

Hierarchical Bayesian Networks: an Approach to Classification and Learning for Structured Data

Elias Gyftodimos and Peter A. Flach

Computer Science Department, University of Bristol, UK
{E.Gyftodimos, Peter.Flach}@bristol.ac.uk

Abstract. Bayesian Networks are one of the most popular formalisms for reasoning under uncertainty. Hierarchical Bayesian Networks (HBNs) are an extension of Bayesian Networks that are able to deal with structured domains, using knowledge about the structure of the data to introduce a bias that can contribute to improving inference and learning methods. In effect, nodes in an HBN are (possibly nested) aggregations of simpler nodes. Every aggregate node is itself an HBN modeling independences inside a subset of the whole world under consideration. In this paper we discuss how HBNs can be used as Bayesian classifiers for structured domains. We also discuss how HBNs can be further extended to model more complex data structures, such as lists or sets, and we present the results of preliminary experiments on the mutagenesis dataset.

1 Introduction

Bayesian Networks [16] are a popular framework for reasoning under uncertainty. However, inference mechanisms for Bayesian Networks are compromised by the fact that they can only deal with propositional domains. Hierarchical Bayesian Networks (HBNs) extend Bayesian Networks, so that nodes in the network may correspond to (possibly nested) aggregations of atomic types. Links in the network represent probabilistic dependences the same way as in standard Bayesian Networks, the difference being that those links may lie at any level of nesting into the data structure [8].

An HBN is a compact representation of the full joint probability distribution on the elements of a structured domain. In this respect, HBNs share some similarities with Stochastic Logic Programs (SLPs) [2, 15]. One of the main differences between the two approaches is that SLPs take clausal logic as a starting point and extend it by annotating clauses with probabilities, whereas HBN rather begin from a probabilistic reasoning formalism (standard Bayesian Networks) and extend it to structured domains. Bayesian Logic Programs [9] are also based on clausal logic, but differ from SLPs in that their probabilistic part corresponds to *degrees of belief* of an agent. Probabilistic Relational Models (PRMs) [10], that are a combination of Bayesian Networks and relational models, are also closely related to HBNs. PRMs are based on an instantiation of a relational schema in order to create a multi-layered Bayesian Network, where layers are derived from different entries in a relational database, and use particular aggregation functions in order to model conditional probabilities between elements of different tables. HBNs adopt a method that is more closely related to the particular data structure, by redefining

the probability distribution on the structured domain. Object-oriented Bayesian Networks [11] also combine Bayesian inference with structured data, exploiting data encapsulation and inheritance.

The outline of the paper is as follows. We begin by presenting preliminary terminology and definitions on HBNs in section 2. In section 3 we present our perspective for the extension of HBNs to first-order and higher-order structures. Section 4 shows an adaptation of a popular algorithm for learning standard Bayesian Networks for the case of HBNs. Section 5 discusses Bayesian classification based on HBNs and presents experimental results on the mutagenesis domain. Finally, we summarise our main conclusions and discuss directions for further work.

2 Hierarchical Bayesian Networks: Preliminaries

A standard Bayesian Network is a graphical model that is used to represent conditional independences among a set of variables. It consists of two parts: the *structural* part, a directed acyclic graph in which nodes stand for random variables and edges for direct conditional dependences between them; and the *probabilistic* part that quantifies the conditional dependences, and in the case of discrete variables is a set of *conditional probability tables* (CPTs), each specifying the conditional probability of each value of a variable given the values of its parents in the graph.

The key property in a Bayesian Network is that a variable is independent of its non-descendants given the values of its parents in the graph. This property can be exploited to decompose the full joint probability of all the variables using the chain rule of probabilities: $P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | \pi_i)$, where π_i denotes the set of parents of x_i in the graph.

