Chapter 2

Trees and their properties

I think that I shall never see
A poem as lovely as a tree.
Joyce Kilmer
–*Trees*

In this section we introduce tree distributions as a subclass of decomposable models and we demonstrate some of the properties that make them attractive from a computational point of view. It will be shown that fundamental operations on distributions: inference, sampling and marginalizing carry over directly from their junction tree counterparts and are order n or less when applied to trees. Learning tree distributions will be formulated as Maximum Likelihood (ML) estimation problem. To solve it, an algorithm will be presented that finds the tree distribution T that best approximates a given target distribution P in the sense of the Kullback-Leibler divergence. The algorithm optimizes over both structure and parameters, in time and memory proportional to the size of the data set and quadratic in the number of variables n. Some final considerations on modeling power and ease of visualization will prepare the introduction of mixtures of trees in the next section.

2.1 Tree distributions

In this section we will introduce the tree model and the notation that will be used throughout the paper. Let V denote the set of variables of interest. As stated before, the cardinality of V is $|V| = n$. For each variable $v \in V$ let r_v denote its number of values, $\Omega(v)$ represent its domain and $x_v \in \Omega(v)$ a particular value of v. Similarly, for subset A of V, $\Omega(A) = \bigotimes_{v \in A} \Omega(v)$ is the domain of A and x_A an assignment to the variables in A. In particular, $\Omega(V)$ is the state-space of all the variables in V; to simplify notation x_V will be denoted by x. Sometimes we shall need the maximum of r_v over V; we shall denote this value by r_{MAX}.

According to the graphical model paradigm, each variable is viewed as a vertex of an (undirected) graph (V, E). An edge connecting variables u and v is denoted by (uv) and its significance will become clear in the following. The graph (V, E) is called a *tree* if it has no cycles[1]. Note that under this definition, a tree can have a number p between 1 and $|V|$ *connected components*. The number of edges $|E|$ and p are in the relationship

$$|E| + p \ = \ |V| \tag{2.1}$$

[1]This definition differs slightly from the definition of a tree in the graph theory literature. There, a tree is required to have no cycles *and* to be connected (meaning that between every two vertices there should exist a path); our definition of a tree allows for disconnected graphs to be trees and corresponds to what in graph theory is called a *forest*.

which means that adding an edge to a tree reduces the number of connected components by 1. Thus, a tree can have at most $|V| - 1 = n - 1$ edges.

Now we define a probability distribution T that is *conformal* with a tree. Let us denote by T_{uv} and T_v the marginals of T for $u, v \in V$ and $(uv) \in E$:

$$T_{uv}(x_u, x_v) = \sum_{x|u=x_u, v=x_v} T(x) \tag{2.2}$$

$$T_v(x_v) = \sum_{x|v=x_v} T(x). \tag{2.3}$$

They must satisfy the consistency condition

$$T_v(x_v) = \sum_{x_u} T(x_v, x_u) \;\; \forall u, v, \; (uv) \in E \tag{2.4}$$

Let $\deg v$ be the *degree* of vertex v, i.e. the number of edges incident to $v \in V$. Then, the distribution T is conformal with the tree (V, E) if it can be factored as:

$$T(x) = \frac{\prod_{(u,v)\in E} T_{uv}(x_u, x_v)}{\prod_{v \in V} T_v(x_v)^{\deg v - 1}} \tag{2.5}$$

The distribution T itself will be called a tree when no confusion is possible. The graph (V, E) represents the *structure* of the distribution T. Noting that for all trees over the same domain V the edge set E alone uniquely defines the tree structure, in the following when no confusion is possible we identify E with the structure. If the tree is connected, i.e. it *spans* all the nodes in V, it is sometimes called a *spanning tree*.

Because a tree is a triangulated graph it is easy to see that a tree distribution is a decomposable model. In fact, the above representation (2.5) is identical to the junction tree representation of T. The cliques identify with the graph's edges (hence all cliques are size two) and the separators are all the nodes of degree larger than one. Thus the junction tree is identical to the tree itself with the clique and separator potentials being the marginals T_{uv} and T_v, $\deg v > 1$ respectively.

This shows a remarkable property of tree distributions: a distribution T that is conformal with the tree (V, E) is completely determined by its edge marginals $\{T_{uv}, \; (uv) \in E\}$.

