Chapter 6

An Approach to Hidden variable discovery

Atâtea clăile de fire stângi!
Găsi-vor oare gest inchis să le rezume,
Să nege, dreaptă, linia ce frângi–
Ochi în virgin triunghi tăiat spre lume?
Ion Barbu
–Grup
So many heaps of tangled straw!
Will they find clean gesture to comprise them,
To straighten them into the golden beam
Shone by the divine triangle eye?

This chapter presents a way of using mixtures of trees and the MixTreeS learning algorithm for discovering structure in a particular class of graphical models with hidden variables. In the process of structure discovery model selection and model validation are important components. With this goal in mind, I introduce and a novel method for validating independencies in graphical models with hidden variables whose scope is much broader than the class of models investigated here.

6.1 Structure learning paradigms

Structure discovery has been one of the most challenging and most fascinating problems in graphical models ever since the beginning of the field. This is to no surprise given that the task of identifying dependencies and independencies in observations is at the core of scientific discovery as an intellectual process. Work in this field can be loosely categorized into three areas of research.

The first category of structure learning methods (and the oldest also) is based on conditional independence tests. [68] introduce an algorithm that constructs a Bayesian network that is consistent with a given list of independence statements. A similar result exists for decomposable models: [16] described a method to learn decomposable models or decomposable approximations to Markov network models using an oracle for conditional independence queries. The algorithm is exponential in the size of the largest separator. The two works are impressive in their success in inverting the relationship between a graphical model structure and its list of independencies. On the other hand, the strength of the results reflects the strength of the underlying assumptions. In other words, the methods assume as input an information that is at least as hard to obtain as the graph itself. Testing the conditional independence $A \perp B \mid C$ is exponential in $|A|.|B|.|C|$. The first method assumes that we know beforehand all independencies in a domain V (and implicitly all the pairs A, B that are *not* independent). But who is to give us such a list of independencies? Moreover, assuming that we were to undertake the intractable task of performing all the required independence tests, we would be faced with uncertain and perhaps contradictory information that none of the above methods are prepared to handle.

[55] presents an algorithm for finding a polytree structure from mutual information measurements when the underlying distribution is known to be a polytree. The same chapter presents a method for learning *star decomposable graphs*[1] the case of binary variables and ideal correlation measurements. These methods have the same drawback of requiring ideal information that cannot be obtained from data.

[6] introduces an algorithm for learning a minimal I-map of a distribution from data. The algorithm is based on evaluation of pairwise mutual informations between variables, and independence tests that require computational effort proportional to the size of the largest separator, but it is shown to work efficiently on sparse graphs. The implicit assumption that complete independence information can be obtained from pairwise independence tests - an assumption that will be examined later in this chapter. It is also assumed that there is sufficient data to ensure a correct answer to all the independence tests, an assumption that brings this paper in the same group with the ones requiring ideal information.

The second category of methods explicitly takes into account the existence of a finite sample of data points from which the structure has to be learned. The most influential is the Bayesian approach pioneered by [9] and developed by Heckerman, Geiger and Chickering in the now classical paper [33]. They state the problem of Bayesian learning of Bayesian belief networks from data, make explicit the assumptions underlying their approach and show how to calculate, for each network structure E the *score* $Pr[\mathcal{D}, E]$. The score is then used for model comparison. They also prove that optimizing the Bayesian score over networks structures is NP-hard when each node is allowed to have more than 1 parent. Therefore, both [9] and [33] use a local search: The search algorithms typically generate proposed additions or deletions of an edge from the graph, and these proposed moves are evaluated by computing (an approximation to) the marginal likelihood of the updated graph (exact calculations can be performed in some cases for graphs without hidden nodes). Local search is the most common approach to structure learning thus far. Methods based on non-Bayesian scores have been proposed by [47]. The cited works assume that data are complete and that there are no hidden variables. [62] develop a method for scoring equivalence classes of DAGs with hidden variables in the special case when the dependence relationships are jointly Gaussian, augmented by a heuristic search algorithm.

The third category of methods includes methods that also consider learning from data, but are explicitly aimed at finding dependencies. Some of the methods do not attempt to construct a complete graphical model, but to find as many dependencies or independencies as possible. [64] and [17] directly find dependencies in discrete databases by counting coincidences in subsamples. [34] constructs minimal Markov nets in a heuristic way. The first two methods are aiming at discovering structure in high-dimensional domains, where applying more sophisticated algorithms is computationally too expensive. This is a little explored area of research, but it is the area where the approach presented here most likely belongs.

[1] A star decomposable graph is a tree over the visible domain augmented with some hidden variables where each hidden variable has a degree at least 3.

The method to be developed in the forthcoming is fundamentally a method of learning from data. It belongs to the third category of the above taxonomy by the fact that it is mainly concerned with finding (a certain class of) dependency structures rather than a fully determined graphical model. But it relates to the second category by the fact that it generates multiple models and compares them by a score that will be further defined.

6.2 The problem of variable partitioning

The approach to structure learning presented in this thesis differs from all the methods mentioned above. Rather than adopting a local search heuristic based on single edge addition or deletion like the methods in the second category, or than looking for dependencies like the methods in the third category, it attempts to find *separators*.

A separator[2] is a variable or group of variables S that, when observed, separates $V \setminus S$ into two independent subsets A_1, A_2. If a separator exists and it is found, then learning the structure of the domain V reduces to learning the structures of $A_1 \bigcup S$ and $A_2 \bigcup S$. Since the number of possible structures grows superexponentially with n, by partitioning the domain the search space is reduced in a superexponential manner. Moreover, if more separators can be found in A_1, A_2, then one can proceed recursively, effectively implementing a divide-and-conquer structure learning algorithm. We call this approach *variable partitioning*.

If the separator S is included in V it is called a *visible* separatore. A visible separator of small size (practically 1 or 2 variables) can be found in a brute-force manner, by trying all possibilities. The number of such trials is $\mathcal{O}(n^{|S|}$ and each trial implies testing the conditional independence of two or more large sets, an exponential task by itself. As it will be shown later in this chapter (section 6.7.3), under certain assumptions the aforementioned independence tests can be reduced to tests on pairs of variables. This makes each trial quadratic in n.

This work focuses on a different situation: the case when the separator is a hidden variable. For this case the brute force approach is not sufficient and a different method needs to be devised. Much of the machinery developed for this case is applicable to visible separators; such that, in some sense, a visible separator is a simple case of hidden variable separator.

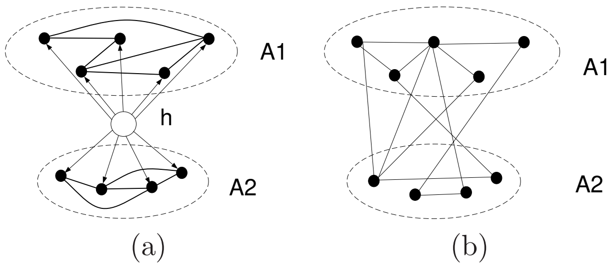
The hidden variable separator, or shortly hidden variable h, is assumed to partition V into several clusters of variables that are mutually independent conditioned on h. To maintain computational tractability, we assume a restrictive architecture for each cluster, in particular we assume that each cluster is a tree. We propose an algorithm whose goal it is to find a good set of trees and to find a good choice of tree structure within each cluster. Later we examine what happens when the true distribution of the data violates this restrictive assumption.

Let V denote as usual the set of variables of interest. Remember also that x_v is a particular value of V, x_A an assignment to the variables in the subset A of V and x is a shorthand for x_V. The variables in V will be referred to as being *visible*. We introduce an additional variable, a *hidden* (unobservable) variable denoted by h. This variable takes values ranging from 1 to m. The joint space $V \cup \{h\}$ is denoted by V^h. The following assumption about the joint distribution P over V^h is essential:

Assumption 1. The hidden variable h partitions V into p mutually disjoint subsets

[2] The term originates in the junction tree literature.