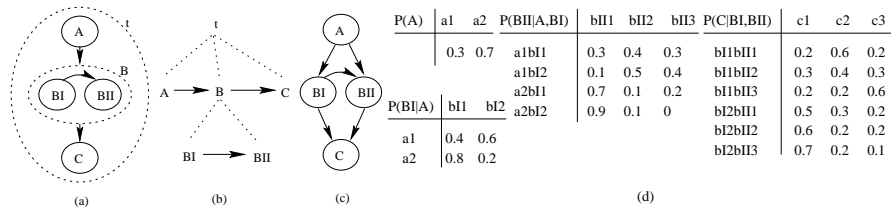


Fig. 1. A simple Hierarchical Bayesian Network. (a) Nested representation. (b) Tree representation. (c) Standard BN expressing the same dependences. (d) Probabilistic part.

Hierarchical Bayesian Networks are a generalisation of standard Bayesian Networks, defined over structured data types. An HBN consists of two parts: the *structural* and the *probabilistic* part. The former (also referred to as the *HBN-tree structure* or simply *HBN structure*) describes the *part-of* relationships and the *probabilistic dependences* between the variables. The latter contains the quantitative part of the conditional probabilities for the variables that are defined in the structural part. In this paper we will restrict our analysis to discrete domains, so the probabilistic part will be a set of

conditional probability tables. Figure 1 presents a simple Hierarchical Bayesian Network. The structural part consists of three variables, A, B and C , where B is itself a pair (BI, BII). This may be represented either using nested nodes (a), or by a tree-like type hierarchy (b). We use the symbol t to denote a top-level composite node that includes all the variables of our world. In (c) it is shown how the probabilistic dependence links unfold if we flatten the hierarchical structure to a standard Bayesian Network.

In an HBN two types of relationships between nodes may be observed: relationships in the type structure (called *t-relationships*) and relationships that are formed by the probabilistic dependence links (*p-relationships*). We will make use of everyday terminology for both kinds of relationships, and refer to *parents, ancestors, siblings, spouses* etc. in the obvious meaning. In the previous example, B has two t-children, namely BI and BII , one p-parent (A) and one p-child (C). The scope of a probabilistic dependence link is assumed to “propagate” through the type structure, defining a set of *higher-level* probabilistic relationships. Trivially, all p-parents of a node are also considered its higher-level parents. For example, the higher-level parents of C are B (as a trivial case), BI and BII (because they are t-children of B and there exists a p-link $B \rightarrow C$).

We will now provide more formal definitions for HBNs. We begin by introducing hierarchical type aggregations, over which Hierarchical Bayesian Networks are defined. Types are recursively defined, in order to represent nested structures, e.g. “a 5-tuple of pairs of booleans”. Currently, the only aggregation operator that we allow for composite types is the Cartesian product, but we plan to extend composite types to include aggregations such as lists and sets, as we discuss in section 3. This will demand a proper definition of probability distribution over these constructs, such as the ones used in the 1BC2 first-order naive Bayesian classifier [12].

Definition 1 (Type). *An atomic type is a domain of constants. If $\{\tau_1, \tau_2, \dots, \tau_n\}$ is a set of types, then the Cartesian product $\tau = \tau_1 \times \tau_2 \times \dots \times \tau_n$ is a composite type. The types $\tau_1, \tau_2, \dots, \tau_n$ are called the component types of τ .*

Definition 2 (Type structure). *The type structure corresponding to a type τ is a tree t such that: (1) if τ is an atomic type, t is a single node labelled τ ; (2) if τ is composite, t has root τ and as children the type structures that correspond to the components of τ .*

Definition 3 (HBN-tree structure). *Let τ be an atomic or composite type, and t its corresponding type structure. An HBN-tree structure T over the type structure t , is a triplet $\langle R, \mathcal{V}, \mathcal{E} \rangle$ where*

- R is the root of the structure, and corresponds to a random variable of type τ .
- \mathcal{V} is a set of HBN-tree structures called the t-children of R . If τ is an atomic type then this set is empty, otherwise it is the set of HBN-tree structures over the component-types of τ . R is also called the t-parent of the elements of \mathcal{V} .
- $\mathcal{E} \subset \mathcal{V}^2$ is a set of directed edges between elements of \mathcal{V} such that the resulting graph contains no directed cycles. For $(v, v') \in \mathcal{E}$ we say that v and v' participate in a p-relationship, or more specifically that v is a p-parent of v' and v' is a p-child of v .

