Chapter 1

Introduction

Cetatea siderală in stricta-i descărnare
Imi dezveleşte-n Număr vertebra ei de fier.
Ion Barbu
–Pytagora
The astral city in naked rigor
Displays its iron spine - the Number.

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. Expressing the domain knowledge as a probability distribution 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"[1].

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 modeling 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 independences 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 Introduction by example

Before discussing graphical models, let us illustrate the task of density estimation by an example. The *domain* represented in figure 1-1 is the DNA splice junction domain (briefly SPLICE) that will be encountered in the Experiments section. It consists of 61 discrete *variables*; 60 of them, called "site 1", … "site 60" represent consecutive sites in a DNA sequence. They can each take 4 values, denoted by the symbols A, C, G, T representing the 4 bases that make up the nucleic acid. The variable called "junction" denoted the fact that, sometimes, the middle of the sequence represents a *splice junction*. A splice junction is the place where a section of non-coding DNA (called *intron*) meets a section of coding DNA (or *exon*). The "junction" variable takes 3 values: EI (exon-intron) when the first 30 variables belong to the exon and the next 30 are the beginning of the intron, IE (intron-exon) when the reverse happens and "none" whe the presented DNA section contains no junction[2]. An *observation* or *data point* is an observed instantiation for all the variables in the domain

[1]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.

[2]The data are such that a junction appears in the middle position or not at all.

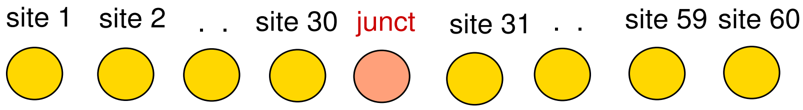

Figure 1-1: The DNA splice junction domain.

(in our case, a DNA sequence together with the corresponding value for the "junction" variable).

In density estimation, the assumption is that the observed data are generated by some underlying probabilistic process and the goal of learning is to reconstruct this process based on the observed data. This reconstruction, or any approximation of thereof, is termed as a *model*. Therefore, in this thesis, a model will be always a joint probability distribution over the variables in the domain. This distribution incorporates our knowledge about the domain. For example, a model for the SPLICE domain, is expected to assign relatively high probability to the sequences that are biologically plausible (among them the sequences observed in the data set) and a probability close to 0 to implausible sequences. Moreover, a plausible sequence coupled with the correct value for the "junction" variable shoud have a high probability, whereas the same sequence coupled with any of the other two values for the "junction" variables should receive a zero probability.

The *state space*, i.e. the set of all possible configurations of the 61 variables, has a size of $3 \times 4^{60} \approx 10^{13}$. It is impossible to explicitly assign a probability to each configuration and therefore we have to construct more compact (from the storage point of view) and tractable (from the computation point of view) representations of probability distributions. To avoid the curse of dimensionality, we require that the models have a number of parameters that is small or slowly increasing with the dimension and that learning the models from data can also be done efficiently. In the next section we shall see that graphical probability models, have the first property but do not fully satisfy the second requirement. Trees, a subclass of graphical models, enjoy both properties but sometimes need to be combined in a mixture to increase their modeling power.

In the present example, there is a natural ordering of the variables in a sequence. In other examples (Digits, Bars) the variables are arranged on a two-dimensional grid. But, in general, it is not necessary to have any spatial relationship between the variables in the domain. In the forthcoming, we will discuss arranging the variables in a graph. In this case, the graph may, but is not required to match the spatial arrangement of the variables.

1.3 Graphical models of conditional independence

1.3.1 Examples of established belief network classes

Here we introduce graphical models of conditional independence or in short *graphical models*. As [55] 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

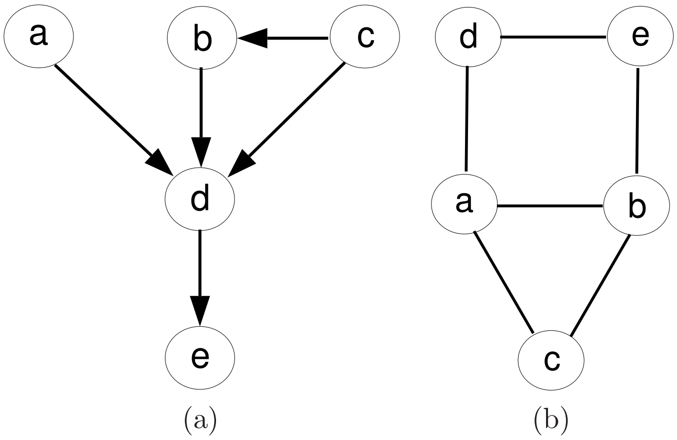

Figure 1-2: An example of a Bayes net (a) and of a Markov net (b) over 5 variables.

