Combining Bayesian Networks with Higher-Order Data Representations

Elias Gyftodimos and Peter A. Flach

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

Abstract. This paper introduces Higher-Order Bayesian Networks, a probabilistic reasoning formalism which combines the efficient reasoning mechanisms of Bayesian Networks with the expressive power of higher-order logics. We discuss how the proposed graphical model is used in order to define a probability distribution semantics over particular families of higher-order terms. We give an example of the application of our method on the Mutagenesis domain, a popular dataset from the Inductive Logic Programming community, showing how we employ probabilistic inference and model learning for the construction of a probabilistic classifier based on Higher-Order Bayesian Networks.

1 Introduction

In the past years there has been an increasing interest on methods for learning and reasoning for structured data. Real-world problem domains often cannot be expressed with propositional, "single-table" relational representations. Probabilistic models, popular on propositional domains, have started being proposed for structured domains, giving rise to a new area of research referred to as probabilistic inductive logic programming or probabilistic/statistical relational learning. At an earlier stage of our research [4, 5] we have introduced Hierarchical Bayesian Networks, which define probability distributions over structured types consisting of nested tuples, lists and sets. In this paper we introduce Higher-Order Bayesian Networks (HOBNs), a probabilistic graphical model formalism which is applying methods inspired by Bayesian Networks to complex data structures represented as terms in higher-order logics. We substantially expand our previous research, presenting a detailed formalism for dealing with a much broader family of higher-order terms. The novelty of our approach with respect to existing research on the field consists in the explicit handling of higher-order structures such as sets, rather than emulating these using first-order constructs.

The outline of the paper is as follows. The following section gives a brief overview of the higher-order logic we use for data representation. Section 3 contains the formal definitions of the proposed model, and section 4 defines the derived probability distribution over higher-order terms. Section 5 presents experimental results on a popular real-world benchmark dataset, briefly explaining how inference, model learning and classification is performed under the proposed framework. Section 6 gives a brief overview of existing related approaches. Finally, we summarise our main conclusions and discuss our perspective for further research.

2 Representation of Individuals

Basic terms[9] are a family of typed higher-order terms that can be used for the intuitive representation of structured individuals. Constructs described by basic terms fall into three categories: The first is called *basic tuples* and includes individuals of the form $(t_1,...,t_n)$, where each of the t_i is also a basic term. The second, *basic structures*, describes first-order structures such as lists or trees. A basic structure is a term of the form $Ct_1 \dots t_n$, where C is a *data constructor* (functor) of arity n and the t_i are basic terms. E.g. in order to define lists as in the LISP programming language, we need two data constructors, "*cons*" and "*nil*" of arity two and zero respectively, and the list with elements 1, 2, and 3 is written as "$(cons\ 1\ (cons\ 2\ (cons\ 3\ nil)))$". The third category, *basic abstractions*, is suitable for the description of higher-order structures such as sets and multisets. A set of elements from a domain D can be viewed as a function of type $f : D \to \{\bot, \top\}$ where $f(x) = \top$ if and only if x is a member of the set. In general, a basic abstraction defines the characteristic function of an individual, and is a term t of the form

$$\lambda x.(if\ x = t_1\ then\ s_1\ else \dots else\ if\ x = t_n\ then\ s_n\ else\ s_0)$$

where the t_i and s_i are basic terms and s_0 is a *default term*, i.e. a special term that is the default value of the characteristic function for a particular kind of basic abstraction (zero for multisets, $\bot$ for sets, etc). The set $supp(t) = \{s_1, \dots, s_n\}$ is called the *support set* of t. The cardinality of $supp(t)$ will be called the *size* of the abstraction. The value of the application of the term t to a term u is denoted as $V(t\ u)$. The formal definition of basic terms also contains a definition of the class of default terms as well as the definition of a total order on basic terms so that a basic abstraction can be written in a unique manner with $t_1 < \cdots < t_n$.

