Chapter 3

Mixtures of trees

The previous section has shown that in the framework of graphical probability models, tree distributions enjoy many properties that make them attractive as modeling tools: they have a flexible topology, are intuitively appealing, sampling and computing likelihoods are linear time, simple efficient algorithms for marginalizing and conditioning ($\mathcal{O}(n^2)$ or less) exist. Fitting the best tree to a given distribution can also be done exactly and efficiently. Trees can capture simple pairwise interactions between variables but they can prove insufficient for more complex distributions. Therefore, this chapter introduces a more powerful model, the *mixture of trees*. As this thesis will show, mixtures of trees enjoy most of the computational advantages of trees and, in addition, they are universal approximators over the space of all distributions.

We define a mixture of trees to be a distribution of the form

$$Q(x) \;=\; \sum_{k=1}^{m} \lambda_k T^k(x) \tag{3.1}$$

with

$$\lambda_k \geq 0,\; k = 1, \ldots, m; \qquad \sum_{k=1}^{m} \lambda_k \;=\; 1. \tag{3.2}$$

The tree distributions T^k are the *mixture components* and λ_k are called *mixture coefficients*. From the graphical models perspective, a mixture of trees can be viewed as a containing an unobserved *choice variable* z, which takes value $k \in \{1, \ldots m\}$ with probability λ_k. Conditioned on the value of z the distribution of the visible variables V is a tree. The m trees may have different structures and different parameters.

Note that because of the variable structure of the component trees, a mixture of trees is neither a Bayesian network nor a Markov random field. Let us adopt the notation

$$A \;\perp_P\; B \mid C \tag{3.3}$$

for "A independent B given C under distribution P". If for some (all) $k \in \{1, \ldots m\}$ we have

$$A \;\perp_{T^k}\; B \mid C \;\text{ with } A, B, C \subset V$$

this will not imply that

$$A \;\perp_Q\; B \mid C.$$

On the other hand, a mixture of trees is capable of representing dependency structures that are conditioned on the value of one variable (the choice variable), something that a usual Bayesian network or Markov net cannot do. Situations where such a model is potentially useful abound in real life: Consider for example bitmaps of handwritten digits.

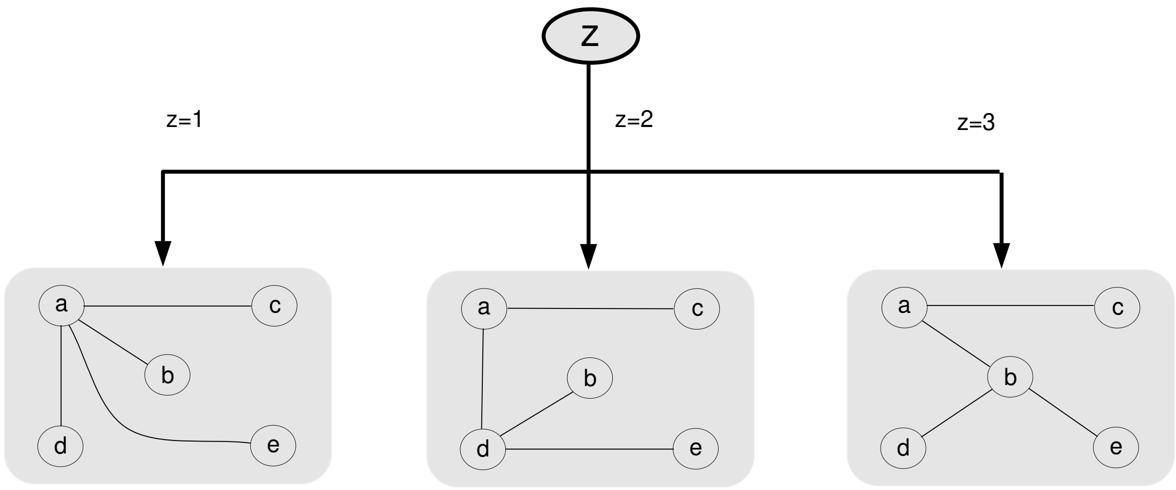

Figure 3-1: A mixture of trees with $m = 3$ and $n = 5$. Note that although $b \perp_{T^k} c \,|\, a$ for all $k = 1, 2, 3$ this does not imply in general $b \perp c \,|\, a$ for the mixture.

