Latent Tree Classifier

Yi Wang[1], Nevin L. Zhang[2], Tao Chen[3], and Leonard K. M. Poon[2]

[1] Department of Computer Science
National University of Singapore
Singapore 117417, Singapore
wangy@comp.nus.edu.sg
[2] Department of Computer Science & Engineering
The Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong
{lzhang,lkmpoon}@cse.ust.hk
[3] Shenzhen Institute of Advanced Technology
Chinese Academy of Sciences
Shenzhen, China
tao.chen@siat.ac.cn

Abstract. We propose a novel generative model for classification called latent tree classifier (LTC). An LTC represents each class-conditional distribution of attributes using a latent tree model, and uses Bayes rule to make prediction. Latent tree models can capture complex relationship among attributes. Therefore, LTC can approximate the true distribution behind data well and thus achieve good classification accuracy. We present an algorithm for learning LTC and empirically evaluate it on 37 UCI data sets. The results show that LTC compares favorably to the state-of-the-art. We also demonstrate that LTC can reveal underlying concepts and discover interesting subgroups within each class.

Keywords: Bayesian network classifier, latent variable model

1 Introduction

Classification is one of the most active areas in machine learning research. The task is to predict the class label of an instance based on a set of attributes that describe the instance. Approaches to this problem divide into two categories: Generative and discriminative [19]. Let C be the class variable and $\mathbf{X}$ be the set of attributes. Generative approaches build models for the joint distribution $P(C, \mathbf{X})$, compute the posterior distribution $P(C|\mathbf{X})$ using Bayes rule, and assign an instance to the most likely class. In contrast, discriminative approaches directly model $P(C|\mathbf{X})$. In this paper, we focus on generative approaches and assume categorical attributes.

The simplest generative model is the naive Bayes (NB) classifier [8]. It assumes that attributes are mutually independent given the class label. All dependencies among attributes are ignored. Despite its simplicity, NB has been shown to be surprisingly effective in a number of domains [7].

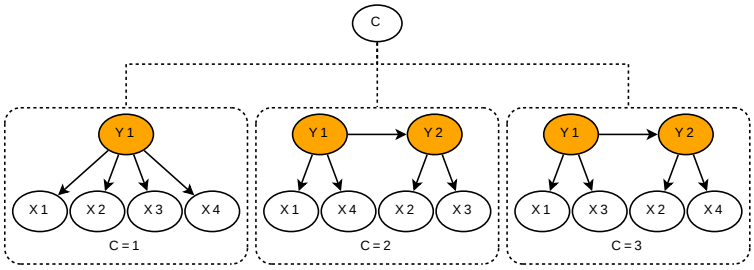


Fig. 1. An example latent tree classifier. C is the class variable with 3 classes, X_1–X_4 are four attributes. Each rectangle contains a latent tree model for a specific class, in which Y_1 and Y_2 are latent variables.

The conditional independence assumption underlying NB is rarely true in practice. Violating this assumption could lead to poor prediction. The past decade has seen a large body of work on relaxing this unrealistic assumption. To mention two successful instances, tree augmented naive Bayes (TAN) [10] builds a Chow-Liu tree [5] to model the attribute dependencies, while averaged one-dependence estimators (AODE) [21] constructs a set of tree models over the attributes and averages them to make prediction.

In this paper, we propose a novel approach to model the relationship among attributes. Our approach is based on latent tree models. A *latent tree model* (LTM) [23] is a tree-structured Bayesian network in which variables at leaf nodes are observed and called *manifest variables*, whereas variables at internal nodes are hidden and called *latent variables*. The model represents a set of complex relationship among the manifest variables in a compact way. To see this point, consider eliminating all the latent variables from the model. This will result in a fully connected Bayesian network over all the manifest variables.

In our approach, we treat attributes as manifest variables and build LTMs to model the relationship among them. The relationship could be different across classes. Therefore, we build an LTM for each class. We refer to the collection of LTMs plus the prior class distribution as a latent tree classifier (LTC). An example is shown in Fig. 1. Each rectangle in the figure contains the LTM for a class. Since the LTMs can model complex relationship among attributes, we expect LTC to approximate the true distribution behind data well and thus to achieve good classification accuracy. We empirically verify this hypothesis in the experiments.

In addition to good classification performance, building LTCs on the basis of LTMs also makes it possible to discover latent structures behind data. In particular, we will demonstrate that the latent variables introduced during the learning process can reveal concepts embedded in data as well as interesting subgroups within each class. This merit can boost user confidence in LTCs.