Types are used to describe domains of basic terms. A basic tuple type is a Cartesian product $\tau_1 \times \cdots \times \tau_n$ of simpler types to which the elements of a tuple belong. A type of basic structures is defined by a *type constructor*, to which are associated a set of data constructors that define terms of that type. E.g. we can define the type L of lists of elements from a type τ, with two associated data constructors *cons* and *nil*. Data constructors are also typed; for instance *cons* has a function type, accepting two arguments of types τ, L respectively, and its value is of type L. This is noted as $cons : \tau \to L \to L$. The type of a basic abstraction is a function type $\alpha \to \beta$, where α is the type of the argument and β is the type of the value domain of the abstraction. The formal definitions of higher-order types and basic terms can be sought at [9] and go beyond the scope of this paper.

Additionally to standard basic terms, in the present paper we will refer to atomic terms and types. An atomic type is a domain of constants. It can be seen as a special case of a type constructor, to which all the associated data constructors have arity zero. An example of an atomic type is the type of booleans. An atomic term is a member of an atomic type.

We will now define a type tree, which is a tree describing the domain of an individual term.

Definition 1 (Type tree). *The type tree corresponding to a type τ is a tree t such that:*

– *If τ is an atomic type, t is a single leaf labelled τ.*
– *If τ is a basic tuple type $(\tau 1 \times \cdots \times \tau_n)$, then t has root τ and as children the type trees that correspond to all the τ_i.*
– *If τ is a basic structure type with associated data constructors $C_i : \tau_{i_1} \to \cdots \to \tau_{i_m} \to \tau, i = 1, \ldots, n$, then t has root τ. The children of τ are a finite set S of type trees. Every argument τ_{i_j} is mapped to either an element of S or to τ. If two arguments τ_{i_j}, τ_{k_l} are equal types they may be mapped to the same $c \in S$. If an argument τ_{i_j} is equal to the type τ it may be mapped to the root τ. The tree corresponding to each τ_{i_j} is denoted as $c_\tau(i, j)$.*
– *If τ is a basic abstraction type $\beta \to \gamma$, then t has root τ and children the type trees corresponding to β and γ.*

3 Higher-Order Bayesian Networks: Preliminaries

A standard Bayesian Network is a graphical model that encodes the 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 dependence between them; and the *probabilistic* part that quantifies the conditional dependence. Higher-Order Bayesian Networks (HOBNs) are a generalisation of standard Bayesian Networks for basic terms. The structural part of an HOBN is a type tree over the domain, and a set of edges between nodes of the type tree that model correlations between them. The probabilistic part contains the parameters that quantify those correlations. We work under the assumption that domains are discrete. Our example domain comes from the Mutagenesis dataset [12]. Instances in this domain are molecular structures classified as “mutagenic” or “non-mutagenic”, 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. Figure 1 shows an HOBN over that domain. Here *AtomMap* is a boolean variable that corresponds to the target domain of the abstraction *Atoms*.

We will refer to two distinct types of relationships between nodes of an HOBN. Firstly, relationships in the type tree called *t-relationships*. Secondly, 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, nephews* etc. in the obvious meaning. The t-parent and the p-parents of a node are subsequently used for defining the sets of *higher-level parents* (h-parents) and *variable parents* (v-parents) for each node, which in turn are used for the definition of the probabilistic part of the model. In figure 1 node *Molecule* has five t-children *(Atoms, IndA, Ind1, Lumo, LogP)* while node *Element* has one p-parent *(ToBonds)* and two p-children *(AtomType, Charge)*.

Definition 2 (HOBN node and HOBN leaf). *An HOBN node associated to a type τ corresponds to a random variable of that type. We will refer to the HOBN node and the corresponding variable interchangeably. If τ is atomic, the associated HOBN node is called an HOBN leaf.*

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

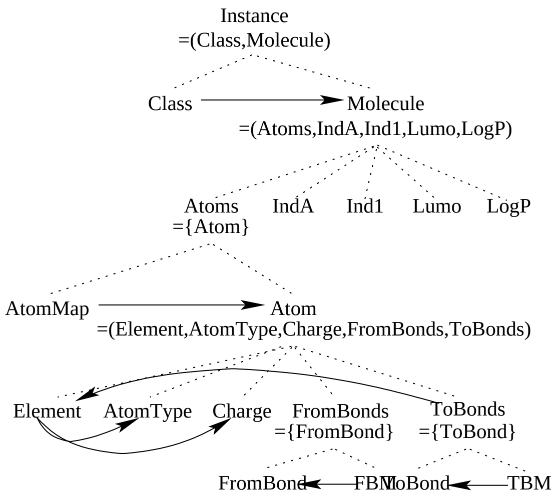

Fig. 1. HOBN structure for the mutagenesis domain.