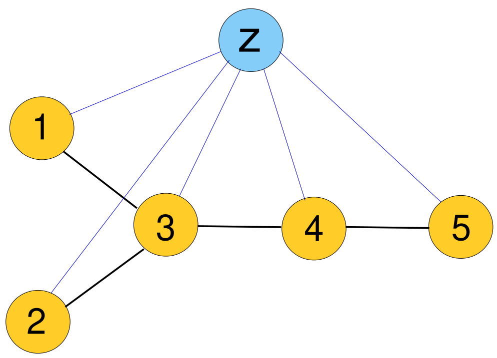

Figure 3-2: A Mixture of trees with shared structure represented as a graphical model.

They obviously contain many dependencies between pixels; however, the pattern of these dependencies will vary across digits. Imagine a medical database recording the body weight and other data for each patient. The body weight could be a function of age and height for a healthy person, but it would depend on other conditions if the patient suffered from a disease or were an athlete. If, in a situation like the ones mentioned above, conditioning on one variable produces a dependency structure characterized by sparse, acyclic pairwise dependencies, then a mixture of trees will be the model most able to uncover and exploit this kind of dependencies.

If we impose that all the trees in the mixture have the same structure we obtain a *mixture of trees with shared structure* (MTSS). Note that even in this case the resulting mixture does not preserve the independence relationships of the component trees. But a MTSS *can* be represented as a Bayes net (after choosing an orientation for the trees) or, more directly, as a *chain graph.* Chain graphs were introduced by [45]; they represent a superclass of both Bayes nets and Markov random fields [44]. A chain graph contains both directed and undirected edges. The representation of the MTSS as a graphical model is given in figure 3-2.

A simple modification of both the mixture of trees and the mixture of trees with shared structure is to have the choice variable be observed as well. Such models will be called *mixtures with visible choice variable* in the forthcoming. Unless otherwise stated, it will be assumed that the choice variable is hidden. Therefore, sometimes it will be referred to as *hidden variable.*

3.1 Representation power of mixtures of trees

Here it will be shown that for discrete variable domains the class of mixtures of trees can represent any distribution.

Theorem Let $V = \{v_1, v_2, \ldots v_n\}$ a set of variables with finite ranges and P a probability distribution over $\Omega(V)$. Then P can be represented as a mixture of trees.

Proof. Let us denote by δ_{x^*} the distribution over $\Omega(V)$ defined by

$$\delta_{x^*}(x) = \begin{cases} 1, & x = x^* \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

δ_{x^*} can be identified with a tree T^{x^*} having n connected components (i.e. a factored distribution) and

$$T^{x^*}_{v_j} = \begin{cases} 1, & v_j = x^*_j \\ 0, & \text{otherwise} \end{cases} \quad \text{for } j = 1, \ldots n. \tag{3.5}$$

Moreover, any distribution P can be represented as a mixture of δ distributions:

$$P(x) \ = \ \sum_{x^*} P(x^*)\delta_{x^*}(x). \tag{3.6}$$

where the sum is taken over the range of V and $P(x^*)$ represent the mixing coefficients. This completes the proof. ∎

3.2 Basic operations with mixtures of trees

Marginalization The marginal distribution of variables subset $A \subset V$ is

$$\begin{aligned} Q_A(x_A) &= \sum_{x_{V\setminus A}} Q(x) \\ &= \sum_{x_{V\setminus A}} \sum_{k=1}^{m} \lambda_k T^k(x) \\ &= \sum_{k=1}^{m} \lambda_k \sum_{x_{V\setminus A}} T^k(x) \\ &= \sum_{k=1}^{m} \lambda_k T^k_A(x_A) \qquad (3.7) \end{aligned}$$

Hence, the marginal of Q is a mixture of the marginals of the component trees.

Inference (in the restricted sense) which is marginalization after conditioning on the evidence is performed in an analog way. Let $\mathcal{E}_B$ be the evidence and $B \subset V$ be the set of variables that the evidence is about. Then

$$\begin{aligned} Q_A(x_A|\mathcal{E}_B) &= \sum_{x_{V\setminus(A\cup B)}} Q(x|\mathcal{E}_B) \qquad (3.8) \\ &= \sum_{k=1}^{m} \lambda_k T^k_A(x_A|\mathcal{E}_B). \qquad (3.9) \end{aligned}$$

