Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

Bayesian Network Classifiers using Ensembles and Smoothing

He Zhang · François Petitjean ·
Wray Buntine

Received: 23 Jul 2019 / Revised: 25 Nov 2019 / Accepted: 03 Feb 2020

Abstract Bayesian network classifiers (BNCs) are, functionally, an interesting class of models, because they can be learnt out-of-core, i.e. without needing to hold the whole training data in main memory. The Selective K-Dependence Bayesian Network Classifier (SKDB) is state-of-the-art in this class of models and has shown to rival Random Forest (RF) on problems with categorical data. In this paper, we introduce an ensembling technique for SKDB, called Ensemble of SKDB (ESKDB). We show that ESKDB significantly outperforms RF on categorical and numerical data, as well as rivalling XGBoost. ESKDB combines three main components: (1) an effective strategy to vary the networks that is built by single classifiers (to make it an ensemble), (2) a stochastic discretization method which allows to both tackle numerical data as well as further increases the variance between different components of our ensemble and (3) a superior smoothing technique to ensure proper calibration of ESKDB's probabilities. We conduct a large set of experiments with 72 datasets to study the properties of ESKDB (through a sensitivity analysis) and show its competitiveness with the state of the art.

Keywords Bayesian Network Classifier · Ensemble Learning · Probability Smoothing · Hierarchical Dirichlet Process · Attribute Discretization

1 Introduction

With the rapid development of Web technologies in the last decades, large datasets are created everywhere, such as in social media, E-commerce and health care. In-core algorithms, e.g. Random Forest (RF) (Breiman, 2001) and Support Vector Machine (SVM) (Hearst, 1998), are less suited to large amounts of data because they require the data to be stored in main memory. Out-of-core learners – i.e. algorithms that can

This research was partially supported by the China Scholarship Council under awards 201506300081 and the Australian Government through the Australian Research Council's Discovery Projects funding scheme (projects DP190100017 and DE170100037).

He Zhang, François Petitjean, Wray Buntine
Faculty of Information Technology, Monash University
E-mail: {he.zhang, francois.petitjean, wray.buntine}@monash.edu

learn from a dataset without holding it fully in the main memory – appear to be more suited to large quantities of data because of their ability to scale. Bayesian Network Classifiers (BNCs) are out-of-core learners and thus show great potential; instances of this class of classifiers include the famous Naïve Bayes (NB) (Lewis, 1998) algorithm, Tree Augmented Naïve Bayes (TAN) (Friedman et al, 1997), K-Dependence Bayes (KDB) (Sahami, 1996), as well as Selective KDB (SKDB) (Martınez et al, 2016), which was shown to be competitive to RF on categorical data.

A BNC is a directed acyclic graph whose nodes represent the variables of the dataset and edges indicate the direct dependencies between those variables. Generally the target or class variable is a parent of all other variables, with several additional connections existing between the other variables. The bias/variance trade-off of BNCs can be easily tuned by putting a limit on the maximum number of parents that a node can have. With k parents, the model then looks at all possible combinations of the $(k+1)$ nodes connected with each other. The higher the value of k, the lower the bias of the algorithm (and usually the higher the variance as well).

SKDB is a highly scalable BNC that achieves a good trade-off between structural complexity and classification performance by efficiently choosing the value of the maximum number of parents ($maxK$). For large datasets, a higher complexity or low-biased model is preferable because it allows the model to capture fine detail in data more precisely. But this low bias model has more parameters and potentially higher variance, leading to poorer predictions. As a result, more data is usually needed (or a superior parameter estimation technique, as we will see later).

In this paper, we propose to ensemble the SKDB algorithm to both increases its accuracy as well as to make it applicable to numerical data. There are two broad frameworks for ensembling, Bayesian model averaging (Hoeting et al, 1999), first implemented for Bayesian networks in (Madigan et al, 1995), and the more frequentist style commonly associated with ensembles (Zhou, 2012) best illustrated by Random Forest (Breiman, 2001). We use the second broad framework because it is more suited to larger amounts of data.