The rest of this paper is structured as follows. We formally define LTC in Sect. 2 and present an algorithm for learning LTC in Sect. 3. In Sect. 4, we

empirically evaluate LTC on 37 UCI data sets, and compare it with a spectrum of generative classifiers as well as C4.5 [18]. In Sect. 5, we demonstrate that LTC can discover appealing latent structures using an example. We discuss some related work in Sect. 6 and finally conclude this paper in Sect. 7.

2 Latent Tree Classifier

We start by briefly reviewing latent tree models (LTMs). An LTM is a pair $\mathcal{M} = (m, \boldsymbol{\theta})$. The first component m denotes the rooted tree and the set of cardinalities of the latent variables. The second component $\boldsymbol{\theta}$ denotes the collection of parameters in $\mathcal{M}$. It contains a conditional probability table (CPT) for each node given its parent.

Let $\mathbf{X}$ and $\mathbf{Y}$ be the set of manifest variables and the set of latent variables in $\mathcal{M}$, respectively. We use $P(\mathbf{X}, \mathbf{Y}|\mathcal{M})$ to denote the joint distribution represented by $\mathcal{M}$. Two LTMs $\mathcal{M}$ and $\mathcal{M}'$ are *marginally equivalent* if they share the same set of manifest variables $\mathbf{X}$ and $P(\mathbf{X}|\mathcal{M}) = P(\mathbf{X}|\mathcal{M}')$.

Let $|Z|$ denote the cardinality of variable Z. For a node Z in $\mathcal{M}$, we denote the set of its neighbors by $\mathbf{nb}(Z)$. An LTM is *regular* if for any latent node Y, $|Y| \leq \frac{\prod_{Z \in \mathbf{nb}(Y)} |Z|}{\max_{Z \in \mathbf{nb}(Y)} |Z|}$, and the inequality strictly holds when Y has only two neighbors. As shown by [23], an irregular model $\mathcal{M}$ is over-complicated and can be reduced to a regular model $\mathcal{M}'$ which is marginally equivalent but contains fewer parameters than $\mathcal{M}$. Henceforth, we consider only regular models.

We consider the classification problem where each instance is described using n attributes $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, and belongs to one of the r classes $C = 1, 2, \ldots, r$. A *latent tree classifier* (LTC) consists of a prior distribution $P(C)$ on C and a collection of r LTMs over the attributes $\mathbf{X}$. We denote the c-th LTM by $\mathcal{M}_c = (m_c, \boldsymbol{\theta}_c)$ and the set of latent variables in $\mathcal{M}_c$ by $\mathbf{Y}_c$. The LTC represents a joint distribution over C and $\mathbf{X}$, $\forall c = 1, 2, \ldots, r$,

$$\begin{aligned} P(C = c, \mathbf{X}) &= P(C = c) P(\mathbf{X}|\mathcal{M}_c) \\ &= P(C = c) \sum_{\mathbf{Y}_c} P(\mathbf{X}, \mathbf{Y}_c|\mathcal{M}_c) \ . \end{aligned} \tag{1}$$

Given an LTC, we classify an instance $\mathbf{X} = \mathbf{x}$ to the class $c^\star$, where

$$\begin{aligned} c^\star &= \arg\max_C P(C|\mathbf{X} = \mathbf{x}) \\ &= \arg\max_C P(C, \mathbf{X} = \mathbf{x}) \ . \end{aligned} \tag{2}$$

Note that, according to (1), this requires us to sum out all the latent variables $\mathbf{Y}_c$ for each class c. Thanks to the tree structures of LTMs, the summation could be done in linear time in the number of attributes, as formalized below.

Proposition 1. *The time complexity of classifying an unlabeled instance with an LTC is $O(rnv^2)$, where r is the number of classes, n is the number of attributes, and v is the maximum cardinality of variables in the LTC.*

Proof. The time complexity of summing out all the latent variables $\mathbf{Y}_c$ from the c-th LTM $\mathcal{M}_c$ is $O\big((|\mathbf{Y}_c|+n)v_c^2\big)$ [17], where v_c denotes the maximum cardinality of the variables in $\mathcal{M}_c$. It is know that a regular LTM contains less than n latent variables [23]. Therefore, the overall time complexity for classifying an instance is $O(rnv^2)$. □

3 A Learning Algorithm

Given a labeled training set $\mathcal{D}$, we consider how to learn a good LTC from $\mathcal{D}$. This amounts to learning the prior distribution $P(C)$ and a good LTM $\mathcal{M}_c$ for each class $c = 1, 2, \ldots, r$.