One can also infer the value of the hidden variable given some evidence $\mathcal{E}_B$ by applying Bayes' rule

$$\begin{aligned} Pr[z=k|\mathcal{E}_B] &= \frac{Pr[\mathcal{E}_B|z=k]Pr[z=k]}{\sum_{k'} Pr[\mathcal{E}_B|z=k']Pr[z=k']} \qquad (3.10) \\ &= \frac{\lambda_k T^k_B(\mathcal{E}_B)}{\sum_{k'} \lambda_{k'} T^{k'}_B(\mathcal{E}_B)} \qquad (3.11) \end{aligned}$$

In particular, when the evidence consists of observing all the visible variables $\mathcal{E} = \{v_1 = x_1,\, v_2 = x_2, \ldots\, v_n = x_n\}$, $B = V$ and $(x_1,\, x_2, \ldots\, x_n) = x$ the *posterior* probability distribution of z is

$$Pr[z=k|\mathcal{E}] \;\equiv\; Pr[z=k|x] \;=\; \frac{\lambda_k T^k(x)}{\sum_{k'} \lambda_{k'} T^{k'}(x)} \qquad (3.12)$$

Sampling Sampling in a mixture of trees, in the presence of evidence or not, is a straightforward extension of sampling from one tree discussed in chapter 2.2. The procedure for sampling from a mixture of trees uses **TreeSample(T,** $\mathcal{E}$**)** the procedure that samples from a tree distribution given evidence $\mathcal{E}$ (possibly void)

Algorithm **MixTreeSample**(Q, $\mathcal{E}$)

Input a mixture of trees Q
evidence $\mathcal{E}$

1. sample k the value of z from $(\lambda_1,\, \lambda_2,\, \ldots \lambda_n)$
2. sample x from T^k component of the mixture and returns a value x.

Output x

Conclusions As this subsection has shown, the basic operations on mixtures of trees, marginalization, conditioning and sampling, are direct extensions of the corresponding operations on tree distributions. Their complexity scales accordingly: Marginalization over a subset $V \setminus A$ in a mixture of trees takes m times the computation for marginalization over the same subset in a single tree. For instance, computing the marginal of a single variable (that is not in a separator) takes $\mathcal{O}(mr^2_{MAX})$. Inference in a mixture of trees takes m junction tree propagation operations, $\mathcal{O}(mnr^2_{MAX})$. Inferring the value of the hidden variable takes mn multiplications. Sampling from a mixture is $\mathcal{O}(m + n.n_\mathrm{s})$, where n_s is the number of operations required to sample a value of a vertex in a clique, conditioned on the value of the other vertex. For the directed tree representation $n_\mathrm{s} \leq r_{MAX}$. The choice of the tree to sample from is an m-way choice requiring $\mathcal{O}(m)$ operations.

3.3 Learning mixtures of trees in the ML framework

3.3.1 The basic algorithm

This section will show how a mixture of trees can be fit to an observed dataset in the Maximum Likelihood paradigm via the EM algorithm [18].

The learning problem is similar to the one formulated for trees in chapter 2. We are given a set of observations $\mathcal{D} = \{x^1, x^2, \ldots, x^N\}$ and we are required to find the mixture of trees Q that satisfies

$$Q = \operatorname*{argmax}_{Q'} \sum_{i=1}^{N} \log Q(x^i). \tag{3.13}$$

Here and in the rest of the thesis we will assume that there are no missing values for the variables in V. As for z we will first assume that it is hidden. We denote the (hidden) values of the choice variable by $\{z^i, i = 1, \ldots N\}$.

Learning the ML mixture of trees model will be done by means of the Expectation-Maximization (EM) algorithm. This is an iterative algorithm devised for fitting maximum likelihood parameters for models with hidden or missing variables. Each iteration consists of two steps, Expectation and Maximization, that will be described below.

The EM algorithm introduces a likelihood function called the *complete log-likelihood* which is the log-likelihood of both the observed and the unobserved data given the current model estimate $\mathcal{M} = \{m,\ T^k,\ \lambda^k,\ k = 1, \ldots m\}$

$$l_c(x^{1,\ldots N}, z^{1,\ldots N}|\mathcal{M}) = \sum_{i=1}^{N}\sum_{k=1}^{m} \delta_{k,z^i}(\log \lambda_k + \log T^k(x^i)) \tag{3.14}$$

The complete log-likelihood depends on the unknown values of the hidden variable and therefore it is not directly computable. The idea underlying the EM algorithm is to instead compute and optimize the *expected* value of l_c.