– *R is the* root *of the structure, and corresponds to a random variable of type τ.*
– *$\mathcal{V}$ is a set of* HOBN structures *called the* t-children *of R. If τ is an* atomic type *then this set is empty, otherwise it is the* set of HOBN structures over the children of τ in t. R is also called the *t-parent* of the elements of $\mathcal{V}$.
– *Let $\mathcal{V}''$ be the set of t-descendants of R. $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}''$ is a set of directed edges between nodes of the HOBN structure. For $(v, v') \in \mathcal{E}$ we say that v is a* p-parent *of v'.*

Intuitively, a node may be the p-parent of its t-siblings and t-nephews, but of no other nodes. There are two additional constraints that a legal HOBN structure must satisfy. One is that if τ is a type $\beta \to \gamma$ corresponding to a basic abstraction, then the set of p-relationships $\mathcal{E}$ for the subtree under τ always contains the p-relationship (γ, β). The second constraint is that the structure needs to be acyclic. This is similar to the acyclicity property in Bayesian Networks, taking into account the propagation of the probabilistic dependence through the type tree. The formal definition of this property follows after the definition of the the notions of higher-level parents and leaf parents, which explain the propagation of the probabilistic dependence introduced by p-relationships down to the leaves of the HOBN structure, so that all dependence can be expressed among pairs of HOBN leaves.

Definition 4 (Higher-level parents and leaf parents). *The higher-level parents (h-parents) of a node t whose t-parent is t', are (i) its p-parents and (ii) the h-parents of t'. The leaf parents (l-parents) of a node are the HOBN leaves that are either (i) its h-parents or (ii) t-descendants of its h-parents.*

It is now possible to define the acyclicity property that was mentioned above:

Definition 5 (Acyclic HOBN structure). *An HOBN structure is acyclic if no HOBN leaf in the structure is an l-ancestor of itself.*

The probabilistic part of an HOBN contains the parameters that quantify the respective joint probabilities, and that are used in deriving a probability over the domain of basic terms that are described by the type tree.

Definition 6. *The probabilistic part related to an HOBN structure T consists of a set of joint probability distributions, defined for some HOBN nodes in T, joint with the respective l-parents of each node. The following probability distributions are contained in the probabilistic part:*

1. *A joint probability distribution over each HOBN leaf X of type α, and its l-parents $X_1,\dots,X_n$, of types $\alpha_1,\dots,\alpha_n$ respectively. For every type $\xi = \alpha_{i_1} \times \dots \times \alpha_{i_m}$, where all i_j are distinct, the conditional probability over the type α given the type ξ is derived by this joint probability and is denoted by $P_{\alpha|\xi}$.*
2. *For each node X associated to a type of basic abstractions τ, a joint probability distribution over the sizes of the basic abstractions that belong to τ , and the l-parents of X, namely $X_1,\dots,X_n$, of types $\alpha_1,\dots,\alpha_n$ respectively. For every type $\xi = \alpha_{i_1} \times \dots \times \alpha_{i_m}$, where all i_j are distinct, the conditional probability over the size of the abstractions of type τ given the type ξ is derived by this joint probability and is denoted by $P_{size_\tau|\xi}$.*
3. *For each node X associated to a type of basic structures τ, a joint probability distribution over the domain $cons_\tau$ which contains the data constructors associated to that type, and the l-parents of X, namely $X_1,\dots,X_n$, of types $\alpha_1,\dots,\alpha_n$ respectively. For every type $\xi = \alpha_{i_1} \times \dots \times \alpha_{i_m}$, where all i_j are distinct, the conditional probability over the size of the abstractions of type τ given the type ξ is derived by this joint probability and is denoted by $P_{cons_\tau|\xi}$.*

As a trivial case when a node has no l-parents the Cartesian product ξ corresponds to the nullary tuple type, and the conditional probability distribution reduces to an unconditional probability distribution.