The prior $P(C)$ can be easily estimated from $\mathcal{D}$ by counting the number of instances belonging to each class. In the following, we focus on the more challenging task of learning the LTMs.

3.1 Model Selection

We first partition $\mathcal{D}$ by class label and obtain r data sets $\{\mathcal{D}_c|c = 1, 2, \ldots, r\}$. Each $\mathcal{D}_c$ contains only the attributes $\mathbf{X}$. We then learn an LTM $\mathcal{M}_c$ from each $\mathcal{D}_c$ independently.

The LTM is of high quality if it is close to the true distribution underlying $\mathcal{D}_c$. Nonetheless, the true distribution is unknown. Therefore, we use AIC score [1] for model selection,

$$AIC(m_c|\mathcal{D}_c) = -2\log P(\mathcal{D}_c|m_c, \boldsymbol{\theta}_c^\star) + 2d(m_c) \ , \tag{3}$$

where $\boldsymbol{\theta}_c^\star$ is the maximum likelihood estimate to the parameters $\boldsymbol{\theta}_c$, and $d(m_c)$ is the number of free parameters in model m_c. The AIC score is an approximation to the expected KL divergence of $\mathcal{M}_c$ from the true distribution. The lower the score, the smaller the difference between $\mathcal{M}_c$ and the true distribution, and the better the LTM.

In literatures, BIC score is used more often for learning Bayesian network classifiers [10]. However, BIC score over-penalizes complex models and can lead to poor approximation to the true distribution. In a preliminary study [20], we empirically compared AIC with BIC for learning LTCs. We observed that AIC produces LTMs that better fit unseen data. The LTCs learned using AIC also achieve better classification accuracy than those learned using BIC.

3.2 Model Search

We adopt a recently developed hill-climbing algorithm called EAST [3] to search for high scoring LTMs. EAST explores the model space using five search operators. They are *node introduction* (NI), *node deletion* (ND), *node relocation* (NR), *state introduction* (SI), and *state deletion* (SD). Given an LTM, NI applies to a latent variable and two of its neighbors. It adds a new latent variable to mediate

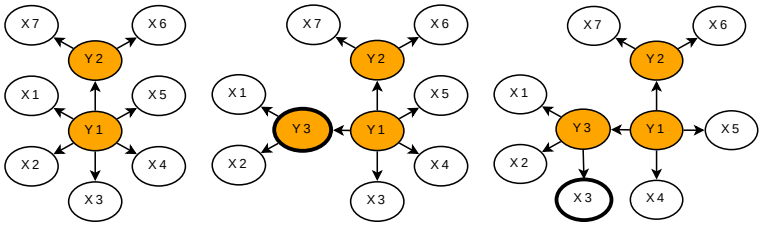


Fig. 2. Illustration of NI, ND, and NR operators.

the latent variable and the two neighbors, and sets its cardinality to the same as the existing latent variable. Figure 2 shows such an example. The model m_2 is obtained from m_1 by introducing a new latent variable Y_3. ND is the reverse of NI. It applies to two neighboring latent variables, removes one of them and links its neighbors to the other. In Fig. 2, by deleting Y_3 one goes from m_2 back to m_1. NR operator adjusts the connections in an LTM. It considers two latent variables, disconnects a neighbor from the first latent variable, and links it to the second latent variable. In Fig. 2, one relocates X_3 in m_2 from Y_1 to Y_3 and obtains m_3. The last two operators modify domains of latent variables. SI adds a new state to a latent variable. SD does the reverse.

Given a data set, EAST starts with the simplest LTM, i.e., the LTM that contains only one latent variable whose cardinality equals to 1, and greedily improves the model. In each search step, it applies some operators to the current model, obtains a collection of candidate models, evaluates them using the AIC score, and picks the best one to seed the next search step. The process repeats itself until the model score ceases to increase. Note that, if EAST never improves the initial model, the final LTC reduces to NB.

In each search step, one could apply all the five operators to the current model. This could, however, produce a large number of candidate models. Evaluating them could take a long time. Therefore, the search procedure in EAST is structured into three phases: Expansion, simplification, and adjustment. In each phase, we consider only one or two operators and thus obtain much fewer candidate models. In the expansion phase, we only apply NI and SI. Both operators make the current model more expressive and thus improve the first term in AIC score. In the simplification phase, we consider only ND and SD. Both operators simplify the current model and thus improve the second term in AIC score. In the adjustment phase, we apply NR to adjust the structure of the current model. It helps avoid local maxima. EAST iteratively goes through these three phases and alternatively improves the two terms in AIC score.

