Machine Learning manuscript No.
(will be inserted by the editor)

Accurate parameter estimation for Bayesian Network Classifiers using Hierarchical Dirichlet Processes

François Petitjean · Wray Buntine · Geoffrey I. Webb · Nayyar Zaidi

Received: date / Accepted: date

Abstract This paper introduces a novel parameter estimation method for the probability tables of Bayesian Network Classifiers (BNCs), using Hierarchical Dirichlet Processes (HDPs). The main result of this paper is to show that proper parameter estimation allows BNCs to outperform leading learning methods such as Random Forest for both 0-1 loss and RMSE, albeit just on categorical datasets.

As data assets become larger, entering the hyped world of "big", accurate classification requires three main elements: (1) classifiers with low-bias that can capture the fine-detail of large datasets (2) out-of-core learners that can learn from data without having to hold it all in main memory and (3) models that can classify new data very efficiently.

The latest Bayesian Network classifiers (BNCs) have these requirements. Their bias can be controlled easily by increasing the number of parents of the nodes in the graph. Their structure can be learned out of core with a limited number of passes over the data. However, as the bias is made lower to accurately model classification tasks, so is the accuracy of their parameters' estimates. In this paper, we introduce the use of Hierarchical Dirichlet Processes for accurate parameter estimation of BNCs.

We conduct an extensive set of experiments on 68 standard datasets and demonstrate that our resulting classifiers perform very competitively with Random Forest in terms of prediction, while keeping the out-of-core capability and superior classification time.

Francois Petitjean
Faculty of Information Technology, Monash University
E-mail: francois.petitjean@monash.edu

Wray Buntine
Faculty of Information Technology, Monash University
E-mail: wray.buntine@monash.edu

Geoffrey I. Webb
Faculty of Information Technology, Monash University
E-mail: geoff.webb@monash.edu

Nayyar Zaidi
Faculty of Information Technology, Monash University
E-mail: nayyar.zaidi@monash.edu

1 Introduction

With the ever increasing availability of large datasets, Bayesian Network Classifiers (BNCs) show great potential because they can be learned out-of-core, i.e. without having to hold the data in main memory. This can be done in a discriminative fashion, for example, TAN (Friedman et al, 1997), kDB (Sahami, 1996) and Selective kDB (Martínez et al, 2016) as well as well as generatively, using fixed-structure models such as Naive Bayes (Lewis, 1998) and Average n-Dependence Estimators – AnDE (Webb et al, 2005, 2012). In contrast Random Forests (RFs) (Breiman, 2001), are not easily learned out-of-core because they require either repeated sorting of the datasets or sampling. For instance, one of RF's state-of-the-art implementation in Mahout (link) side-steps the problem by ensuring the training sets for each tree of the forest is small enough to be in-core.

Constraints on the network structure of BNCs are usually considered to be the main control on their bias-variance trade-off. If the number of parents for nodes is restricted to a relatively low number, then bias will generally be high and the variance on their estimates relatively low (we will actually show in the experiments that the variance can be high even for structures with low complexity). For large datasets, lower bias or higher complexity is preferable because it allows the models to more precisely capture fine detail in the data, translating into higher accuracy (exemplified by the success of deep networks). The number of parameters to estimate increases exponentially with the number of parents allowed for each node; thus, for larger models, accurate estimation of the parameters becomes critical.

We now turn to the aim of this current paper. One of the main issues with low-bias learners is their variance; it is logical that when increasing the number of free parameters, even with the largest possible dataset, there will be a point at which some parameters will not have sufficient examples to be learned with precision. Variance is thus not just a problem for small datasets, but can reappear when designing best-performing learners for large datasets because they require low bias. When the number of examples per parameter decreases, the variance increases because parameter estimation fails to derive accurate estimates. This, of course, is why maximum-likelihood estimates (MLEs) are not often used with low-bias learners unless ensembles are also involved.

Remarkably, experiments in this paper show that for networks as simple as TAN (where each node has two parents at most), which significantly underperform RFs when using Laplace smoothing, can significantly outperform RFs once more careful parameter estimation is performed. This is particularly surprising because one wouldn't expect the variance to be high for models such as TAN. This is due to the fact that the variance is not even among all combinations of feature values and can indeed be relatively high for some of them. We will see that our estimates automatically adapt to cases with high or low variance by making the most of Hierarchical Dirichlet Process (HDP).

Drawing the link between BNCs and HDP: For n-gram models, where one wishes to estimate extremely low-bias categorical distributions and for which very few examples per parameter are available, MLEs have long since been abandoned in favor of sophisticated smoothing techniques such as modified Kneser-Ney (Chen and Goodman, 1996). These, however, have complex back-off parameters that need to be set. For our more general and heterogeneous context of probability table estimation, there exist no techniques to set these parameters. Hierarchical Pitman-Yor Process (HPYP) is the Bayesian version of Kneser-Ney smoothing; it was introduced by Teh (2006) and uses empirical estimates for hyperparameters. This has been demonstrated to be very effective (Wood et al, 2011; Ehsan Shareghi, 2017). HPYP is well-suited for Zipfian contexts. Since we have discrete variables with mostly fewer outcomes (as opposed to n-grams where each word can take 100k+ values) we

do not use the HPYP, and prefer the lower-variance Hierarchical Dirichlet Process (HDP) (Teh et al, 2006) – it is equivalent to HPYP with discount parameter fixed to 0.

In this paper, we propose to adapt the method of Teh (2006) from n-grams and apply it to parameter estimation for BNCs, and use HDPs for the hierarchical probability models. Having showed that our approach outperforms state-of-the-art BNC parameter estimation techniques, we use RF as an exemplar of state-of-the-art machine learning because it is a widely used learning method for the types of tabular data to which our methods are suited which can be used out of the box without need for configuration. We show that our estimator allows BNCs to compete against RFs on categorical datasets. Furthermore, because our method is completely out-of-core, we demonstrate that we can obtain results on large datasets on standard computers with which RF cannot even be trained using standard packages such as Weka. Our models can also classify orders of magnitude faster than RF.

This paper is organized as follows. In Section 2, we review Bayesian Network Classifiers (BNCs). In Section 3 we motivate our use of Hierarchical Dirichlet Processes (HDPs) for BNCs' parameter estimation. We present our method in Section 4 and related work in Section 5. We have conducted extensive experiments, reported in Section 6.

2 Standard Bayesian Network Classifiers

2.1 Notations

The following framework can be found in texts on learning Bayesian networks, such as (Koller and Friedman, 2009). A BN $\mathcal{B} = \langle \mathcal{G}, \Theta \rangle$, is characterized by the structure $\mathcal{G}$ (a directed acyclic graph, where each vertex is an attribute from X), and parameters Θ, that quantifies the dependencies within the structure. The parameter object Θ, contains a set of parameters for each vertex in $\mathcal{G}$: $\theta_{x_i|\Pi_i(\mathbf{x})}$, where $\Pi_i(.)$ is a function which given the datum $\mathbf{x} = \langle x_0, x_1, \ldots, x_n \rangle$ as its input, returns the values of the attributes which are the parents of node i in structure $\mathcal{G}$. Note, each attribute is a Random Variable X_i and x_i represents the value of that random variable. For notational simplicity we write $\theta_{x_i|\Pi_i(\mathbf{x})}$ instead of $\theta_{X_i=x_i|\Pi_i(\mathbf{x})}$, and θ_y instead of $\theta_{y|\Pi_0(\mathbf{x})}$. Also, use $\theta_{X_i|\Pi_i(\mathbf{x})}$ to represent the full vector of values for each x_i. A BN $\mathcal{B}$ computes the joint probability distribution as

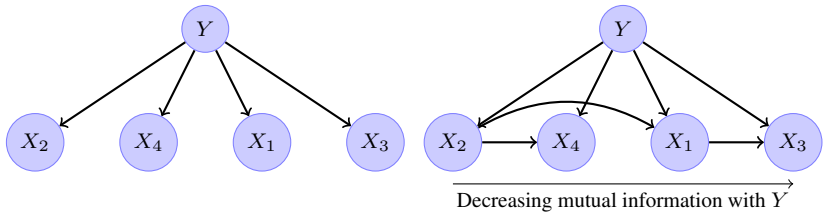
$$\mathrm{P}_{\mathcal{B}}(\mathbf{x}) = \prod_{i=0}^{n} \theta_{x_i|\Pi_i(\mathbf{x})}.$$