Figure 6-1: The H-model (a) and a graphical model of the same distribution marginalized over h (b).



$A_1, \ldots, A_p$, $\bigcup_j A_j = V$ that are conditionally independent given h. We write

$$A_j \perp A_{j'} \mid h \quad j, j' = 1, \ldots m \tag{6.1}$$

Consider the graphical model G corresponding to P (figure 6-1,a). Assumption 1 implies the graph topology $G = (V^h, E \cup E')$[3]. The edge set is the union of E', which contains a *directed* edge from the hidden variable h to each of the visible variables in V, and of E, which contains only undirected edges between pairs of visible variables. Moreover, the induced graph over V only has p connected components. Thus, in the graphical model language, conditional independence given h translates into h *separating* the respective variable clusters in G. We can view this model as a mixture of factorial models over the p hyper-variables $A_{1,\ldots p}$. Therefore, we call a model that satisfies Assumption 1 a *hyper mixture of factorials*, or in short an *H model*. H models are chain graphs.

Note that A_j being mutually independent given h does not imply that A_j are independent after marginalizing over the values of h. Graphically, if A_j are mutually separated (by h) in G, they are in general *not* separated in the graph $G^M = (V, E^M)$ corresponding to the marginal distribution of V

$$Q(x) \;=\; \sum_{k=1}^{m} P(x, h = k) \tag{6.2}$$

Let us compute the number of parameters of the models represented by P and Q under the simplifying assumption that $|v| = 2$ for all v, $p = 2$ and that P, $P_{A_j|h}$, $j = 1, 2$ are represented by probability tables. We have

$$\#pars(P) \;=\; m(2^{|A_1|} + 2^{|A_2|} - 2) \;\ll\; 2^n - 1 \;=\; \#pars(Q) \;\; \text{for } m \ll 2^n/(2^{|A_1|} + 2^{|A_2|}) \tag{6.3}$$

Hence, if the sizes of A_1 and A_2 are close, introducing a hidden variable can result in exponential savings in terms of the number of parameters.

We formulate the *variable partitioning task* as follows: given a domain $V^h = \{h\} \bigcup_j A_j$ such that $A_j \perp A_{j'} | h \;\forall j, j'$ and a set of observations $\mathcal{D} = \{x^1, \ldots, x^N\}$ from $V = \bigcup_j A_j$, find the sets A_j, the number of values of h, the distribution P_h and the distribution $\gamma^i = P(h|x^i)$ over the values of the hidden variable for each of the observed instances x^i in $\mathcal{D}$.

Of course, the final goal is to construct a model P of the domain; to do this, once variable partitioning is completed, one has two choices: either to use the model P that

[3]More precisely, a graph that is consistent with Assumption 1 is equivalent to a structure of type G

our algorithm learns, or to construct by some other method (including a recursive call to our variable partitioning algorithm) a probabilistic model for each of the domains $A_j \cup h$. The sizes of the domains being smaller than n, this task should be considerably easier than constructing a model over V directly.

However, one should be aware of the important difficulties that this quest has to face. Searching over the space of all possible partitions of V (of size $\sim \frac{p^n}{p!}$) is a forbidding task. It has been shown that learning the structure of belief networks from data even in the absence of hidden variables is an intractable problem [33]. Second, the H model is a mixture model and we also know that learning mixtures in certain instances is NP-hard [4]. The above reasons show that all one can do is to provide heuristic procedures for coping with this large-scale but important search problem.

There are theoretical difficulties as well: as it has been shown, graphical models with hidden variables do not belong to the curved exponential family and thus the BIC/MDL criterion is not a consistent estimate of the marginal likelihood of the model structure, as it is for exponential family models [29]. Computing the Bayesian posterior marginal probabilities of the model structure represents an equally intractable computation.

It is also generally known that models with hidden variables are subject to non-identifiability problems. A simple example will illustrate this: Suppose that the observed distribution over the two binary variables a, b is uniform and that we know that this distribution is the marginal of a mixture distribution

$$Q_{ab} = \sum_{k=1,2} P_h(k) P_{ab|h=k}$$

The above is an underdetermined system of equations in the unknowns P_h, $P_{ab|h}$. Two of the possible solutions are

$$P_h^{(1)}(k) = 0.5 \;\; k = 1,2; \quad P_{ab|0}^{(1)} = \begin{cases} 0 & a = b \\ 0.5 & a \neq b \end{cases} \quad P_{ab|1}^{(1)} = \begin{cases} 0.5 & a = b \\ 0 & a \neq b \end{cases}$$

and

$$P_h^{(2)}(k) = 0.5 \;\; k = 1,2; \quad P_{ab|0}^{(2)} = \begin{cases} 0 & a = 0 \\ 0.5 & a = 1 \end{cases} \quad P_{ab|1}^{(2)} = \begin{cases} 0.5 & a = 0 \\ 0 & a = 1 \end{cases}$$

Both of the above distributions have a uniform marginal over a, b and so does any convex combination of $P^{(1)}, P^{(2)}$. Hence there is an infinity of mixture models that have the same marginal over the observed variables.

Non-identifiability can cause severe convergence problems in learning the model. For our task identifiability is even more important, since in its absence we are likely to find several hidden variable solutions (potentially an infinity!) that explain the data equally well. Identifiability of hidden variable models is an area of current research [29, 59, 63]. The existent results are only preliminaries to a general theory. However, in the case of models with binary variables only, [63] gives the following sufficient condition that is highly relevant to the present problem.

Theorem [63] An H-model over a domain V^h consisting of binary variables only is identifiable if either (a) or (b) hold:

(a) $p \geq 3$

(b) $p = 2$, $V = A_1 \bigcup A_2$ and $P_{A_1|h}, P_{A_2|h}$ are not fully connected graphical models.

6.3 The tree H model

In the following we present an algorithm for variable partitioning derived under additional assumptions about the model structure. We shall also assume for now that m and p are known.

We define a *tree H model* over $V^h = \{h\} \bigcup \left(\bigcup_{j=1}^{p} A_j \right)$ as an H model for which $P_{A_j|h=k} \equiv T^k_{A_j}$ are trees for all j, k. The structure of $T^k_{A_j}$ is fixed for all k, but its parameters may vary with k. We denote the structure of each connected component[4] j by E_j. Their union is the edge set $E = \bigcup_j E_j$. Because there are p distinct connected components we have

$$|E| \;=\; n - p \tag{6.4}$$

The conditional distribution

$$P_{V|h}(x|k) \;\equiv\; T^k(x) \;=\; \prod_j T^k_{A_j}(x_{A_j})$$

can be viewed either as the product of p independent trees over the A_js or as one tree T^k over V having p connected components. This remark enables us: (1) to avoid explicitly specifying p from now on by using the notation T^k and (2) to view the H as a special case of mixture of trees with shared structure. This allows us to use the efficient algorithms devised for mixtures of trees for fitting tree H models.

A tree H model is obviously a mixture of trees with shared structure; therefore, learning it can be done by a variant of the MixTreeS learning algorithm. The derivation of the algorithm closely parallels the one given in section 3.3.3, so only the final description thereof is be included here.