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

$$A \perp B \mid C \tag{1.1}$$

Equivalently, two variable sets are (conditionally) independent when knowing one does not affect our knowledge about the other. Graphical models, by taking independences 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 independences a graphical model reduces to a n-way probability table or some other generic function of n variables.

A belief network encodes independences 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. The most common classes of belief networks are:

- Markov nets
- Bayes nets
- decomposable models
- chain graphs

A **Markov net** (better known as a Markov random field) is defined by a structure that is an undirected graph with arbitrary topology. Two variables that are connected by an edge are *neighbors*. The independences that a Markov net expresses are summarized by the *global Markov property*: a variable is independent of all the others given its neighbors. Figure 1-2(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 1-2(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 independences represented by this net.

Thus, the Markov net is specified in two stages: first, a graph describes the structure of the model. The structure implicitly defines the cliques. Second, the probability distribution is expressed as a product of functions of the variables in each clique and some parameters. We say that the distribution factors according to the graph. For example, a distribution that factors according to the graph in figure 1-2 is represented by

$$P \;=\; \phi_{abc}\phi_{ad}\phi_{de}\phi_{be} \tag{1.2}$$

The number of variables in each clique is essential for the efficiency of the computations carried by the model. The fewer variables in any clique, 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.

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 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 1-2(a) depicts an example of a DAG. The independences 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 1-2(a) the families are $\{a\}, \{c\}, \{b, c\}, \{b, c, d\}, \{d, e\}$. The list of independences encoded by this net includes $a \perp b$, $\{a, b, c\} \perp e|d$. Any distribution with this structure can be represented as

$$P \;=\; P_a P_c P_{b|c} P_{d|bc} P_{e|d} \tag{1.3}$$

In short, for both classes of belief nets, the model is specified by first specifying its structure. The structure determines the subsets of "closely connected" variables (cliques in one case, families in the other) on which we further define the factor functions (clique potentials or conditional probabilities) that represent the model's parametrization.

Any probability distribution P that can be factored according to a graph G (DAG or undirected) and thus possesses all the independences 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 independences 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 independences does not have a perfect map.

Decomposable models. 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*. 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* [38, 36] then for each clique $C \subset V$ the marginal probability of C is equal to the clique potential ϕ_C.

The factorization itself has the form

$$P \;=\; \frac{\prod_C \phi_C}{\prod_S \phi_S} \tag{1.4}$$

From now on, any reference to a junction tree should implicitly assume that the tree is consistent. This property endows decomposable models with a certain computational simplicity that is exploited by several belief network algorithms. We will discuss inference in junction trees in section 1.3.4.

Chain graphs [44] are a more general category of graphical models. Their underlying graph comprises both directed and undirected edges. Bayes nets and Markov nets are both subclasses of chain graphs.

1.3.2 Advantages of graphical models

The advantages of a model that is factorized according to the independences 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 unifying 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 operating with other representations (as for example, specifying a joint probability) [55].

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 [55], then a belief network is a tool for *reasoning under uncertainty*.

- **Advantages in learning the parameters from data for a given structure.** More independences mean fewer free parameters compared to a *full* probability model (i.e. a model with no independences) 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[3]. 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.

1.3.3 Structure learning in belief networks

Learning the structure of a graphical model from data, however, is not an easy task. [33] formulates the problem of structure learning in Bayes networks as a Bayesian model selection problem. They show that under reasonable 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 [32]. 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.

[3]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). Another reason is of computational nature: we may introduce a hidden variable because the resulting model explains the observed data well with fewer parameters than the models that do not include the hidden variable. This is often the case for hidden causes: for a second example from the medical domain, a disease is a hidden variable that allows one to describe in a simple way the interplay between a multitude of observation facts called symptoms. In chapter 6 we discuss this issue at length.