The goal of developing a BN classifier is to predict the value of some class variable, say X_0. We will assume that the first attribute is the class attribute and denote it with Y (i.e., $X_0 = Y$), and denote a value for it by y, where $y \in \mathcal{Y}$. For a BN classifier $\mathcal{B}$, we can write:

$$\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) = \frac{\mathrm{P}_{\mathcal{B}}(y, \mathbf{x})}{\mathrm{P}_{\mathcal{B}}(\mathbf{x})} = \frac{\theta_{y|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y,\Pi_i(\mathbf{x})}}{\sum_{y' \in \mathcal{Y}} \theta_{y'|\Pi_0(\mathbf{x})} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_i(\mathbf{x})}}.$$

2.2 Structure Learning for BNCs

Most approaches to learning BNCs learn the structure first and then learn the parameters as a separate step. Numerous algorithms have been developed for learning BNC network structure. The key difference that distinguishes BNC structure learning from normal BN structure



(a) (b)

Fig. 1 Example BNC structures: (a) Naive Bayes, (b) kDB-1

learning is that the precision of the posterior estimates $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x})$ matters rather than the precision of $\mathrm{P}_{\mathcal{B}}(y, \mathbf{x})$. As a result, it is usually important to ensure that all attributes in the class' Markov blanket are connected directly to the class or its children. As a consequence, it is common for BNCs to connect all attributes to the class.

Naive Bayes (NB - see eg (Lewis, 1998)) is a popular BNC that makes the class the parent of all other attributes and includes no other edges. The resulting network is illustrated in Figure 1(a) and assumes conditional independence between all attributes conditioned on the class. As a consequence, $\mathrm{P}_{\mathcal{B}}(y|\mathbf{x}) \propto \theta_y \prod_{i=1}^{n} \theta_{x_i|y}$. Tree Augmented Naive Bayes (TAN) (Friedman et al, 1997) adds a further parent to each non-class attribute, seeking to address the greatest conditional interdependencies. It uses the Chow-Liu (Chow and Liu, 1968) algorithm to find the maximum-likelihood tree of dependencies among the attributes in polynomial time.

K-Dependence Bayes (kDB) (Sahami, 1996) allows each non-class attribute to have up to k parents, with k being a user-set value. It first sorts the attributes on mutual information with the class. Each attribute x_i is assigned the k parent attributes that maximize conditional mutual information (CMI) with the class, $\mathrm{CMI}(y, x_i|\Pi_i(\mathbf{x}))$, out of those attributes with higher mutual information with the class. Figure 1(b) shows kDB-1 (for $k = 1$).

Selective kDB (SKDB) (Martínez et al, 2016) selects values $n^* \leq n$ and $k^* \leq k$ such that a kDB over the n^* attributes with highest mutual information with the class and using k^* in place of k maximizes some user selected measure of performance (in the current work, RMSE) assessed using incremental cross validation over the training data.

Other discriminative scoring schemes have been studied, see for example the work by Carvalho et al (2011). A recent review of BNCs was written by (Bielza and Larrañaga, 2014).

2.3 Maximum Likelihood Estimates

Given data points $\mathcal{D} = \{(y^{(1)}, \mathbf{x}^{(1)}), \ldots, (y^{(N)}, \mathbf{x}^{(N)})\}$, the log-likelihood of $\mathcal{B}$ is:

$$\sum_{j=1}^{N} \log \mathrm{P}_{\mathcal{B}}(y^{(j)}, \mathbf{x}^{(j)}) = \sum_{j=1}^{N} \left(\log \theta_{y^{(j)}|\Pi_0(\mathbf{x}^{(j)})} + \sum_{i=1}^{n} \log \theta_{X_i^{(j)}|y^{(j)},\Pi_i(\mathbf{x}^{(j)})} \right), \quad (1)$$

$$\text{with} \sum_{y \in \mathcal{Y}} \theta_{y|\Pi_0(\mathbf{x})} = 1, \quad \text{and} \sum_{X_i \in \mathcal{X}_i} \theta_{X_i|y,\Pi_i(\mathbf{x})} = 1. \quad (2)$$

Maximizing the log-likelihood to optimize the parameters (Θ) yields the well-known MLEs for Bayesian networks. Most importantly, MLEs factorize into independent distributions for each node, as do most standard maximum aposterior estimates (Buntine, 1996).

Theorem 1 *(Wermuth and Lauritzen, 1983) Within the constraints in Equation 2, Equation 1 is maximized when $\theta_{x_i|\Pi_i(\mathbf{x})}$ corresponds to empirical estimates of probabilities from the data, that is, $\theta_{y|\Pi_0(\mathbf{x})} = \mathrm{P}_{\mathcal{D}}(y|\Pi_0(\mathbf{x}))$ and $\theta_{X_i|\Pi_i(\mathbf{x})} = \mathrm{P}_{\mathcal{D}}(X_i|\Pi_i(\mathbf{x}))$.*

Thus our algorithms decompose the problem into separate sub-problems, one for each $\theta_{X_i|y,\Pi_i(\mathbf{x})}$.

2.4 Efficiency of BNC Learning

One often under-appreciated aspect of many BNC learning algorithms is their computational efficiency. Many BNC algorithms can be learned out-of-core, avoiding the overheads associated with retaining the training data in memory.

NB requires only a single pass through the data to learn the parameters, counting the joint frequency of each pair of a class and an attribute value. TAN and kDB require two passes through the data. The first collects the statistics required to learn the structure, and the second the joint frequency statistics required to parameterize that structure. SkDB requires three passes through the data. The first two collect the statistics required to learn structure and parameters, as per standard kDB. The third performs an incremental cross validation to select a subset of the attributes and the k^* to be used in place of k.

3 Why and How are we using HDPs?

The key contribution of this paper is to use hierarchical Dirichlet priors for each categorical distribution $\theta_{X_i|\Pi_i(\mathbf{x})}$, which yields back-off estimates that naturally smooth the empirical estimates at the leaves.

The intuition for our method is that estimation of conditional probabilities should share information with their near neighbours.Suppose you wish to estimate a conditional probability table (CPT) for $p(y|x_1, x_2, x_3)$ from data where the features x_1, x_2, x_3 take on values $\{1, 2, 3, 4\}$. This CPT can be represented as a tree: the root node branches on the values of x_1 and has 4 branches, the 2^{nd} and 3^{rd} level nodes test x_2 and x_3 and have 4 branches. The 4^{th} level consists of leaves and each node has a probability vector for y that we wish to estimate. The sharing intuition says that the leaf node representing $p(y|x_1 = 1, x_2 = 2, x_3 = 1)$ should have similar values to the leaf for $p(y|x_1 = 1, x_2 = 2, x_3 = 2)$ because they have a common parent, but should not be so similar to $p(y|x_1 = 3, x_2 = 1, x_3 = 2)$, which only shares a great-great grandparent.

We achieve this sharing by using a hierarchical prior. So we have vectors $p(Y|x_1 = 1, x_2 = 2, x_3 = X_3)$ (for $X_3 = 1, 2, 3, 4$) are generated from the same prior which has a common probability vector, say $q(Y|x_1 = 1, x_2 = 2)$, as a mean. Now $p(y|x_1, x_2, x_3)$ can often be similar to $p(y|x_1, x_2)$ which in turn can often be similar to $p(y|x_1)$ and in turn to $p(y)$. However, strictly speaking, $p(y|x_1, x_2)$, $p(y|x_1)$ and $p(y)$ are aggregate values here derived from the underlying model which specifies $p(Y|X_1, X_2, X_2)$. So, to model hierarchical similarity with a HDP, instead of using the derived $p(y|x_1, x_2)$, $p(y|x_1)$ and $p(y)$ in the hierarchical prior, we use introduced some latent (hierarchical) parameters, say $q(y|x_1, x_2)$, $q(y|x_1)$ and $q(y)$. This indeed is the innovation of (Teh, 2006). In our case we

use hierarchical Dirichlet distributions because the variables are all discrete and finite, but the algorithm relies on methods developed for a HDP (Lim et al, 2016).