The difficulty in creating an ensemble of a base classifier lies in varying the results of the original classifier without raising the bias of the base classifier, and while keeping the covariance between the (varied) classifiers low. To do this, we combine two sources of stochasticity: (1) we vary the order in which the variables are considered, which controls what combination of attributes will be considered and (2) we vary the discretization for numerical attributes, which allows different 'elemental' classifiers of the ensemble to consider different cut-points. We will detail in Section 3 how these stochasticities are defined to obtain both high accuracy and diversity of the elemental classifiers composing the ensemble. Finally, we add a third component to ESKDB in using an advanced smoothing technique based on Hierarchical Dirichlet Processes (HDP): it allows to control the variance of ESKDB further and, as we will show, substantially improves accuracy and probability calibration.

We carry out an extensive set of experiments on 72 datasets with data quantity up to 5M examples. We start by performing a large sensitivity analysis to show the influence of each of the three contributions on our final ESKDB algorithm (varying the attribute order, varying the discretization and advanced smoothing). Having shown that all three components indeed bring significant improvement to the classifier, we then proceed by showing how it compares with the state of the art. We start by comparing ESKDB to existing Bayesian Network Classifiers (BNCs) – NB, TAN, KDB, KDF, AODE and SKDB. We show that ESKDB significantly outper-

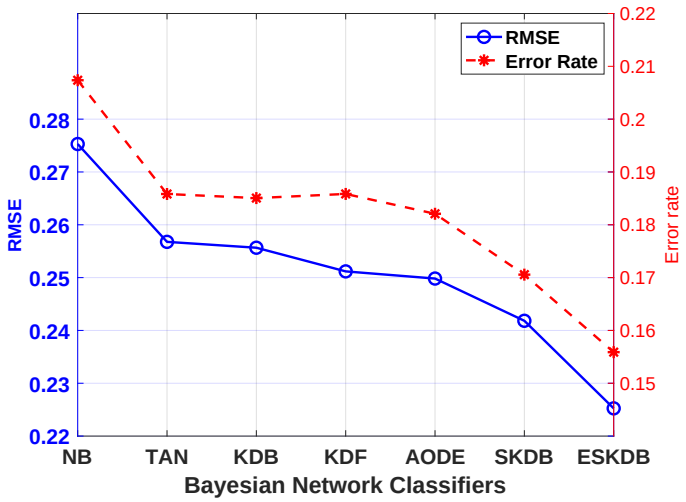


Fig. 1 Average RMSE and error rate for NB, TAN, KDB, SKDB, KDF, AODE and our proposed ESKDB classifier over the 72 datasets (list in Table 1)

.

forms state-of-the-art BNCs.This is highlighted in Figure 1. We then proceed to compare ESKDB to non-BNCs. Our results show that ESKDB significantly outperforms both XGBoost (Chen and Guestrin, 2016) with default parameterization and Random Forest (Breiman, 2001) and that its performance is not significantly different from XGBoost with highly tuned parameters. We believe these are strong results because ESKDB can handle both numerical and categorical data while being able to perform with a limited number of passes over the data (and hence not needing to load the data in main memory such as RF and XGBoost). We finally complete our experiments' section by studying the running time of ESKDB and give guidance on using different smoothing methods on ESKDB.
The main contributions of this paper are as follows:

- A novel ensembling method for Bayesian Network classifiers – ESKDB – with two novel elements:
  - we vary the attribute order over the variables by sampling orders following the mutual information with the class;
  - we vary the number of cut-points for the discretisation of each numerical attribute by sampling the information gain (and combined with the MDL stopping criterion).
- An improved probability estimation technique: we add a prior to existing Hierarchical Dirichlet Process smoothing techniques. The result is a sampler that requires 10x fewer samples than the state-of-the-art one proposed in (Petitjean et al, 2018).

Put together, these three components make ESKDB become the most accurate Bayesian Network Classifier to date. It achieves better accuracy, better probability calibration, can handle numerical attributes and does all these much faster than the state-of-the-art BNC – SKDB-HDP (Petitjean et al, 2018). We show that our

classifier runs virtually parameter-free and significantly outperforms Random Forest and scores just behind a highly-tuned XGBoost algorithm.