The Expectation (E) step consists of estimating the posterior probability of the hidden variable for each of the observations. In our case this means estimating the probability of each tree generating data point x^i

$$Pr[z^i = k|x^i, \mathcal{M}] = \gamma_k(i) = \frac{\lambda_k T^k(x^i)}{\sum_{k'} \lambda_{k'} T^{k'}(x^i)} = E[\delta_{k,z^i}] \tag{3.15}$$

One uses these posterior probabilities to compute the expectation of l_c, which is a linear function of the $\gamma_k(i)$ values.

$$E[l_c(x^{1,\dots N}, z^{1,\dots N}|\mathcal{M})] \;=\; \sum_{i=1}^{N}\sum_{k=1}^{m}\gamma_k(i)(\log\lambda_k + \log T^k(x^i)) \tag{3.16}$$

Let us introduce the following quantities:

$$\Gamma_k \;=\; \sum_{i=1}^{N}\gamma_k(x^i), \quad k = 1, \dots m \tag{3.17}$$

$$P^k(x^i) \;=\; \frac{\gamma_k(i)}{\Gamma_k} \tag{3.18}$$

The sums $\Gamma_k \in [0, N]$ can be interpreted as the total number of data points that are generated by component T^k. By normalizing the posteriors $\gamma_k(i)$ with Γ_k we obtain a probability distribution P^k over the data set. In the following it will be shown that P^k acts as a target distribution for T^k. For now, let us express the expected complete log-likelihood in terms of P^k and Γ_k.

$$E[l_c\,|\,x^{1,\dots N},\mathcal{M}] \;=\; \sum_{k=1}^{m}\Gamma_k\log\lambda_k + \sum_{k=1}^{m}\Gamma_k\sum_{i=1}^{N}P^k(x^i)\log T^k(x^i) \tag{3.19}$$

The Maximization (M) step of the EM algorithm reestimates the parameters of the model so as to maximize $E[l_c\,|\,x^{1,\dots N},\mathcal{M}]$. It can be proved [18] that the iteration over these two steps converges a local maximum of the log-likelihood of the visible data given the model.

By inspecting equation (3.19) one can see that the expression of $E[l_c]$ is a sum whose terms depend on disjoint subsets of the model's parameters. Hence, one can maximize separately each term of the sum w.r.t. to the part of the model that it depends on.

Thus, by maximizing the first term of (3.19) subject to the constraint

$$\sum_{k=1}^{m}\lambda_k \;=\; 1$$

one obtains the new values for the parameters λ

$$\lambda_k \;=\; \frac{\Gamma_k}{N} \text{ for } k = 1, \dots m \tag{3.20}$$

To obtain the new distributions T^k, we have to maximize for each k the expression that is the negative of the cross-entropy between P^k and T^k.

$$\sum_{i=1}^{N}P^k(x^i)\log T^k(x^i) \tag{3.21}$$

This problem is equivalent to the problem of fitting a tree to a given distribution and can be solved exactly as shown in section 2.3.2. Here I will give a brief reminder of the procedure. First, one has to compute the mutual information between each pair of variables in V under the target distribution P^k

$$I^k_{uv} \;=\; \sum_{x_u x_v}P^k_{uv}(x_u, x_v)\log\frac{P^k_{uv}(x_u, x_v)}{P^k_u(x_u)P^k_v(x_v)}, \quad u, v \in V, u \neq v. \tag{3.22}$$

Second, the optimal tree structure E_{T^k} is found by a Maximum Weight Spanning Tree algorithm using I^k_{uv} as the weight for edge $(u, v), \forall u, v \in V$. Once the tree is found, its marginals T^k_{uv} (or $T^k_{u|v}$), $(uv) \in E_{T^k}$ are exactly equal to the corresponding marginals P^k_{uv} of the target distribution P^k. They are already computed as an intermediate step in the computation of the mutual informations I^k_{uv} (2.26). The algorithm is summarized here:

Algorithm **MixTree** – outline

MixTree$(\mathcal{D}, \mathcal{M}_0)$

Input: Dataset $\{x^1, \ldots x^N\}$
Initial model $\mathcal{M}_0 = \{m,\ T^k,\ \lambda^k,\ k = 1, \ldots m\}$
Procedure **TreeLearn**(P)
Iterate until convergence:
E step: compute $\gamma^i_k,\ P^k(x^i)$ for $k = 1, \ldots m,\ i = 1, \ldots N$ by (3.15), (3.17), (3.18)
M step: for $k = 1, \ldots m$
Ml. $\lambda_k \leftarrow \Gamma_k/N$
MT. T^k = **TreeLearn**(P^k)
Output model $\mathcal{M} = \{m,\ T^k,\ \lambda^k,\ k = 1, \ldots m\}$

Now we show the steps of the algorithm again in more detail. Next to each step is displayed its running time. Time and memory requirements will be discussed in the next subsection.

