Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science

Proposal for Thesis Research in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Title: **Mixtures of Trees for Density Estimation and Classification**

Submitted by:	**Marina Meilă-Predoviciu**	____________________
	Center for Biological and Computational Learning	(Signature of author)
	Rm E25-201	

Date of Submission: July 20, 1998

Expected Date of Completion: December, 1998

Laboratory where thesis will be done: Center for Biological and Computational Learning

Abstract

One of the challenges of density estimation as it is used in machine learning is that usually the data are multivariate and often the dimensionality is large. Operating with joint distributions over multidimensional domains raises specific problems that are not encountered in the univariate case. Graphical models are representations of joint densities that are specifically tailored to address these problems. They do so by focusing on expressing the dependencies between the variables in the domain, without constraining either the topology of these dependencies, or their functional form. The property of separating the dependency structure from its detailed functional form makes graphical models an excellent support for human intuition and enables one to design general inference algorithms for this model class. However, both inference and learning from data in the most commonly used classes of graphical models of general topology are provably NP-hard.

The purpose of this thesis is to propose and to study a class of graphical models that is rich enough to be useful in practical applications yet admits tractable inference and learning algorithms. This class is the class of mixtures of trees models.

Mixtures of trees inherit from graphical models and the perspective of the reasoning under uncertainty community on one side and from mixture models and the perspective of the classical machine learning community on the other side, thus contributing to bridging the space between these two fields of research.

My thesis will introduce a set of efficient algorithms for inference and learning from data in mixtures of trees models. The emphasis will be on learning from a statistical perspective and in this respect in addition to the basic algorithms: (1) I will formulate and study the learning problem in the Bayesian framework, finding classes of priors over structures and parameters that are compatible with tractable learning algorithms; (2) I will implement and study the performance of mixtures of trees in classification, a common task for a statistical model, and show that tree models implement an implicit variable selection mechanism; (3) I shall evaluate the quality of the learned models on real data sets; (4) I shall approach structure learning in belief networks, an important and provably hard problem, and, based on the mixture of trees learning algorithm, I shall introduce a heuristic for learning a special class of belief networks with hidden variables. This work will be done in the context of discrete variable models, but in some cases, classes of continuous variable models will be considered as well.

1 Searching for computationally tractable graphical models

1.1 Density estimation in multidimensional domains

Probability theory is a powerful and general formalism that has successfully been applied in a variety of scientific and technical fields. In the field of machine learning, and especially in what is known as *unsupervised learning*, the probabilistic approach has proven to be particularly fruitful.

The task of unsupervised learning is that, given a set of observations, or *data*, of producing a *model* or description of the data. It is often the case that the data is assumed to be generated by some [stationary] process and building a model represents building a description of that process. In any definition of learning is present an implicit assumption of redundancy: the assumption that the description of the data is more compact than the data themselves or that the model constructed from the present data can predict [properties of] future observations from the same source.

As opposed to *supervised learning*, where learning is performed in view of a specified task and the data presented to the learner are labeled consequently as "inputs" and "outputs" of the task, in unsupervised learning there are no "output" variables and the the envisioned usage of the model is not known at the time of learning. For example, after *clustering* a data set, one may be interested only in the number of clusters, or in the shapes of the clusters for analysis purposes, or one may want to classify future observations as belonging to one (or more) of the discovered clusters (as in document classification), or one may use the model for lossy data compression (as in vector quantization).

Density estimation is the most general form of unsupervised learning and provides a fully probabilistic approach to unsupervised learning. In density estimation, the knowledge about the domain is represented as a joint probability distribution over the variables in the domain. This allows us to formulate the learning problem in a principled way, as a data compression problem or (equivalently) as a maximum likelihood estimation problem. When prior knowledge exists, it is specified as a prior distribution over the class of models, and the task is one of Bayesian model selection or Bayesian model averaging.

Parametric density estimation with its probabilistic framework also enables us to separate what we consider as "essential" in the description of the data from what we consider inessential or "random".

[The last statement reveals the ill-definedness of "task-free" or purely unsupervised learning: what is essential for one task may be superfluous for another. Probability theory cannot overcome this difficulty but it can provide us with a better understanding of the assumptions underlying the models that we are constructing.]

One of the challenges of density estimation as it is used in machine learning is that usually the data are multivariate and often the dimensionality is large. Examples of domains with typically high data dimensionality are pattern recognition, image processing, text classification, diagnosis systems, computational biology and genetics. Dealing with joint distributions over multivariate domains raises specific problems that are not encountered in the univariate case. Distributions over domains with more than 3 dimensions are hard to visualize and to represent intuitively. If the variables are discrete, the size of the state-space grows exponentially with the number of dimensions. For continuous (and bounded) domains, the number of data points necessary to achieve a certain density of points per unit volume also increases exponentially in the number of dimensions. One other way of seeing this is that if the radius of the neighborhood around a data point is kept fixed while the number of dimensions, in the forthcoming denoted by n, is increasing the relative volume of that neighborhood is exponentially decreasing. This constitutes a problem for non-parametric models. While the possibilities of gathering data are usually limited by physical constraints, the increase in the number of variables leads, in the case of a parametric model class, to an increase in the number of parameters of the model and consequently to the phenomenon of *overfitting*. Moreover, the increased dimensionality of the parameter space may lead to an exponential increase in the computational demands for finding an optimal set of parameters. This ensemble of difficulties related to modelling multivariate data is known as *the curse of dimensionality*.

Graphical models are models of joint densities that, without attempting to eliminate the curse of dimensionality, limit its effects to the strictly necessary. They do so by taking advantage of the independencies existing between (subsets of) variables in the domain. In the cases when the dependencies are sparse (in a way that will be formalized later on) and their pattern is known, graphical models allow for efficient inference algorithms. In these cases, as a side-result, the graphical representation is intuitive and easy to visualize and to manage by humans as well.