To evaluate a candidate model, one needs to run the EM algorithm [6] to compute the MLE $\boldsymbol{\theta}^\star$. EM is known to be expensive. To speed up the evaluation process, we run local EM instead. The key observations are that (1) the parameters of the current model have already been optimized, and (2) a can-

didate model only differs from the current model in a small part. Therefore, in local EM, we fix the parameters of the unaltered part, and optimize only the parameters that are foreign to the current model. Consider the model m_2 in Fig. 2 that is obtained from m_1 using NI. In local EM, we only optimize the CPTs of Y_3, X_1, and X_2. In our implementation, we run local EM for a predetermined number of iterations. It might not converge, but the obtained estimation is accurate enough for ranking candidate models. After we obtain the best candidate model, we optimize its parameters using full EM before comparing it with the current model.

4 Empirical Evaluation

In this section, we empirically compare LTC with a spectrum of generative classifiers, ranging from the simplest NB, to more advanced TAN and AODE, and to the most general *Bayesian network augmented naive Bayes* (BAN) [10]. We also include C4.5 decision tree [18] in the comparison as a reference.

4.1 Experimental Settings

We used the 37 UCI data sets [2] that are recommended by WEKA [22] in our experiments. The learning algorithms of TAN and AODE proposed by [10] and [21] do not handle missing values. Thus, we removed incomplete instances from the data sets. TAN, AODE, BAN, and LTC require discrete attributes. Therefore, we discretized the data sets using the supervised discretization method proposed by [9]. Table 1 summarizes the characteristics of the data sets obtained after preprocessing.

We implemented LTC in Java. The detailed settings are as follows. We ran 40 iterations of local EM to evaluate each candidate model. For the best candidate model, we ran full EM to optimize its parameters. The EM was terminated if the improvement in loglikelihoods is smaller than 0.01, or the number of iterations reaches 500. For both local and full EM, we adopted the pyramid strategy proposed by [4] to avoid local maxima. The number of starting points was set at 16 and 64, respectively.

We used the WEKA implementations of the other classification algorithms in our experiments. Some details are given below:

- **AODE**: We set the frequency limit on super parents at 30 as suggested by [21].
- **BAN**: We set the initial models to be naive Bayes and used hill-climbing to search for good BANs with high AIC scores. The Markov blanket correction built in WEKA was conducted on the final models to ensure every attribute is in the Markov blanket of the class variable.

Following the common practice in machine learning, we smoothed the parameters for all the trained generative classifiers using Laplace correction. We set the smoothing factor $\alpha = 1$. Preliminary experimental results show that the

Table 1. The 37 data sets used in the experiments, their characteristics (columns 2-4), and the classification accuracy of various algorithms (columns 5-10). For each data set, the best accuracy is highlighted in boldface.