Algorithm **HMixTreeS**

Input: Dataset $\mathcal{D} = \{x^1, \ldots x^N\}$
    Procedure Kruskal(p, weights) that fits a maximum weight tree with $n - p$ edges over V
    p a minimum number of connected components
    Initial model $\mathcal{M}_0$
Initialize EM with $\mathcal{M} = \{E, \lambda_k, T^k,\ k = 1, \ldots m\}$
Iterate until convergence
    E step: compute $\gamma^i_k,\ P^k(x^i)$ for $k = 1, \ldots m,\ i = 1, \ldots N$
    M step:
        M1. $\lambda_k \leftarrow \Gamma_k / N,\ k = 1, \ldots m$
        M2. compute marginals $P^k_v,\ P^k_{uv},\ u, v \in V,\ k = 1, \ldots m$
        M3. compute mutual information $I^k_{uv} u, v \in V,\ k = 1, \ldots m$
        M4. call Kruskal($p, \{\ I_{uv|h}\}$) to generate E
        M5. $T^k_{uv} \leftarrow P^k_{uv},\ ; T^k_v \leftarrow P^k_v$ for $(u, v) \in E$
Output $\mathcal{M}$

As one can see, the only change w.r.t the standard MixTreeS algorithm is in step **M4** where we need to construct the maximum weight tree with the prescribed maximum number of edges. Kruskal's algorithm fits this purpose easily, since it adds the edges in the order of

[4]The reader should be aware of the distinction between *mixture components* indexed by the values of h and *connected components* which are subtrees of each tree mixture component and are indexed by $j = 1, \ldots p$.

their decreasing weight. It can be proved[5] that stopping Kruskal's algorithm after it adds $n - p$ edges results in the maximum weight spanning tree with (at most) that number of edges.

6.4 Variable partitioning in the general case

6.4.1 Outline of the procedure

We have presented a tractable algorithm for learning a special case of H models, the tree H models. Our next step is to apply the same algorithm to data that are not known to be generated by a tree H model. Hence, we need to proceed with caution and to use model selection and validation criteria to evaluate the output of the algorithm.

The procedure I used can be outlined as a two stage process: The first stage generates models with different values for m and p and different structures and parameters by repeatedly calling the **HMixTreeS** algorithm. The second stage validates the models produced in the first stage by testing that the independencies they imply are not contradicted by the data and computes a score for each (valid) model. The different models can be then compared based on this score. The method is summarized as follows:

Algorithm **H-learn** – outline

Input: Dataset $\mathcal{D} = \{x^1, \ldots x^N\}$
Confidence level δ
Parameters $m_{MAX}, p_{MAX}, N^{trials}$
Procedure **HMixTreeS**
for $m = 2, 3, \ldots, m_{MAX}$
for $p = 2, 3, \ldots p_{MAX}$
for $t = 1, \ldots, N^{trials}$
Initialize randomly $\mathcal{M}_0$
$\mathcal{M}_{m,p,t}$=**HMixTreeS**$(\mathcal{D}, \mathcal{M}_0, p)$
Validate $\mathcal{M}_{m,p,t}$ with confidence δ
if $\mathcal{M}_{m,p,t}$ valid
Compute $score(\mathcal{M}_{m,p,t})$
Output $\mathcal{M} = \underset{\mathcal{M}_{m,p,t}\text{valid}}{\text{argmin}}\ score(\mathcal{M}_{m,p,t})$ or the l models with best score

The rest of the chapter studies and develops this approach. For two of the subtasks involved - evaluating the description length of a model and testing independence - detailed descriptions are given in sections 6.6 and 6.7 respectively.

6.4.2 Defining structure as simple explanation

When we set our goal to be learning about a real domain that has inherently unobserved variables, having little or no prior knowledge about it, how can we know when we have discovered the "right" interactions between variables ? We shall *define* a good structure in this case as a structure that is both simple and likely in view of the data, simplicity being measured by the description length of the model. The reader may have realized already

[5]By a proof similar to the proof of Theorem 8.2.18. in [71].

that we what we have stated is a version of the well known Minimum Description Length (MDL) principle [57].

$$\mathcal{M}^* \;=\; \underset{\mathcal{M}}{\text{argmin}}\; [\overbrace{DL(\mathcal{M}) \;+\; DL(\mathcal{D}|\mathcal{M})}^{DL(\mathcal{D},\mathcal{M})}] \tag{6.5}$$

The last term in (6.5) is $-\log Pr[\mathcal{D}|\mathcal{M}]$ the negative base 2 log-likelihood. The first term can be decomposed as

$$DL(\mathcal{M}) \;=\; \text{bits}(m) + \text{bits}(\text{structure}) + \sum_{k=1}^{m} \text{bits}(\text{parameters}\,|\,\text{structure}) \tag{6.6}$$

Note that any tree structure requires the same number of bits[6], independently of p (1 for each variable, specifying its parent or ϕ). The first term is also constant over all models[7]. Hence, computing the description length amounts to computing the description length of the parameters for the distributions T^k. However, this is not a straightforward task.

We can approach it in two ways. The first is to use the approximation to the model description length derived by Rissanen [57] and also known as the Bayes Information Criterion (BIC). It assumes $\frac{1}{2}n_{pars}\log N$ as the description length of the parameters. According to BIC, each edge $(uv) \in E_T$ contributes

$$\Delta_{uv}^{BIC} \;=\; \frac{m}{2}(r_u - 1)(r_v - 1)\log N \tag{6.7}$$

bits in addition to the description of a "tree" with no edges that we take as bottom line. The minimization of the DL over tree structures can be automatically incorporated into the EM algorithm, by assigning the weigths

$$W_{uv} = NI_{uv|h} - \Delta_{uv}^{BIC}. \tag{6.8}$$

The method has the advantage of indirectly controling the number of connected components p. A drawback of equation (6.8) is that it penalizes parameters in each component equally. To see why this is a problem, imagine that there is one component that is responsible for no data at all; the parameters of this component will have no influence on the data likelihood but the model will still be penalized for them. So, it would be sensible to replace m in the above formula with something representing an "effective number of parameters".

A heuristic remedy that I tried for this problem is to take into account the fact that parameters are estimated from different amounts of data for each k. Hence, I considered that each edge $(uv) \in E_T$ contributes to the total penalty by

$$\Delta_{uv}^{BICh} \;=\; \frac{1}{2}(r_u - 1)(r_v - 1)\sum_{k=1}^{m} \max(\log \Gamma_k, 0) \tag{6.9}$$

which gives the $BICh$ criterion.

A second approach is to directly approximate the DL of a model. To achieve this, we represent the probability values on a number of bits b (smaller than the machine precision) obtaining an approximate model $\mathcal{M}_b$. For each b in a chosen range, $DL(\mathcal{M}_b)$ is easily

[6] Unless we have a nonuniform prior over tree structures.

[7] If m is small,as it is implicitly assumed throughout the paper and if there is no prior on m. Otherwise, the first term will vary with m.

computed and so is $Pr[\mathcal{D}|\mathcal{M}]$. Then we approximate the true DL by $\operatorname{argmin}_b DL(\mathcal{D}, \mathcal{M}_b)$. A full description of this procedure will be included in section 6.6. This method being a purely evaluatory method, it can be used even for models learned using the penalty from equations (6.8) or (6.9) above.

Number of connected components p As a consequence of the weight modification (6.8), the weights can now be negative. Kruskal's algorithm will stop adding edges when it first encounters a weight ≤ 0, so this implicitly controls p. Moreover, the penalty will be likely to increase with m thus favoring sparser structures. Alternatively, one can make p vary over a previously fixed range (e.g. between 2 and n) and compare all the resulting models by their description lengths.

Selecting m. The description length DL provides us with a consistent criterion for comparing between model structures with different numbers of trees, thus enabling us to select m.

Local optima We deal with this problem in the usual way: by restarting the EM algorithm several times, each time from different random points in the model space. Consequently, for each new run of the EM algorithm we may end up with a different model structure (and parameters) but this is a help inasmuch as it is a problem, since the role of the EM algorithm is to present us with good, but also diverse candidate structures.

6.5 Experiments

6.5.1 Experimental procedure

The experiments described in the following aim to assess the ability of the **HMixTreeS** algorithm to discover the correct structure and the hidden variable in two cases: when the data are generated by a mixture of trees and when they are not. In both cases, we will also evaluate the description length and the alternative scores discussed above as model selection criteria for this task.