3.1 Intuition developed for Naive Bayes

Imagine a simple Naive Bayes structure such as illustrated in Figure 1(a): the class is the sole parent of every node in $\mathcal{G}$. In this case, we use a (non-hierarchical) Dirichlet as suggested for Bayesian Naïve Bayes (Rennie et al, 2003), for $i = 1, ..., n$ and all y

$$\theta_{X_i|y} \sim \text{Dir}\left(\phi_{X_i}, \alpha_i\right) \ , \tag{3}$$

where α_i is a concentration parameter for node i (we will later develop how we *tie* these parameters in different configurations in the hierarchical case). Note the non-standard notation for the Dirichlet: for convenience we separate the vector probability ϕ_{X_i} and the concentration α_i, making it a 2-argument distribution[1].

We can think of this model in two ways: we add a bias to the parameter estimation that encourages parameter estimates of each $\theta_{X_i|y}$ to have a common mean ϕ_{X_i}. Alternatively, we expect $\theta_{X_i|y}$ for different values y to be similar. If they are similar, it is natural to think that they have a common mean, in this case ϕ_{X_i}. Note, however, that ϕ_{X_i} is a *prior parameter*, introduced above as $q(\cdot)$, and does not correspond to the mean estimated by marginalising, $\sum_y \hat{p}(y)\theta_{X_i|y}$, readily estimated from the data. The ϕ_{X_i} is a latent variable and a Bayesian hierarchical sampler is required to estimate it.

The hyperparameter α_i controls how similar the categorical distributions $\theta_{X_i|y}$ and ϕ_{X_i} should be: if α_i is large, then each $\theta_{X_i|y}$ virtually reproduces ϕ_{X_i}; conversely, $\theta_{X_i|y}$ can vary more freely as α_i tends to 0. Estimation also involves estimating the hyperparameters; generally they get larger as one goes further down the tree.

3.2 Intuition developed for kDB-1

As described in Section 2.2, kDB-1 relaxes Naive Bayes' assumption about the conditional independencies (given y) between the attributes by allowing one extra-parent per node as presented in Figure 1(a). The structure learning process starts from the NB structure. Then it orders the nodes by highest mutual information with the class to be ranked first, *e.g.*, $\langle x_2, x_4, x_1, x_3 \rangle$ in Figure 1(a). Finally, it considers all candidate parents with higher mutual information with the class, and chooses the one that offers the highest mutual information between the class and the child node when conditioned on it. We keep the same idea for the estimation of $\theta_{X_i|\Pi_i(\mathbf{x})}$ as in the NB case, Eq 3, except that now X_i has 2 parents: the class and another covariate. This translates into the following, for $i = 1, ..., n$ and all $y, \Pi(i)$

$$\theta_{X_i|y,\Pi(i)} \sim \text{Dir}\left(\phi_{X_i|y}, \alpha_{i|y}\right) \ , \tag{4}$$

where $\Pi(i)$ only comprises a single node for all $i > 1$ (the first node has only y as parent). Now we could have used ϕ_{X_i} as the latent parent, so it is independent of y, but this would mean all leaves in the tree have similar probability vectors. This is a stronger statement than we need; rather we prefer adjacent nodes on the tree to be similar, not all nodes. With a

[1] Some papers would use the notation Dir $\left(\alpha_i \phi_{X_i}\right)$ or separate the vector $(\alpha_i \phi_{X_i})$ into its $|X_i|$ arguments.

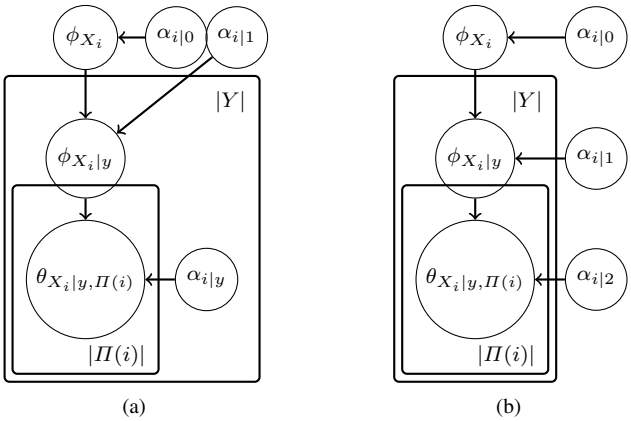


Fig. 2 Our parameter structure model for one X_i and kDB-1. (a) Tying the concentration at the parent. (b) Tying the concentration at the level.

hierarchical model we add another level of complexity, making the dependence on y and require a further parent above for $i = 1, ..., n$ and all y

$$\phi_{X_i|y} \sim \mathrm{Dir}\left(\phi_{X_i}, \alpha_{i|1}\right) \ . \tag{5}$$

This means that different branches in the tree can have different means, and thus the model is more flexible (and has hence relatively low bias). Our Bayesian estimation handles these additional parameters and limits the effect of variance on the model.

The model naturally defines the hierarchical structure given in Figure 2, with the formula above represented by the graphical model given in Figure 2(a).

3.3 Intuition – general framework

The intuition of the framework for kDB-1 naturally extends to BNs with higher numbers of parents. We structure the estimation of the conditional probability of each factor "child given parents" to have a hierarchy with as many levels as the node has parents. At each level, the hierarchy branches on the different values that the newly introduced parent takes: on the different values of y at the first level, on the different values of the first parent at the second level, etc. Once the structure is set, all we need is thus to have an order between the parents. For Naive Bayes, there is only one parent – y. For Tree-Augmented Naive Bayes (TAN), as nodes cannot have more than a single parent apart from the class, we place the class first and its other parent second. For all other structures, we place y as the first parent and then order the parents Π_i by highest mutual information between them and child conditioned on the class. This follows both the NLP framework for n-gram estimation and kDB structure learning: position first in the hierarchy the nodes that are most likely to have an influence on the estimate. Positioning the class first allows us to pull the estimates to be most accurate in the probability space that is near $\mathrm{P}(y|\mathbf{x})$, which is our final target for classification, as we are not really interested in obtaining accurate estimates of $\mathrm{P}(X_i|y, \Pi(i))$ in parts of the probability space that are unrelated to y.

Note that the latent/prior probability vectors $\phi_{X_i|y,\Pi_i(\mathbf{x})}$ do not model observed data, as the $\theta_{X_i|y,\Pi_i(\mathbf{x})}$ do. We represent them with different symbols (ϕ versus θ) to highlight this fundamental difference.

3.4 Summary of the intuition behind HDPs

This is the intuition behind the hierarchical Pitman-Yor language model proposed as a Bayesian interpretation of Kneser-Ney smoothing (Teh, 2006). This framework uses hierarchical priors to model the intuition, and automatically decides how much these hierarchical probability distributions should be related by. This allows the parameter estimation to 'share' information through their ancestors, which significantly improves the estimation of BNCs' parameters. We adapted the Bayesian version because the complex hyperparameter settings of modified Kneser-Ney (Chen and Goodman, 1996) have not been determined for the more general context of clique trees (used in Bayesian networks), whereas the Bayesian framework gives a ready means for estimating them.

Note that in the finite discrete context, DPs are equivalent to Dirichlet distributions (Ferguson, 1973), so we present our models in terms of Dirichlets, but the inference is done efficiently using a collapsed Gibbs sampler for HDPs (Du et al, 2010; Gasthaus and Teh, 2010; Buntine and Mishra, 2014). These recent collapsed samplers for the hierarchical Bayesian algorithms are considerably more efficient and accurate and so do not suffer the well-known algorithmic issues of original hierarchical Chinese restaurant algorithms (Teh et al, 2006). Note however that, unlike some applications of HDPs, there are no "atoms" generated at the root of the HDP hierarchy. The root of the hierarchy is a simple Dirichlet, as are the subsequent nodes. The HDP formalism is used to provide an efficient algorithm as a collapsed version of a Gibbs sampler.

4 Our Framework: HDPs for BNCs

This section reviews our model and sampling approach.