Since every tree T is a decomposable model it can be represented in terms of conditional probabilities

$$T(x) = \prod_{v \in V} T_{v|\mathrm{pa}(v)}(x_v | x_{\mathrm{pa}(v)}) \tag{2.6}$$

We shall call the representations (2.5) and (2.6) the *undirected* and *directed* tree representations respectively. The form (2.6) is obtained from (2.5) by choosing an arbitrary root in each connected component and directing each edge away from the root. In other words, if for (uv) in E, u is closer to the root than v, then u becomes the parent of v and is denoted by $\mathrm{pa}(v)$. Note that in the *directed tree* thus obtained, each vertex has at most one parent. Consequently, the families of a tree distribution are either of size one (the root or roots) or of size two (the families of the other vertices). After having transformed the structure into a directed tree one computes the conditional probabilities corresponding to each directed edge by recursively substituting $\frac{T_{v\mathrm{pa}(v)}}{T_{\mathrm{pa}(v)}}$ by $T_{v|\mathrm{pa}(v)}$ starting from the root. $\mathrm{pa}(v)$ represents the parent of v in the thus directed tree or the empty set if v is the root of a connected component. The directed tree representation has the advantage of having independent parameters. The total number of free parameters in either representation is

$$\begin{aligned}\sum_{(u,v)\in E} r_u r_v - \sum_{v\in V}(\deg v - 1) r_v - p \ &= \\ = \ \sum_{(u,v)\in E}(r_u-1)(r_v-1) + \sum_{v\in V} r_v - \ n & \qquad (2.7)\end{aligned}$$

The r.h.s. of (2.7) shows that each newly added edge (uv) increases the number of parameters by $(r_u - 1)(r_v - 1)$.

Now we shall characterize the set of independences represented by a tree. In a tree there is at most one path between every two vertices. If node w is on the path between u and v we say that w *separates* u and v. Correspondingly, if we set w as a root, the probability distribution T can be decomposed into the product of two factors, each one containing only one of the variables u, v. Hence, u and v are independent given w:

$$u \perp v \mid w$$

Therefore we conclude that in a tree two variables are separated by any set that intersects the path between them. Two subsets $A, B \subset V$ are independent given $C \subset V$ if C intersects every path between $u \in A$ and $v \in B$.

We can also verify that for a tree the undirected Markov property holds. The set of variables connected by edges to a variable v, namely its neighbors separates v from all the other variables in V.

2.2 Inference, sampling and marginalization in a tree distribution

This section discusses the basic operations: inference, marginalization and sampling for tree distributions. It demonstrates that the algorithms for performing these operations are direct adaptations of their counterparts for junction trees and are heavily relying on the generic Junction Tree algorithm. It is also shown that all basic operations are linear in the number of variables n. This fact is important because the analog operations on mixtures of trees use the algorithms for trees as building blocks. The details of the algorithms however, although presented here for the sake of completeness, can be skipped without prejudice for the understanding of the rest of the thesis.

2.2.1 Inference.

As shown already, tree distributions are decomposable models and the generic inference algorithm for trees is an instance of the inference algorithm for decomposable models called the Junction Tree algorithm. We shall define it as a procedure, **PropagateEvidence**$(T, \mathcal{E})$ that takes as inputs a tree having structure (V, E) presented in the undirected factored representation and some categorical evidence $\mathcal{E}$ on a set $A \subset V$. The procedure performs the first two steps of the inference process as defined in section 1.3.4. It enters the categorical evidence by multiplying the original distribution T with a $\{0, 1\}$-values function representing the categorical evidence. Then it calibrates the resulting tree by local propagation of information between the cliques (edges) of the tree. Finally, it outputs the tree T^* that is factored conformally to the original T and represents the posterior distribution $T_{V|\mathcal{E}}$. The algorithm is described in detail in the last section of this chapter. It is also demonstrated that it takes a running time of the order $\mathcal{O}(r_{MAX}^2|E|)$.

The last step, extracting the probabilities of variables of interest from the above mentioned representation is done by marginalization and will be discussed in the next section. In particular, obtaining the posterior probability of any single variable involves marginalization over at most one other variable which takes $\mathcal{O}(r_{MAX})$ operations.

2.2.2 Marginalization

In a junction tree, computing the marginal distribution of any group of variables that are contained in the same clique can be performed by marginalization within the respective clique. For a tree, whose cliques are size two, the marginal of each variable v can be obtained either directly as T_v (if v is a separator and separator potentials are stored explicitly) or by marginalizing in the potential T_{uv} of the edge incident to v (if v is a leaf node). Thus, the marginal for any single variable can be obtained in $\mathcal{O}(r_{MAX})$ additions.

The pairwise marginals for all the variables that are neighbors in T are directly available as the clique potentials T_{uv}. The above enumeration exhausts all the cases where the marginals are directly available from the tree distribution. In the following I describe an algorithm that can efficiently compute the marginal distributions for arbitrary pairs of variables. The algorithm can be generalized to marginal distributions over arbitrary subsets of V.