Domain	Attr.	Class	Size	LTC	NB	TAN	AODE	BAN	C4.5
anneal	38	6	898	97.78±1.96	96.10±2.54	98.66±1.15	98.33±1.59	97.99±1.47	**98.77±0.98**
australian	14	2	690	85.36±4.29	85.51±2.65	85.22±5.33	**86.09±3.50**	85.51±4.10	85.65±4.07
autos	25	7	159	**85.50±8.44**	72.88±10.12	79.25±8.84	81.08±7.39	77.29±6.40	78.58±8.54
balance-scale	4	3	625	70.24±3.10	70.71±4.08	**71.03±3.51**	69.59±4.01	70.24±4.44	69.59±4.27
breast-cancer	9	2	277	72.87±9.31	75.41±6.44	71.11±5.14	**76.49±7.96**	71.42±6.54	74.39±7.34
breast-w	9	2	683	**97.51±1.54**	97.51±2.19	96.63±2.08	97.36±2.04	97.07±1.95	95.76±2.61
corral	6	2	128	**100.00±0.00**	85.96±7.05	99.23±2.43	89.10±8.98	97.69±3.72	94.62±8.92
credit-a	15	2	653	86.38±4.38	87.29±3.53	86.84±3.02	**87.59±3.51**	85.31±3.44	86.99±4.48
credit-g	20	2	1000	73.20±3.97	75.80±4.32	74.00±4.40	**77.10±4.38**	74.90±3.54	72.10±4.46
diabetes	8	2	768	76.44±2.44	77.87±3.50	78.77±3.32	78.52±4.11	**78.91±3.62**	78.26±3.97
flare	10	2	1066	**83.21±2.77**	80.30±3.42	82.84±2.27	82.46±2.31	82.93±2.33	82.09±1.80
glass	9	7	214	76.19±7.41	74.37±8.97	**76.19±9.88**	**76.19±7.41**	74.46±11.28	73.94±9.76
glass2	9	2	163	**85.18±9.44**	83.97±8.99	85.18±9.89	83.97±9.91	85.18±9.89	84.01±7.32
heart-c	13	5	296	82.46±4.66	**84.11±7.85**	82.80±5.74	83.10±7.17	83.10±6.99	74.66±6.49
heart-statlog	13	2	270	81.85±9.63	**83.33±6.36**	82.22±6.94	81.85±6.86	80.00±7.65	81.85±5.91
hepatitis	19	2	80	88.75±12.43	85.00±15.37	88.75±13.76	85.00±12.91	87.50±11.79	**90.00±14.19**
ionosphere	34	2	351	**94.31±2.99**	90.60±3.83	93.17±3.60	92.31±2.34	93.17±4.07	89.17±5.35
iris	4	3	150	94.00±5.84	94.00±5.84	**94.67±5.26**	93.33±5.44	94.00±5.84	94.00±4.92
kr-vs-kp	36	2	3196	96.62±1.19	87.89±1.81	92.21±2.30	91.18±0.83	97.06±0.92	**99.44±0.48**
letter	16	26	20000	**92.71±0.47**	74.04±1.04	85.61±0.63	88.91±0.50	85.01±0.84	78.63±0.62
lymph	18	4	148	**89.14±6.65**	83.67±6.91	85.10±7.01	85.62±8.66	87.05±9.99	78.33±10.44
mofn-3-7-10	10	2	1324	94.48±2.48	85.35±1.53	91.16±1.79	89.05±2.53	**100.00±0.00**	**100.00±0.00**
mushroom	22	2	5644	**100.00±0.00**	97.41±0.72	99.81±0.26	**100.00±0.00**	99.95±0.09	**100.00±0.00**
pima	8	2	768	77.35±3.92	78.13±4.24	78.65±4.62	**78.65±3.81**	77.87±4.53	78.38±2.90
primary-tumor	17	22	132	45.60±11.04	**47.14±11.59**	41.04±12.56	46.37±10.12	45.60±8.64	43.24±10.55
satimage	36	6	6435	**89.57±1.24**	82.42±1.51	88.50±0.89	89.26±0.59	87.91±1.01	84.37±1.34
segment	19	7	2310	**95.67±1.47**	91.52±1.60	95.32±1.74	95.63±1.23	95.06±1.80	95.32±1.63
shuttle-small	9	7	5800	**99.88±0.14**	99.34±0.27	99.81±0.15	99.84±0.13	99.86±0.11	99.59±0.19
sonar	60	2	208	82.31±9.19	85.62±5.41	86.60±7.72	**87.07±6.31**	80.29±8.85	79.81±8.14
soybean	35	19	562	**93.78±2.52**	91.64±4.44	93.41±3.38	91.99±4.22	92.17±4.70	91.82±3.75
splice	61	3	3190	94.51±2.07	95.36±1.00	95.30±1.41	**96.21±1.07**	94.48±1.40	94.36±1.58
vehicle	18	4	846	**74.94±4.29**	62.65±4.15	73.99±4.44	73.06±4.65	72.94±5.00	71.99±3.45
vote	16	2	232	**95.86±3.38**	89.91±4.45	94.03±4.72	94.03±4.07	93.81±4.93	95.18±4.48
vowel	13	11	990	80.30±3.02	67.07±6.14	**87.37±2.94**	81.92±4.11	82.22±3.96	80.91±2.31
waveform-21	21	3	5000	86.02±1.99	81.76±1.49	83.10±1.46	**86.60±1.26**	83.10±1.25	75.44±2.10
waveform-5000	40	3	5000	86.06±1.39	80.74±1.38	82.02±1.26	**86.36±1.65**	82.44±1.21	76.48±1.47
zoo	17	7	101	94.18±6.60	93.18±7.93	95.18±8.15	95.09±5.18	**96.09±5.05**	92.18±8.94
Mean				**86.49±4.26**	83.12±4.72	85.80±4.43	85.85±4.40	85.66±4.41	84.32±4.59
# Wins				**15**	3	4	11	3	5

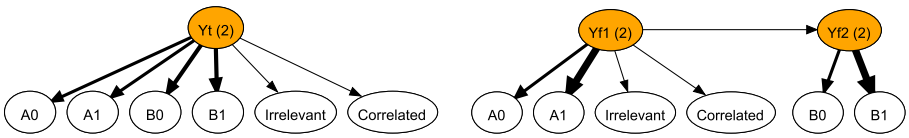
parameter smoothing leads to significant improvement in classification accuracy [20].