If τ is an atomic type, an HBN-tree structure over t will be called an *HBN-variable*. We will use the term *HBN-variable* to refer also to the random variable of type τ that the root of the structure is associated to.

Definition 4 (Higher-level parents and children). Given an HBN-tree structure $T = \langle R, \mathcal{V}, \mathcal{E} \rangle$ and a t-child of R , $\langle R', \mathcal{V}', \mathcal{E}' \rangle \in \mathcal{V}$, then for any v_P, v_C, v_i such that $(v_P, v) \in \mathcal{E}$, $(v, v_C) \in \mathcal{E}$, $v_i \in \mathcal{V}'$, we say that v_P is a higher-level parent of v_i , and that v_i is a higher-level parent of v_C . Furthermore, if v_{HLP} is a higher-level parent of v , then v_{HLP} is also a higher-level parent of v_i , and if v is a higher-level parent of v_{HLC} then v_i is also a higher-level parent of v_{HLC} .

For an HBN structure we can construct a standard Bayesian Network that maps the same independences between variables. The nodes in the Bayesian Network correspond to variable nodes of the HBN, and links in the Bayesian Network correspond to higher-level links of the HBN. We will call the resulting structure the *corresponding Bayesian Network* of the original HBN.

Definition 5. The HBN-Probabilistic Part related to an HBN-structure T consists of: (1) a probability table for each HBN-variable in T that does not have any p-parents or higher-level parents; (2) a conditional probability table for each other HBN-variable, given the values of all HBN-variables that are its p-parents or higher-level parents.

Definition 6. A Hierarchical Bayesian Network is a triplet $\langle T, \mathcal{P}, t \rangle$ where

- t is a type structure
- $T = \langle R, \mathcal{V}, \mathcal{E} \rangle$ is an HBN-tree structure over t
- \mathcal{P} is the HBN-Probabilistic Part related to T

Definition 7 (Probability distributions over types). If τ is an atomic type, $P_\tau(x)$, $x \in \tau$, is the probability distribution over τ . If $\tau = \tau_1 \times \dots \times \tau_n$ and $x \in \tau$, then $P(x) = P(x_1, \dots, x_n)$, where $x_i \in \tau_i$ are the components of x .

An HBN maps the conditional independences between its variable nodes, in a way that the value of an atomic variable is independent of all atomic variables that are not its higher-level descendants, given the value of its higher-level parents. The independences that an HBN describes can be exploited using the chain rule of conditional probability, to decompose the full joint probability of all the atomic types into a product of the conditional probabilities, in the following way:

$$P(x) = \begin{cases} P_{\tau_x}(x) & \text{if } x \in \tau_x \text{ is atomic} \\ \prod_{i=1}^n P(x_i | \pi_i) & \text{otherwise} \end{cases} \quad (1)$$

where x_1, x_2, \dots, x_n are the components of x and π_i are the p-parents of x_i in the structure.

Example 1. For the HBN structure of Figure 1, we have:

$$\begin{aligned} P(t) &= P(A, B, C) && \text{by Definition 7} \\ &= P(A)P(B|A)P(C|B) && \text{by the chain rule of probability} \\ &= P(A)P(BI, BII|A)P(C|BI, BII) && \text{by Definition 7} \\ &= P(A)P(BI|A)P(BII|BI, A)P(C|BI, BII) && \text{by Equation (1)} \end{aligned}$$