First, it will be shown that the marginal T_{uv} depends only on the potentials of the edges on the path between u and v in E. Let $\text{path}(u,v) = (w^0 = u, w^1, w^2, \ldots, w^d = v)$ be the vertices on the path between u and v in the tree (V,E) and let $d \geq 1$ be its length. Then, the marginal T_{uv} can be expressed as

$$\begin{aligned}
T_{uv} &= \sum_{x_{V\setminus\{u,v\}}} T \\
&= \sum_{x_{V\setminus\{u,v\}}} \left(T_u \prod_{i=1}^{d} T_{w^i|w^{i-1}} \prod_{w'\in V\setminus\text{path}(u,v)} T_{w'|\text{pa}(w')} \right) && (2.8)\\
&= \sum_{x_{V\setminus\{u,v\}}} T_u \prod_{i=1}^{d} T_{w^i|w^{i-1}} && (2.9)\\
&= \sum_{x_{w^i}, i=1,d-1} \frac{\prod_{i=1}^{d} T_{w^{i-1}w^i}}{\prod_{i=1}^{d-1} T_{w^i}} && (2.10)
\end{aligned}$$

The first form (2.8) is obtained as the directed tree representation with a root at u. Summation over each $x_{w'}$ not on the path from u to v can be done recursively starting from the leaves of the tree and results in a factor of 1. Thus the form (2.9) is obtained. The third form, (2.10) is the rewriting of the previous equation in the undirected representation.

Each of the intermediate variables w^i appears in only two factors of (2.9); consequently, summation over the values of the intermediate variables can be done one variable at a time, as indicated by the following telescoped sum:

$$T_{vu} = \sum_{x_{w^{d-1}}} T_{v|w^{d-1}} \ldots \underbrace{T_{w^3|w^2} \sum_{x_{w^2}} \underbrace{(T_u \sum_{x_{w^1}} T_{w^1|u} T_{w^2|w^1})}_{T_{w^2 u}}}_{T_{w^3 u}} \tag{2.11}$$

The computation of one value of T_{uv} takes $\sum_{i=1}^{d-1} r_{w^i}$ additions and multiplications; to compute the whole marginal probability table we need to perform

$$r_u r_v \sum_{i=1}^{d-1} r_{w^i} = \mathcal{O}((d-1) r_{MAX}^3)$$

operations.

As the above equation shows, the intermediate sums involved in the computation of T_{uv} are themselves marginal distributions. This suggests that if the intermediate sums are stored and if the pairs $(uv) \notin E$ are enumerated in a judiciously chosen order, one can compute *all* the pairwise marginal tables by summing over 1 variable only for each of them; in other words, all the pairwise marginals can be computed with $\mathcal{O}((n-1)(n-2)r_{MAX}^3)$ operations.

Marginalization in the presence of evidence represents the third and last step of inference, as presented in the previous subsection. This problem can be approached in various ways, one of them being the polytree algorithm of [55]. The approach we present here follows directly from the procedures for entering evidence, calibration and marginalization introduced above. To find $T^*_{uv} = T_{uv|\mathcal{E}_A}$ for $u, v \in V$, $A \subset V$ one has to

1. enter and propagate evidence by $T^* \leftarrow$ **PropagateEvidence**$(T, \mathcal{E}_A)$
2. compute the marginal T^*_{uv}

2.2.3 Sampling

Sampling in a tree is best performed using the directed tree representation. The value of the root node(s) is sampled from its (their) marginal distribution. Then, the value of each of the other nodes is sampled from the conditional distribution given its parent $P(v|\mathrm{pa}(v))$, recursively, starting from the root(s). This simple algorithm is the specialization of the algorithm presented in [14] for sampling from a junction tree. In [12] an algorithm is presented for sampling without replacement in a junction tree that can be immediately specialized for trees.

Sampling in a tree in the presence of evidence is done, just like marginalization in the presence of evidence, in two steps. First, one incorporates the evidence by the **PropagateEvidence** algorithm; second, sampling by the procedure described above is performed on the resulting conditional distribution.

2.3 Learning trees in the Maximum Likelihood framework

2.3.1 Problem formulation

First, I shall formulate the learning problem as a Maximum Likelihood (ML) estimation task.

Assume a domain V and a set of observations from V called a *dataset* $\mathcal{D} = \{x^1, x^2, \ldots x^N\}$. We further assume that these data were generated by sampling independently from an unknown tree distribution T^0 over V. The learning problem consists in finding the generative model T^0. According to the Maximum Likelihood principle the estimate of T^0 from $\mathcal{D}$ is the model that maximizes the probability (or likelihood) of the observed data. Equivalently, one can search to optimize the logarithm of the likelihood, called *log-likelihood*, which leads us to formulate the *ML Learning Problem* for trees as follows:

Given a domain V and a set of complete observations $\mathcal{D}$, find a tree distribution T^* for which

$$T^* = \operatorname*{argmax}_{T} \sum_{x \in \mathcal{D}} \log T(x). \tag{2.12}$$

2.3.2 Fitting a tree to a distribution

The solution to the ML Learning Problem has been published in [7] in the broader context of finding the tree that best fits a given distribution P over the dataset $\mathcal{D}$. The goodness of fit is evaluated by the Kullback-Leibler (KL) divergence [43] between P and T :

$$KL(P\,||\,T) = \sum_{x \in \mathcal{D}} P(x) \log \frac{P(x)}{T(x)} \tag{2.13}$$

Since the Chow and Liu algorithm will constitute a building block for the algorithms that will be developed in my thesis, I shall present it and its derivation here. The impatient reader can skip to the next subsection.