1.2 Graphical models of conditional independence. Examples of established belief network classes

Here we introduce graphical models of conditional independence or in short *graphical models*. As [Pearl, 1988] describes them, graphical models are "a computation-minded interpretation of probability theory, an interpretation that exposes the qualitative nature of this centuries-old formalism, its compatibility with human intuition and, most importantly, its amenability to network representation and to parallel and distributed computation.". Graphical models are also known as *belief networks* and in the forthcoming the two terms shall be used interchangeably.

We define *probabilistic conditional independence* as follows: If A, B, C are disjoint sets of variables, we say that A and B are *independent given* C, if the following relationship holds: $P_{AB|C} = P_{A|C} P_{B|C}$ for all configurations of A, B and C. In the graphical models language we say that C *separates* the sets A, B and we write

$$A \perp B \mid C \tag{1}$$

Equivalently, two variable sets are (conditionally) independent when knowing one does not affect our knowledge about the other. Graphical models, by taking independencies into account, avoid processing irrelevant information where possible. This means that using a graphical model is at one's advantage in the measure that the variables in the domain can be grouped into (conditionally) independent subsets. For a domain where there are no independencies a graphical model reduces to a n-way probability table or some other generic function of n variables. But for such a domain, where everything depends on everything, no simple model would do.(true?)

A belief network encodes independencies by means of a graph in the following way: each variable v is associated with a vertex of the graph. The graph's topology is used to represent dependencies. More rigorously, the absence of an edge indicates an independence relationship. This graph is called the *structure* of the belief network. There are two main classes of belief networks:

- Bayes nets
- Markov nets

Bayes nets are belief nets whose structure is a directed graph that has no directed cycles; such a structure is called a *directed acyclic graph*, or shortly a DAG. We denote by $\vec{uv}$ an edge directed from u to v and in this case we

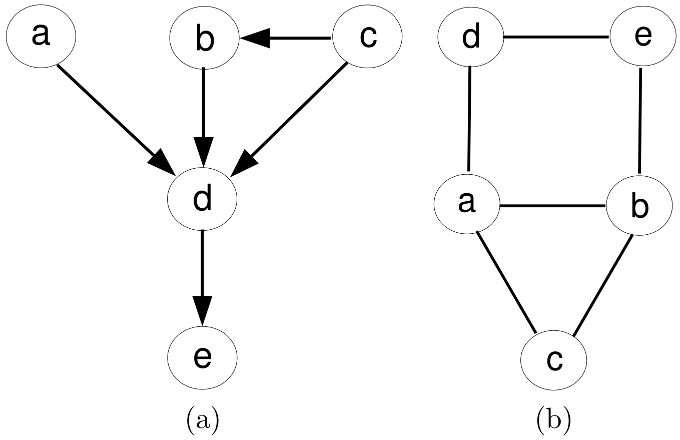

Figure 1: An example of a Bayes net (a) and of a Markov net (b) over 5 variables.

call u the *parent* of v. A *descendent* of u is a variable w that can be reached by a directed path starting at u; hence, the children of u, their children, and so on, are all descendents of u. Figure 1(a) depicts an example of a DAG. The independencies encoded by a Bayes net are summarized by the *directed Markov property*, that states that in a Bayes net, any variable is independent of its non-descendents given its parents. The probability distribution itself is given in a factorized form that corresponds to the independence relationships expressed by the graph. Each factor is a function that depends on one variable and all its parents (we call this set of variables a *family*). For example, in figure 1(a) the families are $\{a\}, \{c\}, \{b, c\}, \{b, c, d\}, \{d, e\}$. The list of independencies encoded by this net includes $a \perp b$, $\{a, b, c\} \perp e|d$.

The second kind of belief nets that are in wide use are Markov nets (better known as Markov random fields). For a Markov net the structure is an undirected graph with arbitrary topology. Two variables that are connected by an edge are *neighbours*. The independencies that a Markov net expresses are summarized by the *global Markov property*: a variable is independent of all the others given its neighbours. Figure 1(b) shows a Markov net and illustrates some of the independence relationships that it encodes. We define a *clique* of an undirected graph to be a maximal subset of variables that are all neighbors of each other. The probability distribution is a product of functions (called *clique potentials*) defined each on a clique of the graph. In the example of figure 1(b) the cliques are $\{a, b, c\}, \{a, d\}, \{d, e\}, \{b, e\}$; $c \perp d, e|a, b$, $d \perp b|a, e$, $e \perp a|b, d$ are among the independencies represented by this net.

In short, for both classes of belief nets, the model is specified in two stages: first, a graph defines the structure of the model and implicitly a set of subsets of “closely connected” variables (families in one case, cliques in the other); second, the probability distribution is expressed as a product of functions of the previosly defined variable subsets and some parameters. We say that the distribution factors according to the graph.

The number of variables in each factor is essential for the efficiency of the computations carried on by the model. The fewer variables in any subset, the more efficient the model. The totality of the parameters corresponding to the factor functions forms the parameter set associated with the given graph structure.

Any probability distribution P that can be factored according to a graph G (DAG or undirected) and thus possesses all the independencies encoded in it is called *conformal* to G. The graph G is called an *I-map* of P. If G is an I-map of P then only the independencies represented in G will be useful from the computational point of view even though a belief net with structure G will be able to represent P exactly. If the distribution P has no other dependencies then those represented by G, G is said to be a *perfect map* for P. For any DAG or undirected graph there is a probability distribution P for which G is a perfect graph. The converse is not true: there are distributions whose set of independencies does not have a perfect map.