Algorithm **MixTree**

Input: Dataset $\{x^1, \ldots x^N\}$
Initial model $\mathcal{M} = \{m,\ T^k,\ \lambda^k,\ k = 1, \ldots m\}$
Procedure MWST(weights) that fits a maximum weight spanning tree over V
Iterate until convergence:

E step:	compute $\gamma^i_k,\ P^k(x^i)$ for $k = 1, \ldots m,\ i = 1, \ldots N$ by (3.15), (3.17), (3.18)	$\mathcal{O}(mnN)$
M step:	for $k = 1, \ldots m$	
M1.	$\lambda_k \leftarrow \Gamma_k/N$	$\mathcal{O}(m)$
M2.	compute marginals $P^k_v,\ P^k_{uv},\ u, v \in V$	$\mathcal{O}(mn^2N)$
M3.	compute mutual information $I^k_{uv}\ u, v \in V$	$\mathcal{O}(mnr^2_{MAX})$
M4.	call MWST($\{ I^k_{uv} \}$) to generate E_{T^k}	$\mathcal{O}(mn^2)$
M5.	$T^k_{uv} \leftarrow P^k_{uv},\ ; T^k_v \leftarrow P^k_v$ for $(u, v) \in E_{T^k}$	$\mathcal{O}(mnr^2_{MAX})$

3.3.2 Running time and storage requirements

Running time. Computing the likelihood of a data point under a tree distribution (in the directed tree representation, where the parameters represent conditional distributions) takes $n - 1 = \mathcal{O}(n)$ multiplications. Hence, the E step should require $\mathcal{O}(mnN)$ floating point multiplications/divisions.

For the M step the situation is: m divisions for computing the λ_k values in step **M1**; $\mathcal{O}(mn^2N)$ for the marginals of P^k in step **M2**; another $\mathcal{O}(mn^2r^2_{MAX})$ for the mutual informations in step **M3**; $\mathcal{O}(mn^2)$ for running the MWST algorithm m times in step **M4**; finally $\mathcal{O}(mnr^2_{MAX})$ for computing the tree parameters in the directed representation in step **M5** of the algorithm. These values are presented at the right of the **MixTree** algorithm above.

The total running time per EM iteration is thus

$$\mathcal{O}(mn^2N + mnr^2_{MAX})) \tag{3.23}$$

The dominating terms in this cost correspond to the computation of the marginals P^k_{uv} and of the $mn(n-1)/2$ mutual informations under the distributions P^k.

Storage requirements Storing the data takes mN integer numbers; additionally, to represent the model one needs $\mathcal{O}(mnr^2_{MAX})$ real parameters. This storage is necessary independently of the learning algorithm used.

The intermediate values that are needed by the EM algorithm are:

- the γ_k values that require mN storage locations. The P^k values can overwrite the γ_k values so that no additional storage is needed for them.
- the single variable and pairwise marginals of P^k that require $\frac{n(n+1)}{2}r^2_{MAX}$ storage. They can be overwritten for successive values of k.
- the mutual informations I^k_{uv} between pairs of variables that require $n(n-1)/2$ storage locations. They can also be overwritten for each k.
- the additional temporary storage required by the MWST algorithm. In the Kruskal algorithm, this amount is proportional to the number of candidate edges, namely $\mathcal{O}(n^2)$. Other algorithms, [10] require only linear space.

The total storage is

$$\mathcal{O}(mN + mnr^2_{MAX} + n^2r^2_{MAX}) \tag{3.24}$$

Again, the dominant cost corresponds to the computation of the pairwise marginals of the distributions P^k.

3.3.3 Learning mixtures of trees with shared structure

It is possible to constrain the m trees to share the same structure, thus constructing a truly Bayesian network. The steps of the EM algorithm that learn mixtures of trees with shared structure, called **MixTreeS**, are given below. Most of the reasoning and calculations parallel the ones describing the fitting of a tree to a distribution in section 2.3.2.