3 Extending HBNs to First-Order and Higher-Order Structures

So far, we have only considered tuples as type aggregations. We will now discuss how more complex type constructors can be embedded into HBNs to allow for handling first-order and higher-order structures, such as lists, trees, sets, etc. We address this issue by introducing what we call *distributions over generic types* for HBN composite nodes. Informally, a generic type is an aggregation of elements of a single component type A , that can be defined as any grammar G over the alphabet A . The notion of a generic type is broad enough to represent data structures of any complexity. It may seem that adopting such a general definition does not serve a practical purpose, but in fact it is necessary in order not to impose an a priori restriction on the expressiveness of the model. Information in the real world may come in very complex structures, such as DNA-sequences, web pages, computer programs or natural language. Furthermore, what is needed in order to build a probabilistic model on such a domain is a proper probability distribution over its elements. In order to deal with a particular domain in HBNs, one needs to define first a specific generic type that represents the domain, and second a probability distribution on the domain, possibly based on a probability distribution of the component type. The probabilistic part of the (extended) HBNs will also include these probability distributions for the generic types they contain. Here follows an example of how such definitions could be provided for lists and sets. The λ -abstraction definition of a set (actually, this defines a finite subset of a possibly infinite domain) is adapted from [14] and the distribution on the set type comes from [3].

Example 2. Let A be a type and $P_A(x)$ a probability distribution over its elements.

Lists The set of lists of elements of A is a type $\tau_{[A]}$ such that

1. $[] \in \tau_{[A]}$
 2. $\forall m \in \mathbb{N}, m > 0, x_1, x_2, \dots, x_m \in A \rightarrow [x_1, x_2, \dots, x_m] \in \tau_{[A]}$.
- A probability distribution over the elements of $\tau_{[A]}$ is given by

$$P_{[A]}([x_1, x_2, \dots, x_l]) = P_{len}(l) \times \prod_i P_A(x_i)$$

where P_{len} is a distribution over integers, that stands for the probability over the lengths of a list (for example, the geometric distribution $P_{len}(l) = p_0(1 - p_0)^l$).

Sets The set of finite sets of elements of A is a type $\tau_{\{A\}}$ such that for all $m \in \mathbb{N}$,

$$\lambda.x.(if\ x \in \{x_1, x_2, \dots, x_m\}\ then\ TRUE\ else\ FALSE)$$

is a member of $\tau_{\{A\}}$. A probability distribution over the elements of $\tau_{\{A\}}$ is given by

$$P_{SS}(S) = \sum_{S' \subseteq S} (-P_{A'}(S'))^{l-l'} \varepsilon \frac{P_{A'}(S')^l}{1 - P_{A'}(S')}$$

where l is the cardinality of S , l' is the cardinality of S' , $P_{A'}(S) = \sum_{x \in S} P_A(x)$, and ε is a parameter determining the probability of the empty set.

The definition of such probability distributions for composite domains shows also how conditional probabilities can be computed for an HBN. Different cases of conditional probabilities may occur in an HBN, with a composite structure conditioned upon another variable, or being on the conditional part itself.

Example 3. Consider a domain of lists labeled according to a class, where the distribution on lists is defined as above and $P_C(c)$ is the distribution on the class attribute, and an HBN modeling that domain with two nodes *List* and *Class*. In case of the p-link *Class*→*List*, we have:

$$P_{[A]}([x_1, x_2, \dots, x_l] | c) = P_{len}(l | c) \times \prod_i P_A(x_i | c)$$

For the conditional probability that corresponds to the link *List*→*Class*, we use Bayes' theorem to obtain:

$$\begin{aligned} P_C(c | [x_1, x_2, \dots, x_l]) &= \frac{P_{[A]}([x_1, x_2, \dots, x_l] | c) \times P_C(c)}{P_{[A]}([x_1, x_2, \dots, x_l])} \\ &= \frac{P_{len}(l | c) \times \prod_i P_A(x_i | c) \times P_C(c)}{P_{len}(l) \times \prod_i P_A(x_i)} \end{aligned}$$