The experiments therefore were run on artificial data generated in two steps. First, an H-model structure is created (figures 6-2 and 6-3 show the structures used here). The domain has 12 visible variables and one hidden variable, all taking 3 values. Second, for each structure, 10 sets of random parameters were sampled; each resulting model was used to generate a data set of 3,000 examples, of which 2,000 were used for training and 1,000 were used for testing. The values of the hidden variable were recorded but served only for testing purposes. The first structure (figure 6-2) is a tree H model, the other two (figure 6-3) are general H models. In the latter case, because pilot experiments have shown that discovering the variable partitioning is relatively easy when the true partitioning is visible in the marginal distribution of the data, all the data sets were checked for this property and accepted only if the true structure could not be recovered by fitting a single tree.

6.5.2 Experiments with tree H models

This group of experiments demonstrates that the **HMixTreeS** algorithm is capable of recovering structure when the generative model is a tree H model. The model structure is shown in figure 6-2. Two random parametrizations for this structure were generated and from each them a training set ($N = 3,000$) and a test set ($N_{test} = 1000$).

For each training set, models with $m = 2, 3, 4$ were learned using the MDL edge penalty to choose the number of connected components p. For comparison the case $m = 3,\ p =$

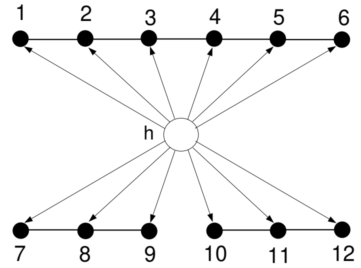


Figure 6-2: The tree H model used for training.

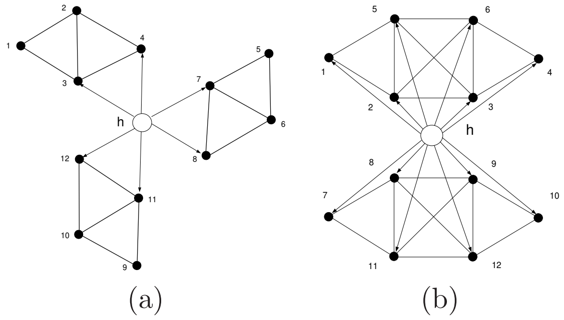


Figure 6-3: The models used for training: (a) S1, (b) S2.

4, $\beta = 0$ was tried too. For each set of parameters, the algorithm is run 20 times with different random initializations.

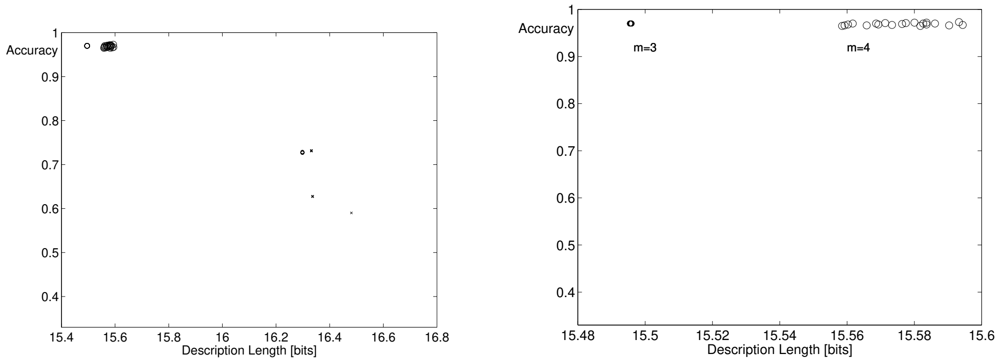
The results for one of the data sets are displayed in figure 6-4. From the figure we can see that the correct structure $p = 3$ is found in all 20 cases corresponding to $m = 3$ and MDL edge penalty. The high accuracy means that the values of the hidden variable are correctly estimated. It can also be seen that for $m = 2$ and 4 and for $p = 4$, the description length is higher than for the true model $m = 3, p = 3$. Hence, the number of values of the hidden variable is correctly selected too. The results for the second data set are similar, except that there the correct structure is found in only 19 out of 20 trials.

Another example of successful structure learning and hidden variable discovery is the Bars learning task that will be presented in section 7.1.2.

6.5.3 General H models

The following experiments were run on 10 data sets generated from structure S1. For every parameter setting, the learning algorithm was run 20 times from different initial points. The initial points were chosen randomly without including any knowledge about the desired structure. Unless otherwise stated, the empirical description length DL was used as model selection criterion. We diverge a little from the algorithm outlined previously by leaving model validation last, after examining the properties of all the resulting models.

The first series of experiments were run on models with $m = 3$ and fixed $p = 3$ (the true values). As figure 6-5 shows, in 8 out of 10 trials the correct structure prevailed both in terms of the number of times it appeared and in terms of its description length. For these 8 cases, according to the DL criterion, the best model had the correct structure and in many



(a) (b)

Figure 6-4: Description lengths and accuracies for models with fixed $p = 2$ and variable $m = 2, 3, 4$ learned from data generated by the tree H model in figure 6-2. For $m = 3$ and 4 all structures are correct. (a) general, (b) detail of the upper left corner of (a).

cases all five best models were correct. The accuracy of the hidden variable retrieval is also relatively high (the baseline for accuracy is 0.33), but not as high. For the remaining two data sets (c and j in figure 6-5) the correct structure is present but it overtaken in score by other structures (however, at least one of the best 5 models has the correct structure).

I have run a series of similar experiments with $m = 3$ but applying the MDL edge penalty to set p. In this situation, the learning algorithm consistently overestimated p by 1, but otherwise found a structure compatible with the correct one. Therefore, I searched the space of p values in a manner similar to the search for m. For this purpose, models with $m = 3, 4$ and $p = 2, 4$ were trained. The results were not encouraging: For $p = 2$ both $m = 3$ and $m = 4$ produced a significant number of models with shorter description length. This suggests the presence of residual dependences between components that are taken up by the additional link (especially in the case $m = 3$), something to be investigated in the model validation phase. But, as a conclusion so far, estimating p appears to be a hard task.

The next set of experiments studies the effect of varying m with p fixed at the true value 3. H models with $m = 2, 4, 5, 6$ were trained on all the data sets. Figure 6-6 summarizes the results for set d. One sees that the models for $m = 2$ cluster separately in a region of higher description lengths. For $m = 4, 5, 6$ three things happen: the proportion of good structures is increased (for all data sets); the description length of the models decreases until $m = 5$ and remains about the same for $m = 6$ (all but 1 case, where it settles at $m = 4$); the hidden variable accuracy remains stationary (all cases). Moreover, in all cases, the minimum DL model has the best structure (but $m > 3$) and except for data set c, all 5 best models are correct. The explanation for this behavior is that, since the true mixture components are not tree distributions, adding more components to the mixture of trees helps model the true distribution more closely: some of the original three mixture components are allocated more than one tree. The explanation is sustained by examining the confusion matrices: for each of the m trees the data it represents comes mostly from one component of the original mixture.