4.1 Model

Consider the case of estimating $\mathrm{P}(x|y, x_1, ..., x_n)$ where the variables $x_1, ..., x_n$ for $n \geq 0$ are ordered as described previously, resulting in a full joint table for all values of the discrete variables. We can present this as a decision tree where the root node banches on y, all nodes at the 1^{st} level test x_1, at the 2^{nd} level test x_2 and so forth. A node at the leaf (the n-th level) has the parameter vector $\theta_{X|y,x_1,...,x_n}$ for values of $y, x_1, ..., x_n$ given by its branch on the tree. A node at the i-th level (for $i = 1, \ldots, n-1$) has the parameter, a latent prior parameter, $\phi_{X|y,x_1,...,x_i}$ where again values of $y, x_1, ..., x_i$ are given by its branch on the tree. The full hierarchical model is given by, for $i = 1, \ldots, n-1$

$$\theta_{X|y,x_1,...,x_n} \sim \mathrm{Dir}\left(\phi_{X|y,x_1,...,x_{n-1}}, \alpha_{y,x_1,...,x_{n-1}}\right) \tag{6}$$

$$\phi_{X|y,x_1,...,x_i} \sim \mathrm{Dir}\left(\phi_{X|y,x_1,...,x_{i-1}}, \alpha_{y,x_1,...,x_i}\right) \tag{7}$$

$$\phi_{X|y} \sim \mathrm{Dir}\left(\phi_X, \alpha_y\right) \tag{8}$$

$$\phi_X \sim \mathrm{Dir}\left(\frac{1}{|X|}\mathbf{1}, \alpha_0\right) . \tag{9}$$

We discuss below, in Section 4.3.2, how we can tie the hyperparameters α_* so that they are not all distinct. Experience has shown us that there should not be just one value in the entire tree, nor should there be a different value per node.

4.2 Context Tree – Data Structure

In the collapsed Chinese Restaurant Process (CRP) used here (Du et al, 2010; Gasthaus and Teh, 2010), one only needs to store the number of tables at each node, not the full configuration of customers at tables. This eliminates the need for dynamic memory. This section presents the essential theory supporting our approach.

The intuition of this algorithm is that each node $\theta_{X|y,x_1,\ldots,x_n}$ or $\phi_{X|y,x_1,\ldots,x_i}$ passes up some fraction of its own data as a multinomial likelihood to its parent. So the nodes will have a vector of sufficient statistics $n_{X|y,x_1,\ldots,x_i}$ recorded for each node. These have a virtual CRP with which we only record the number of tables $t_{X|y,x_1,\ldots,x_i}$, which we refer to as *pseudo-counts*. The counts $t_{X|y,x_1,\ldots,x_i}$ represents the fraction of $n_{X|y,x_1,\ldots,x_i}$ that is passed (in a multinomial likelihood) up to its parent node. The mechanics of this comes from the collapsed Chinese Restaurant Process theory.

As with hierarchical CRPs, these statistics are related for $i \geq 0$:

$$n_{x|y,x_1,\ldots,x_{i-1}} = \sum_{x_i} t_{x|y,x_1,\ldots,x_i}, \tag{10}$$

and moreover the base case $n_x = \sum_y t_{x|y}$. The likelihood for the data with this configuration can be represented with θ and all but the root ϕ marginalised out:

$$\mathrm{P}(\mathcal{D}, n, t|\phi_X, \alpha) = \prod_x \phi_x^{n_x} \tag{11}$$
$$\prod_{i=0}^{n} \prod_{y,x_1,\ldots,x_i} \frac{\alpha_{y,x_1,\ldots,x_i}^{t_{\cdot|y,x_1,\ldots,x_i}}}{\alpha_{y,x_1,\ldots,x_i}^{(n_{\cdot|y,x_1,\ldots,x_i})}} \prod_x S_{t_{x|y,x_1,\ldots,x_i}}^{n_{x|y,x_1,\ldots,x_i}},$$

where S_t^n is an unsigned Stirling number of the first kind, $\alpha^{(n)} = \alpha(\alpha+1)\cdots(\alpha+n-1)$ is a rising factorial, and the 'dot' notation is used to represent totals, so $n_{\cdot|y}$ is the sum of $n_{x|y}$. The Stirling number is a combinatoric quantity that is easily tabulated (Du et al, 2010) and simple asymptotic formula exist (Hwang, 1995). The multinomial likelihood on ϕ_X can also be marginalised out with a Dirichlet prior.

4.3 Gibbs Sampling

Note, in Equation 11, the counts n_* are derived quantities (summed from their child pseudo-counts) and all pseudo-counts t_* are latent variables that are sampled using a Gibbs algorithm. Moreover, the parameters $\phi_{x|y,x_1,\ldots,x_i}$ are estimated recursively from $\phi_{x|y,x_1,\ldots,x_{i-1}}$ and the corresponding counts $n_{x|y,x_1,\ldots,x_i}$ using standard Dirichlet parameter estimation. The final estimates for $\theta_{x|y,x_1,\ldots,x_i}$ can then be obtained again using Dirichlet parameter estimation. This is done periodically to obtain an MCMC estimate during the Gibbs sampling of the pseudo-counts t_* and the concentration parameters α_*. This section then discusses how the Gibbs sampling of these are done.

4.3.1 Sampling Pseudo-counts t_*

We use a direct strategy for sampling the t_*, sweeping through the tree sampling each pseudo-count individually using a formula derived from Equation 11:

$$\mathrm{P}(t_{x|y,x_1,\ldots,x_i}|\mathcal{D},n,t^{-x|y,x_1,\ldots,x_i},\phi_X,\alpha) \propto \frac{\alpha_{y,x_1,\ldots,x_i}^{t_{x|y,x_1,\ldots,x_i}}}{\alpha_{y,x_1,\ldots,x_{i-1}}^{(n_{\cdot|y,x_1,\ldots,x_{i-1}})}} S_{t_{x|y,x_1,\ldots,x_{i-1}}}^{n_{x|y,x_1,\ldots,x_{i-1}}} S_{t_{x|y,x_1,\ldots,x_i}}^{n_{x|y,x_1,\ldots,x_i}} .$$

Note that $t_{x|y,x_1,\ldots,x_i}$ exists in the two sums $n_{\cdot|y,x_1,\ldots,x_{i-1}}$ and $n_{x|y,x_1,\ldots,x_{i-1}}$. This is made efficient because computing the Stirling numbers is a table lookup. Note the Stirling numbers are shared among the different trees, so only calculated once for all nodes of the BNC.

The base case, $i = 0$ is different because the root parameter vector ϕ_X is marginalised using the Dirichlet integral:

$$\mathrm{P}(t_{x|y}|\mathcal{D},n,t^{-x|y},\alpha) \propto \frac{\Gamma\left(n_{x|y}+\alpha_0/|X|\right)}{\Gamma\left(n_{\cdot|y}+\alpha_0\right)} \alpha_y^{t_{x|y}} S_{t_{x|y}}^{n_{x|y}} .$$

These two sampling formula, as they stand, are also inefficient because $t_{x|y,x_1,\ldots,x_i}$ ranges over $1,\ldots,n_{x|y,x_1,\ldots,x_i}$ when $n_{x|y,x_1,\ldots,x_i} > 0$.

From DP theory, we know that the pseudo-counts $t_{x|y,x_1,\ldots,x_i}$ have a standard deviation given by $O(\log^{1/2} n_{x|y,x_1,\ldots,x_i})$, which is very small, thus in practice the full range is almost certainly never used. Moreover, note the mean of $t_{x|y,x_1,\ldots,x_i}$ changes with the concentration parameter, so in effect the sampler is coupled and large moves in the "search" may not be effective. As a safe and efficient option, we only sample the pseudo-counts within a window of ± 10 of their current value. We have tested this empirically, and due to the standard deviations, it is safer as the Monte Carlo sampling converges and smaller moves are typical.

Moreover, to initialise pseudo-counts in the Gibbs sampler, we use the expected value of the pseudo-count for a HDP given the current count and the relevant concentrations:

$$t \leftarrow \begin{cases} n & \text{if } n \leqslant 1 \\ \max(1, \lfloor \alpha\left(\psi_0(\alpha+n) - \psi_0(\alpha)\right) \rfloor & \text{if } n > 1 \end{cases} \tag{12}$$

This requires sweeping up the tree from the data at the leaves.