Bayes nets and Markov nets represent distinct but intersecting classes of distributions. A probability distribution that can be mapped perfectly as both a Bayes net and a Markov net is called a *decomposable model*. Decomposable models enjoy a certain computational simplicity that is exploited by several belief network algorithms and in particular, by the algorithms that this thesis introduces.

1.3 Advantages of graphical models

The advantages of a model that is factorized according to the independencies between variables are of several categories:

- **Flexibility and modeling power.** The flexible dependence topology, complemented by the freedom in the choice of the factor functions, makes belief network a rich and powerful class among probabilistic models. In particular, belief networks encompass and provide a unifing view of several other model classes (including some that are used for supervised learning): hidden Markov models, Boltzmann machines, stochastic neural networks, Helmholtz machines, decision trees, mixture models, naive-Bayes models.

More important perhaps than the flexibility of the topology is the flexibility in the usages that graphical models admit. Because a belief network represents a probability distribution, any query that can be expressed as a function of probabilities over subsets of variables is acceptable. This means that in particular, a belief network can be used as a classifier for any variable of its domain, can be converted to take any two subsets of variables as inputs and outputs respectively and to compute probabilities of the outputs given the inputs or can be used for diagnostic purposes by computing the most likely configuration of a set of variables given another set.

This flexibility is theoretically a property of any probability density model, but not all density representations are endowed with the powerful inference machine that allows one to compute arbitrary conditional probabilities and thus to take advantage of it.

- **The models are easier to understand and to interpret.** A wide body of practical experience shows that graphical representations of dependencies are very appealing to the (non-technical) users of statistical models of data. Bayes nets, which allow for the interpretation of a directed edge $\vec{uv}$ as a causal effect of u on v, are particularly appreciated as a means of *knowledge elicitation* from human domain experts. Moreover, in a Bayes net the parameters represent conditional probabilities; it is found that specifying or operating with conditional probabilities is much easier for a human than operationg with other representations (as for example, specifying a joint probability) [Pearl, 1988].

  The outputs of the model, being probabilities, have a clear meaning. If we give these probabilities the interpretation of *degrees of belief*, as advocated by [Pearl, 1988], then a belief network is a tool for *reasoning under uncertainty.*

- **Avantages in learning the parameters from data for a given structure.** More independencies mean fewer free parameters compared to a *full* probability model (i.e. a model with no independencies) over the same set of variables. Each parameter appears in a function of only a subset of the variables, thus depends on fewer variables. Under certain parametrizations that are possible for any Bayes net or Markov net structure, this allows for independent estimation of parameters in different factors. In general, for finite amounts of data, a smaller number of (independent) parameters implies an increased accuracy in the estimation of each parameter and thus a lower model variance.

- **Hidden variables and missing data.** A *hidden variable* is a variable whose value is never observed[1]. Other variables may on some occasions be observed, but not in others. When the latter happens, we say that the current observation of the domain has a missing value for that variable. If in an observation (or *data point*) no variable is missing, we say that the observation is *complete*. The graphical models framework allows variables to be specified as observed or unobserved for any data point, integrating thus supervised learning and naturally handling both missing data and hidden variables.

1.4 Structure learning in belief networks

Learning the structure of a graphical model from data, however, is not an easy task. [Heckerman et al., 1995] formulates the problem of structure learning in Bayes networks as a Bayesian model selection problem. He shows that under reasonable simplifying assumptions the prior over the parameters of a Bayes network over a discrete variable domain has the form of a Dirichlet distribution. Moreover, given a set of observations, the posterior probability of a network structure can be computed in closed form. Similar results can be derived for continuous variable models with jointly Gaussian distributions [Heckerman, 1995]. However, finding the structure with the highest posterior probability is an intractable task. For general DAG structures, there are no known algorithms for finding the optimal structure that are asymptotically more efficient than exhaustive search.

Therefore, in the majority of the applications, the structure of the model is either assessed by a domain expert, or is learned by examining structures that are close to one elicited from prior knowledge. If neither of these is the case, then usually some simple structure is chosen.

[1] One can ask: why include in the model a variable that is never observed? One reason is that our physical model of the domain postulates the variable, although in the given conditions we cannot observe it directly (e.g. the state of a patients liver is only assessed indirectly, by certain blood-tests). This paper should provide a second reason for why one may want to introduce hidden variables in a model.

1.5 Inference and decomposable models

The task of using a belief network in conjuction with actual data is termed *inference*. In particular, inference means answering a generic query to the model that has the form: Q="what is the probability of variable v having value x_v given that the values of the variables in the subset $V' \subset V$ are known?". The variables in V' and their observed values are referred to as *categorical evidence* and denoted by $\mathbf{e}$. In a more general setting, one can define evidence to be a probability distribution over V' (called a *likelihood*) but this case will not be considered here. Thus, inference in the restricted sense can be formally defined as computing the probabilily $P(v = x_v|\mathbf{e})$ in the current model. This query is important for two reasons: One, the answers to a wide range of common queries can be formulated as a function of one or more queries of type Q or can be obtained by modified versions of the inference algorithm that solves the query Q. Two, Q serves as a benchmark query for the efficiency of the inference algorithms for a given class of belief networks.

Inference in Bayes nets. In [Pearl, 1988] Pearl introduced an algorithm that performs exact inference in *singly connected* Bayes networks, called by him *polytrees*. A singly connected Bayes net is a network whose underlying undirected graph has no cycles. In such a net there will always be at most one (undirected) path between any two variables u and v. Pearl's algorithm, as it came to be named, assumes that each node can receive/send messages only from/to its *neighbors* (i.e. its parents and children) and that it performs computations based only on the information locally present at each node. Thus it is a *local* algorithm. Pearl proves that it is also asynchronous, exact and that it terminates finite time. The minimum running time is bounded above by the diameter of the (undirected) graph, which in turn is less or equal tothe number of variables n.