We can now give the definition of an HOBN:

Definition 7. *A* Higher Order Bayesian Network *is a triplet $\langle T, \Theta, t\rangle$ where t is a type tree, $T = \langle R, \mathcal{V}, \mathcal{E}\rangle$ is an HOBN structure over t and Θ is the probabilistic part related to T.*

4 Probability over basic terms

In this section a probability function over basic terms belonging to a type α is defined, using an HOBN over that type and exploiting the conditional independence assumptions that it introduces. As with standard Bayesian Networks, the joint probability is decomposed to a product of conditional probabilities using the chain rule, and the independent variables are eliminated from the posterior. In HOBNs the probability is defined recursively from the root to the leaves of the type tree, and the probability of a type in each node is expressed using the probabilities of its t-children. Before the formal definition, a short intuitive explanation of the distribution is given, describing how different cases of basic terms are treated.

We are using the notation $P_{\alpha|\xi}(t|c)$ as the probability of a term t of type α, conditional on some context c of type ξ. The definition has two parts: In the first part it is shown how to decompose the probability of t as a product of probabilities of simpler terms $P_{\alpha_i|\xi'}(t_i|c')$, which have types that correspond to the children of α in the type tree. At this stage the conditional part is possibly augmented with additional knowledge on the p-parents of t_i. In the second part of the definition it is shown how in a similar way the probability under the conditional context c is expressed as a product of probabilities under simpler contexts whose types correspond to the children of ξ in the type tree.

The first part of the definition has three different cases, according to t being a basic tuple, basic structure or basic abstraction. In the first case where $t = (t_1, \dots, t_n)$, the probability of the term t is defined as the product of the probabilities of the terms t_i of type $\alpha_i, i = 1, \dots, n$. The conditional independence statements that are derived from the HOBN structure are employed in order to simplify each posterior. In the second case where $t = Ct_1 \dots t_n$, a similar decomposition as with the tuple case is taking place. Each t_i is of a type $\alpha_i, i = 1, \dots, n$, which is either one of the t-children of α or is α itself, for recursive data structures. The probability of t is defined as the product of the probabilities of the t_i conditional on the values of the respective p-parents $P_{\alpha_i|\xi'}(t_i|c, \pi(t_i))$, also multiplied by the probability of the constructor $C, P_{cons_{\alpha}|\xi}(C|c)$. In the third case, where $t = \lambda x.(if\ x = t_1\ then\ v_1 \dots\ else\ if\ x = t_n\ then\ v_n\ else\ v_0)$ the result is based on the product of the probabilities of each t_i conditional on the respective v_i, as is determined by the p-relationship in the HOBN structure.

The second part of the definition assumes that α is an atomic type. The conditional probability is recursively decomposed to a product where the context is replaced by its t-children, until the leaves of the type tree are reached. At each point when deriving the probability $P_{\alpha|\xi}(t|u, c)$, the context is a tuple of terms. The definition introduces a rotation of the tuple elements, by selecting the first element u in the tuple, and creating a new context which is the rest of the tuple with the t-children of the first element attached to the end. This gives a systematic way of reaching the point where the context is a tuple of atomic types. As in the first part of the definition, there are separate cases according to the type of the term in the context that is being decomposed, i.e. the element that is in the first position of the context tuple. If this term is a tuple $u = (u_1, \dots, u_n)$, then the probability $P_{\alpha|\xi}(t|u, c)$ is simplified to $P_{\alpha|\xi'}(t|c, u_1, \dots, u_n)$, where α, ξ, ξ' are the appropriate types. If the term is a basic structure $u = C\ u_1, \dots, u_n$, then the probability is defined using the probabilities $P_{\alpha|\xi_i}(t|c, u_i)$. If the term is a basic abstraction $u = \lambda x.(if\ x = u_1\ then\ v_1 \dots\ else\ if\ x = u_\ell\ then\ v_\ell\ else\ v_0)$, then the probability is defined using the probabilities $P_{\alpha|\xi'}(t|c, u_i, v_i)$. In the course of applying the first part of the definition, a conditional context is introduced for the variable t. This conditional context contains the p-parents of the node corresponding to t. Subsequently t is decomposed to its t-descendents that lay on the leaves of the type tree. The p-parents of t are h-parents of those leaves. Finally, each of those h-parents is replaced by its t-descendents down to the leaves of the type tree. Therefore, after all the decomposition steps, the context is a tuple of atomic types which are the l-parents of α, so the respective conditional probability is contained in the probabilistic part of the model, least a permutation of the l-parents in the context.