4.3.2 Tying and Sampling Concentrations α_*

Tying: Rather than using a separate concentration parameter $\alpha_{x|y,x_1,\ldots,x_i}$ for every node, experience on other models alerts us that significant improvements should be possible by judicious sampling of the concentration parameters (Buntine and Mishra, 2014). Figures 2(b) and Figure 2(b) represent two different tyings of concentration parameters. Experiments on the tying of these hyperparameters are presented in Section 6.2.

The tying strategy does not affect the method for sampling concentration parameters, explained next.

Sampling: We use an auxiliary variable sampler detailed in Section 4.3 of (Lim et al, 2016). This introduces an auxiliary variable for each node, and then a Gamma sample can be taken for the tied variable after summing the statistics across the tied nodes. The general

form of the likelihood for a concentration, α, from Equation 11 is $\prod_j \frac{\alpha^{t_j}}{\alpha^{(n_j)}}$ where j runs over the tied nodes and (n_j, t_j) are the corresponding counts at the nodes. We use a prior of the form $\alpha \sim \text{Gamma}(1, 2)$. To sample from the posterior for α we need to augment the denominator terms $\alpha^{(n_j)}$, because they have no match to a known distribution. This is done by introducing $q_j \sim \text{Beta}(\alpha, n_j)$, then the joint posterior becomes

$$\begin{aligned} \text{P}(\alpha, q|\mathcal{D}, n, t) &\propto 2e^{-2\alpha} \prod_j \frac{\alpha^{t_j}}{\alpha^{(n_j)}} \prod_j q_j^{\alpha-1}(1-q_j)^{n_j} \frac{\Gamma(\alpha+n_j)}{\Gamma(\alpha)\Gamma(n_j)} \\ &\propto e^{-2\alpha} \prod_j \alpha^{t_j} q_j^{\alpha-1}(1-q_j)^{n_j} \ . \end{aligned}$$

Looking closely at this, one can see α has a gamma distribution. Thus, a sampling algorithm for α is as following:

1. sample $q_j \sim \text{Beta}(\alpha, n_j)$ for all j, then
2. sample $\alpha \sim \text{gamma}\left(1 + \sum_j t_j,\ 2 + \sum_j \log 1/q_j\right)$.

Note we set $\alpha = \frac{2}{\#TargetOutcomes}$ initially.

5 Related Work

Extensive discussions of methods for DP and PYP hierarchies are presented by Gasthaus and Teh (2010); Lim et al (2016). Standard Chinese restaurant process (CRP) samplers (Teh et al, 2006) use dynamic memory so are computationally demanding, and not being collapsed also makes them considerably slower. Lim et al (2016) deal with the case where the counts at the leaves of the tree are latent, so not applicable to our context. The direct samplers of Du et al (2010), which are also *collapsed* CRP samplers, are more efficient than CRP samplers and those of Lim et al (2016) in the current context. Gasthaus and Teh (2010) dealt with a PYP where the discount parameters change frequently so direct samplers were inefficient because the cache of Stirling numbers needed constant recomputation. On-the-fly samplers have also been developed by Shareghi et al (2017) for PYP hierarchies, making it possible to use PYP for deep trees and large dataset sizes. This however does not change the issue of constant recomputation of Stirling numbers, which is why initialisations based on Modified Kneser-Ney have been developed by Shareghi et al (2016).

The use of DP and PYP hierarchies for regression and clustering – as opposed to classification in our case – has been studied by Nguyen et al (2015); Huynh et al (2016) , respectively.

Related work for BNCs was discussed in 2.2. There are other methods for improving BNCs. A simple back-off strategy, backing off to the root, is proposed by Friedman et al (1997). Moreover, for some simple classes of networks, such as TAN, a disciminative generalisation of logistic regression can be used because the optimisation surface is convex (Roos et al, 2005; Zaidi et al, 2017). Neither techniques are applicable to the more complex BNCs we consider.

Bayesian model averaging methods are common for Bayesian network learning (Friedman and Koller, 2003). Average n-Dependence Estimators – AnDE (Webb et al, 2005, 2012), another ensemble method, is competitive for smaller data sets but cannot compete against SkDB for larger data sets (Martínez et al, 2016).

Either way, these invariably use the same Laplacian prior as the m-estimates reported here in Section 6.

6 Experiments

The aim of this section is to assess our HDP-based estimates for Bayesian Network Classifiers (BNCs). In Section 6.1, we give the general settings that are necessary to understand and reproduce our experiments. Then, in Section 6.2, we start by studying how to parameterize our method: i.e. by studying the influence of number of iterations and the tying strategy used. In Section 6.3, we demonstrate the superiority of our estimates over the state of the art across 8 different BNC structures. Finally, having obtained significant improvements over the state-of-the-art, we then turn to comparing the best-performing configuration (TAN and Selective kDB with HDP estimates) with Random Forest (RF) in Section 6.4. We show that our estimate allows even models as simple as TAN to significantly outperform RF (with statistical significance), while standard approaches to parameter estimation are beaten by RF. We conclude the experiments with a demonstration of our system's out-of-core capability and show results obtained on the Splice dataset with 50 million training examples, a quantity that RF cannot handle on most machines.

6.1 Experimental Design and Setting

Design: All experiments are carried out on a total of 68 datasets from the UCI archive (Lichman, 2013); 38 datasets with less than 1000 instances, 23 datasets with instances between 1000 and 10000, and 7 datasets with more than 10000 instances. The list and description of the datasets is given in Table 4 at the end of this paper. For all methods, numeric attributes are discretized by using the Minimum Description Length (MDL) discretization method (Fayyad and Irani, 1992). A missing value is treated as a separate attribute value and taken into account exactly like other values. Each algorithm is tested on each dataset using 2-fold cross validation repeated 5 times. We assess the results by reporting 0-1 Loss and RMSE, and report Win-Draw-Loss (W-D-L) results when comparing the 0-1 Loss and RMSE of two models. A two-tail binomial sign test is used to determine the significance of the results, using $p \leq 0.05$.

Note the RMSE is related to the Brier score, which is a proper scoring rule for classifiers and thus generally referable to error, especially in the context of unequally occurring classes or unequal costs. It measures how well calibrated the probability estimates are. We use it because we suspected that our methods could improve probability estimates but not necessarily errors.

Software: Note that, to ensure reproducibility of our work and allow other researchers to easily build on our research, we have made our source code for HDP parameter estimation available on Github.

Compared methods: We assess our estimates for 8 BNC structures with growing complexity. Our BNC structures are: naive Bayes (NB), Tree-Augmented naive Bayes (TAN) (Friedman et al, 1997), k-Dependence Bayesian Network (kDB) (Sahami, 1996) with $k = 1$ to 5 and Selective KDB (SkDB) (Martínez et al, 2016) with maximum k set to 5 also. When comparing to Random Forest (RF), we use the Weka default parameterization, i.e. selecting $\log_2(n) + 1$ attributes in each tree,[2] no minimum leaf size and using 100 decision trees in this work.

[2] Selecting $\sqrt{n}$ attributes produces similar results and conclusion, so the results are left out of this paper for concision.

For BNCs, we compare our HDP estimates to so-called m-estimates[3] (Mitchell, 1997) as follows:

$$\hat{p}(x_i|\Pi(i)) = \frac{counts(x_i, \Pi(i)) + \frac{m}{|X_i|}}{counts(\Pi(i)) + m} \tag{13}$$

where $\Pi(i)$ are the parent-values of X_i. The value of m is set by cross-validation on a holdout set of size $\min(N/10, 5000)$ among with $m \in \{0, 0.05, 0.2, 1, 5, 20\}$.

Count statistics are stored in a prefix tree; for m-estimates, if zero counts are found, we back off as many levels in the tree as necessary to find at least one count. For instance, if $counts(x_4, x_0, x_3)$ is equal to zero, then $\hat{p}(x_4|x_0)$ is considered instead of $\hat{p}(x_4|x_0, x_3)$. Note that not using this strategy significantly degrades the performance of BNCs with using m-estimates (for our HDP estimates, the intermediate nodes ϕ are considered latent and thus inferred directly during sampling).