The singly-connectedness of the graph is essential for both the finite termination and the correctness of the algorithm's output[2]. For general multiply connected Bayes nets, inference is provably NP-hard [Cooper, 1990].

The standard way of performing inference in a Bayes net of general topology is to transform it into a decomposable model; this is always possible by

[2] It is worth mentioning, however, that recently some impressively successful applications of Pearl's algorithm to Bayes networks with loops have been published. Namely the Turbo codes [Berrou and Glavieux, 1996], Gallager [Gallager, 1963] and Neal-MacKay [MacKay and Neal, 1997] codes that are all based on belief propagation in multiply connected networks. Why Pearl's algorithm performs well in these cases is a topic of intense current research [Weiss, 1997].

a series of edge additions combined with removing the edges' directionality. Adding edges to a graphical model does not change the probability distribution that it represents but will "hide" (and thus make computationally unusable) some of its independencies. Once the decomposable counterpart of the Bayes net is constructed, inference is perfomed via the standard inference algorithm for decomposable models, the *Junction Tree Algorithm* that will be described below.

Inference in decomposable models As defined before, a decomposable model is a belief network whose structure (or, equivalently, qhose set of independencies) can be represented either as a DAG or as an undirected graph. Tree distributions, which will be introduced in the next section, are examples of decomposable models. In a decomposable model, the cliques of the undirected graph representation play a special role. They can be arranged in a tree (i.e acyclic graph) structure called *junction tree*. The vertices of the junction tree are the cliques. The intersection of two cliques that are neighbors in the tree is always non-void and is called a *separator*. Any probability distribution that is factorized according to a decomposable model can be refactorized according to the junction tree into clique potentials ϕ_C and separator potentials ϕ_S in a way that is called *consistent* and that ensures the following important property:

The junction tree consistency property If a junction tree is *consistent* [Jensen et al., 1990, Jensen, 1996] then for each clique $C \subset V$ the marginal probability of C is equal to the clique potential ϕ_C.

From now on, any reference to a junction tree should implicitly assume that the tree is consistent. Consequently, in a consistent junction tree the marginal of any variable v can be computed by marginalization in any ϕ_C for which $v \in C$. If the clique sizes are small relative to n, this represents an important saving w.r.t. the time for computing the marginal but also for other operations necessary in inference. Based on the consistency property of the junction tree, [Jensen et al., 1990] has introduced an inference algorithm known as the *Junction tree algorithm*. The Junction tree algorithm requires only local computations (involving only the variables within one clique). Similarly to Pearl's algorithm for polytrees, information is propagated along the edges of the tree, by means of the separators. The algorithm is exact and finite time; it requires a number of basic clique operations proportional to the number of cliques in the tree. If all the variables of the domain V take values in finite sets, the time required for the basic inference operations in a clique is proportional to the size of *total state space* of the clique (hence it is exponential in cardinality of the clique). The total time required for inference in a junction tree is $\mathcal{O}($ the sum of these state space sizes $)$.

Returning to Bayes nets, it follows that the time required for inference by the junction tree method is exponential in the size of the largest clique of the resulting junction tree. It is hence desirable to minimize this quantity (or alternatively the sum of the state space sizes) in the process of constructing the decomposable model. However, the former objective is provably NP-hard (being equivalent to solving a max-clique problem) [Becker and Geiger, 1996]. It is expected, but not yet known, that the alternative objective is also an NP-hard problem. Moreover, it has been shown by [Jensen and Jensen, 1994] that any exact inference algorithm based on local computations is as least as hard as the junction tree algorithm and thus also NP-hard.

Inference in Markov nets. [Geman and Geman, 1984] showed that inference in Markov random fields with arbitrary topology is also intractable. They introduced a Markov chain Monte Carlo sampling technique for computing approximate values of marginal and conditional probabilities in such a network that is known as *simulated annealing*.

Similar approaches of approximate inference by Monte Carlo techniques have been devised and used for Bayes nets of small size also [Gilks et al., 1996, Spiegelhalter et al., 1994].

Approximate inference in Bayes nets is a topic of current research. The existent approaches include: pruning the model and performing exact inferenced on the reduced model [Kjærulff, 1993], cutting loops and bounding the incurred error [Draper and Hanks, 1994], variational methods to bound the node probabilities in sigmoidal belief networks [Jaakkola, 1997, Jordan et al., 1998].

1.6 Goal of this thesis

The previous subsections have presented the challenge of density estimation for multidimensional domains and have introduced graphical models as tools specifically designed for this purpose.

It has been shown that graphical models focus on expressing the dependencies between the variables of the domain, without constraining neither the topology of these dependencies, nor their functional form. The property of separating the dependencey structure from its detailed functional form makes them an excellent support for human intuition, whithout compromising the modelling power of this model class.

The probabilistic semantics of a graphical model makes it possible to separate model learning from its usage: once a belief network is constructed from data, one can use it in any way that is consistent with the laws of probability.

We have seen also that inference in graphical models can be performed efficiently when the models are simple, but that in the general case it is an NP-hard problem. The same holds for learning a belief network structure from data: learning is hard in general, but we shall see that there are classes of structures for which both learning the structure and the parameters can be done efficiently.

It is the purpose of this thesis to propose and to describe a class of graphical models that is rich enough to be useful in practical applications yet admits tractable inference and learning algorithms.