Definition 8. *Let t be a basic term whose type α is associated to a node A of the HOBN. By $\pi(t)$ we denote a basic tuple that contains the values of the p-parents of A. The conditional probability function $P_{\alpha|\xi}(t|c)$, where c is a term of type ξ (initially a nullary tuple, then determined by earlier recursive steps), is defined as follows:*

1 *If α is a non-atomic type, corresponding to either a basic tuple, basic structure or basic abstraction domain, then:*

1.a *If $\alpha = \alpha_1 \times \cdots \times \alpha_n$ and $t = (t_1, \ldots, t_n)$, then*

$$P_{\alpha|\xi}(t|c) = \prod_{i=1}^{n} P_{\alpha_i|\xi'}(t_i|c, \pi(t_i))$$

where ξ' is the type of the tuple $(c, \pi(t_i))$.

1.b *If t is a basic structure, $t = C_i t_1 \ldots t_n$, where C_i is the i-th data constructor associated to the type α and is of type $\alpha_1 \to \cdots \to \alpha_n \to \alpha$, then*

$$P_{\alpha|\xi}(t|c) = P_{cons_\alpha|\xi}(C_i|c) \prod_{j=1}^{n} P_{c_\alpha(i,j)|\xi'}(t_j|c, \pi(t_j))$$

where ξ' is the type of the tuple $(c, \pi(t_j))$ and $c_\alpha(i, j)$ is the t-child of α which is mapped to the argument t_j of the data constructor.

1.c *If t is a basic abstraction of type $\alpha = \beta \to \gamma$ with $t = \lambda x.(if\ x = t_1\ then\ v_1 \ldots\ else\ if\ x = t_\ell\ then\ v_\ell\ else\ v_0)$, then*

$$P_{\alpha|\xi}(t|c) = \sum_{\ell'=\ell}^{+\infty} \sum_{*} \ell'! P_{size_\alpha|\xi}(\ell'|c) \prod_{i=1}^{\ell} \frac{(P_{\beta|\xi'}(t_i|c, v_i) P_{\gamma|\xi}(v_i|c))^{x_i}}{x_i!}$$

where ξ' is the type of the tuples (c, v_i) and the summation marked with $(*)$ is over all different integer solutions of the equation $x_1 + \cdots + x_\ell = \ell'$ under the constraints $x_i > 0, i = 1, \ldots, \ell$.

2 *If α is either atomic, abstraction size, or the domain of associated data constructors of a type, then $P_{\alpha|\xi}(t|c)$ is defined as follows (c' may trivially be a nullary tuple):*

2.a *If c is a tuple of atomic types or a nullary tuple, then $P_{\alpha|\xi}(t|c)$ is given in the HOBN probabilistic part.*

2.b *If $c = (u, c'), \xi = \upsilon \times \xi'$, where u is atomic but c' contains non-atomic terms, then*

$$P_{\alpha|\upsilon\times\xi'}(t|u, c') = P_{\alpha|\xi'\times\upsilon}(t|c', u)$$