Let us start by examining (2.13). It is known [11] that for any two distributions P and Q, $KL(P||Q) \geq 0$ and that equality is attained only for $Q \equiv P$. The KL divergence can be rewritten as

$$\begin{aligned} KL(P\,||\,Q) &= \sum_{x} P(x)[\log P(x) - \log Q(x)] \\ &= \sum_{x} P(x) \log P(x) - \sum_{x} P(x) \log Q(x) \end{aligned} \tag{2.14}$$

Notice that the first term above does not depend on Q. Hence, minimizing the KL divergence w.r.t. Q is equivalent to maximizing the second term of (2.14) (called the *cross-entropy* between P and Q) and we know that this is achieved for $Q = P$.

Now, let us return to our problem of fitting a tree to a fixed distribution P. Finding a tree distribution requires finding its structure (represented by the edge set E) and the corresponding parameters, i.e. the values of $T_{uv}(x_u, x_v)$ for all edges $(uv) \in E$ and for all values x_u, x_v.

Assume first that the structure E is fixed and expand the right-hand side of (2.13):

$$\begin{aligned} &KL(P\,||\,T) = \\ &= \sum_{x \in \mathcal{D}} P(x)[\log P(x) - \log T(x)] \\ &= -\sum_{x \in \mathcal{D}} P(x) \log \frac{1}{P(x)} - \sum_{x \in \mathcal{D}} P(x) \log[\prod_{v \in V} T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})] \\ &= -H(P) - \sum_{v \in V} \sum_{x \in \mathcal{D}} P(x) \log T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) \end{aligned} \tag{2.15}$$

$$= \quad -H(P) - \sum_{v \in V} \sum_{x_v, x_{\mathrm{pa}(v)}} P_{v,\mathrm{pa}(v)}(x_v, x_{\mathrm{pa}(v)}) \log T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$$

$$= \quad -H(P) - \sum_{v \in V} \sum_{x_{\mathrm{pa}(v)}} P_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) \sum_{x_v} P_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) \log T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$$

In the above, $H(P)$ denotes the entropy of the distribution P, a quantity that does not depend on T, and P_{uv}, P_v represent respectively the marginals of $\{u, v\}$, v under P. The inner sums in the last two lines are taken over the domains of v and $\mathrm{pa}(v)$ respectively. When v is a root node, $\mathrm{pa}(v)$ is the void set and its corresponding range has, by convention, one value with a probability of $P_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)}) = 1$. Moreover, note that the terms that depend on T are of the form

$$-\sum_{x_v} P_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)}) \log T_{v|\mathrm{pa}(v)}(x_v|x_{\mathrm{pa}(v)})$$

which differs only by a constant independent of T from the KL divergence

$$KL(P_{v|\mathrm{pa}(v)} \,||\, T_{v|\mathrm{pa}(v)})$$

We know that the latter is minimized by

$$T_{v|\mathrm{pa}(v)}(.\,|\,x_{\mathrm{pa}(v)}) \;\equiv\; P_{v|\mathrm{pa}(v)}(.\,|\,x_{\mathrm{pa}(v)}) \;\; \forall v \in V. \tag{2.16}$$

Hence, for a fixed structure E, the best tree parameters in the sense of the minimum KL divergence are obtained by copying the corresponding values from the conditional distributions $P_{v|\mathrm{pa}(v)}$. Let us make two remarks: first, the identity (2.16) can be achieved for all v and $x_{\mathrm{pa}(v)}$ because the distributions $T_{v|\mathrm{pa}(v)=x_{\mathrm{pa}(v)}}$ are each parameterized by its own set of parameters. Second, from the identity (2.16) it follows that

$$T_{uv} \;\equiv\; P_{uv} \;\; \forall (u, v) \in E \tag{2.17}$$

and subsequently, that the resulting distribution T is the same independently of the choice of the roots. For each structure E we denote by T^E the tree with edge set E and whose parameters satisfy equation (2.17). T^E achieves the optimum of (2.13) over all tree distributions conformal with (V, E).

Now, with the previous results in mind, we shall proceed to the minimization of $KL(P\,||\,T)$ over the tree structures. First, notice that this task is equivalent to maximizing the objective

$$J(E) \;=\; \sum_{x \in \mathcal{D}} P(x) \log T^E(x). \tag{2.18}$$

over all structures E.