This class is the class of mixtures of trees models. Trees and mixtures of trees will be formally defined in the next section. For now, a simple definition of a tree is a Bayes net in which each node has at most one parent. Thus mixtures of trees inherit from both graphical models and mixture models. My thesis will introduce this model and demonstrate it's usefulness as a tool for density estimation and classification. I will show that mixture of trees models can approximate arbitrarily closely any distribution over a discrete domain. Moreover, a tree is a simple and suggestive representation. In certain cases, a mixture of trees can inherit the intuitiveness and suggestive power from the trees that it is composed of.

Tree distributions admit effective inference and learning algorithms and my thesis will show how these convenient properties transfer to the much richer class of mixtures of trees. Thus, I shall provide a set of computationally efficient algorithms for learning this class of models from data and for using it in tasks like inference or classification.

The difference between the model that I am introducing and each of its "parent" model classes - decomposable belief nets and mixture models - is not large. But the contribution lies partly in the direction of the approach: to construct classes of graphical models with good computational properties. The mixtures of trees extend the graphical models with a new class and, by combining the flexible topology, dependence oriented perspective of belief nets with the rigid toplolgy of mixture models, they bridge two fields that were distinct until recently.

For most of my proposed work, I shall consider domains comprising discrete variables only. By discrete variable it will always be understood a variable taking values in a finite set.

The reason for this restriction lies rather with the general belief network theory then with the direction of this planned research and it of a rather practical then conceptual nature, as it will be explained now. The nature of the operations required by inference algorithms makes them usually impossible to be performed in closed form representable by a computer in the case of continuous domains.

The advantage of a discrete variable model is that its state space is finite and, consequently, the set of all probability measures defined over it is finite-dimensional. Hence, in this fortunate case, the hypotetical model class can include the whole space. In contrast, the space of all probability measures over a continous domain is infinite-dimensional. However, any representation of practical use must be finite-dimensional and thus, must be restricted to some finite dimensional familiy of the whole space. One must be able to maintain a set of members of this family (i.e. the factor functions) and to perform on them the operations required by the inference algorithms. For example, marginalization is basic operation for the junction tree algorithm. To apply this algorithm to a continuous variable model we need to integrate our density model of a clique potential over an arbitrary subset of variables in closed form. Moreover, in order to ensure consistency, the marginal density obtained this way must be identical to the marginal density of the same variables in any other clique that contains them. This imposes strong constraints on the families of densities that can be used, ruling out most of the common ones.

One notable exception to the above is represented by the jointly Gaussian models. For these models marginalization and conditioning can be done in closed form, while consistency is automatically ensured. An extension to this class of models is the Gaussian conditional models [Lauritzen, 1992] that are Bayes nets that admit discrete variables either as neighbors to other discrete variables or as parents for the continuous ones.

2 Trees and their properties

In this section we introduce tree distributions as a subclass of graphical 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 are order n or less when applied to trees. Learning tree distributions will be formulated as Maximum Likelihood (ML) estimation problem. An algorithm will be presented that learns the most likely tree distribution, both structure and parameters, in time and memory proportional to the size of the data set and quadratic in the number of variables n. Then, incorporating prior knowledge is considered and learning is reformulated as a Bayesian estimation problem. Some final considerations on modelling 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 $|V| = n$. Let r_v denote the number of values of variable $v \in V$, x_v a particular value of v, x_A an assignment to the variables in the subset A of 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). The graph (V, E) is called a *tree* if it has no cycles[3]. 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

$$|E| + p \;=\; |V| \tag{2}$$

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 all $v \in V$ and $(uv) \in E$:

$$T_{uv}(x_u, x_v) \;=\; \sum_{x | u = x_u, v = x_v} T(x) \tag{3}$$

$$T_v(x_v) \;=\; \sum_{x | v = x_v} T(x). \tag{4}$$

They must satisfy the consistency condition

$$T_v(x_v) \;=\; \sum_{x_u} T(x_v, x_u) \;\; \forall u, v, \;\; (uv) \in E \tag{5}$$

[3]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 corresponds to what in graph is called a *forest*.

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 factorized as:

$$T(x) \;=\; \frac{\prod_{(u,v)\in E} T_{uv}(x_u, x_v)}{\prod_{v\in V} T_v(x_v)^{\deg v - 1}} \tag{6}$$

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 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*.

It is possible to represent T in terms of conditional probabilities

$$T(x) \;=\; \prod_{v\in V} T_{v|\mathrm{pa}(v)}(x_v | x_{\mathrm{pa}(v)}) \tag{7}$$

The form (7) is obtained from (6) 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. 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

$$\sum_{(u,v)\in E} r_u r_v - \sum_{v\in V} (\deg\, v - 1) r_v - p \tag{8}$$

From the second representation of T one can notice that for a distribution that is conformal with a tree, all the separation properties induced by the tree structure have a corresponding conditional independence property of the distribution. More concretely, 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.

The existence of equivalent DAG and Markov field representations proves also that tree distributions are decomposable models. 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. 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.

2.2 Inference, sampling and marginalization in a tree distribution

Inference. Since a tree distribution is also a junction tree, it follows that inference can be done via the junction tree algorithm. The junction tree representation is identical to (6) whith T_{uv} and T_v,deg$v > 1$ being the clique potentials and the separator potentials respectively.

In the cases in which the evidence is factorable according to the tree, which include categorical evidence, the junction tree propagation in a tree distribution takes order $\mathcal{O}(r_{MAX}^2|E|)$ time. This results in the junction tree representatio of the distribution $T(x_{V\setminus V'}|e)$. After that, finding the marginal probability of any variable involves marginalization over one other variable which takes $\mathcal{O}(r_{MAX})$ operations.

But, since a tree admits also a Bayes net representation which makes it a polytree, one can use Pearl's algorithm for polytrees as well. The advantage is that Pearl's algorithm can take any type of evidence as input.