4 Learning HBNs

One important area of concern is the problem of learning HBNs, i.e., given a database of observations, to construct an HBN that fits the data in a satisfactory way. There are several levels at which this problem can possibly be addressed. For instance, we can learn the HBN-probabilistic part for a given HBN-structure; or we can learn both the HBN-probabilistic and structural part, given the type structure. Although trying to learn the type structure as well is in theory possible, we do not address this problem for two reasons. First, the complexity of the task is significant, and combined with learning the HBN structure would give no computational gains over learning a standard Bayesian Network directly. More importantly, we see the type structure as available prior knowledge that is an inherent characteristic of the domain and should be exploited. In our analysis, we assume that there are no missing values in the database, and that different observations in the database occur independently. Learning the probabilistic part can be achieved in a straightforward manner, using the relative frequencies of events in the database in order to estimate the values of the respective conditional probabilities. Given the independence of different instances, the relative frequencies will converge to the actual probability values when the database is sufficiently large. So, when the HBN structure is known and there is enough data available, estimating the conditional probabilities is a relatively straightforward process. We use the Laplace estimate to ensure that even unobserved events will be assigned a non-zero probability.

Deriving the HBN structure from the database is a more complex task. Knowledge of the type structure is exploited in HBNs as a declarative bias, as it significantly reduces the number of possible network structures. For example, the number of possible structures for a Bayesian Network with 10 nodes exceeds 10^{18} , whereas given a type

structure that consists of a pair of 5-tuples the total number of possible HBN structures is approximately 2.3×10^{10} , and for a type structure of a 5-tuple of pairs the number drops to approximately 7×10^6 . Clearly, HBNs subsume standard Bayesian Networks for which the type structure consists of a single level, so the actual gains achieved in a domain depend on the type structure. However, we argue that many domains are naturally structured in a hierarchical way, and it is exactly in those domains that the application of HBNs is a powerful tool. We will discuss two different approaches to the learning problem: a Bayesian scoring criterion, and a minimal description length method.

The first approach to learning the HBN structure is an adaptation of the method described in [1]. We use a Bayesian method to compute the likelihood of a structure given the data, and search for the structure that maximises that likelihood. A restriction of this method is that it requires a set of training data in propositional form, i.e. a single-table database. In [1] a formula is derived to compute $P(B_S, D)$ for a Bayesian Network structure B_S and a database D , depending on the prior $P(B_S)$. That result is based on the assumptions that (a) the variables in the database are discrete, (b) different instances occur independently given the structure, (c) there are no missing values, and (d) that before seeing the database, we consider all the possible conditional probability values setups for a given structure equally likely.

Theorem 1 (Cooper and Herskovits). *Let B_S be a Bayesian Network structure containing n discrete variables x_i , each associated to a domain $(v_{i1}, \dots, v_{ir_i})$, and π_i be the set of parents of x_i in B_S . Suppose D is a database of m instantiations of the variables x_i , and let $(w_{i1}, \dots, w_{iq_i})$ be all the unique instantiations of π_i in D . Let N_{ijk} be the number of cases where $x_i = v_{ik}$ and π_i is instantiated to w_{ij} , and let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. The joint probability of having the structure B_S and the database D is given by:*

$$P(B_S, D) = P(B_S) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

Definition 8. *Let B_{HS} be an HBN structure, and B_S the corresponding Bayesian Network structure of B_{HS} . We define the joint probability of the structure B_{HS} and the database D as $P(B_{HS}, D) = \alpha P(B_S, D)$, where α is a normalising constant such that $\sum_{HS} P(B_{HS}, D) = 1$.*

This means that in order to compute the above joint probability, instead of parents (in the standard Bayesian Network case) we consider higher-level parents. The constant α is needed because there are fewer possible HBN structures than standard BN structures containing the same variables, given the type structure. Additionally, we assume equal prior probabilities among different structures. So, in order to compute the likelihood for an HBN structure, we simply need to extract the corresponding BN structure from it and apply the above formula.

As mentioned above, the application of the Bayesian scoring function requires the data to be in a single-table propositional form. This is in general possible to achieve when the instances are composed of nested tuples, since the total ‘‘length’’ of all instances will be the same, and therefore can be represented in a single relation. The introduction of first-order and higher-order aggregation operators, which is a natural

extension of HBNs as discussed in Section 3, would present a problem since these constructs do not have fixed size and therefore are not representable in a propositional way. For this reason, we discuss another scoring function, based on the minimal description length principle, that deals with data instances regardless from the form of representation. The minimal description length principle (MDL) [17] is based on finding the model that provides the shortest description of the data. The aim is (a) to minimise the size of the model, i.e. the number of parameters needed by the HBN, and (b) find the parameter values that achieve the shortest description of the original data. Here we provide an MDL likelihood function for HBNs, based on a measure used for standard Bayesian Networks [13].