Expanding the above formula and using (2.17) we obtain successively:

$$J(E) \;= \tag{2.19}$$

$$= \;\; \sum_{i=1}^{N} P(x^i) \log T^E(x^i)$$

$$= \;\; \sum_{i=1}^{N} P(x^i)[\sum_{(u,v) \in E} \log T^E_{uv}(x^i_u x^i_v) - \sum_{v \in V} (\deg v - 1) \log T^E_v(x^i_v)] \tag{2.20}$$

$$
\begin{aligned}
&= \sum_{i=1}^{N} P(x^i)[\sum_{(u,v)\in E} \log P_{uv}(x_u^i x_v^i) - \sum_{v\in V}(\deg v - 1)\log P_v(x_v^i)] && (2.21)\\
&= \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)[\log P_{uv}(x_u^i x_v^i) - \log P_u(x_u^i) - \log P_v(x_v^i)]\\
&\quad + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x_v^i) && (2.22)\\
&= \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)\log\frac{P_{uv}(x_u^i x_v^i)}{P_u(x_u^i)P_v(x_v^i)} + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x_v^i) && (2.23)\\
&= \sum_{(u,v)\in E}\sum_{x_u x_v} P_{uv}(x_u, x_v)\,\log\frac{P_{uv}(x_u,x_v)}{P_u(x_u)P_v(x_v)} + \sum_{v\in V}\sum_{x_v} P_v(x_v)\log P_v(x_v) && (2.24)\\
&= \sum_{(u,v)\in E} I_{uv} - \sum_{v\in V} H(P_v) && (2.25)
\end{aligned}
$$

Equation (2.20) follows from the undirected tree representation (2.5) of T^E, (2.21) is obtained from (2.20) by taking into account (2.17); equation (2.23) follows from (2.22) by performing a summation over all $x \in \mathcal{D}$ that have the same x_u, x_v and using the definitions of P_{uv} and P_v; finally in equation (2.24) the terms I_{uv} under the first sum sign represent the *mutual information* between the variables u and v under the distribution P:

$$
I_{uv} \;=\; \sum_{x_u x_v} P_{uv}(x_u, x_v)\,\log\frac{P_{uv}(x_u,x_v)}{P_u(x_u)P_v(x_v)} \tag{2.26}
$$

The mutual information between two variables is a quantity that is always non-negative and equals 0 only when the variables are independent.

Remark two important facts about equation (2.25): first, the second sum does not depend on the structure E; second and more importantly, the dependence of $J(E)$ from E is additive w.r.t. to the elements of the set E. In other words, each edge in $(u, v) \in E$ contributes a certain positive amount to $J(E)$ and this amount I_{uv} is always the same, independently of the presence or absence of other edges and of the size of their contributions!

In this situation, maximization of J over all structures can be performed efficiently via a *Maximum Weight Spanning Tree* (MWST) algorithm [10] with weights $W_{uv} = I_{uv},\ u, v \in V$.

The MWST problem is formulated as follows: given a graph $(V, \overline{E})$ and a set of real numbers, called *weights* each corresponding to an edge of the graph, find a tree (V, E), $E \in \overline{E}$ for which the sum of the weights corresponding to its edges is maximized. This problem can be solved by a greedy algorithm that constructs the tree by adding one edge at a time, in decreasing order of the weights $\{W_{uv}\}$. There are several variants of the algorithm: the simplest one, called Kruskal's algorithm, runs in $\mathcal{O}(n^2 \log n)$ time. Note that if all the weights are strictly positive, a tree with the maximum number of edges and $p = 1$ connected components will result. If some of the weights W_{uv} are zero, it is possible to obtain trees with more than one connected component. More sophisticated MWST algorithms exist (see for example [10, 65, 25, 20, 60]) and they improve on Kruskal's algorithm on both running time and memory requirements. However, the running time of all published algorithms is at least proportional to the number of candidate edges ($|\overline{E}|$). In our case, this number is equal to $n(n-1)/2$ since all pairs of variables have to be considered. Hence, the best running time achievable for a MWST algorithm will be $\mathcal{O}(n^2)$. Henceforth, we will assume that the MWST algorithm runs in $\mathcal{O}(n^2)$ time and will not further specify the implementation.

All of the above are summarized in the **TreeLearn** algorithm

Algorithm **TreeLearn**

Input	Probability distribution P over domain V
	Procedure MWST(weights) that fits a maximum weight spanning tree over V.
1.	Compute marginals $P_v,\ P_{uv}$ for $u,\ v \in V$.
2.	Compute mutual informations I_{uv} for $u,\ v \in V$.
3.	Call MWST($\{I_{uv}\}$) that outputs the edge set E of a tree distribution T.
4.	Set $T_{uv} \equiv P_{uv}$ for $(uv) \in E$.
Output	T

The algorithm takes as input a probability distribution P over a domain V and outputs a tree distribution T that minimizes the KL divergence $KL(P\,||\,T)$.

The running times for the algorithm's steps are as follows:

steps 1,2. For steps 1. and 2. (computing the marginals and the mutual informations for all pairs of variables) the running time is dependent on the representation of P. But, generally, it should be expected to be $\mathcal{O}(n^2)$, since there are $n(n-1)/2$ mutual information values to be computed.

step 3. The MWST algorithm takes $\mathcal{O}(n^2)$ operations (or $\mathcal{O}(n^2 \log n)$ in Kruskal's variant).

step 4. This step comprises only $\mathcal{O}(nr_{MAX}^2)$ assignments (remember that $|E| < n$).

