00**
credit-a	.23 ± .01	.24 ± .01	.24 ± .01	**.42 ± .01**	.12 ± .02
diabetes	.09 ± .02	.08 ± .00	.09 ± .02	**.16 ± .00**	.12 ± .00
heart-statlog	.35 ± .01	.33 ± .01	.33 ± .01	**.39 ± .00**	.30 ± .00
flare	.06 ± .02	.06 ± .02	.05 ± .01	**.07 ± .00**	**.07 ± .00**
glass	.47 ± .00	.43 ± .02	.45 ± .03	**.48 ± .00**	.47 ± .02
ionosphere	.12 ± .00	.23 ± .07	.20 ± .10	**.46 ± .01**	.38 ± .01
kr-vs-kp	.01 ± .01	.02 ± .02	.02 ± .01	**.10 ± .04**	.06 ± .01
letter	.34 ± .19	.44 ± .00	.44 ± .00	**.48 ± .01**	.44 ± .03
mushroom	.22 ± .14	.23 ± .14	.29 ± .10	**.55 ± .03**	.51 ± .03
pima	.08 ± .00	.07 ± .02	.07 ± .01	**.16 ± .04**	.12 ± .00
shuttle-small	.36 ± .12	.29 ± .04	.38 ± .04	**.57 ± .04**	.48 ± .02
vehicle	.17 ± .03	.17 ± .05	.14 ± .02	**.32 ± .01**	.31 ± .01
vote	.54 ± .01	.54 ± .00	.54 ± .00	**.62 ± .00**	.43 ± .00
waveform-21	.38 ± .00	.37 ± .00	.37 ± .00	**.48 ± .00**	.47 ± .00
sick	.30 ± .22	.03 ± .01	.03 ± .01	**.35 ± .10**	.12 ± .00
glass2	.21 ± .02	.15 ± .00	.15 ± .00	**.31 ± .00**	**.31 ± .00**
iris	.71 ± .05	.77 ± .08	.68 ± .11	**.83 ± .00**	**.83 ± .00**
segment	.62 ± .04	.59 ± .05	.55 ± .05	**.65 ± .04**	**.68 ± .02**
waveform-5000	.39 ± .04	.37 ± .00	.37 ± .00	**.47 ± .00**	**.47 ± .00**
autos	**.34 ± .01**	.37 ± .01	.31 ± .05	.23 ± .00	.23 ± .02
mofn-3-7-10	.04 ± .04	**.07 ± .01**	.03 ± .03	.03 ± .03	.03 ± .03
balance-scale	.13 ± .02	.13 ± .04	**.15 ± .03**	.09 ± .00	.09 ± .00
vowel	**.24 ± .01**	**.24 ± .02**	.21 ± .02	.20 ± .00	.18 ± .01
satimage	**.62 ± .00**	**.62 ± .00**	**.62 ± .00**	.52 ± .03	.59 ± .01
breast-w	**.83 ± .00**	**.83 ± .00**	**.83 ± .00**	.71 ± .00	.68 ± .00
corral	**.24 ± .05**	.15 ± .03	.17 ± .06	.19 ± .00	.19 ± .00
credit-g	**.11 ± .02**	.03 ± .01	.06 ± .04	.03 ± .01	.01 ± .00
heart-c	**.32 ± .01**	.26 ± .00	.25 ± .03	.29 ± .02	.30 ± .00
lymph	**.28 ± .04**	.21 ± .05	.19 ± .03	.20 ± .03	.22 ± .01
sonar	**.34 ± .00**	.32 ± .00	.32 ± .00	.24 ± .00	.25 ± .00
soybean	**.71 ± .02**	.66 ± .02	.66 ± .02	.58 ± .03	.70 ± .05
splice	**.69 ± .01**	.55 ± .00	.53 ± .00	.60 ± .08	.09 ± .01
zoo	**.86 ± .04**	.82 ± .05	.79 ± .03	.78 ± .00	.64 ± .00
hypothyroid	**.25 ± .02**	.20 ± .02	.18 ± .04	.22 ± .02	.18 ± .01

6.4. A caveat with the use of labeled data

When it come to the evaluation of multidimensional clustering, case studies on unlabeled data (such as those given in the previous two sections) are more important than indices calculated against labeled data. There are two reasons. First, when we evaluate a clustering method using labeled data, we are asking whether it can recover some known class partitions. If it can, then we can conclude that the method is indeed capable of finding meaningful partitions. If it cannot, however, we cannot conclude the opposite. In fact, the premise of multidimensional clustering is that data can be meaningfully clustered in multiple ways. The known class partitions might be only some of the ways. There might be others, especially when only one class partition is given. To properly determine whether a method can produce meaningful clusterings, all its results should be examined by domain experts.