4.2 Classification Accuracy

We estimated the classification accuracy of an algorithm using stratified 10-fold cross validation [12]. All the algorithms were run on the same training/test splits. The mean and the standard deviation of accuracy are shown in Table 1. For each data set, the best accuracy is highlighted in boldface. For each algorithm, Table 1 also reports its average accuracy over all the data sets and the number of wins, i.e., the number of data sets on which it achieves the best accuracy.

Table 2. The number of times that LTC significantly wins, ties with, and loses to the other algorithms.

	NB	TAN	AODE	BAN	C4.5
# Wins	17	8	5	5	11
# Ties	19	27	29	29	24
# Loses	1	2	3	3	2



(a) $\mathcal{M}_t$ (b) $\mathcal{M}_f$

Fig. 3. The structures of the LTMs for corral data. The numbers in the parentheses denote the cardinalities of the latent variables. The width of an edge denote the mutual information between the incident nodes.

From Table 1, we can see that LTC achieves the best overall accuracy, followed by AODE, TAN, BAN, C4.5, and NB, in that order. In terms of the number of wins, LTC is also the best (15 wins), with AODE (11 wins) and C4.5 (5 wins) being the two runners-up.

To compare LTC with the other algorithms, we also conducted two-tailed paired t-test with $p = 0.05$. The number of significant wins, ties, and loses is given in Table 2. It shows that LTC significantly outperforms NB (17 wins/1 loses) and C4.5 (11/2). LTC is also better than TAN (8/2), AODE (5/3), and BAN (5/3).

5 Discovery of Latent Structures

One advantage of LTC is that it can capture concepts underlying domains and automatically discover interesting subgroups within each class. In this section, the readers will see one such example. More examples will be given in an extended version of this paper.

The example is involved with the corral data [11]. It contains two classes *true* and $false$, and six boolean attributes A_0, A_1, B_0, B_1, $Irrelevant$, and $Correlated$. The target concept is $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. $Irrelevant$ is an irrelevant random attribute, and $Correlated$ matches the class label 75% of the time.

We learned an LTC from the corral data and obtained two LTMs, one for each class. We denote the LTMs by $\mathcal{M}_t$ and $\mathcal{M}_f$, respectively. Their structures are shown in Fig. 3. $\mathcal{M}_t$ contains one latent variable Y_t, and $\mathcal{M}_f$ contains two latent variables Y_{f1} and Y_{f2}. All the latent variables are binary.

5.1 Main Findings

We first observe that in both models, the four attributes A_0, A_1, B_0, and B_1 are closely correlated to their latent parents. In contrast, $Irrelevant$ and $Correlated$ are almost independent of their parents (notice the difference in edge widths in Fig. 3). This is interesting as both models correctly pick the four relevant attributes to the target concept.

We further studied the meanings of the latent variables and obtained more appealing findings. The latent variable Y_t in $\mathcal{M}_t$ takes two values. Therefore, Y_t represents a soft partition over the samples in the *true* class into two groups, each group corresponding to one value of Y_t. We refer to those groups as *latent groups*, and denote them by the corresponding values of Y_t. The latent variables Y_{f1} and Y_{f2} in $\mathcal{M}_f$ also take two values. Similarly, each latent variable represents a peculiar soft partition over the samples in the $false$ class into two latent groups.

Our analysis in the next subsection will show that:

1. The latent groups $Y_t = 1$ and $Y_t = 2$ correspond to the two components of the concept, $A_0 \wedge A_1$ and $B_0 \wedge B_1$, respectively;
2. The latent groups $Y_{f1} = 1$ and $Y_{f1} = 2$ correspond to $\neg A_0$ and $\neg A_1$, while the latent groups $Y_{f2} = 1$ and $Y_{f2} = 2$ correspond to $\neg B_0$ and $\neg B_1$;
3. The latent variables Y_{f1} and Y_{f2} jointly enumerate the four cases when the target concept $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$ does not satisfy.