2.c *If $c = (u, c'), \xi = \upsilon \times \xi'$, where u is a basic tuple $(u_1, \ldots, u_n)$ of type $\upsilon = \upsilon_1 \times \cdots \times \upsilon_n$, then*

$$P_{\alpha|\upsilon\times\xi'}(t|u, c') = P_{\alpha|\xi'\times\upsilon_1\times\cdots\times\upsilon_n}(t|c', u_1, \ldots, u_n)$$

2.d *If $c = (u, c'), \xi = \upsilon \times \xi'$, where u is a basic structure $C_i u_1 \ldots u_n$, and C_i is the i-th data constructor associated to the type υ and is of type $\upsilon_1 \to \cdots \to \upsilon_n \to \upsilon$, then*

$$P_{\alpha|\upsilon\times\xi'}(t|u, c') = P_{\alpha|\xi'}(t|c') \prod_{j=1}^{n} \frac{P_{\alpha|\xi'\times c_\upsilon(i,j)\times\upsilon'_j}(t|c', u_j, \pi(u_j))}{P_{\alpha|\xi'\times\upsilon'_j}(t|c', \pi(u_j))}$$

where υ'_j is the type of $\pi(u_j)$ and $c_\upsilon(i, j)$ is the t-child of υ which is mapped to the argument u_j of the data constructor.

2.e *If $c = (u, c')$, $\xi = \upsilon \times \xi'$, where u is a basic abstraction of type $\upsilon = \beta \to \gamma$ with $u = \lambda x.(if\ x = u_1\ then\ v_1 \dots\ else\ if\ x = u_\ell\ then\ v_\ell\ else\ v_0)$, then*

$$P_{\alpha|\upsilon\times\xi'}(t|u,c') = P_{\alpha|\xi'}(t|c') \sum_{\ell'=\ell}^{+\infty} \sum_{*} \prod_{i=1}^{\ell} \left(\frac{P_{\alpha|\xi'\times\beta\times\gamma}(t|c', u_i, v_i)}{P_{\alpha|\xi'}(t|c')}\right)^{x_i}$$

where the summation marked with $(*)$ is over all different integer solutions of the equation $x_1 + \cdots + x_\ell = \ell'$ under the constraints $x_i > 0, i = 1, \dots, \ell$.

This completes the definition for distributions over basic terms based on an HOBN.

Proposition 1. *The function $P_{\alpha|\xi}$ given in definition 8 is a well-defined probability over basic terms of type α, under the assumption that the conditional independence statements employed hold given the relevant context.*

The proof of the above proposition cannot be presented here due to space constraints. It is derived by applying in each case the chain rule of conditional probability, Bayes' theorem, and using standard combinatorics.

5 Experimental evaluation

We present here the results of the application of our method on a real-world dataset, the Mutagenesis domain [12] described earlier, consisting of a total of 188 instances. The task is to predict whether particular molecules are mutagenic or not. We used 10-fold cross-validation which is customary on this dataset.

The first important issue concerning the application of HOBNs on data analysis is the problem of probabilistic inference, i.e. the calculation of a probability $P(Q|E)$ where Q and E ("query" and "evidence", respectively), are instantiated subsets of the problem domain. The method we are using in HOBNs is a straightforward extension of an approximate inference method for standard BNs: The graphical model is used as a generator of random instances on the domain. If we generate a sufficiently large number of such instantiations, the relative frequency of the cases where both Q and E hold divided by the relative frequency of the cases where E holds will converge to $P(Q|E)$. Probabilistic classification is a direct application of inference, where for each possible class C_i for a test instance T, the value of $P(C_i|T)$ is computed and the C_i which maximises that expression is the respective predicted value of the class. Finally, an area of great interest concerns the construction of an appropriate model given a set of training observations on the domain. In our analysis, we assume that the type tree is an inherent characteristic of the data and therefore is known. What we are learning is the p-relationships between the HOBN nodes, and the parameters of the model. When the HOBN structure is known (i.e. both the type tree and the p-relationships), training the parameters of the model is straightforward when there are no missing values in the data, using the relative frequencies of events in the database in order to estimate the values of the respective joint probabilities. Our approach for structure learning is based on a scoring function for candidate structures. Given such a function we employ a greedy best-first search method, starting from an initial structure (either empty or containing some p-links which are a priori known to be useful) and adding at a time the p-link which optimises the most the scoring function, until no further improvement occurs. The scoring function used in the present experiment was the accuracy of the model on the training data, using cross-validation to avoid over-fitting. The initial structure corresponds to a "naive Bayes" assumption, i.e. that all attributes are mutually independent given the class. This is established by the p-link $Class \rightarrow Molecule$ in the HOBN structure. Table 1 summarises the accuracies obtained in this dataset. We conclude that HOBNs approach the performance of state-of-the-art algorithms in the domain. It is also important that the learning algorithm employed gives a significant improvement compared to a naive Bayes approach.