6.2 Tying and Number of Iterations

Before proceeding with the comparison of our method to the state of the art, it is important to study two elements: (1) for how many iterations to run the sampler and (2) how to tie the concentration parameters. These two elements are directly related because the less tying, the more parameters to infer, which means that we expect to have to run the sampler for more iterations.

We consider three different tying strategies:

1. Same Parent (SP): children of each node share the same parameter – illustrated in Figure 2(a).
2. Level (L): we use one parameter for each level of the tree – illustrated in Figure 2(b).
3. Single (S): all parameters tied together.

Number of iterations: Asymptotically, the accuracy of the estimates improves as we increase the number of iterations. The question is how quickly they asymptote. We thus studied the performance of our two flagship classifiers – TAN and SkDB – on all datasets as we increase the number of iterations from 500 to 50,000. For each combination of classifier$\times$tying strategy, we assess win-loss profile for x iterations versus 50,000. The resulting win-loss plot in Figure 3 shows that across all tying strategies and models, running our sampler for 50,000 iterations is significantly better than with fewer iterations. Even for models as simple as TAN with a Single concentration parameter, running the sampler for 5,000 iterations wins 13 times and loses 42 times as compared to running it for 50,000 iterations. Unless specified otherwise, we thus run the sampler for 50,000 iterations. We surmise that even more iterations could further improve accuracy but leave this for future research.

Tying strategy: Having seen that 50,000 iterations seems important regardless of the tying strategy, we here show that tying per Level seems to the best default strategy. It is important to note that we do not intend to give a definitive answer valid for all domains here, but are simply giving a reasonable 'default' parameterization. The Level strategy was illustrated for kDB-1 in Figure 2(b). To illustrate this we compare TAN and SkDB parameterized with the Same Parent (SP) and Single (S) strategies versus using the Level (L) tying strategy across different numbers of iterations. Figure 4 gives the win-loss plot. We see that L provides a uniformly good solution providing both the best results with 50,000 iterations

[3] Also known as Schurmann-Grassberger's Law when $m = 1$, which is a particular case of Lidstone's Law (Lidstone, 1920; Hardy, 1920) with $\lambda = \frac{1}{|X_i|}$, also based on a Dirichlet prior.

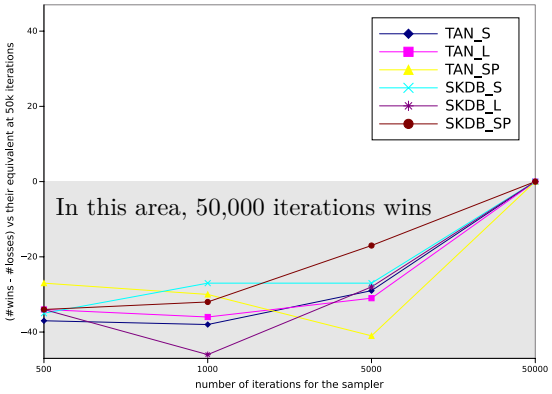


Fig. 3 Win/loss plot on RMSE for each combination of (flagship classifier) $\times$ (tying strategy). Comparison is for running each combination for x iterations vs 50,000 and include **S**ingle, **L**evel and **S**ame**P**arent.

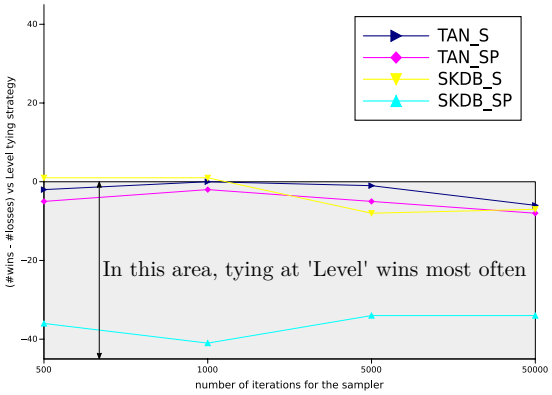


Fig. 4 Win/loss plot of each combination of (flagship classifier) $\times$ (S or SP tying strategy) versus tying at level (L).

but also providing solid performances as early as 500 iterations. It is worth noting that for TAN, the L and S strategies are very similar, only differing by one concentration parameter. The SP strategy seems to clearly underperform L, all the more when the complexity of the model increases, which makes sense given that the number of concentration parameters to estimate increases exponentially with the depth of the prefix tree, which is mostly controlled by the number of parents for each node i. It is possible that for large amounts of data, the SP strategy would offer a better bias/variance tradeoff but such a study falls out of the scope of this paper. We thus use L as a tying strategy for the remainder of this paper.

6.3 HDP vs m-estimates for Bayes Network Classifiers

So far, we have only assessed the relative performance of HDP estimates with different parameterizations. Having settled on 50,000 iterations and per Level tying, we now turn to full comparison with the state-of-the-art in smoothing Bayesian network classifiers: using m-estimates with the value of m cross-validated on a holdout set. We also remind the reader

Table 1 Win/Draw/Loss for 8 BNCs for our HDP estimate vs m-estimate. Stat. sig. ($p < 0.05$) results are depicted in boldface.

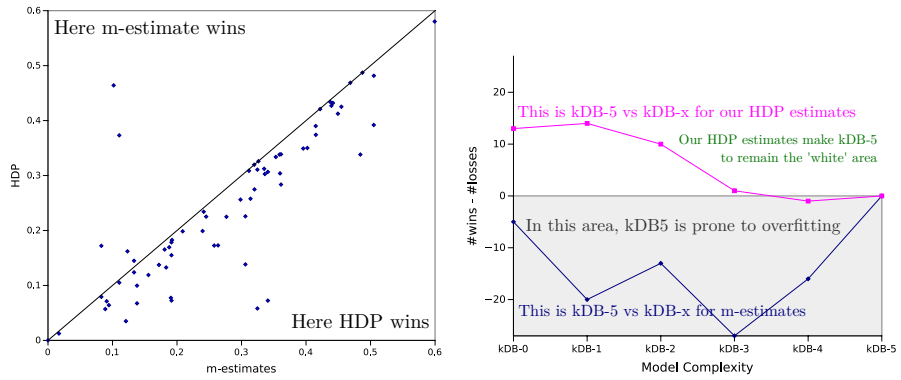
Classifier	Win–draw–loss for HDP vs m-estimate	
	0/1-loss	RMSE
Naive Bayes	**41–4–23**	40–0–28
TAN	**45–4–19**	**52–1–15**
kDB-1	**45–4–19**	**50–1–17**
kDB-2	**54–2–12**	**54–0–14**
kDB-3	**52–4–12**	**53–2–13**
kDB-4	**56–4– 8**	**56–0–12**
kDB-5	**60–4– 4**	**60–2– 6**
SkDB	**45–4–19**	**54–0–14**

that, to provide the best competitor, we also added the back-off strategy described above, without which m-estimates cannot compete at all.

We report in Table 1 the win-draw-loss of our HDP estimates versus m-estimates across 8 different BNCs from Naive Bayes and TAN to kDB with $1 \leqslant k \leqslant 5$ and SkDB. It is clear from this table that our HDP estimates are far superior to m-estimates. It is even quite surprising to see our estimates outperform m-estimates with models as simple as Naive Bayes, where our hierarchy only has one single level. Moreover, as the model complexity increases (the maximum number of parents for each node), this difference increases. The scatter-plot for kDB-5 HDP vs m-estimate is given in Figure 5(a) and shows again the same trend with HDP significantly outperforming m-estimate. As usual when dealing with a broad range of datasets, there are a few points for which HDP loses. Interestingly, the most important loss is for the `Cylinder-Bands` dataset, which contain only 540 samples, and thus for which we would have expected that smoothing would be important; detailed introspection of this dataset show that the 540 cases seem to be relatively similar to each other (in which case the cross-validation used for m-estimates help discover this).