Second, although the evaluation criterion used in this section is natural and the only criterion used in the literature, it has one drawback. It can give good evaluation to a naïve method that generates a huge number of random partitions.[4] So, it is important to test multidimensional clustering methods on real-world data and see whether they can help find meaningful clusters. On the ICAC data, we found 9 partitions. By inspecting their information curves, we were able to quickly identify a few 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.

7. Other related work

The work presented in this paper is related to subspace clustering (including pattern-based clustering, bi-clustering, and correlation clustering) [31,24]. Both multidimensional clustering and subspace clustering produce clusters that are characterized by subsets of attributes. However, there are important differences. First, multidimensional clustering produces one

[4] One such method is proposed by Caruana et al. [3].

partition for each subset of attributes. On the other hand, subspace clustering might produce one single cluster for a given subset of attributes. Even when multiple clusters are produced for a given subset, they usually do not form a partition. Second, the work we propose is a model-based approach, while most subspace clustering methods are distance-based. In a model-based approach, clusters are defined by the choice of a model class and a model selection criterion. In contrast, the existing subspace clustering methods usually lack a clear task definition [24]. Hoff [21] proposes a model-based method for subspace clustering. It produces unidimensional partitions rather than multidimensional clusterings.

8. Conclusions and future directions

This paper is concerned with cluster analysis of categorical data. Complex data sets such as the ICAC data can usually be meaningfully partitioned in multiple ways, each being based on a subset of the attributes. Unidimensional clustering is unable to uncover such partitions because it seeks one partition that is jointly defined by all the attributes. Feature selection does not help either because there does not exist one 'true' partition. Instead there are multiple 'true' partitions. To discover those partitions, one needs to consider multidimensional clustering.

We propose one method for multidimensional clustering, namely latent tree analysis. An algorithm for latent tree analysis named EAST is described. Empirical results are presented to show that latent tree analysis can indeed obtain rich and meaningful clustering results on complex data. On the ICAC data, for instance, latent tree analysis produced several meaningful partitions and revealed interesting relationships among them. In contrast, none of the alternative methods considered were able to yield any meaningful partitions at all.

On the other hand, the EAST algorithm is rather slow. To analyze the ICAC data, it took around 5 hours on a top end personal computer, while the alternative methods took only dozens of seconds. On another dichotomous data set with 100 attributes and 600 records, it took 23.4 hours. In some applications, it is acceptable for data analysis to take days or weeks, while it is not in other cases. One future direction is to develop faster algorithms for latent tree analysis. A second direction is to extend the work presented in this paper to cover continuous and mixed data. A third direction is to apply multidimensional clustering to various domains.

Acknowledgements

We thank Tianfang Wang, Yan Zhao, and Xiuyan Wu for useful discussions on model interpretation. 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 2011CB505101 and the Shenzhen New Industry Development Fund #CXB201005250021A.

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (1974) 716–723.
[2] J.D. Banfield, A.E. Raftery, Model-based gaussian and non-gaussian clustering, Biometrics 49 (3) (1993) 803–821.
[3] R. Caruana, M. Elhawary, N. Nguyen, C. Smith, Meta clustering, in: Proceedings of the Sixth IEEE International Conference on Data Mining, 2006, pp. 107–118.
[4] P. Cheeseman, J. Stutz, Bayesian classification (autoclass): Theory and results, in: Advances in Knowledge Discovery and Data Mining, 1996, pp. 153–180.
[5] T. Chen, Search-based learning of latent tree models, PhD thesis, 2009, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, http://www.cse.ust.hk/faculty/lzhang/paper/thesis/ChenTao.pdf.
[6] T. Chen, N.L. Zhang, Y. Wang, Efficient model evaluation in the search-based approach to latent structure discovery, in: Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM-08), vol. 8, 2008, pp. 57–64.
[7] D.M. Chickering, Optimal structure identification with greedy search, Journal of Machine Learning Research 3 (2002).
[8] D.M. Chickering, D. Heckerman, Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables, Machine Learning 29 (2–3) (1997) 181–212.
[9] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, New York, 1991.
[10] R.G. Cowell, A.P. David, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer, 1999.
[11] Y. Cui, X.Z. Fern, J.G. Dy, Non-redundant multi-view clustering via orthogonalization, in: Proceedings of the Seventh IEEE International Conference on Data Mining, 2007, pp. 133–142.
[12] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society B 39 (1) (1997) 1–38.
[13] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[14] C. Fraley, A.E. Raftery, MCLUST: Software for model-based cluster analysis, Journal of Classification 16 (1999) 297–306.
[15] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (2–3) (1997) 131–163.
[16] D. Geiger, D. Heckerman, C. Meek, Asymptotic model selection for directed networks with hidden variables, in: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, 1996, pp. 283–290.
[17] L.A. Goodman, The analysis of systems of qualitative variables when some of the variables are unobservable. Part I—a modified latent structure approach, The American Journal of Sociology 79 (5) (1974) 1179–1259.
[18] P. Green, Penalized likelihood, in: Encyclopaedia of Statistical Science, Update Vol. 3, 1999, pp. 578–586.
[19] S. Guindon, O. Gascuel, A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, Systematic Biology 52 (5) (2003) 696–704.
[20] J.A. Hagenaars, Latent structure models with direct effects between indicators: Local dependence models, Sociological Methods and Research 16 (3) (1988) 379–405.