1.3.4 Inference and decomposable models

The task of using a belief network in conjunction 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 $\mathcal{E}$. 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 probability $P(v = x_v|\mathcal{E})$ 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 [55] 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 to the 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[4]. For general multiply connected Bayes nets, inference is provably NP-hard [8].

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 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 independences. Once the decomposable counterpart of the Bayes net is constructed, inference is performed via the standard inference algorithm for decomposable models, the *Junction Tree Algorithm* that will be described below.

Inference in junction trees. As defined before, a decomposable model (or junction tree) is a belief network whose cliques form a tree. Tree distributions, which will be introduced in the next section, are examples of decomposable models.

Inference in a graphical model has 3 stages. Here these stages are described for the case of the junction tree and they represent what is known as the *Junction Tree algorithm* [38, 36]:

> **Entering evidence.** This step combines a joint distribution over the variables (which can be thought of as a prior) with evidence (acquired from a different source of information) to produce a posterior distribution of the variables given the evidence.

[4] 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 [3], Gallager [26] and Neal-MacKay [46] 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 [70].

The expression of the posterior is in the same factorized form as the original distribution, but at this stage it does not satisfy all the consistency conditions implicit in the graphical model's (e.g. junction tree's) definition.

Propagating evidence. This is a stage of processing whose final result is the posterior expressed as a consistent and normalized junction tree. Often this phase is referred to as the Junction Tree algorithm proper. 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 $)$.

Extracting the probabilities of the variables of interest by marginalization in the joint posterior distribution obtained at the end of the previous stage. 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. This is also the reason for the locality of the operations necessary in the previous step of the inference procedure.

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) [2]. It is expected, but not yet known, that the alternative objective is also an NP-hard problem. Moreover, it has been shown by [37] 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. [30] 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 [31, 61].

Approximate inference in Bayes nets is a topic of current research. The existent approaches include: pruning the model and performing exact inference on the reduced model [40], cutting loops and bounding the incurred error [19], variational methods to bound the node probabilities in sigmoidal belief networks [35, 39].

1.4 Why, what and where? Goal, contributions and road map of the thesis

The previous sections have presented the challenge of density estimation for multidimensional domains and have introduced 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 dependency structure from its detailed functional form makes them an excellent support for human intuition, whitout compromising the modeling 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 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*.

1.4.1 Contributions

Mixture of trees models . A tree can be defined as a Bayes net in which each node has at most one parent. But trees can represent only acyclic pairwise dependencies and thus have a limited modeling power. By combining tree distributions in a mixture, one can represent any distribution over discrete variables. The number of trees (or mixture components) is a parameter controlling the model's complexity. Mixtures of trees can represent a different set of dependencies than graphical models. From the algorithmic perspective, this thesis shows that the properties of tree distributions, namely the ability to perform the basic operations of computing likelihoods, marginalization and sampling in linear time, directly extend to mixtures.

An efficient learning algorithm. This thesis introduces an efficient algorithm for estimating mixtures of trees from data. The algorithm builds upon a fundamental property of trees, the fact that unlike almost any other class of graphical models, a tree's structure and parameters can be learned efficiently. Embedding the tree learning algorithm in an Expectation-Maximization search procedure produces an algorithm that is quadratic in the domain dimension n and linear in the number of trees m and in the size of the data set N. The algorithm finds Maximum Likelihood estimates but can serve for Bayesian estimation as well. To preserve the algorithm's efficiency in the latter case, one needs to use a restricted class of priors. The thesis characterizes this class as being the class of *decomposable* priors and shows that the restrictions that this class imposes are not stronger than the assumptions underlying the mixture of trees learning algorithm itself. It is also shown that many widely used priors belong to this class.