Definition 9. *Let B be a Hierarchical Bayesian Network, formed by the HBN structure B_{HS} and probabilistic part \mathcal{P} . Suppose $D = u_1, u_2, \dots, u_n$ is a set of training data instances, and that the conditional probabilities in \mathcal{P} are estimated by the frequencies of events in D . The MDL scoring of the structure B_{HS} and the database is*

$$M(B_{HS}, D) = \frac{\log n}{2} |B| - \sum_{i=1}^n \log(P_B(u_i))$$

where $|B|$ is the number of parameters in the network and P_B is the distribution over instances that is defined by B .

The first term in the above formula penalises structures with more p-links, while the second is the (negated) log-likelihood measure of the probabilistic part given the data. The MDL function presented here is easily applied to complex data types, as long as we can compute proper probability distributions over them using the HBN.

The next step is to find the HBN structure that optimises the respective measure, i.e. maximises $P(B_{HS}, D)$ or minimises $M(B_{HS}, D)$. Since probabilistic links occur only between siblings in the type structure, this introduces a limit to the number of possible parents we need to consider for each node. The search space of the possible structures is significantly reduced compared to the standard Bayesian Network case. Furthermore, if we apply an ordering on the nodes of the type structure (in fact, we only need an ordering for each subset of siblings) the number of possible structures decreases dramatically. In [1] it is shown that with an ordering on the nodes and a sufficiently tight limit on the number of parents for each node, the derivation of the structure that maximises $P(B_S, D)$ is computationally feasible. Allowing p-links only between t-siblings reduces the maximum number of parents, but with HBNs this is achieved in a domain-specific way, instead of simply imposing a hard limit for all the nodes.

The learning algorithm we introduce is a recursive search for the best possible p-link setup among a set of t-siblings, beginning from the root of the type structure and proceeding towards the leaves. In our current experiments we have used the Bayesian scoring function, but the MDL criterion can be used in a similar way.

Algorithm 1 (HBN-structure learning)

Given: a type structure T ; a node k in the type structure; a (partial) setup E of p-links of the HBN-structure; and a database D of instantiations of the atomic variables.

Output: a setup of p-links over the HBN-structure.

To compute $\text{learn}(T, k, E, D)$:

1. if k is a variable node, return E
2. if k is composite:
 3. find the set E' of p-links between the t -children of k that maximises $P(E \cup E', D)$
 4. for each t -child k_i of k , let E_i be the result of $\text{learn}(T, k_i, E \cup E', D)$
 5. return the set $E \cup E' \cup (\cup_i E_i)$ of all the p-links derived

As an example, suppose we want to reconstruct the HBN structure of Figure 1. The type structure would be known, and a database of instances of values for A, BI, BII and C would be available. The algorithm initially would search among all possible p-link configurations between A, B and C , find the optimal top-level structure, and subsequently extend it by searching among possible configurations of p-links between BI and BII . Typically, for a type structure T with root t , the initial query is $\text{learn}(T, t, \{\}, D)$. However, prior knowledge about p-links that we want to assert that will occur in the solution can be provided at this stage, substituting the empty set with the set of those p-links.

Currently, in order to maximise $P(E \cup E', D)$, we perform an exhaustive search among possible structures. An ordering on the nodes may be taken into account, although one is not necessary. Note that such a search would normally be infeasible if conducted on a standard Bayesian Network. The reduction of the search space, due to hierarchy, allows for a more detailed search. However, if the families in the type structure grow too large, exhaustive search will eventually be rendered infeasible. In such cases, the search space of possible sets of p-links could be traversed using any traditional search method, such as steepest line hill climbing (which is used by Cooper and Herskovits in their $K2$ algorithm), beam search, iterative deepening etc.