Hence, the total running time of **TreeLearn** is $\mathcal{O}(n^2 + nr_{MAX}^2)$ or $\mathcal{O}(n^2)$ if we consider r_{MAX} to be a constant.

2.3.3 Solving the ML learning problem

The previous subsection has presented the algorithm **TreeLearn**(P) that finds the tree distribution T^* closest in KL divergence to a given distribution P,

$$T^* \;=\; \operatorname*{argmin}_T \, KL(P\,||\,T).$$

To solve the initial Maximum Likelihood estimation problem it is sufficient to call **Treelearn**$(\overline{P})$ where $\overline{P}$ is the uniform distribution over the data

$$\overline{P}(x) = \begin{cases} \frac{1}{N}, & x \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

To see this, note that

$$KL(\overline{P}\,||\,T) \;=\; -\frac{1}{N}\sum_{i=1}^{N} \log T(x^i) - \log N. \tag{2.28}$$

Hence, minimizing the above expression is equivalent to maximizing the r.h.s. of equation (2.12).

For this case, the first step of the **TreeLearn** algorithm, computing the marginals, takes $\mathcal{O}(Nn^2r_{MAX}^2)$ time. This time dominates the times required for each of the following steps (unless $Nr_{MAX}^2 < \log n$, which is usually not the case) and thus, the running time of the ML tree estimation algorithm is $\mathcal{O}(Nn^2r_{MAX}^2)$.

2.4 Representation capabilities

If graphical representations are easy to grasp by means of human intuition, then the subclass of tree graphical models will be even more intuitive. For once, they are sparse graphs, having $n-1$ or fewer edges. More importantly and more precisely, between each two variables there is at most one path, or, in other words, the separation relationships between subsets of variables, which are not easy to read out in a general Bayes net topology, are obvious in a tree. Thus, in a tree, an edge corresponds to the simplest common sense notion of direct dependency and is the natural representation for it. However, the very simplicity that makes tree models intuitively appealing also limits their modeling power. In other words, the class of dependency structures representable by trees is a relatively small one. For instance, over a domain of dimension n there are n^{n-2} distinct (undirected) spanning trees [71], but a total of $2^{n(n-1)/2}$ undirected graphs[2]. Mixture of trees models, which will be presented next, are a way of circumventing this problem. This thesis will show that mixtures of trees can arbitrarily increase the representation power of tree distributions while sacrificing only little of their computational advantages and, to a certain extent, even preserving their appeal to human intuition.

2.5 Appendix: The Junction Tree algorithm for trees

This section will present the mechanisms for entering evidence in a tree density model and for maintaining consistent representations in the presence of evidence.

Since a tree distribution is also a junction tree, inference will be done via the Junction Tree algorithm using the undirected tree representation (2.5) described previously.

Entering evidence In the general sense, *evidence* on a subset A of variables is defined to be a (possibly unnormalized) probability distribution

$$\mathcal{E}_A \, : \, \Omega(A) \to [0, \infty). \tag{2.29}$$

A *finding* is a special type of evidence: $A = \{v\}$ for some $v \in V$ and $f_v \equiv \mathcal{E}_{\{v\}}$ takes values in the set $\{0, 1\}$ only. A finding on v represents a statement that v cannot take certain values. Any finding f_v can be expressed as a sum of δ functions[3] with as many non-zero terms as the number of non-zero values of f_v.

$$f_v(x_v) \;=\; \sum_{\bar{x}_v \in \Omega v} f_v(\bar{x}_v)\delta_{\bar{x}_v, x_v} \tag{2.30}$$

If $\mathcal{E}_A$ consists of a collection of findings, one for each variable $v \in A$, such that:

$$\mathcal{E}_A \;=\; \prod_{v \in A} f_v \tag{2.31}$$

the evidence is said to be *categorical.* Here and in the rest of the thesis, only categorical evidence will be considered. For handling non-categorical evidence that is contained in one clique, the reader is referred to [38]. [55] discusses another special case of non-categorical evidence that is compatible with the polytree algorithm.

[2] To see that $\alpha = n^{n-2} < \beta = 2^{n(n-1)/2}$ note that α grows like n^n while β grows like $(2^{n/2})^n$ and that $n < 2^{n/2}$ for $n > 2$.