It is also interesting to study the capacity of HDP to prevent overfitting as compared to the m-estimate (with m cross-validated). In Figure 5(b), we report for m-estimates the win-loss plot for kDB-5 compared to kDBs with increasing complexity from 0 (kDB-0 is NB) to 4. Given that kDB-5 has generally lower bias than kDB $\forall k \leqslant 4$, we can typically attribute its losses to overfitting. Starting with the bottom line, which represents the behaviour of using m-estimates, we can see that kDB-5 generally loses to lower complexity kDBs. The maximum difference is with kDB-3 which seems to globally have a nice bias/variance tradeoff on this collection of datasets.

Conversely, we can see that HDP estimates (top-curve in Figure 5(b)) allows us to nicely control for overfitting. What happens is that we make the most of the low-biased structure offered by kDB, while not being overly prone to overfitting. In some sense, our hierarchical process makes it possible to pull the probability estimates towards higher-level nodes for which we have more data, and this automatically depending on the dataset. It seems that it makes it possible to be less strict about the structure and to be powerful at controlling for the variance. In fact, controlling for overfitting is what Selective kDB (SkDB) tries to achieve; in our experiments, kDB5-HDP has a slight edge over SkDB5-HDP with a win-draw-loss



(a) (b)

Fig. 5 (a) Scatter plot on RMSE for kDB-5 for HDP vs m-estimate. (b) Win/loss plot of kDB-5 vs kDB-x for m-estimates vs our HDP ones.

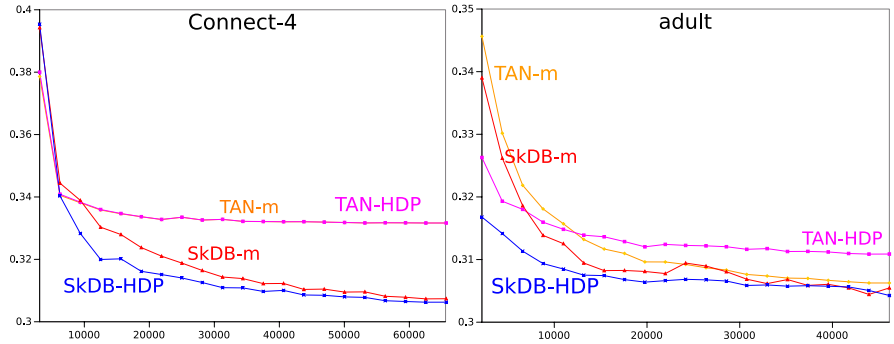


Fig. 6 Learning curves on RMSE for HDP and m-estimate. The x-axis is dataset size, the y-axis is RMSE.

of 33–5–30 on RMSE. Nevertheless, it remains that HDP largely outperforms m-estimates with a win-loss – for SkDB – of 60 to 8.

Finally, we present some learning curves for TAN and SkDB on a some larger datasets in Figure 6. Each point corresponds the mean RMSE for quantity of data x over 10 runs. Globally, we can see that our HDP estimates seem to 'learn' faster, i.e. overfit less. For the `connect-4` dataset, SkDB-HDP dominates all the way through with the difference in RMSE getting smaller as the quantity of data increases. For `adult`, we can observe the same behaviour for SkDB. Interestingly, for TAN on this dataset, although HDP estimates do learn faster, they are overtaken by m-estimates after 10,000 datapoints.

6.4 BNCs with HDP vs Random Forest

Having shown that our approach outperforms the state of the art for BNCs parameter estimation, we compare BNCs using our HDP estimates against Random Forest (RF). The aim of this section is not to suggest that BNCs should replace RF, but rather to that BNCs can perform competitively.

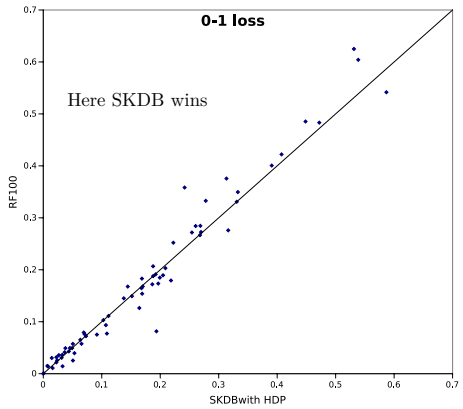


Fig. 7 0-1 loss scatter plot of SKDB with our HDP parameter estimate vs Random Forest

Table 2 Win/Draw/Loss m-estimates and our HDP estimates, as compared with Random Forest. We use our 2 flagship classifiers TAN and SkDB. Stat. sig. results ($p < 0.05$) are depicted in boldface.

Compared classifiers	Win–draw–loss	
	0/1-loss	RMSE
TAN-m vs RF	26–3–39	**25–0–43**
SkDB-m vs RF	27–3–38	29–1–38
TAN-HDP vs RF	**42–3–23**	**42–0–26**
SkDB-HDP vs RF	35–3–30	**44–0–24**

Before proceeding, it is important to recall that RF is run on the same datasets as our BNCs with HDP estimates, i.e., with attributes discretized when necessary.

We report in Table 2 and Figure 7 the results of TAN and SkDB. From this table we can see that RF is more often more accurate than the BNCs with m-estimates. Conversely, we can see that BNCs with HDP outperform RF more often, even with a model as simple as TAN. This result is important because our techniques are all completely out-of-core and do not need to retain the data in main memory, as do most state-of-the-art learners. Note that comparing 0-1 loss is fairer to RF, which is not a probabilistic model, although, reportedly, plain RF estimates as we do outperform other RF variations in terms of RMSE (Boström, 2012).

Obviously, for the larger datasets, RF catches up to TAN-HDP (which has a high-bias structure) but for the 10 largest datasets we considered, TAN-HDP still wins 6 times (1 draw) and SkDB-HDP is extremely competitive with a win-draw-loss of 7–0–3.

6.5 Out-of-core Capacity

Our last set of experiments aims at showcasing the out-of-core capacity of our system. We run SkDB on the Splice dataset (Sonnenburg and Franc, 2010), which contains 50 million training examples, and on which RF did not run using Weka defaults (requiring more than

Table 3 Results on the Splice dataset on which RF cannot run.

Classifier	0/1-loss	RMSE
SkDB5-m	1.50%	0.011
SkDB5-HDP	0.32%	0.005

our limit of 138GB of RAM). Note that this dataset is provided with a test dataset with 5M samples. The results are reported in Table 3 and show that HDP dramatically improves both 0-1 loss and RMSE. Note that predicting majority class yields an error rate of 1% so SkDB5-m fails on this dataset.

6.6 Running Time

Although running time is not directly a focus of this paper, we give below some associated observations:

- Training time complexity increases linearly with the number of iterations the sampler runs for, linearly with the number of covariates and linearly with the number of nodes in the trees (which increases exponentially with depth).
- Training time is reasonable. As an example, training of *SkDB*$_5$*-HDP* (with $maxK = 5$) on Splice with 50 million samples took under 4 hours, among which 1.5 hours are spent to learn the structure of the BN. *SkDB*$_5$ implied that the 140 independent hierarchies have a depth of 6 and we run 5,000 iterations of the sampler.
- For the Adult dataset training *SkDB*$_5$ with 25k samples and 50,000 iterations with level tying took 86 seconds, for the Abalone dataset training with 2k samples took 6 seconds – classification time takes less than 1s to classify 25k samples, which is one of the strength of BNCs: once learned, classification is a simple look-up for each factor. This classification time is actually under 1s for all models considered in this paper for the Adult dataset.

7 Conclusions

This paper presents accurate parameter estimation for Bayesian Network Classifiers using Hierarchical Dirichlet Processes estimates, combining these well-researched areas for the first time. We have demonstrated that HDPs are not only capable of outperforming state-of-the-art parameter estimation techniques, but do so while functioning completely out-of-core. We have also showed that, for categorical data, this makes it possible to make BNCs highly competitive with Random Forest. We note that while BNCs are not currently state of the art for classification, they are still popular in applications. With this improvement in performance, and usable implementations in packages such as R, BNCs will be far more useful in real-world applications because they are readily implemented on high performance desktops, not requiring clusters.