An accelerated learning algorithm for sparse binary data. If one has in mind application over high-dimensional domains like document categorization and retrieval, preference data or image compression, a quadratic algorithm may not be sufficiently fast. Therefore, I introduce an algorithm that exploits a property of the data – sparsity – to construct a family of algorithms that are jointly subquadratic. In controlled experiments on artificial data

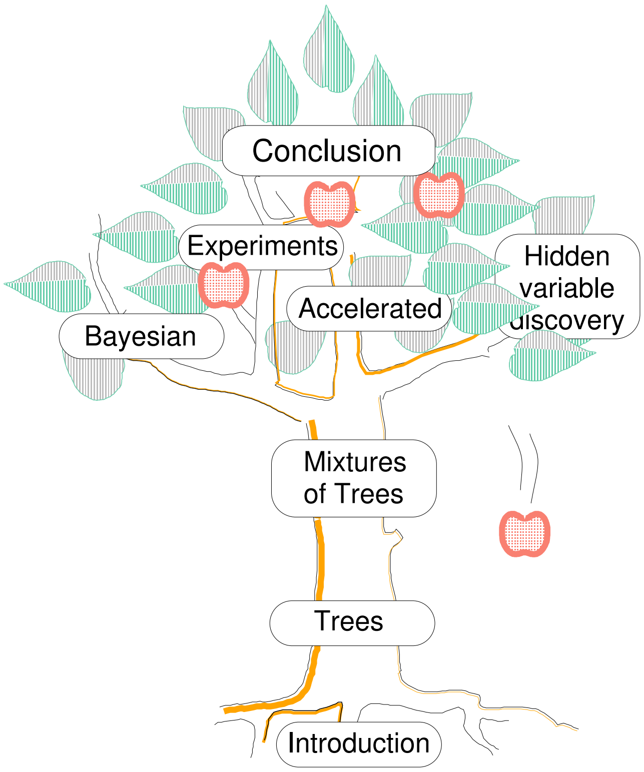

Figure 1-3: Structure of the thesis

the new algorithms achieve speedup factors of up to 3000; the performance is practically independent of n as long as the sparsity and the size of the data set remain constant.

A top-down approach to hidden variable discovery Learning the structure of a graphical model from data is a hard but important problem. The vast majority of algorithms in this field pursue a bottom up strategy in which the basic unit is the graph edge. There is no fundamental reason to make the graph edge play a special role; in fact, it is not the edges but the families or cliques that are the building parts of a graphical model. Therefore, in this thesis I pose the question: can structure search be performed in a top-down manner? I show that there is a hidden variable model called here the H model for which this approach is natural and produces a partition of the variable set into clusters on which structure search can be performed independently. I show then that the ideas underlying the mixture of trees learning algorithm can serve as a basis for variable partitioning as well. The result is a heuristic algorithm for discovering hidden variables in H models. Motivated by the need to validate the models obtained by the hidden variable discovery algorithm, I investigated the use of large deviation theory for testing probabilistic independence between discrete variables and obtained an alternate **distribution free independence test** based on Sanov's theorem. The optimality of the test is under study.

1.4.2 A road map for the reader

The structure of the thesis can be seen in the diagram 1-3. The introduction and chapter 2 lay the foundation by defining the fundamental concepts and by reviewing tree distributions and their inference and learning algorithms.

Chapter 3 builds on this to define the mixture of trees with its variants, to introduce the algorithm for learning mixtures of trees from data in the Maximum Likelihood framework and to show that mixtures of trees can approximate arbitrarily closely any distribution over a discrete domain.

The following chapters will each develop this material into a different direction and thus can be read independently in any order. Chapter 4 discusses learning mixtures of trees in a Bayesian framework. The basic learning algorithm is extended to take into account a class of priors called decomposable priors. It is shown that this class is rich enough to contain important priors like the Dirichlet prior and Minimum Description Length type priors. In this process, the assumptions behind decomposable priors and the learning algorithm itself are made explicit and discussed.

Chapter 5 aims to improve the computational properties of the tree learning algorithm. It introduces a data representation and two new algorithms that exploit the properties of those domains to construct exact Maximum Likelihood trees in subquadratic time. The compatibility of the new algorithms with the EM framework and with the use of decomposable priors are also discussed.

The next chapter, 6, introduces the top-down method for learning structures with a hidden variable, methods for scoring the obtained models and a discussion of the independence test approach to validating them.

Finally there comes the chapter devoted to experimental assessments. Chapter 7 demonstrates the performance of mixtures of trees on various tasks. There are very good results in density estimation, even for data that are not generated from a mixture of trees. The last part of this chapter discusses classification with mixtures of trees. Although the model is a density estimator, its performance as a classifier in my experiments is excellent, even in competition with classifiers trained in supervised mode. I analize the behavior of the single tree classifier and demonstrate that it acts like an implicit feature selector.

The last chapter, 8 contains the concluding remarks.