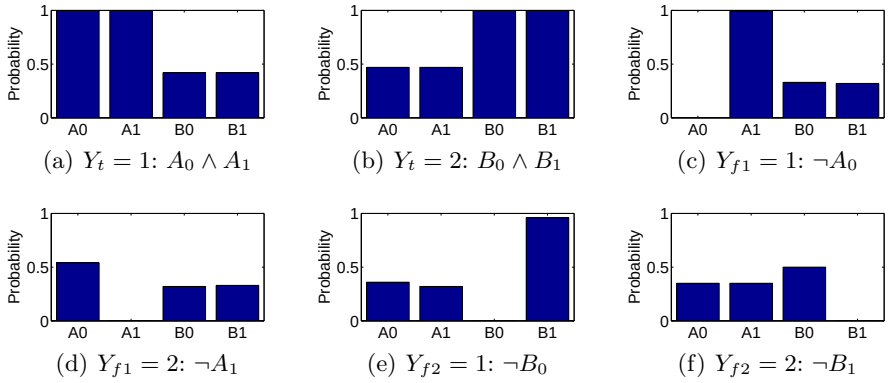
Before diving into the details, we would like to point out that LTC successfully discovering the underlying concept and intra-class subgroups gives rise to its perfect classification result on the corral data (see Table 1). We argue that the capability of discovering such latent patterns is one reason why LTC achieves good classification accuracy.

5.2 Detailed Analysis

To understand the characteristics of each latent group, we examine the conditional distribution of each attribute, i.e., $P(X|Y = 1)$ and $P(X|Y = 2)$ for all $X \in \{A_0, A_1, B_0, B_1\}$ and $Y \in \{Y_t, Y_{f1}, Y_{f2}\}$. Those distributions are plotted in Fig. 4. The height of a bar indicates the corresponding probability value.

We start by the latent groups associated with Y_t. In latent group $Y_t = 1$, A_0 and A_1 always take value *true*, while B_0 and B_1 emerge at random. Clearly, this group of instances belong to class *true* because they satisfy $A_0 \wedge A_1$. In contrast, in latent group $Y_t = 2$, B_0 and B_1 always take value *true*, while A_0 and A_1 emerge at random. Clearly, this group corresponds to the concept $B_0 \wedge B_1$.

We next examine the two latent variables in $\mathcal{M}_f$. It is clear that A_0 never occurs in latent group $Y_{f1} = 1$, while A_1 never occurs in latent group $Y_{f1} = 2$. Therefore, the two latent groups correspond to $\neg A_0$ and $\neg A_1$, respectively. Y_{f1} thus reveals the two cases when $A_0 \wedge A_1$ does not satisfy. Similarly, we find that B_0 never occurs in latent group $Y_{f2} = 1$, while B_1 never occurs in latent group $Y_{f2} = 2$. Therefore, the two latent groups correspond to $\neg B_0$ and $\neg B_1$, respectively. Y_{f2} thus reveals the two cases when $B_0 \wedge B_1$ does not satisfy. Consequently, Y_{f1} and



(a) $Y_t = 1$: $A_0 \wedge A_1$ (b) $Y_t = 2$: $B_0 \wedge B_1$ (c) $Y_{f1} = 1$: $\neg A_0$

(d) $Y_{f1} = 2$: $\neg A_1$ (e) $Y_{f2} = 1$: $\neg B_0$ (f) $Y_{f2} = 2$: $\neg B_1$

Fig. 4. The attribute distributions in each latent group and the corresponding concept.

Y_{f2} jointly represent the four cases when the target concept $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$ does not satisfy.

6 Related Work

There are a large body of literatures that attempt to improve classification accuracy by exploiting attribute dependencies. They mainly divide into two categories: Those that directly model relationship among attributes, and those that capture such relationship using latent variables. TAN, AODE, and BAN fall into the first category. Another representative from this category is Bayesian multinet [10]. It learns a Bayesian network for each class and uses them jointly to make prediction. Our method is based on the similar idea, but we learn an LTM to represent the joint distribution of each class.

Our method falls into the second category. In this category, various latent variable models have been tested for continuous data. To give two examples, [16] combine finite mixture model with naive Bayes classifier. [13] propose latent classification model. It uses a mixture of factor analyzers to represent attribute dependencies.

In contrast, we are aware of much less work on categorical data. The one that is the most closely related to ours is the hierarchical naive Bayes model (HNB) [24, 14]. HNB also exploits LTM to model the relationship among attributes. However, it differs from LTC in two aspects. First, attributes in HNB are usually partitioned into disjoint subsets, while each subset is modeled using a separate LTM. The root (latent) nodes of those LTMs can be treated as features extracted from different subsets of attributes, and are put together with the class variable to form a naive Bayes model for classification. In contrast, LTC builds one single LTM to connect all attributes for each class. The LTM as a whole gives a generative model for that class, which is used in the prediction phase to compute the likelihood of new data points.