The remainder of this paper is organised as follows. We review the background and related work in Section 2. Section 3 describes our ESKDB algorithm. Results of our extensive set of experiments are reported in Section 4. We conclude this paper in Section 5.

2 Related Work and Background

2.1 Bayesian Network Classifier

Let capital letters $\mathbf{X} = (X_1, X_2, ...X_n)$ represent n attributes in a dataset. Lower case letters $\mathbf{x} = (x_1, x_2, ..., x_n)$ represent specific data values taken by these attributes for a particular datum. In particular, Y is the class variable, and y is one of the possible class labels that data $\mathbf{x}$ belongs to. The basic task of a classification problem is to compute the conditional class probability distribution $P(y|\mathbf{x})$ over all the possible classes and assigns the class to $\mathbf{x}$ most suited given the context, for instance depending on the trade-off between recall and precision desired.

A Bayesian Network (BN) $\mathscr{B} = \langle \mathscr{G}, \Theta \rangle$ is characterized by two parameters $\mathscr{G}$ and Θ. $\mathscr{G}$ is a directed acyclic graph whose nodes and edges represent random variables $X_1, X_2, ...X_n$ and direct dependencies between those variables, respectively. We say X_i is the parent of X_j if X_i is pointing directly to X_j via a single edge. Another parameter Θ quantifies the network structure with a set of parameters $\theta_{x_i|\Pi_i(x)} = P_{\mathscr{B}}(x_i|\Pi_{x_i})$ for each possible value x_i of X_i, and Π_{x_i} of Π_{X_i}, where Π_{X_i} denotes the set of parents of X_i in $\mathscr{G}$. A BN defines the joint probability distribution over $\mathbf{x}$ given by

$$P_{\mathscr{B}}(\mathbf{x}) = \prod_{i=1}^{n} P_{\mathscr{B}}(x_i|\Pi_{x_i}) = \prod_{i=1}^{n} \theta_{x_i|\Pi_{x_i}} \tag{1}$$

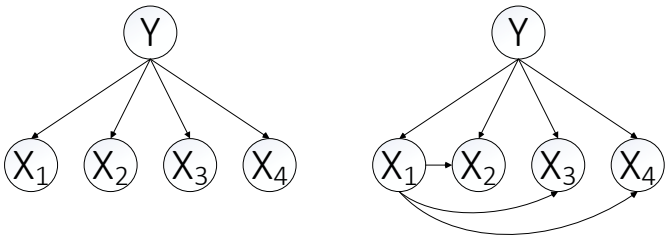
When using a Bayesian Network as a classifier, the precision of posterior estimates $P_{\mathscr{B}}(y|\mathbf{x})$ matters rather than the precision of $P_{\mathscr{B}}(y, \mathbf{x})$. As a result, it is usually important to ensure that all variables in the class' Markov blanket are connected directly to the class. As a consequence, Y is the common parent for all the random variables as shown in Figure 2. The conditional class probability of a BNC can then be written as

$$P_{\mathscr{B}}(y|\mathbf{x}) = \frac{P_{\mathscr{B}}(y, \mathbf{x})}{P_{\mathscr{B}}(\mathbf{x})} = \frac{\theta_y \prod_{i=1}^{n} \theta_{x_i|y,\Pi_{x_i}}}{\sum_{y' \in Y} \theta_{y'} \prod_{i=1}^{n} \theta_{x_i|y',\Pi_{x_i}}} \propto \theta_y \prod_{i=1}^{n} \theta_{x_i|y,\Pi_{x_i}} \tag{2}$$

There are some basic BNCs that have been developed and gained popularity, including NB, TAN, KDB and SKDB. We use Figure 2 to introduce these BNCs more clearly.

NB (Lewis, 1998) is the simplest BNC with a strong independence assumption that each attribute is conditionally independent of every other attribute given the class label. This makes the class the parent of all other attributes and includes no other edges (See Figure 2a).