Sampling in a tree is best performed using the directed tree representation. The value of the root node(s) is sampled from it's marginal distribution. Then, the value of each of the other nodes is sampled from the conditional distribution given its parent $P(v|\text{pa}(v))$, recursively, starting from the root(s). This simple algorithm is the specialization of the algorithm presented in [Dawid, 1992] for sampling from a junction tree. In [Cowell, 1997] is presented an algorithm for sampling without replacement in a junction tree that can be immediately specialized for trees.

Sampling in a tree in the presence of categorical or noncategorical evidence is a more delicate task. A method for achieving this will be presented in this thesis.

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) or by marginalizing in the potential

T_{uv} of the edge incident to v (if v is a leaf node). 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. My thesis will present an algorithm that can efficiently compute pairwise marginal distributions for arbitrary pairs of variables. The algorithm shall be generalized to marginal distributions over arbitrary subsets of V.

Like sampling, computing marginals in the presence of evidence over arbitrary subsets V' is a more complex task. My thesis will show what are the sources of difficulty for this task and will present methods for performing it.

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 unkown 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

$$T^* \;=\; \operatorname*{argmax}_{T} \sum_{x \in \mathcal{D}} \log T(x). \tag{9}$$

2.3.2 Fitting a tree to a distribution

The solution to the ML Learning Problem has been published in [Chow and Liu, 1968] 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 [Kullback and Leibler, 1951] between P and T :

$$KL(P\,||\,T) \;=\; \sum_{x\in\mathcal{D}} P(x)\log\frac{P(x)}{T(x)} \tag{10}$$

Since the Chow and Liu algorithm will constitute a building block for the algorithms that I am proposing to develop 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 (10). It is known that for any two distributions P and Q, $KL(P||Q) \geq 0$ and that the minimum is attained when $Q \equiv P$. The KL divergence can be rewritten as

$$\begin{aligned} KL(P\,||\,Q) &= \sum_x P(x)[\log P(x) - \log Q(x)] \\ &= \sum_x P(x)\log P(x) - \sum_x P(x)\log Q(x) \end{aligned} \tag{11}$$

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 (11) and we know that this is achieved for $Q = P$.

Now, let us return to our original 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 righ-hand side of (10):

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

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

$$T_{v|\mathrm{pa}(v)}(.\,|\,x_{\mathrm{pa}(v)}) \;\equiv\; P_{v|\mathrm{pa}(v)}(.\,|\,x_{\mathrm{pa}(v)}) \;\; \forall v \in V. \tag{12}$$

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, it is always possible to do this, because in the directed representation of a tree the parameter values can be assigned independently. Second, from the identity (12) it follows that

$$T_{uv} \;\equiv\; P_{uv} \;\; \forall (u, v) \in E \tag{13}$$

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 (13).

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

$$J(E) \;=\; \sum_{x \in \mathcal{D}} P(x) \log T^E(x). \tag{14}$$

over all structures E.

Expanding the above formula and using (13) we obtain successively:

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

$$
\begin{aligned}
&= \sum_{i=1}^{N} P(x^i)[\sum_{(u,v)\in E} \log T^E_{uv}(x^i_u x^i_v) - \sum_{v\in V}(\deg v - 1)\log T^E_v(x^i_v)] && (15)\\
&= \sum_{i=1}^{N} P(x^i)[\sum_{(u,v)\in E} \log P_{uv}(x^i_u x^i_v) - \sum_{v\in V}(\deg v - 1)\log P_v(x^i_v)] && (16)\\
&= \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)[\log P_{uv}(x^i_u x^i_v) - \log P_u(x^i_u) - \log P_v(x^i_v)] \\
&\quad + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x^i_v) && (17)\\
&= \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)\log\frac{P_{uv}(x^i_u x^i_v)}{P_u(x^i_u)P_v(x^i_v)} + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x^i_v) && (18)\\
&= \sum_{(u,v)\in E}\sum_{x_u x_v} P_{uv}(x_u, x_v)\ \log\frac{P_{uv}(x_u, x_v)}{P_u(x_u)P_v(x_v)} + \sum_{v\in V}\sum_{x_v} P_v(x_v)\log P_v(x_v) && (19)\\
&= \sum_{(u,v)\in E} I_{uv} - \sum_{v\in V} H(P_v) && (20)
\end{aligned}
$$

Equation (16) is obtained from (15) by taking into account (13); equation (18) follows from (17) 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 (19) the terms I_{uv} under the first sum sign represent the *mutual information* between the variables u and v under the distribution P. The mutual information is a quantity that is always non-negative and equals 0 only when the two variables are independent.

Remark two important facts about equation (20): 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 the *Maximum Weight Spanning Tree* (MWST) algorithm [Cormen et al., 1990] with weights $W_{uv} = I_{uv},\ u, v \in V$.

The MWST algorithm is a greedy algorithm that constructs the tree by adding one edge at a time, in decreasing order of the weigths $\{W_{uv}\}$. 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.

All of the above are summarized in the **TreeLearn** algorithm presented in figure 2. 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 in Kruskal's variant takes $\mathcal{O}(n^2 \log n)$ operations.

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

2.3.3 Solving the ML learning problem

The previous subsection has presented the algorithm **TreeLearn**(P), summarized in figure 2, 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

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

To see this, remark that

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

Hence, minimizing the above expression is equivalent to maximizing the r.h.s. of equation (9).

Figure 2: The **TreeLearn**(P) algorithm.