The E step here is identical to the E step of the **MixTree** algorithm. So are the expression of the expected complete log-likelihood

$$E[l_c \,|\, x^{1,\dots N}, \mathcal{M}] \;=\; \sum_{k=1}^{m} \Gamma_k \log \lambda_k + \sum_{k=1}^{m} \Gamma_k \sum_{i=1}^{N} P^k(x^i) \log T^k(x^i) \tag{3.25}$$

One can easily see that the reestimation of $\{\lambda_k,\; k = 1, \dots m\}$ is decoupled from the estimation of the rest of the model and can be performed in the same way as before, i.e.

$$\lambda_k \;=\; \frac{\Gamma_k}{N} \;\text{ for } k = 1, \dots m \tag{3.26}$$

The difference is in reestimating the k tree distributions, since now they are constrained to have the same structure. Hence, the maximization over the tree distributions cannot be decoupled into m separate tree estimations but must be performed simultaneously for all the trees. A reasoning similar to one in section 2.3.2 shows that for any given structure the optimal parameters of each tree edge T^k_{uv} are equal to the parameters of the corresponding marginal distribution P^k_{uv} (equation (2.17),reproduced here)

$$T^k_{uv} \;\equiv\; P^k_{uv} \;\; \forall (u,v) \in E \tag{3.27}$$

It remains only to find the optimal structure. The expression to be optimized is the second sum in the r.h.s. of equation (3.25):

$$\sum_{k=1}^{m} \Gamma_k \sum_{i=1}^{N} P^k(x^i) \log T^k(x^i) \tag{3.28}$$

By making the substitution (3.27) in the above expression and following a series of steps similar to (2.20–2.25) we obtain that (3.28) is equal to

$$\begin{aligned}
& \sum_{k=1}^{m} \Gamma_k [\sum_{(u,v)\in E} I^k_{uv} - \sum_{v\in V} H(P^k_v)] && (3.29)\\
= \;\; & N \sum_{k=1}^{m} \lambda_k \sum_{(u,v)\in E} I^k_{uv} + \text{constant independent of structure} && (3.30)\\
= \;\; & N \sum_{(u,v)\in E} I_{uv|z} + \text{constant} && (3.31)
\end{aligned}$$

Hence, optimizing the structure E can be again done by a MWST algorithm with the edge weights represented by $I_{uv|z}$ the *conditional mutual information* between the variables adjacent to it given z. This algorithm is essentially the same as the TANB learning algorithm of [24, 22]. The expressions in the right column represent the running time for each step of the algorithm.

Algorithm **MixTreeS**

Input: Dataset $\{x^1, \ldots x^N\}$
Initial model $\mathcal{M} = \{m,\ T^k,\ \lambda^k,\ k = 1, \ldots m\}$
Procedure MWST(weights) that fits a maximum weight spanning tree over V
Iterate until convergence:

E step:		compute $\gamma^i_k,\ P^k(x^i)$ for $k = 1, \ldots m,\ i = 1, \ldots N$ by (3.15), (3.17), (3.18)	$\mathcal{O}(mnN)$
M step:	**M1.**	for $k = 1, \ldots m$ compute marginals $P^k_v,\ P^k_{uv},\ u, v \in V$	$\mathcal{O}(mn^2N)$
	M2.	compute the conditional mutual information $I_{uv\vert z}\ u, v \in V$	$\mathcal{O}(mnr^2_{MAX})$
	M3.	call MWST({ $I_{uv\vert z}$ }) to generate E_T	$\mathcal{O}(n^2)$
	M4.	for $k = 1, \ldots m$: $\lambda_k \leftarrow \Gamma_k/N$	$\mathcal{O}(m)$
	M5.	for $k = 1, \ldots m$: $T^k_{uv} \leftarrow P^k_{uv},\ ; T^k_v \leftarrow P^k_v$ for $(u,v) \in E_T$	$\mathcal{O}(mnr^2_{MAX})$

3.3.4 Remarks on the learning algorithms

The above described procedures for learning mixtures of trees from data are based on one important assumption that should be made explicit now. It is the