TAN (Friedman et al, 1997) adds a single parent to each non-class attribute, seeking to address the greatest conditional inter-dependencies. It uses the Chow-Liu (Chow and Liu, 1968) algorithm to find the maximum-likelihood tree of dependencies



(a) NB (b) KDB-2

Fig. 2 Example BNC structures: Naïve Bayes on left and KDB-2 on right.

in polynomial time. The type of network learned by the TAN algorithm is called a one-dependence estimator (ODE).

KDB (Sahami, 1996) allows each non-class attribute to have up to K parents, where K is user-defined (See Figure 2b). It first sorts the attributes on mutual information with the class. Each attribute x_i is assigned the K parent attributes that maximise conditional mutual information (CMI) with the class out of those attributes with higher mutual information with the class. SKDB (Martınez et al, 2016) extends KDB by selecting values $n^* \leq n$ and $k^* \leq maxK$ for a KDB over n^* attributes with k^* parents using incremental cross-validation.

AODE (Webb et al, 2005) is an augmentation of NB that relaxes the strong independence assumption of NB by averaging over multiple structures that relax some of the independence assumptions. In each of these structures, a different attribute is set to be the parent of all other attributes. Then, at prediction time, class probability estimates from the different structures are simply averaged. AODE is an efficient classifier with the same simplicity as NB.

KDF (Duan and Wang, 2017) is an ensemble model combining multiple KDBs by changing the predictive attribute orders. The ensemble size of KDF is equal to the number of attributes in the dataset, where each base KDB estimator has a different first attribute. KDF considers not only the mutual information between attribute and the class but also the conditional mutual information between prior selected attributes. The classification accuracy of KDF outperforms single BNCs and AODE.

2.2 Bayesian Model Averaging

Bayesian model averaging for Bayesian networks was first suggested in (Buntine, 1991) for the unsupervised problem. An MCMC approach (Madigan et al, 1995) works for smaller data sets for the unsupervised case and more recent variants use the k-best structures (Tian et al, 2010). A variant of the problem is using model averaging to predict the existence of individual arcs. When done as a whole, this gives Bayesian structure discovery (Koivisto and Sood, 2004). For classification, an approximate model averaging (AMA) scheme is proposed (Dash and Cooper, 2004). This demonstrates, for classification, the effectiveness of model averaging versus model selection, demonstrating a marginal improvement over NB. Model averaging

techniques have also been developed for decision tree classification (Buntine, 1993; Chipman et al, 1998), but these techniques generally do not scale for large data and Random Forest is usually used in their place.

Having viewed the model averaging approach, which did not have the desirable scaling properties of SKDB, we instead choose to use classical ensemble methods, reviewed next. While for the Bayesian approach, the ensembles are sampled according to the posterior, in the classical ensemble approach, custom techniques are needed to sample ensembles.

2.3 Ensemble Learning

Ensemble learning improves the performance of single learners by training a set of learners to solve the same problem and combining them. An ensemble is stronger than a single learner for several reasons. First, ensemble methods can apply different information from the training data by combining some equally performing single learners. Second ensembles are more likely to include a better hypothesis than a single learner. Third, ensembles can give better approximations to the true target function than single ones (Zhou, 2015). Representatives of ensemble learning are boosting (Freund and Schapire, 1995) and bagging (Breiman, 1996).

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Although, boosting can be viewed as an additive model rather than ensembling (Friedman et al, 2000). Models are added sequentially until no further improvements can be made. XGBoost (Chen and Guestrin, 2016), short for eXtreme Gradient Boosting, is a scalable tree boosting system that has recently dominated applied machine learning and Kaggle competitions for structured and tabular data. XGBoost has two major improvements: speeding up the tree construction and proposing a new distributed algorithm for tree searching.

Bagging (Breiman, 1996) constructs an ensemble by generating multiple base learners on different sub-training sets. These subsets are sampled with replacement from the original training dataset, where the sizes of these subsets are the same as the whole training set. RF (Breiman, 2001) is a variant of bagging that is considered one of the most powerful ensemble methods. There are two aspects of randomness for each decision tree in the forest: (1) random sampling from the original training set to generate the subsets for each decision tree, as bagging does, and (2) randomly selecting a subset of attributes for use by the conventional split selection procedure to build the decision tree.