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

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

2.4 Learning trees in the Bayesian framework

The ML paradigm assumes that the model is estimated based on data only excluding the incorporation of other sources of knowledge. If the latter, also termed as *prior knowledge*, exist and if they can be formulated as a probability distribution over the hypothesized model class, then one can resort to the Bayesian formulation of learning. In this formulation, one considers the *posterior probability* of a tree distribution model T given the data $\mathcal{D}$

$$P(T\,|\,\mathcal{D}) \;\propto\; P(T,\mathcal{D}) \;=\; P(T)\prod_{x\in\mathcal{D}} T(x) \tag{21}$$

In the above, $P(T)$ represents the prior distribution over the class of tree models and the second factor is $P(\mathcal{D}\,|\,T)$, the likelihood of the data given the model T.

In this framework, finding the best model means finding the tree(s) having the highest posterior probability, an approach which is called *Maximum A Posteriori (MAP) estimation.*

I have not found any literature reference to the MAP estimation of tree distributions in particular. However, the theory developed for Bayesian learning of belief networks by [Heckerman et al., 1995] can be specialized for the case of tree distributions. Since this, and everything that pertains to the Bayesian learning of trees applies to the learning of mixtures of trees as well, I shall defer the presentation of this problem to my thesis.

2.5 Representation capabilities

If graphical representations are easy to grasp by means of the human intuition, then the subclass of tree graphical models will be even more intuitive. For once, they are sparse graphs, having $n-1$ or less 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 but a total of $2^{n(n-1)/2}$ undirected graphs[4]. Mixture of trees models, which will be presented next, are a way of circumventing this problem. My 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.

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 modelling tools: they have a flexible topology, are intuitively appealing, sampling and computing likelihoods are linear time, simple efficient algorithms for marginalizing and conditioning ($\mathcal{O}(|V|^2)$ 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 section introduces a more powerful model, the *mixture of trees*. As my 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.

[4]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 $n > 2$.

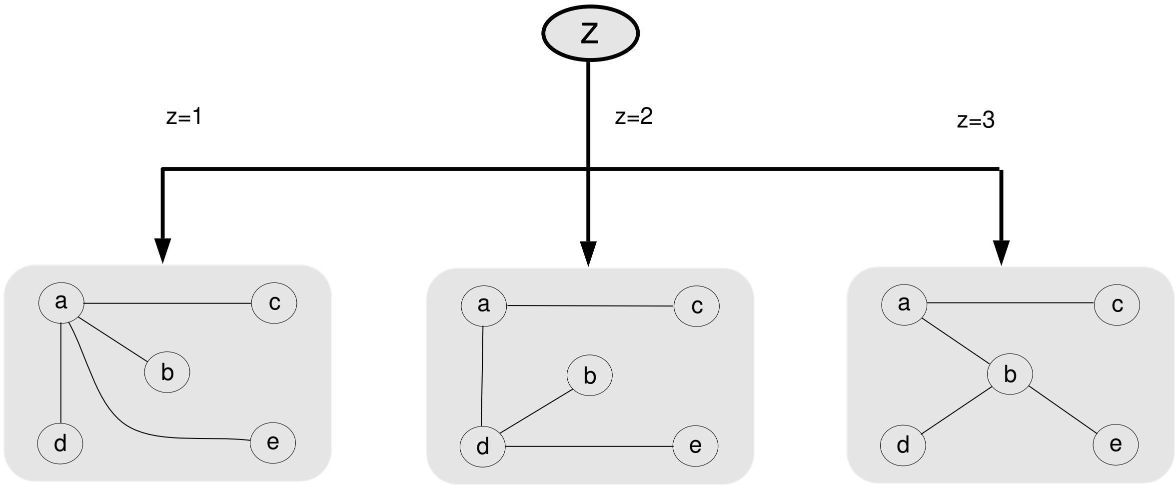

Figure 3: A mixture of trees with $m = 3$ and $n = 5$. Note that although $b \perp_{T^k} c \,|, a$ for all $k = 1, 2, 3$ this does not imply in general $b \perp c \,|\, a$ for the mixture.

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

$$Q(x) \;=\; \sum_{k=1}^{m} \lambda_k T^k(x) \tag{22}$$

with

$$\lambda_k \geq 0,\; k = 1, \ldots, m; \qquad \sum_{k=1}^{m} \lambda_k \;=\; 1. \tag{23}$$

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, taking value $k \in \{1, \ldots m\}$ with probability λ_k. Conditioned on the value of z the distribution of the visible variables x 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 not properly a belief network, but most of the results here owe to the belief network perspective. This means that reading independence relationship from a graphical representation of a mixture of trees like the one in figure 3 (which is obviously NOT the cannonical representation of a Bayes network or a Markov network) should be done cautiously.

Let us adopt the notation

$$A \perp_P B \mid C \tag{24}$$

for "A independent B given C under distribution P". Then, 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. 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 was 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.

4 Plan of the thesis

This thesis proposes to introduce the mixture of trees as a new model for density estimation in multidimensional domains; along with it, it will introduce the necessary learning and inference algorithms for this model class. The properties of the mixture of trees models and algorithms will be studied from the following points of view

- representation power
- computational properties
- quality and usefulness of the learned models on real data and tasks

In the rest of the section I am detailing this plan.

Representation power. I shall prove that any discrete distribution over a domain V can be represented as a mixture of trees.

Basic operations: inference, marginalization and sampling. It has been shown how these operations can be performed in a tree distribution. This thesis will provide efficient algorithms for the same operatins in mixtures of trees. Moreover, it will be shown how to perform marginalization and sampling on trees in the presence of evidence, something for which I haven't found any reference in the literature, and how to extend these methods to mixtures of trees.