Here is a confusion matrix for data set d, $m = 5$. The columns are the trees, the rows are the original components, and the matrix element i, j represents the proportion of data

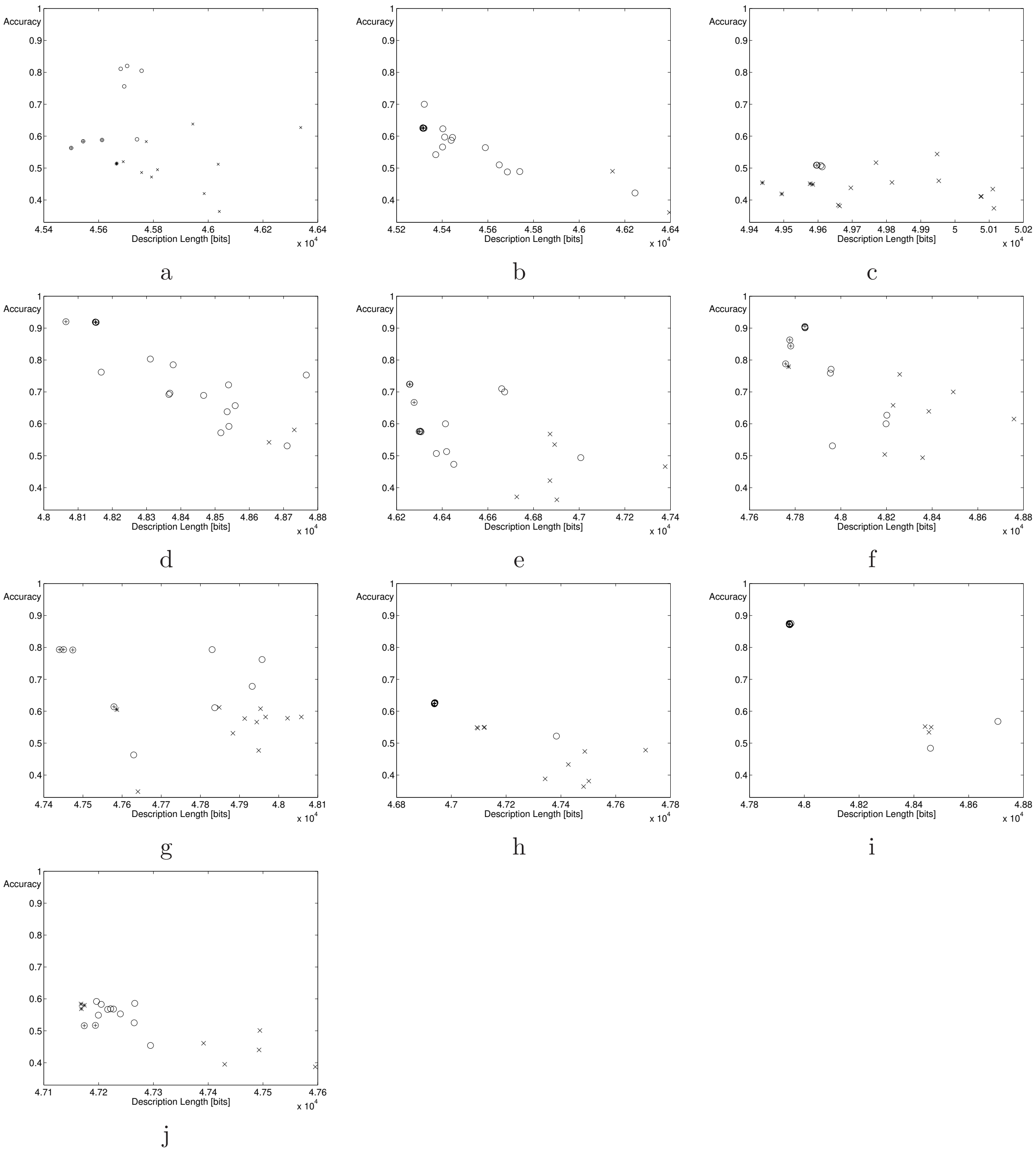


Figure 6-5: Model structures and their scores and accuracies, as obtained by learning tree H models with $m = 3$ and $p = 3$ on 10 data sets generated from structure S1. Circles represent good structures and x's are the bad structures. A + marks each of the 5 lowest DL models. The x-axes measure the empirical description length in bits/example, the vertical axes measure the accuracy of the hidden variable retrieval, whose bottom line is at 0.33

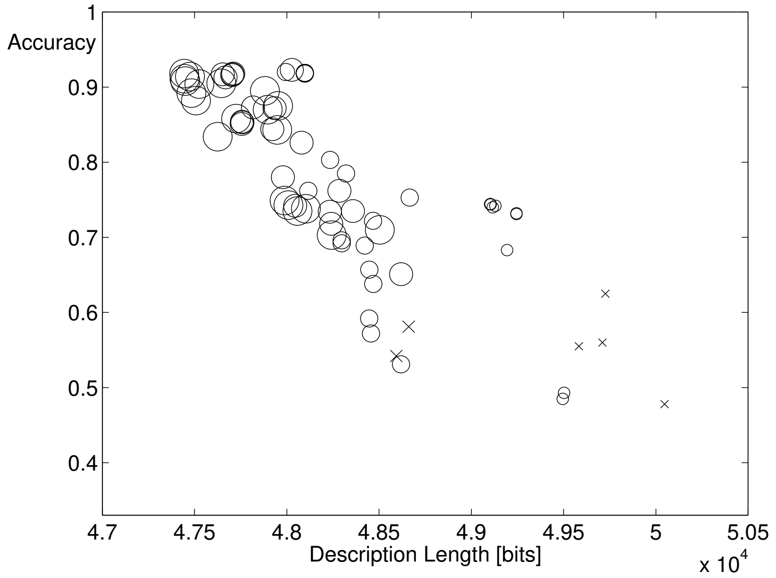


Figure 6-6: Model structures and their scores and accuracies, as obtained by learning tree H models with $p = 3$ and variable $m = 2, 3, 4, 5$ on a data set generated from structure S1. Circles represent good structures and x's are the bad structures. The sizes of the symbols are proportional to m. The x-axis measures the empirical description length in bits/example, the vertical axis measures the accuracy of the hidden variable retrieval, whose bottom line is at 0.33. Notice the decrease in DL with increasing m.

from component i modeled by tree j.

```
      0     1     2     3     4
0: 0.504 0.391 0.063 0.018 0.025
1: 0.010 0.018 0.659 0.012 0.301
2: 0.050 0.023 0.045 0.837 0.045

Cluster Accuracy = 0.914
```

The experiments show that choosing m solely by minimizing the description length can lead to an overestimation of the number of components. Based on the gap in the DL dimension between the models with $m = 2$ and the rest of the models (all data sets except c) one could study the possibility of using it as a criterion for selecting m.

Comparison between the different model section criteria The above results were plotted again (not shown), replacing in turn the BIC, the modified BIC and the test set log-likelihood as the horizontal axis, in order to see if there are differences in selecting the correct structure. We found that the test log-likelihood fails to select the best model in 3 cases when DL does, however we think that this is not sufficient evidence that the log-likelihood is less good than the description length. The other differences seemed also to favor the empirical description length, but in an even less significant way.

The results obtained for the second structure are qualitatively similar.

6.6 Approximating the description length of a model

Here is described the method used to approximate the DL of a mixture of trees model. It is a general enough method to be used with any probability model, graphical or not. The idea is to encode the model $\mathcal{M}$ as best we can on a fixed number of bits B and then to compute the description length of the data under this approximated model called $\mathcal{M}_B$. If B is smaller than the length of the original representation of $\mathcal{M}$ and if $\mathcal{M}$ is a local maximum of the likelihood, then the term $DL(\mathcal{D}\,|\,\mathcal{M}_B)$ will be larger then the original $DL(\mathcal{D}\,|\,\mathcal{M})$; however, the model DL will be equal to B and thus smaller than the original $DL(\mathcal{M})$. By varying B over a reasonable range and minimizing the total description length we obtain

$$DL^{emp} \;=\; \operatorname*{argmin}_B DL(\mathcal{D}, \mathcal{M}_B), \tag{6.10}$$

an upper bound to the local minimum of the desired description length.

The better the encoding for each B (in the sense of preserving the likelihood of each observed data point), the closer is the above minimization to the true $DL(\mathcal{D}, \mathcal{M})$. In practice, we realize a tradeoff between encoding quality and computational cost. We want cheap encoding schemes that preserve the represented distribution reasonably well.