5 Classification with HBNs

Bayesian classifiers [7] compute the most likely class of an instance that is described by a vector of attributes (X_1, X_2, \dots, X_n) , i.e. derive the class value that maximises $P(\text{Class}|X_1, X_2, \dots, X_n)$, using Bayes' theorem to invert the conditional probability and then applying a series of independence assumptions to decompose the joint probability to a product of simpler probabilities. The most widely known member of this family is the *Naive Bayes* classifier, that assumes that all attributes are independent of each other given the class, and therefore $P(\text{Class}|X_1, X_2, \dots, X_n) = \alpha \prod_i P(X_i|\text{Class})$ (where α is a normalising constant).

Extensions of the Naive Bayes classifier have been proposed in two orthogonal directions: on the one hand, lifting the "naive" independence assumption and using Bayesian Networks in order to model more complex conditional independences (e.g. *tree-augmented naive Bayesian classifiers* [6]), and on the other hand, using first-order and higher-order representations for classification of structured data (e.g. the *first-order naive Bayesian classifiers* 1BC and 1BC2 [5, 12]). Preliminary experiments show that HBNs can successfully combine both these two directions, using similar probability distributions on structured domains as 1BC2, but with independence assumptions that

are based on Bayesian Network-like directed acyclic graphs. An HBN based classifier uses the decomposition of the posterior probability $P(X_1, X_2, \dots, X_n | Class)$ to a product of simpler probabilities according the independences derived from the HBN structure. In the case of composite nodes, the distribution over the composite type needs to be used as well (e.g. as described in definition 7 and example 2).

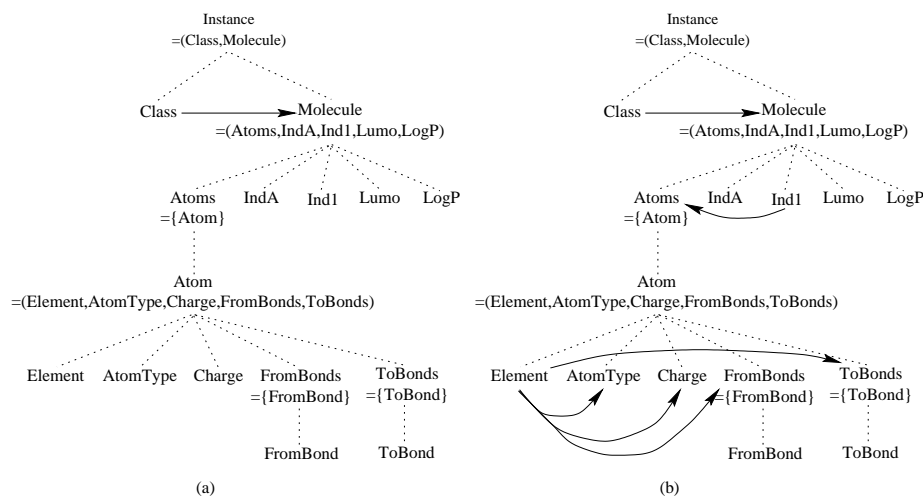


Fig. 2. HBN structures for the mutagenesis domain. (a) Under the naive Bayes assumption. (b) Extended structure.

We have tested our approach on the “regression friendly” Mutagenesis dataset [18]. Instances in this domain are molecular structures, and each one is described by four propositional attributes and a set of atoms. The atoms themselves are characterised by three propositional attributes and two sets of “incoming” and “outgoing” chemical bonds. The task is to predict whether particular molecules are mutagenic or not. For our experiments, we have constructed several HBNs based on the same type structure for instances, and tested different sets of p-links between the nodes. We have employed lists as aggregation operators for atoms and bonds, and used the distribution defined in section 3 to compute the respective conditional probabilities. We present here the results for two such structures, one corresponding to the “naive” assumption of attribute independence given the class, and the other containing a set of p-links that achieved a relatively high accuracy (Figure 2). Reported results correspond to accuracy over 10-fold cross validation. Note that the “naive” HBN is equivalent to the 1BC2 approach. The difference in the accuracy is probably due to a better discretisation of the numeric attributes. Table 1 summarises the results achieved by HBNs and some other approaches. Results for regression, Progol, 1BC and 1BC2 are quoted from [4].