Classifier	Accuracy
Default	66.5%
Best	89.9%
Naive HOBN	77.7%
Extended HOBN	88.8%

Table 1. Accuracy on the Mutagenesis dataset.

6 Related research

Several logic-based probabilistic models have been proposed in the past. Stochastic Logic Programs [2, 10] use clausal logic, where clauses of a prolog program are annotated with probabilities in order to define a distribution over the Herbrand base of the program. Bayesian Logic Programs [6] are also based on clausal logic, and associate predicates to conditional probability distributions. Independent Choice Logic [11] is another combination of Bayesian Networks and first-order logics. Probabilistic Relational Models (PRMs) [7], 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. Ceci et al. [1] have proposed a naive Bayes classification method for structured data in relational representations. Flach and Lachiche [3, 8] have proposed the systems `1BC` and `1BC2` which implement naive Bayes classification for first-order domains. To our knowledge, Higher-Order Bayesian Networks is the first attempt to build a general probabilistic reasoning model over higher-order logic-based representations.

7 Conclusions and Further Work

In this paper we have introduced Higher Order Bayesian Networks, a framework for inference and learning from structured data. We demonstrated how inference and learning methods can be employed for probabilistic classification under this framework.

Current results are encouraging for further development of HOBNs. More efficient methods for inference, inspired from methods applied to standard Bayesian Networks need to be researched, since these may boost the computational efficiency of the approach. Gradient methods for model training, generalising on the EM method that we are currently investigating, are likely to improve the performance of the model under the presence of missing values.

References

1. Michelangelo Ceci, Annalisa Appice, and Donato Malerba. Mr-sbc: A multi-relational naive bayes classifier. In Nada Lavrač, Dragan Gamberger, Ljupčo Todorovski, and Hendrik Blockeel, editors, *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003)*. Springer, 2003.
2. James Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
3. 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.
4. Elias Gyftodimos and Peter Flach. Learning hierarchical bayesian networks for human skill modelling. In Jonathan M. Rossiter and Trevor P. Martin, editors, *Proceedings of the 2003 UK workshop on Computational Intelligence (UKCI-2003)*. University of Bristol, 2003.
5. Elias Gyftodimos and Peter Flach. Hierarchical bayesian networks: An approach to classification and learning for structured data. In George A. Vouros and Themis Panayiotopoulos, editors, *Methods and Applications of Artificial Intelligence, Third Hellenic Conference on AI (SETN 2004), Proceedings*. Springer, 2004.
6. Kristian Kersting and Luc De Raedt. Bayesian logic programs. Technical report, Institute for Computer Science, Machine Learning Lab, University of Freiburg, Germany, 2000.
7. Daphne Koller. Probabilistic relational models. In Sašo Džeroski and Peter A. Flach, editors, *Inductive Logic Programming, 9th International Workshop (ILP-99)*. Springer Verlag, 1999.
8. 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.
9. John W. Lloyd. *Logic for Learning: Learning Comprehensible theories from Structured Data*. Springer, 2003.
10. Stephen Muggleton. Stochastic logic programs. In Luc de Raedt, editor, *Advances in inductive logic programming*, pages 254–264. IOS press, 1996.
11. David Poole. Logic, knowledge representation, and bayesian decision theory. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Computational Logic, First International Conference (CL-2000), Proceedings*. Springer, 2000.
12. 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.