To evaluate the DL of a mixture of trees, we use the above idea only to approximate the DL of the tree parameters within the mixture's DL. We assume that the structure description takes the same number of bits for any tree (n pointers, one – possibly null – for the parent of each node) and thus can be ignored from the DL. The DL of the λ parameters is approximated by

$$b_\lambda \;=\; \frac{m-1}{2}\log N.$$

The parameters of the tree distributions are encoded with the same number of bits b. This number is a sum of two precisions, b_m and b_M, that will be defined below. The number of parameters is given by equation (2.7). Thus, the total number of bits (excluding the constant for structure) is

$$DL(Q) \;=\; b_\lambda + m(b_m + b_M)[\sum_{(u,v)\in E}(r_u-1)(r_v-1) + \sum_{v\in V} r_v - n] + \sum_{(u,v)\in E} r_u \log r_v \tag{6.11}$$

Now we show how to encode the parameters of the trees. In our directed tree based implementation, the parameters represent the values of the conditional probabilities $T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$ (where, of course, sometimes $\mathrm{pa}(v)$ is the empty set). For each fixed value of $\mathrm{pa}(v)$, this distribution is a multinomial. Therefore, in the following we describe a way of encoding multinomial distribution.

6.6.1 Encoding a multinomial distribution

Encoding a set of real numbers on a finite number of bits always involves some sort of quantization. So, to encode the parameters of a distribution one needs to first quantize them. The quantization of the multinomial distribution parameters is done in the natural parametrization introduced in section 4.2.2.

Let the distribution values be $\theta_j,\ j = 1\ldots r$ and let j_{MAX} be the index of the largest θ. The natural parameters are given by (4.27), reproduced here in slightly modified form

$$\phi_i \;=\; -\frac{\log\theta_i}{\log\theta_r} \quad i = 1,\ldots r,\ i \neq j_{MAX}. \tag{6.12}$$

Adding a minus sign in front of the logarithm function ensures that all the ϕ parameters are positive. Since the θ parameters are allowed to be 0, their range is in $[0, \infty]$.

We are going to represent the ϕ values on a finite number of bits. This is done in two stages: First, the large ϕ values are "truncated" to the constant 2^{b_M} if they exceed it. This corresponds to bounding the smallest θ values away from 0. Then, the resulting values are represented as b_m bit precision binary numbers, or equivalently, they are multiplied by 2^{b_m} and stored as integers. Note that while b_M controls the representation precision for small probabilities, b_m control the precision of large probabilities. Both precisions are relative to the largest value of θ. The full encoding of $\theta_{1\ldots r}$ is represented by r, the encoding precision b_m, the position of the maximum j_{MAX} and the values

$$\phi_i^{enc} \;=\; 2^{b_m} \min(\phi_i, 2^{b_M}), \quad i = 1, \ldots r, \; i \neq j_{MAX} \tag{6.13}$$

To decode the values of θ from the above representation one has to apply formula (4.28) to $\phi_i^{enc} 2^{b_m}$ and replace the maximum θ in position j_{MAX}. Note that the value b_M is not necessary for decoding.

If the value of r is given by a separate table (remember that we are encoding several distributions over the same domain) and that b_m is also encoded separately, the total number of bits to encode $\theta_{1,\ldots r}$ is

$$\log\lceil r \rceil + (b_m + b_M)(r-1) \;\approx\; \log r + (b_m + b_M)(r-1) \tag{6.14}$$

By adding up the right-hand sides of (6.14) values for all $T_{v|\mathrm{pa}(v)}$ one obtains the last two terms of (6.11). To see this remark that: (1) each free parameter is encoded on $b_M + b_m$ bits and (2) $r_u \log r_v = r_v \log r_u$ for all positive numbers r_u, r_v. To make the search tractable, the values of b_m and b_M are common to all distributions in Q and thus b_m is recorded only once.

6.7 Model validation by independence testing

6.7.1 An alternate independence test

Assume that we have two independent discrete variables u and v taking r_u, r_v values respectively, and an i.i.d. sample of size N drawn from their joint distribution. Since the variables are independent, their mutual information is obviously zero, but its estimate from the sample $\hat{I}_{uv}$ does not generally exactly equal zero. In this section we want to bound the probability that the *sample mutual information* $\hat{I}_{uv}$ is larger than a threshold $M > 0$.

Of course, $\hat{I}_{uv}$ will never exceed $\max(\log r_u, \log r_v)$ so it will be assumed that M is below this value. We shall use large deviation theory, and in particular Sanov's theorem [11] to obtain this bound.

Let us denote by $\hat{P}_{uv}$, $\hat{P}_u$, $\hat{P}_v$ respectively the sample joint distribution of u, v and its marginals. We shall consider the following two sets of probability distributions over the domain $\Omega(uv)$

$$E \;=\; \{P \mid I_{uv}^P \geq M\} \tag{6.15}$$

$$F \;=\; \{Q \mid I_{uv}^Q = 0\} \tag{6.16}$$

The set F represents all the factorized distributions over $\Omega(uv)$. With this notation our problem can be formulated as: what is the probability of the sample distribution $\hat{P}$ to belong to E given that the true distribution is in F?

Sanov's theorem gives us the following result: For any distribution Q and any closed set of probability distributions E, the probability that the sample distribution $\hat{P}$ of an i.i.d. sample of size N drawn from Q is in E is denoted by $Q^N(E)$ and satisfies

$$Q^N(E) \;\leq\; (N+1)^r 2^{-N.KL(P^*_Q\,||\,Q)} \tag{6.17}$$

where r is the size of Q's domain and

$$P^*_Q \;=\; \operatorname*{argmin}_{P\in E} KL(P\,||\,Q) \tag{6.18}$$

is the distribution in E that is closest to Q in KL-divergence[8]. We also know [72] that for any distribution P over $\Omega(uv)$

$$\min_{Q\in F} KL(P\,||\,Q) \;=\; KL(P\,||\,P_uP_v) \;=\; I^P_{uv}. \tag{6.19}$$

Hence

$$\min_{P\in E, Q\in F} KL(P\,||\,Q) \;=\; \min_{P\in E} I^P_{uv} \;=\; M \tag{6.20}$$

Using this result in Sanov's theorem above we conclude that for any $Q \in F$

$$Q^N(E) \;\leq\; (N+1)^{r_u r_v} 2^{-NM} \tag{6.21}$$

In other words, for any two independent variables u, v, the probability that their sample mutual information $\hat{I}_{uv}$ exceeds a threshold $M > 0$ is bounded by

$$(N+1)^{r_u r_v} 2^{-NM} \;=\; \delta.$$

Conversely, given a probability δ we can derive the corresponding M:

$$M \;=\; \frac{1}{N}[\log\frac{1}{\delta} \,+\, r_u r_v \log(N+1)] \tag{6.22}$$

This bound is distribution free: it requires no assumption about the distributions of u and v. As expected, the value of M for fixed δ decreases with the sample size N. Contrast this with the χ^2 test, that *does* make assumptions about the underlying distribution but whose p-values (the analogues of M) are independent of the sample size.