[21] P.D. Hoff, Model-based subspace clustering, Bayesian Analysis 1 (2) (2006) 321–344.
[22] L.A. Hunt, M.A. Jorgensen, Mixture model clustering: a brief introduction to the MULTIMIX program, Australian and New Zealand Journal of Statistics 41 (2) (1999) 153–171.
[23] P. Jain, R. Meka, I.S. Dhillon, Simultaneous unsupervised learning of disparate clusterings, Statistical Analysis and Data Mining 1 (3) (2008) 195–210.
[24] H.-P. Kriegel, P. Kröger, A. Zimek, Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Transactions on Knowledge Discovery from Data (TKDD) 3 (1) (2009) 1–58.
[25] M.H.C. Law, M.A.T. Figueiredo, A.K. Jain, Simultaneous feature selection and clustering using mixture models, IEEE Transaction on Pattern Analysis and Machine Intelligence 26 (2004) 1154–1166.
[26] P.F. Lazarsfeld, N.W. Henry, Latent Structure Analysis, Houghton Mifflin, Boston, 1968.
[27] Y. Li, M. Dong, J. Hua, Simultaneous localized feature selection and model detection for gaussian mixtures, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 953–960.
[28] G.J. McLachlan, K.E. Basford, Mixture Models: Inference and Applications to Clustering, Marcel Dekker, New York, 1988.
[29] G.J. McLachlan, D. Peel, Mixtures of factor analyzers, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 599–606.
[30] I. Nachmana, G. Elidan, N. Friedman, "Ideal parent" structural learning for continuous variable networks, in: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, 2004, pp. 400–409.
[31] L. Parson, E. Haque, H. Liu, Subspace clustering for high dimensional data-a review, SIGKDD Explorations 5 (1) (2004) 90–105.
[32] J.M. Peña, J.A. Lozano, P. Larrañaga, Performance evaluation of compromise conditional gaussian networks for data clustering, International Journal of Approximate Reasoning 28 (1) (2001) 23–50.
[33] J. Pearl, Probabilistic Reasoning in Intelligence Systems, Morgan Kaufmann, San Mateo, 1988.
[34] Z. Qi, I. Davidson, A principled and flexible framework for finding alternative clusterings, in: KDD-09, 2009.
[35] A.E. Raftery, N. Dean, Variable selection for model-based clustering, Journal of American Statistical Association 101 (2006) 168–178.
[36] J. Rissanen, Stochastic complexity (with discussion), Journal of the Royal Statistical Society, Series B 49 (3) (1987) 223–239.
[37] J. Rissanen, Fisher information and stochastic complexity, IEEE Transactions on Information Theory 42 (1) (1996) 40–47.
[38] R.Y. Rubinstein, Simulation and the Monte Carlo Method, Wiley, New York, 1981.
[39] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (2) (1978) 461–464.
[40] J. Uebersax, Latent class analysis frequently asked questions, http://ourworld.compuserve.com/homepages/jsuebersax/faq.htm, 2008.
[41] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, San Francisco, 2005.
[42] N.L. Zhang, Hierarchical latent class models for cluster analysis, Journal of Machine Learning Research 5 (6) (2004) 697–723.
[43] S. Zhong, J. Ghosh, A unified framework for model-based clustering, Journal of Machine Learning Research 4 (2003) 1001–1037.