This work naturally opens up a number of opportunities for future research. First, we would like to perfect our sampler by assessing the influence of the different runtime configurations of our system including: how often should we sample concentration, widening

Table 4 Datasets

Domain	Case	Att	Class	Domain	Case	Att	Class
Connect-4Opening	67557	43	3	PimaIndiansDiabetes	768	9	2
Statlog(Shuttle)	58000	10	7	BreastCancer(Wisconsin)	699	10	2
Adult	48842	15	2	CreditScreening	690	16	2
LetterRecognition	20000	17	26	BalanceScale	625	5	3
MAGICGammaTelescope	19020	11	2	Syncon	600	61	6
Nursery	12960	9	5	Chess	551	40	2
Sign	12546	9	3	Cylinder	540	40	2
PenDigits	10992	17	10	Musk1	476	167	2
Thyroid	9169	30	20	HouseVotes84	435	17	2
Mushrooms	8124	23	2	HorseColic	368	22	2
Musk2	6598	167	2	Dermatology	366	35	6
Satellite	6435	37	6	Ionosphere	351	35	2
OpticalDigits	5620	49	10	LiverDisorders(Bupa)	345	7	2
PageBlocksClassification	5473	11	5	PrimaryTumor	339	18	22
Wall-following	5456	25	4	Haberman'sSurvival	306	4	2
Nettalk(Phoneme)	5438	8	52	HeartDisease(Cleveland)	303	14	2
Waveform-5000	5000	41	3	Hungarian	294	14	2
Spambase	4601	58	2	Audiology	226	70	24
Abalone	4177	9	3	New-Thyroid	215	6	3
Hypothyroid(Garavan)	3772	30	4	GlassIdentification	214	10	3
Sick-euthyroid	3772	30	2	SonarClassification	208	61	2
King-rook-vs-king-pawn	3196	37	2	AutoImports	205	26	7
Splice-junctionGeneSequences	3190	62	3	WineRecognition	178	14	3
Segment	2310	20	7	Hepatitis	155	20	2
CarEvaluation	1728	8	4	TeachingAssistantEvaluation	151	6	3
Volcanoes	1520	4	4	IrisClassification	150	5	3
Yeast	1484	9	10	Lymphography	148	19	4
ContraceptiveMethodChoice	1473	10	3	Echocardiogram	131	7	2
German	1000	21	2	PromoterGeneSequences	106	58	2
LED	1000	8	10	Zoo	101	17	7
Vowel	990	14	11	PostoperativePatient	90	9	3
Tic-Tac-ToeEndgame	958	10	2	LaborNegotiations	57	17	2
Annealing	898	39	6	LungCancer	32	57	3
Vehicle	846	19	4	Contact-lenses	24	5	3

the window of pseudo-counts at the start of the system and burn-in. Second, we would like to extend this work to Pitman-Yor Processes, which offer an exciting avenue for research, in particular for variables with high cardinality. Third, we are interested in performing an extensive empirical assessment of our system for very large datasets against other big data algorithms. Fourth, we would like to extend this framework to the general class of Bayesian Networks.

References

Bielza C, Larrañaga P (2014) Discrete bayesian network classifiers: a survey. ACM Computing Surveys 47(1):5

Boström H (2012) Forests of probability estimation trees. Int Jnl of Pattern Recognition and Artificial Intelligence 26(02):1251,001

Breiman L (2001) Random forests. Machine Learning 45:5–32

Buntine W (1996) A guide to the literature on learning probabilistic networks from data. IEEE Transactions on Knowledge and Data Engineering 8(2):195–210

Buntine W, Mishra S (2014) Experiments with non-parametric topic models. In: 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '14, pp 881–890

Carvalho AM, Roos T, Oliveira AL, Myllymäki P (2011) Discriminative learning of bayesian networks via factorized conditional log-likelihood. Journal of machine learning research 12(Jul):2181–2210

Chen S, Goodman J (1996) An empirical study of smoothing techniques for language modeling. In: 34th Annual Meeting on Association for Computational Linguistics, ACL '96, pp 310–318

Chow C, Liu C (1968) Approximating discrete probability distributions with dependence trees. IEEE Trans on Information Theory 14(3):462–467

Du L, Buntine W, Jin H (2010) A segmented topic model based on the two-parameter Poisson-Dirichlet process. Machine Learning 81(1):5–19

Ehsan Shareghi TC Gholamreza Haffari (2017) Compressed nonparametric language modelling. In: IJCAI, p Accepted 23/04/2017

Fayyad U, Irani K (1992) On the handling of continuous-valued attributes in decision tree generation. Machine Learning 8(1):87–102

Ferguson T (1973) A Bayesian analysis of some nonparametric problems. The Annals of Statistics 1:209–230

Friedman N, Koller D (2003) Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. Machine Learning 50(1–2):95–125

Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Machine Learning 29(2):131–163

Gasthaus J, Teh Y (2010) Improvements to the sequence memoizer. In: Advances in Neural Information Processing Systems 23, pp 685–693

Hardy G (1920) Correspondence. Insurance record (1889). Transactions of the Faculty Actuaries 8

Huynh V, Phung DQ, Venkatesh S, Nguyen X, Hoffman MD, Bui HH (2016) Scalable nonparametric bayesian multilevel clustering. In: UAI

Hwang HK (1995) Asymptotic expansions for the Stirling numbers of the first kind. Jnl of Combinatorial Theory, Series A 71(2):343–351

Koller D, Friedman N (2009) Probabilistic Graphical Models – Principles and Techniques. Adaptive Computation and Machine Learning, The MIT Press

Lewis D (1998) Naive Bayes at forty: The independence assumption in information retrieval. In: 10th European Conf. on Machine Learning, Springer-Verlag, London, UK, UK, ECML '98, pp 4–15

Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml

Lidstone G (1920) Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. Transactions of the Faculty Actuaries 8:182–192

Lim K, Buntine W, Chen C, Du L (2016) Nonparametric Bayesian topic modelling with the hierarchical Pitman–Yor processes. Int Jnl of Approximate Reasoning 78:172–191

Martínez A, Webb G, Chen S, Zaidi N (2016) Scalable learning of Bayesian network classifiers. Journal of Machine Learning Research 17(44):1–35

Mitchell T (1997) Machine Learning. McGraw-Hill, New York

Nguyen V, Phung DQ, Venkatesh S, Bui HH (2015) A bayesian nonparametric approach to multilevel regression. In: PAKDD (1), pp 330–342

Rennie J, Shih L, Teevan J, Karger D (2003) Tackling the poor assumptions of naive Bayes text classifiers. In: 20th Int. Conf. on Machine Learning, AAAI Press, ICML'03, pp 616–

623
Roos T, Wettig H, Grünwald P, Myllymäki P, Tirri H (2005) On discriminative Bayesian network classifiers and logistic regression. Machine Learning 59(3):267–296
Sahami M (1996) Learning limited dependence Bayesian classifiers. In: Second Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, pp 334–338
Shareghi E, Cohn T, Haffari G (2016) Richer interpolative smoothing based on modified kneser-ney language modeling. In: Empirical Methods in Natural Language Processing, pp 944–949
Shareghi E, Haffari G, Cohn T (2017) Compressed nonparametric language modelling. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp 2701–2707, DOI 10.24963/ijcai.2017/376
Sonnenburg S, Franc V (2010) COFFIN: A computational framework for linear SVMs. In: Fürnkranz J, Joachims T (eds) ICML, pp 999–1006
Teh Y (2006) A Bayesian interpretation of interpolated Kneser-Ney. Tech. Rep. TRA2/06, School of Computing, National University of Singapore
Teh Y, Jordan M, Beal M, Blei D (2006) Hierarchical Dirichlet processes. Journal of the American Statistical Association 101(476):1566–1581
Webb G, Boughton J, Zheng F, Ting K, Salem H (2012) Learning by extrapolation from marginal to full-multivariate probability distributions: Decreasingly naive Bayesian classification. Machine Learning 86(2):233–272
Webb GI, Boughton J, Wang Z (2005) Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning 58(1):5–24
Wermuth N, Lauritzen S (1983) Graphical and recursive models for contigency tables. Biometrika 70(3):537–552
Wood F, Gasthaus J, Archambeau C, James L, Teh Y (2011) The sequence memoizer. Communications of the ACM 54(2):91–98
Zaidi NA, Webb GI, Carman MJ, Petitjean F, Buntine W, Hynes M, De Sterck H (2017) Efficient parameter learning of Bayesian network classifiers. Machine Learning pp 1–41