Because E is the closure of its interior, Sanov's theorem and a reasoning similar to the above one enables us to establish a lower bound for $Q^N(E)$

$$Q^N(E) \;\geq\; \frac{1}{(N+1)^{r_u r_v}} 2^{-NM} \tag{6.23}$$

It is interesting to compare the test devised here, call it *Sanov* test with the well-known χ^2 test for independence. It is easy to show that the χ^2 statistic is approximating a KL divergence. For this, let us compute the quantity $KL(\hat{P}_u\hat{P}_v\,||\,\hat{P}_{uv})$ as a function of the counts N_{ij} =#times $u=i$, $v=j$, N_i =#times $u=i$, N_j =#times $v=j$ and

$$n_{ij} \;=\; \frac{N_i N_j}{N}, \quad i = 1,\ldots r_u,\; j = 1,\ldots r_v.$$

[8] A somewhat tighter bound for the r.h.s of equation (6.17) is $[r + N^{r-1}]2^{-N.KL(P^*_Q\,||\,Q)}$.

$$
\begin{aligned}
KL(\hat{P}_u\hat{P}_v \,||\, \hat{P}_{uv}) &= \sum_{ij} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N_{ij}} && (6.24)\\
&= -\frac{1}{N\ln 2} \sum_{ij} n_{ij} \ln \frac{N_{ij}}{n_{ij}} && (6.25)\\
&= -\frac{1}{N\ln 2} \sum_{ij} N_{ij} \ln(1 + \frac{N_{ij}-n_{ij}}{n_{ij}}) && (6.26)\\
&\approx -\frac{1}{N\ln 2} \sum_{ij} [n_{ij} \frac{N_{ij}-n_{ij}}{n_{ij}} - \frac{1}{2} n_{ij} \frac{(N_{ij}-n_{ij})^2}{n_{ij}^2} + \ldots] && (6.27)\\
&= \frac{1}{N\ln 2} \frac{1}{2} \sum_{ij} \frac{(N_{ij}-n_{ij})^2}{n_{ij}} && (6.28)\\
&= \frac{1}{2\ln 2} \chi^2 && (6.29)
\end{aligned}
$$

Hence, whereas the Sanov test is based on $KL(\hat{P}_{uv} \,||\, \hat{P}_u\hat{P}_v)$, the χ^2 independence test is an approximation of the reverse divergence $KL(\hat{P}_u\hat{P}_v \,||\, \hat{P}_{uv})$. Of course, the motivation for the χ^2 test is not in this divergence but in Central Limit Theorem arguments.

6.7.2 A threshold for mixtures

Now we undertake to estimate an approximate confidence interval for the sample mutual information when the variables u and v are independent given a third variable z. In this case the true joint distribution of u, v is

$$
Q = \sum_{k=1}^{m} \lambda_k Q_u^k Q_v^k \qquad (6.30)
$$

where, as usual, the values λ_k represent the probabilities of variable z being in state $k \in \{1, \ldots m\}$. Again, we assume an i.i.d. sample of size N; given this sample, we group the data points by the values of z in m groups of size N_k respectively. Therefore, we have

$$
\sum_{k=1}^{m} N_k = N.
$$

We denote by $\hat{I}_{uv}^k$ the sample mutual information between u and v in group k. Let us further denote by $\rho(N, M)$ the probability that the sample mutual information from a size N sample does not exceed $M > 0$.

$$
\rho(N, M) = 1 - Q^N(E) \qquad (6.31)
$$

Of course, ρ is also a function of r_u, r_v and the true distribution Q. To simplify the notation we drop these variables. This is acceptable because r_u and r_v are fixed and because the dependence on Q will be masked by the use of bounds that hold for any Q.

Our goal is now to fix a confidence level δ and to find the thresholds M_k, $k = 1, \ldots m$ depending on δ such that $\hat{I}_{uv}^k < M_k$ for all k with probability δ. We start by expressing the probability of the event $\mathcal{E}$="$\hat{I}_{uv}^k < M_k$ for all k" as a function of the values M_k and after we arrive at a convenient expression we will equate that with δ.

Let us first introduce some notation: let Z be the i.i.d. sample of size N representing the values of z, and $\underline{N} = (N_1, \ldots N_m)$ an m-tuple of N_k values as defined above. By

$$\binom{N}{\underline{N}} = \frac{N!}{N_1! N_2! \ldots N_m!} \tag{6.32}$$

we denote "N choose $N_1, \ldots N_m$", the multinomial coefficient indexed by $\underline{N} = (N_1, \ldots N_m)$. Then,

$$\begin{aligned}
Pr[\mathcal{E}] &= \sum_Z Pr[Z] \prod_k Pr[\hat{I}^k_{uv} < M_k \mid Z] \\
&= \sum_Z \prod_k \lambda_k^{N_k(Z)} \rho(N_k(Z), M_k) \\
&= \sum_{\underline{N}} \binom{N}{\underline{N}} \prod_k \lambda_k^{N_k} \prod_k \rho(N_k, M_k)
\end{aligned}$$

If the factors $\rho(N, M)$ vary slowly with N (further on we show when this happens), then the above sum is dominated by the term corresponding to the largest $\binom{N}{\underline{N}} \prod_k \lambda_k^{N_k}$, a value attained for

$$N_k^* \approx \lambda_k N, \quad k = 1, \ldots m. \tag{6.33}$$

Moreover, using Stirling's approximation $N! \approx \left(\frac{N}{e}\right)^N$ we have

$$\begin{aligned}
\binom{N}{\underline{N}^*} \prod_k \lambda_k^{N_k^*} &= \frac{N!}{\prod_k (\lambda_k N)!} \prod_k \lambda_k^{\lambda_k N} \\
&= \frac{N!}{N^N} \prod_k \frac{(\lambda_k N)^{\lambda_k N}}{(\lambda_k N)!} \\
&\approx e^{-N + \sum_k N \lambda_k} \\
&= 1
\end{aligned}$$

This allows us to write

$$Pr[\mathcal{E}] \approx \prod_{k=1}^m \rho(N\lambda_k, M_k) \tag{6.34}$$

If we now choose M_k according to (6.22) i.e.

$$M_k = \frac{1}{\lambda_k N} [\log \frac{1}{\tilde{\delta}} + r_u r_v \log(\lambda_k N + 1)] \tag{6.35}$$

we have

$$1 - \tilde{\delta} \le \rho(\lambda_k N, M_k) \le 1 - \frac{\tilde{\delta}}{(\lambda_k N + 1)^{r_u r_v}} \tag{6.36}$$

and

$$Pr[\mathcal{E}] \approx (1 - \tilde{\delta})^m = 1 - \delta \tag{6.37}$$

For a given δ the corresponding $\tilde{\delta}$ is

$$\tilde{\delta} = 1 - (1 - \delta)^{\frac{1}{m}} \approx \frac{\delta}{m} \tag{6.38}$$

To conclude, we have to prove our assumption that the functions $\rho(N_k, M_k)$ vary slowly with the first argument. More precisely, we are interested in upper bounding them. But

$$\frac{\rho(N_k, M_k)}{\rho(N_k^*, M_k)} \leq \frac{1}{\rho(N_k^*, M_k)} \leq \frac{1}{1-\tilde{\delta}} \tag{6.39}$$

For small values of δ the above quantity is close to unity validating our previous derivation.

6.7.3 Validating graphical models with hidden variables

In this section we will apply the previous test to the models of interest to us: graphical models with a hidden variable that separates the observed ones into two conditionally independent subsets. Thus, we will assume that we have a set of variables V, a dataset $\mathcal{D} = \{x^1, \dots x^N\}$ of observations from V and the model

$$Q(x) = \sum_{k=1}^{m} \lambda_k T_A^k(x_A) T_B^k(x_B) \tag{6.40}$$

where A, B is a partition of V. Thus Q assumes that there is a hidden variable z taking values $k = 1, \dots m$ with probabilities λ_k and that $A \perp B \mid z$. We want to test whether the data $\mathcal{D}$ supports the independence implied by Q. In the following we will be referring to the Sanov test developed in section 6.7.1, but arguments similar to those presented here hold for the χ^2 independence test.

There are two issues that need to be addressed: First, in this case the variable z is not observed. Therefore, we use Monte Carlo sampling from the posterior $\gamma_k(i)$ to obtain sequences $Z = (z^1, \dots z^N)$. For each of them we compute $\hat{I}_{uv}^k$, $k = 1, \dots m$ and compare them to the respective thresholds.