Parameter independence assumption: *For any* k, v *and value of* $\mathrm{pa}(v)$ *the distribution* $T^k_{v|\mathrm{pa}(v)}$ *is a multinomial with* $r_v - 1$ *free parameters that are independent both of the tree structure and of any other parameters of the mixture.*[1]

The parameter independence assumption is essential to the computational efficiency of the **MixTree** and **MixTreeS** algorithms. The independence of the parameters of $T^k_{v|\mathrm{pa}(v)}$ of any other parameters allows for the simple form of step **M5** of the EM algorithm. The independence from tree structures allows us to use the MWST algorithm to globally maximize over tree structures at each iteration. As a practical remark, note that although equation (3.27) calls for copying the marginals of the target distributions P^k, the algorithms should implement the directed tree representation and thus copy in its parameters the conditional probabilities $P^k_{v|\mathrm{pa}(v)}$.

Note also that, although the **TreeLearn** algorithm always attains a global solution, learning a mixture of trees by the EM algorithm will converge to a *local* maximum of the likelihood.

The tree structures obtained by the basic algorithm are connected. In the following chapters we will show reasons and ways to obtain disconnected tree structures.

Missing variables are handled elegantly by trees. Any number of nonadjacent missing variables can be marginalized out in $\mathcal{O}(r_{MAX})$ time and this bound grows exponentially with l, the size of the largest connected subset of missing variables.

Observed but unknown choice variable An interesting special case is the situation where the choice variable is in fact one of the observed variables (or a small subset thereof), but we don't know which one? To discover it, one can either: build several mixtures by conditioning on each one of the observables and then compare their posteriors, or: build one standard mixture model and then compare the mutual information between the structure variable and each of the others to identify the most likely candidate.

3.4 Summary and related work

This chapter has introduced the mixture of trees model with its variants and has shown that the basic operations on mixtures of trees are direct extensions of the same operations on tree distributions and thus inherit their excellent computational properties. It has also presented a tractable algorithm to learn the model from data in the Maximum Likelihood framework.

Trees have been noticed for their flexibility and computational efficiency as early as 1968 by [7] who also used them in a classification task, in what is called here a mixture of trees with visible choice variable (i.e. by fitting a different tree to each class). [55] builds on the Chow and Liu algorithm an exact algorithm for learning a polytree *when the true distribution is known to be a polytree*. Fitting the best polytree to an arbitrary distribution is NP-hard [33, 13]. [27] considered the same model, now with an explicit and observed choice variable as a special case of the *Bayesian multinet* [28]. The MTSS appeared in the work of [24] as the Tree Augmented Naive Bayes classifier and was further developed in [22, 23]. The latter work also considers continuous variables and suggests an ingenious heuristic for deciding whether the continuous variables should be discretized or whether their density should be approximated (e.g. by a mixture of Gaussians). The mixture of trees as it was presented here encompasses the aforementioned models as special cases; it

[1] In [33] this assumption is stated as three distinct assumptions called: parameter modularity, global parameter independence and local parameter independence.

was introduced in [48, 49]. Classification results with the mixture of trees (which will be presented in chapter 7 were first published in [49].

Mixture models, the second "parent" of mixtures of trees, have come an even longer way. Therefore I will only cite mixture models that are closely related to the present work. The mixture of factorial distribution in the form of the Naive-Bayes model (i.e with the choice variable being a class variable in a classification task), sometimes known as Auto-Class was given the name and an attentive study in [5] and was heavily used ever since for its excellent cost/performance ratio[2]. [41] successfully used a mixture of factorial distributions with a hidden variable for classification and this line was further followed by [51] who combined the two. The idea of learning tractable but simple belief networks and superimposing a mixture to account for the remaining dependencies was developed independently of this work by [66] into mixtures of Gaussian belief networks. Their work interleaves EM parameter search with Bayesian model search in a heuristic but general algorithm.

From here we shall go on to examine learning the same model in the Bayesian framework and show that the algorithms introduced here can be extended to that case as well. The quadratic time and storage complexity of the learning algorithm is satisfactory for medium scale problems but may become prohibitive for problems of very high dimensionality. We shall show how these requirements can be reduced under certain conditions of practical importance. We shall also examine how a mixture of trees can be used for classification. Finally we study the use of the mixture of trees learning algorithm as a heuristic for solving a challenging problem: hidden variable discovery.

[2] Section 7.3.5 shows that the Naive Bayes classifier has little sensitivity to irrelevant attributes, just like the single tree classifier, which partly explains its success.