[3] Dirac's symbol $\delta_{\bar{x},x}$ is 1 if $x = \bar{x}$ and 0 otherwise

For any prior distribution P_V of the variables in V and for any categorical evidence $\mathcal{E}_A = \delta_{\bar{x}_A}$, $A \subset V$,

$$P^*_V(x_{V\setminus A}, x_A) \;=\; P_V(x_{V\setminus A}, x_A)\mathcal{E}_A(x_A) \tag{2.32}$$

represents the probability of $x_{V\setminus A}, x_A$ after observing the evidence $\mathcal{E}_A$. This probability is of course 0 for $v \neq \bar{x}_v$. Moreover, the conditional probability of the remaining variables given the evidence is proportional to the above product

$$P^*_{V\setminus A|\mathcal{E}_A}(x_{V\setminus A}) \;\propto\; P_V(x).\mathcal{E}_A(x_A) \tag{2.33}$$

Using the convention $P^*(x_A|\mathcal{E}_A) \;=\; \mathcal{E}_A(x_A)$ we express the posterior conditional distribution of all the variables given the evidence $\mathcal{E}_A$ as

$$P^*_{V|\mathcal{E}_A}(x) \;\propto\; P_V(x).\mathcal{E}_A(x_A). \tag{2.34}$$

The above equations and discussion easily generalizes to the case where $\mathcal{E}_A$ is a product of sums of δ functions, i.e. the case of categorical evidence. Hence, for any categorical evidence $\mathcal{E}_A$ we define the operation of *entering evidence* by equation (2.34).

The normalization constant in equation (2.34) is

$$\sum_x P^*_V(x) \tag{2.35}$$

which represents the prior probability of the categorical evidence $P(\mathcal{E}_A)$ [38, 36]. Hence, the conditional probability of V given the (categorical) evidence $\mathcal{E}_A$ is given by:

$$P^*_{V|\mathcal{E}_A}(x) \;=\; \frac{P_V(x)\mathcal{E}_A(x_A)}{P(\mathcal{E}_A)} \tag{2.36}$$

For a tree distribution T, equation (2.34) rewrites as

$$T^*_{V|\mathcal{E}_A}(x) \;\propto\; T_V(x)\mathcal{E}_A(x_A) \tag{2.37}$$

and the prior of $\mathcal{E}_A$, given by (2.35) with P replaced by T, can be efficiently computed as it will be shown later.

Note also that because the posterior probability obtained by (2.34) is unnormalized, the evidence can be an unnormalized function as well.

Propagating evidence. When (categorical) evidence is entered in a tree T (or in a decomposable model), the resulting distribution $T^*_{V\setminus A|\mathcal{E}_A}$ is factored according to the graph of T, but its factors (i.e. the clique potentials) are neither consistent (as a junction tree) nor normalized. The following step, called tree *calibration* has the purpose to make T^* a normalized and consistent distribution again. The calibration represents the Junction Tree algorithm proper and is applicable in general to any decomposable model, but what will be presented here is its specialization for tree distributions. For the general case and the proofs of consistency, see [36] or [38].

It is important to keep in mind that calibration does not add any information to the model, but merely reorganizes the information already present in order to represent it in a form that is convenient for subsequent use (e.g. marginalizations).

The basic operation in evidence propagation is called *absorption.* We say that edge (vw) absorbs from neighboring edge (uv) when the following procedure is called:

Algorithm **Absorb**

Absorb(u, v, w)

Input	edge potentials T^*_{uv}, T_{vw}, $(uv), (vw) \in E$
1.	$T^*_v \ = \ \sum_{x_u} T^*_{uv}$
2.	$T_v \ = \ \sum_{x_u} T_{uv}$
3.	$T^*_{vw} \ \leftarrow \ T_{vw} \frac{T^*_v}{T_v}$
Output	T^*_{vw}

Note that absorption is an asymmetric operation: **Absorb**(u, v, w) and **Absorb**(w, v, u) produce different results. After an absorption, the potentials T^*_{vw} and T^*_{uv} are consistent. If T^*_{uv} is normalized, so will be T^*_{vw}. If the potentials are already consistent, then an absorb operation in either direction changes nothing. These facts can be easily proved:

$$\begin{aligned}
\sum_{x_w} T^*_{vw}(x_v, x_w) &= \sum_{x_w} T_{vw}(x_v, x_w) \frac{T^*_v(x_v)}{T_v(x_v)} \\
&= \frac{T^*_v(x_v)}{T_v(x_v)} \sum_{x_w} T_{vw}(x_v, x_w) \\
&= \frac{T^*_v(x_v)}{T_v(x_v)} T_v(x_v) \\
&= T^*_v(x_v) \\
&= \sum_{x_u} T^*_{uv}(x_u, x_v)
\end{aligned}$$

The above derivation proves consistency after absorption. Moreover, if the potentials are consistent before absorption, then the fraction $\frac{T^*_v(x_v)}{T_v(x_v)}$ equals 1 for all x_v and absorption in either direction leaves the potentials unchanged. Assume now that T^*_{uv} is normalized. Then

$$\begin{aligned}
\sum_{x_v, x_w} T^*_{vw}(x_v, x_w) &= \sum_{x_v} \sum_{x_w} T^*_{vw}(x_v, x_w) \\
&= \sum_{x_v} T^*_v(x_v) \\
&= \sum_{x_v} \sum_{x_u} T^*_{uv}(x_u, x_v) \\
&= 1
\end{aligned}$$