Second, establishing independence between two sets of variables implies a summation over $\Omega(A) \times \Omega(B) = \Omega(V)$. This configuration space is usually much larger than the size N of the available sample. With $r_u r_v = |\Omega(V)|$ the threshold M becomes giant (6.22) and the comparison meaningless. And, if N becomes comparable to $\Omega(V)$ then the calculations become intractable. Therefore, we will test the independence of A and B by performing independence tests on pairs of variables $u \in A$, $v \in B$. In general it is **not true** that

$$u \perp v \mid z \;\; \forall u \in A,\, v \in B \quad \Longrightarrow \quad A \perp B \mid z. \tag{6.41}$$

But (6.41) is true for a broad class of distributions. In particular, if the *true* distribution of the data is a chain graph (6.41) holds. Now we have to test the independence of $|A||B|$ pairs of variables, something that is computationally intensive, but tractable. However, we will be performing simultaneous tests and we have to take this into account.

We use therefore a Bonferroni inequality [42] that states that if each hypothesis $\mathcal{H}_{uv}$ is true with confidence $1 - \epsilon_{uv}$, then their conjunction is true with confidence at least

$$1 - \sum_{uv} \epsilon_{uv}.$$

In the present case, $\mathcal{H}_{uv}$ represent the hypotheses "$u \perp v \mid z$"; by $\mathcal{E}_{uv}$ we denote the events "$\hat{I}_{uv}^k(Z, \mathcal{D}) < M_k \, \forall k$". There is no special reason to treat any of them differently; hence

$$\epsilon_{uv} = \frac{\epsilon}{|A||B|} = \tilde{\epsilon} \quad \forall u \in A,\; v \in B \tag{6.42}$$

The test procedure can be summarized as follows:

Algorithm **TestIndependence**

Input model Q, data set $\mathcal{D}$, confidence ϵ

1. $\tilde{\epsilon} = \epsilon/(|A||B|)$
   fix δ, compute $\tilde{\delta}$ by (6.38)
2. compute $\underline{\gamma} = (\gamma_k(i),\ k = 1, \dots m,\ i = 1, \dots N)$ the posterior probability that x^i was generated by component k of the mixture
3. for $u \in A$, $v \in B$
   3.1 compute the thresholds M_k by (6.22)
   3.2 repeat
      3.2.1 sample Z from $\underline{\gamma}$
      3.2.2. for $k = 1, \dots m$ compute $\hat{I}^k_{uv}$ and compare to M_k until
      $Pr[\mathcal{E}_{uv}]$ <> $\tilde{\delta}$ with confidence $1 - \tilde{\epsilon}$; ">" means reject $\mathcal{H}_{uv}$, "<" means accept it

Output accept independence hypothesis if all the tests in step 3. accept it, otherwise reject

The crux of the method is step 3 where we need to estimate the low probabilities $Pr[\mathcal{E}_{uv}]$ with *very* high confidence $\tilde{\epsilon}$ for each pair u, v. We have of course the freedom to choose a $\tilde{\delta}$ closer to 0.5 in order to require fewer iterations of 3.1–3.3 to compare $Pr[\mathcal{E}_{uv}]$ with $\tilde{\delta}$ at the required confidence level. But a larger δ means a lower threshold M_k and this in turn means loosening the approximation in Sanov's theorem[9] and thus increasing the probability of type I error (false acceptance).

The choice of ϵ reflects our tradeoff between type I and type II errors. For a small ϵ the probability of rejecting a model (i.e. rejecting the independence hypothesis) that is correct is low, but we are likely to accept independence when this is not true. This latter probability is inflated in our case by further dividing ϵ between the $|A||B|$ tests in (6.42). Therefore it is better to accept a somewhat larger false rejection risk by choosing a larger ϵ lest the test becomes too permissive.

Let us examine if the assumptions our tests are based upon are true in the above procedure. The $|A||B|$ individual independence tests are dependent on each other because they share the observed values $(x^1, \dots x^N)$, but the Bonferroni inequality accounts for that. The dataset $\mathcal{D}$ is not the dataset the model was trained on.

It is in using an independence test based on $Pr[\mathcal{E}_{uv}]$ that we depart from the assumptions. Because the data $\mathcal{D}$ are already observed, the quantity that is actually estimated by repeating 3.2.1 – 3.2.2 is not $Pr[\mathcal{E}_{uv}]$ but $Pr[\mathcal{E}_{uv}|\mathcal{D}]$. It can be expressed as

$$
\begin{aligned}
Pr[\mathcal{E}_{uv}|\mathcal{D}] &= \sum_Z Pr[Z|\mathcal{D}] \prod_k Pr[I^k_{uv} < M_k | Z, \mathcal{D}] &(6.43)\\
&= \sum_Z \prod_i \gamma_{z^i}(i) \prod_k \delta_{I^k_{uv}(Z,\mathcal{D}) < M_k} &(6.44)\\
&= \sum_{Z: I^k_{uv}(Z,\mathcal{D}) < M_k\, \forall k} \prod_i \gamma_{z^i}(i) &(6.45)
\end{aligned}
$$

Hence, a correct test would predict or bound the above probability. But this involves the intractable summation over $\{I^k_{uv}(Z, \mathcal{D}) < M_k\, \forall k\}$.

Finally, a general fact about independence tests and their utility. The independence test can only reject independence and provide an estimate of the risk in doing it. It can

[9]This can be seen by examining the proof of Sanov's theorem.

never accept independence and estimating the risk of error associated with this decision is usually difficult.

6.8 Discussion

This chapter has been an exploration in to the domain of learning the structure of graphical models with hidden variables. We proposed to use the mixture of trees learning algorithm in order to uncover the presence of a hidden variable that partitions the visible variables set into mutually independent subsets. Such a model is useful because (if the number of values of the hidden variable is small) it offers the possibility to drastically reduce the complexity of the resulting density. Moreover, if such a hidden variable and the corresponding separation are discovered, the model's structure can be refined by subsequent structure learning separately in each of the variable clusters. It is, to my knowledge, the first top down approach to the problem of structure learning. However, as we stressed, the task that we attempt to solve is known to be a difficult one. Searching the space of possible structures is proven to be NP-hard. We have transformed it into a continuous domain optimization problem via the EM algorithm and are relying on multiple initializations to obtain good candidate structures. Empirically we showed that this is not unreasonable: the EM algorithm is indeed capable of uncovering the hidden structure and to converge to it even when this structure is not detectable in the marginal distribution of the visible variables.

The problem of model selection for domains with hidden variable is a topic of current research. We introduced a method to empirically evaluate the description length of a distribution. The empirical description length and a number of other criteria (BIC, holdout set likelihood and modified BIC) were compared on the task of selecting the best H model structure. It was found that the examined criteria are relatively similar in accuracy, with slight advantage for the empirical description length. None of the criteria proved completely reliable in selecting the good structure. One reason for this is that the task is such that the scoring function, no matter which one, is in impossibility to distinguish between the failure to discover the independency structure (the information of interest) on one side, and inaccuracy in the representation of the conditional distributions once the structure is found (information irrelevant to our task). In the context of comparing models by score, we found by experiment that determining the correct number of connected components p is not an easy task. In particular, for the data used in the experiments, the MDL edge penalty is systematically too strong, biasing towards models with false independencies.

An alternate approach is to test the obtained models for the independencies they assume. Testing independence in models with hidden variables is a yet unsolved problem. Here we propose a heuristic based on Monte Carlo sampling, together with a novel approach to independence testing based on large deviation theory. Independence tests allow to eliminate the structures in which the hypothesised variable clusters are not independent given the hidden variables, but they cannot help if the model has fewer connected components than the correct one (however, in some cases this can be considered a minor structure error). Moreover, the risk of discarding a structure that is correct but has residual dependencies because the trees are too simple to represent the distributions in each variable cluster and thereby induce inaccuracy in the values of the hidden variable.

By contrast, the case when the working assumption – that the data was generated by a tree H model – was correct has proved to be a much easier one. The rate of structure recovery by the EM algorithm, when the structure parameters m and p are correct is very high, often 100%. Likewise, the accuracy of the values of the hidden variable is very good. For model selection with fixed m and p, as well as for search across values of m, p, the model selection criteria, be they empirical DL or others, are found to be reliable.