¹ Regression result is based on the propositional attributes only.

Classifier	Accuracy
Majority class	66.5%
Regression ¹	89%
Progol	88%
1BC	87.2%
1BC2	82.4%
Naive HBN	77.7%
Extended HBN	88.3%

Table 1. Accuracy on the Mutagenesis dataset.

6 Conclusions and Further Work

In this paper we have presented Hierarchical Bayesian Networks, a framework for learning and classification for structured data. We have defined a learning method for HBNs based on the Cooper and Herskovits structure likelihood measure.

Presently, we are working towards extending HBNs by introducing more aggregation operators for types, such as lists and sets. Preliminary experiments show that using lists high accuracy can be achieved in comparison to approaches that employ the naive Bayes assumption. Further research is essential in order to create and test more generic type constructors. This will allow the application of our framework to structures of arbitrary form and length, such as web pages or DNA sequences.

Acknowledgements

Part of this work was funded by the EPSRC project *Efficient probabilistic models for inference and learning*. Thanks are due to Koichi Furukawa and Ken Ueno for providing the cello data. Thanks also to Mark Crean for designing and implementing an XML interface for HBNs. Finally, the authors would like to thank the anonymous reviewers for providing useful comments.

References

1. Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
2. James Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
3. Peter A. Flach, Elias Gyftodimos, and Nicolas Lachiche. Probabilistic reasoning with terms. *Linköping Electronic Articles in Computer and Information Science*, 7(011), 2002. Submitted. Available at <http://www.ida.liu.se/ext/epa/cis/2002/011/tcover.html>.
4. Peter A. Flach and Nicolas Lachiche. Naive bayesian classification of structured data. Submitted, 2003.
5. Peter A. Flach and Nicolas Lachiche. 1BC: a first-order Bayesian classifier. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Conference on Inductive Logic Programming*, pages 92–103. Springer-Verlag, 1999.

6. Nir Friedman, Dan Geiger, and Moisés Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
7. Ashutosh Garg and Dan Roth. Understanding probabilistic classifiers. In Luc De Raedt and Peter Flach, editors, *Proceedings of the 12th European Conference in Machine Learning (ECML 2001)*, pages 179–191. Springer, 2001.
8. Elias Gyftodimos and Peter A. Flach. Hierarchical bayesian networks: A probabilistic reasoning model for structured domains. In Edwin de Jong and Tim Oates, editors, *Proceedings of the ICML-2002 Workshop on Development of Representations*. University of New South Wales, 2002.
9. Kristian Kersting and Luc De Raedt. Bayesian logic programs. Technical report, Institute for Computer Science, Machine Learning Lab, University of Freiburg, Germany, 2000.
10. Daphne Koller. Probabilistic relational models. In Sašo Džeroski and Peter A. Flach, editors, *Inductive Logic Programming, 9th International Workshop (ILP-99)*. Springer Verlag, 1999.
11. Daphne Koller and Avi Pfeffer. Object-oriented bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 302–313, 1997.
12. Nicolas Lachiche and Peter A. Flach. 1BC2: a true first-order Bayesian classifier. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, pages 133–148. Springer-Verlag, 2002.
13. Wai Lam and Fahiem Bacchus. Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10:269–294, 1994.
14. J. W. Lloyd. Higher-order computational logic. In A. Kakas and F. Sadri, editors, *Computational Logic: From Logic Programming into the Future*. Springer-Verlag, 2002.
15. Stephen Muggleton. Stochastic logic programs. In Luc de Raedt, editor, *Advances in inductive logic programming*, pages 254–264. IOS press, 1996.
16. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems — Networks of Plausible Inference*. Morgan Kaufmann, 1988.
17. Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
18. A. Srinivasan, S. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237, pages 217–232. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.