Learning. An important part of this work will be devoted to learning mixtures of trees from data. First, learning will be formulated as a **ML estimation** problem and an algorithm will be presented that learns mixtures of trees over discrete domains. The algorithm will be an instance of the EM algorithm for mixtures. I shall show that in the case of mixtures of trees both the E and the M steps can be performed exactly and thus the algorithm's running time for a mixture of m trees is of the order $\mathcal{O}(mn^2r_{MAX}^2N)$ per iteration. In the basic algorithm, each tree in the mixture is allowed to have a different structure. Then, a variation of the same algorithm will be introduced, in which all the trees are constrained to have the **same structure**. Other extension of the same algorithm deal with the case of a **visible choice variable** and with learning **mixtures of Gaussian tree distributions** over continuous domains.

In the second stage, I shall formulate the learning problem in a Bayesian framework as a **MAP estimation** task. I shall show that in principle, the EM approach can be used for this task as well, but that for reasons of computational efficiency the class of priors that can be practically accomodated is restricted to priors that are *factored* in a certain form. I shall show that important classes of priors, like the Dirichlet and MDL priors are members of this class. I shall also present other, non-factorable methods for complexity control, like **smoothing**.

Classification. An important application of a probability density models is pattern recognition or classification. Therefore, this task will receive special attention in my thesis. For any density model and any variable in the domain, classification w.r.t. that variable given all the others can be performed simply by a likelihood ratio method. I shall compare empirically the performance of mixture of trees in classification tasks and of other commonly used classifiers like the naive-Bayes classifier. In the special case when the mixture consists of one tree only, it will be shown that the learned tree distribution provides for an implicit **variable selection method**, whose efficiency will be assessed by experiments.

Hidden variable discovery. A mixture of trees is a model that contains a hidden variable, the choice variable, in a special position w.r.t. the others. Thus, learning a mixture of trees can be viewed as a simple instance of *hidden variable discovery*. My thesis will show how to use a modified version of the mixtures of trees learning algorithm in order to solve a more general hidden variable discovery task: finding a hidden variable that *separates* the visible variables into two or more disjoint and conditionally independent subsets. To my knowledge, this will be the first method to search over belief network structures with hidden variables.

Experiments. Last but not least, the mixtures of trees models and learning algorithms will be evaluated by experiments. The first set of experiments will be aimed at the ability of the learning algorithm to recover the true model, in the limit of large amounts of data. Thus, this set of experiments will be performed with synthetic models and data. Then, using real data sets with unknown generative models I shall compare the mixtures of trees models to other models of common use on density estimation and classification tasks. Finally, I shall test the hidden variable discovery algorithm mentioned above on both synthetic and real datasets.

References

[Becker and Geiger, 1996] Becker, A. and Geiger, D. (1996). A sufficiently fast algorithm for finding close to optimal junction trees. In *UAI 96 Proceedings*.

[Berrou and Glavieux, 1996] Berrou, C. and Glavieux, A. (1996). Near-optimum error correcting coding and decoding Turbo codes. *IEEE Transactions on Communications*, 44:1261–1271.

[Chow and Liu, 1968] Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *"IEEE Transactions on Information Theory"*, IT-14(3):462–467.

[Cooper, 1990] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405.

[Cormen et al., 1990] Cormen, T. H., Leiserson, C. E., and Rivest, R. R. (1990). *Introduction to Algorithms*. MIT Press.

[Cowell, 1997] Cowell, R. (1997). Sampling without replacement in junction trees. Statistical Research Paper 15, City University, London.

[Dawid, 1992] Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36.

[Draper and Hanks, 1994] Draper, D. L. and Hanks, S. (1994). Localized partial evaluation of belief networks. In *Proceedings of the 10th Conference on Uncertainty in AI*. Morgan Kaufmann Publishers.

[Gallager, 1963] Gallager, R. G. (1963). *Low-density parity-check codes*. MIT Press.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

[Gilks et al., 1996] Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., editors (1996). *Markov chain Monte Carlo in practice*. Chapman & Hall, London.

[Heckerman, 1995] Heckerman, D. (1995). A tutorial on learning Bayesian networks. Technical Report MSR–TR–95–06, Microsoft Research.

[Heckerman et al., 1995] Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learining*, 20(3):197–243.

[Jaakkola, 1997] Jaakkola, T. S. (1997). *Variational methods for inference and estimation in graphical models*. PhD thesis, Massachusetts Institute of Technology.

[Jensen, 1996] Jensen, F. V. (1996). *An introduction to Bayesian networks*. Springer.

[Jensen and Jensen, 1994] Jensen, F. V. and Jensen, F. (1994). Optimal junction trees. In *Proceedings of the 10th Conference on Uncertainty in AI*. Morgan Kaufmann Publishers.

[Jensen et al., 1990] Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.

[Jordan et al., 1998] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). *Learning and inference in graphical models*, chapter An introduction to variational methods for graphical models, pages 75–104.

[Kjærulff, 1993] Kjærulff, U. (1993). Approximation of Bayesian networks through edge removals. Technical Report Report, Aarlborg University, Department of Mathematics and Computer Science, 94–2007.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Stat.*, 22:79–86.

[Lauritzen, 1992] Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108.

[MacKay and Neal, 1997] MacKay, D. J. C. and Neal, R. M. (1997). Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33:457–458.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman Publishers, San Mateo, CA.

[Spiegelhalter et al., 1994] Spiegelhalter, D. J., Thomas, A., Best, N. G., and Gilks, W. R. (1994). *BUGS: Bayesian inference Using Gibbs Sampling, Version 3.0.* Medical Research Council Biostatistics Unit, Cambridge.

[Weiss, 1997] Weiss, Y. (1997). Belief propagation and revision in networks with loops. Technical Report AIM–1616,CBCL–155, Massachusetts In-Morgan Kaufmann Publishersstitute of Technology, Artificial Intelligence Laboratory.