In the following it will be necessary to use absorption from multiple edges. This is defined as:

Algorithm **Absorb**

Absorb$(u_1, \ldots, u_m, v, w)$

Input	edge potentials $T^*_{u_i v}$, T_{vw}, $(u_i v), (vw) \in E$, $i = 1, \ldots m$
1.	for $i = 1, \ldots, m$
	$\quad T^i_v \ = \ \sum_{x_{u_i}} T^*_{u_i v}$
2.	$T_v \ = \ \sum_{x_u} T_{uv}$
3.	$T_{vw} \ \leftarrow \ T_{vw} \prod_{i=1}^m \frac{T^i_v}{T_v}$
Output	T_{vw}

We say that edge (vw) absorbs from adjacent edges $(u_1 v), \ldots (u_m v)$. Unlike the previous case, after absorption from multiple edges, the potentials involved in the operation are not necessarily consistent. To make them consistent, it would be sufficient for each of the edges $(u_i v)$ to absorb from (vw). As we shall see shortly, this is the idea behind the Junction Tree propagation algorithm: the whole tree is made consistent by a series of absorptions from the periphery towards a "root edge" (or root clique) followed by a second series of absorptions in the opposite direction. The mechanism is implemented by two recursive procedures, **CollectEvidence** and **DistributeEvidence** which are described below.

When **CollectEvidence** from w is called for edge $(v, w) \in E$, then the edge calls **CollectEvidence** from v for all the other edges adjacent to v and then absorbs from them.

Algorithm **CollectEvidence**

CollectEvidence(T, v, w)

Input	factored representation of tree T
	nodes $v, w,\ (vw) \in E$
	let $\mathrm{n}(v) = \{w, u_1, \ldots u_m\}$, $m \geq 0$
1.	for $i = 1, \ldots, m$
	CollectEvidence(T, u_i, v)
2.	**Absorb**$(u_1, \ldots u_m, v, w)$
Output	T

CollectEvidence(v, w) fulfills the task of recursively absorbing from the subtree rooted in node v of edge (vw). **DistributeEvidence**(w, v) has the reverse effect: when it is called from w for edge $(vw) \in E$, all the other edges adjacent to v absorb from (vw) then call **DistributeEvidence** from v.

Algorithm **DistributeEvidence**

DistributeEvidence(T, w, v)

Input	factored representation of tree T
	nodes $v, w,\ (vw) \in E$
	let $\mathrm{n}(v) = \{w, u_1, \ldots u_m\}$, $m \geq 0$
1.	for $i = 1, \ldots, m$
	1.1. **Absorb**(w, v, u_i)
	1.2. **DistributeEvidence**(T, v, u_i)
Output	T

With the procedures above, we can introduce the Junction Tree algorithm, whose specialization for trees (junction trees of cliquesize two) is called here **PropagateEvidence**. The algorithm takes as input a tree distribution T over the space $\Omega(V)$ represented as a set of consistent, normalized marginal distributions T_{uv}, and some categorical evidence $\mathcal{E}_A$, $A \subset V$. It outputs the distribution $T^*_{V|\mathcal{E}_A}$ represented as a consistent, normalized tree distribution with the same structure as T. In this distribution, the marginals corresponding to the variables in A (the "observed" variables) will be δ functions.

Algorithm **PropagateEvidence**

PropagateEvidence$(T, \mathcal{E}_A)$

Input a tree distribution T in factored representation conformal to (V, E)

categorical evidence $\mathcal{E}_A$ on a subset $A \subset V$

1. enter evidence: $T^* \leftarrow T\mathcal{E}_A$
2. choose an edge $(vw) \in E$ as *root edge*
3. **CollectEvidence**(T^*, w, v)
   CollectEvidence(T^*, v, w)
4. normalize: $Z = \sum_{x_v x_w} T^*_{vw}$ $\quad T^*_{vw} \leftarrow \dfrac{T^*_{vw}}{Z}$
5. **DistributeEvidence**(T^*, w, v)
   DistributeEvidence(T^*, v, w)

Output T^*

The proofs of correctness for this algorithm are given in [38, 36]. The normalization constant Z computed in step 4 of the algorithm represents the probability of the evidence $T(\mathcal{E}_A)$. To see this, notice that the potential T^*_{vw} of the root edge does not change after the normalization. Moreover, all the other tree edges absorb, directly or indirectly, from (vw). Therefore, at the end of the algorithm, all the edge potentials will be normalized. Since the tree is consistent, by simple function identification we conclude that indeed each $T^*_{v'w'}, (v'w') \in E$ represents the

The propagation of evidence in a tree distribution takes order $\mathcal{O}(r^2_{MAX}|E|)$ time.