Second, HNB assumes homogeneous latent structure, i.e., the LTMs in HNB are identical throughout all classes. This assumption could be unrealistic in real world applications. See, for example, the corral data presented in Sect. 5. Violating this assumption could lead to degenerated classification performance and failure in latent structure discovery. In contrast, LTC describes different classes using different LTMs. Therefore, it can accommodate the variance across different classes.

We did not include HNB in our empirical comparison. The learning algorithm proposed by [24] can only deal with several attributes and does not scale up to most data sets used in our experiments. [14] developed a more efficient learning algorithm but we have not been able to gain access to their implementation.

Recently, [15] extend the latent classification model to discrete domain. The proposed model only handles binary attributes. Its generalization to multi-valued categorical attributes is non-trivial.

7 Conclusions

We propose a novel generative classifier, namely, latent tree classifier. It builds upon the powerful yet compact representation of latent tree models, and respects the inter-class heterogeneity of the relationships among attributes. We empirically show that LTC compares favorably to NB, TAN, AODE, BAN, and C4.5 in classification accuracy. We also demonstrate that the learned LTC can reveal underlying concepts and discover interesting subgroups within each class. As far as we know, the second feature is unique to our method. We argue that the capability of discovering such latent patterns is one reason why LTC achieves good classification performance.

We used a hill-climbing algorithm, EAST, to learn LTC. For most of the data sets used in the experiments, the training finished within a few seconds to a few hours. For the 5 large data sets, kr-vs-kp, letter, mushroom, satimage, and splice, the training took up to a few days. Thus, LTC is currently most suitable for applications which allow a long offline training phase but demand good online classification performance. On the other hand, we believe that the promising results presented in this paper warrant future research on fast learning algorithms for LTCs.

Acknowledgments. We are grateful to Gang Wang for his inspiring discussions. Research on this work was supported by Hong Kong Research Grants Council GRF Grant #622408, MDA GAMBIT Grant R-252-000-398-490, the National Basic Research Program of China (aka the 973 Program) under project 2011CB505101, and the Shenzhen New Industry Development Fund #CXB201005250021A.

References

1. Akaike, H.: A new look at the statistical model identification. IEEE T. Automat. Contr. 19(6), 716–723 (1974)

2. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
3. Chen, T., Zhang, N.L., Wang, Y.: Efficient model evaluation in the search-based approach to latent structure discovery. In: 4th European Workshop on Probabilistic Graphical Models, pp. 57–64 (2008)
4. Chickering, D.M., Heckerman, D.: Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. Mach. Learn. 29, 181–212 (1997)
5. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE T. Inform. Theory 14(3), 462–467 (1968)
6. Dempster, A.P., Laird, N.M., Rubin, D.R.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. B Met. 39(1), 1–38 (1977)
7. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Mach. Learn. 29, 103–130 (1997)
8. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley (1973)
9. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th International Joint Conference on Artificial Intelligence, pp. 1022–1027 (1993)
10. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. 29(2-3), 131–163 (1997)
11. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: 11th International Conference on Machine Learning, pp. 121–129 (1994)
12. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: 14th International Joint Conference on Artificial Intelligence, pp. 1137–1145 (1995)
13. Langseth, H., Nielsen, T.D.: Latent classification models. Mach. Learn. 59(3), 237–265 (2005)
14. Langseth, H., Nielsen, T.D.: Classification using hierarchical naive Bayes models. Mach. Learn. 63(2), 135–159 (2006)
15. Langseth, H., Nielsen, T.D.: Latent classification models for binary data. Pattern Recogn. 42, 2724–2736 (2009)
16. Monti, S., Cooper, G.F.: A Bayesian network classifier that combines a finite mixture model and a naive Bayes model. In: 11th Annual Conference on Uncertainty in Artificial Intelligence, pp. 447–456 (1995)
17. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
18. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
19. Rubinstein, Y.D., Hastie, T.: Discriminative vs informative learning. In: 3rd International Conference on Knowledge Discovery and Data Mining, pp. 49–53 (1997)
20. Wang, Y.: Latent Tree Models for Multivariate Density Estimation: Algorithms and Applications. Ph.D. thesis, Hong Kong University of Science & Technology (2009)
21. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: Aggregating one-dependence estimators. Mach. Learn. 58, 5–24 (2005),
22. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
23. Zhang, N.L.: Hierarchical latent class models for cluster analysis. Journal of Mach. Learn. Research 5(6), 697–723 (2004)
24. Zhang, N.L., Nielsen, T.D., Jensen, F.V.: Latent variable discovery in classification models. Artif. Intell. Med. 30(3), 283–299 (2004)