For an ensemble of T learners $h_1, ...h_T$, the well known bias-variance decomposition can be further expanded, yielding the bias-variance-covariance decomposition (Zhou, 2012). Without loss of generality, suppose that the individual learners are combined with equal weights. The bias-variance-covariance decomposition of the mean squared error of the ensemble is

$$MSE(H) = \overline{bias}(H)^2 + \frac{1}{T}\overline{variance}(H) + \left((1 - \frac{1}{T}\right)\overline{covariance}(H) \tag{3}$$

where averaged bias and averaged variance are the averages over the ensemble and the average covariance is the averaged covariance over all possibly pairs ($T(T-1)/2$ covariances). This shows that ensembles can reduce the variance while retaining the bias, but the individual learners need low average correlation. Thus to use ensembling

well, we need good performing learners with lower correlation. Note, this intuition also reflects Breiman's experience in developing RF: a significant part of the effort of developing an ensemble method is developing an appropriate algorithm for generating alternatives for the ensemble.

2.4 Probability Smoothing

Maximum Likelihood Estimation (MLE) for partitioning algorithms gives probability estimates only depending on the observed counts $n_1, n_2, ..., n_C$ for each class. The probabilities can be written as $p_c = \frac{n_c}{N}$, where C is the number of class labels. MLE often give unreliable probability estimates, especially when n_c has few counts or zero counts, decreasing the accuracy and ranking performance of classifiers.

Laplace correction (Provost and Domingos, 2003) is the simplest smoothing method. The strategy of this method is to add one to each count to turn zero counts into non-zero ones. M-estimation (Zadrozny and Elkan, 2001) modifies Laplace correction, by computing the probabilities using $p_c = \frac{n_c + M \times b}{N + M}$, where b is the base rate usually assumed to be $b = \frac{1}{C}$. M is a parameter that controls how much scores are shifted towards b, which is usually set to 1. Laplace correction is a special case of M-estimation where $M = C$.

2.5 HDP Smoothing

HDP has recently proven to be very useful for language model smoothing (Shareghi et al, 2017). Language models traditionally use hierarchical backoff smoothing, but the techniques are highly customised for n-gram models and do not suit more general trees. HDPs are combined with these (Shareghi et al, 2017) as a "finishing" method to improve probability estimates. HDP smoothing is also applicable to hierarchical models, like decision trees and BNCs (Petitjean et al, 2018). HDP smoothing assumes that the conditional distribution of a leaf node in the hierarchical model is similar to its parent node and the sibling nodes who share the common parent to it. This is achieved using a hierarchical Dirichlet prior to all the parameters in the tree.

This section briefly reviews this hierarchical Dirichlet smoothing. Please refer to (Petitjean et al, 2018) for more details about the theory, algorithm and their HDP tree smoothing library in Java, the basis of our implementation.

Consider the case of estimating $P(x|y, x_1, x_2, ...x_n)$ where the variables $x_1, x_2, ...x_n$ for $n \geq 0$ are the parents of x with a given order, resulting in a full joint table for all the values of these variables. This full joint table can be represented as a decision tree where the root node tests on y, all nodes at the first level split on x_1, the second level tests on x_2 and so forth. A node at the leaf has the parameter vector $\theta_{X|y,x_1,x_2,...,x_n}$. A node at the i^{th} level has a latent prior parameter vector $\phi_{X|y,x_1,x_2,...,x_i}$. The full hierarchical model is given by, for $i = 1, ..., n-1$

$$\begin{aligned}
\theta_{X|y,x_1,...,x_n} &\sim Dir(\phi_{X|y,x_1,...,x_{n-1}}, \alpha_n) \\
\phi_{X|y,x_1,...,x_i} &\sim Dir(\phi_{X|y,x_1,...,x_{i-1}}, \alpha_i) \\
\phi_{X|y} &\sim Dir(\phi_X, \alpha_y) \\
\phi_X &\sim Dir\left(\frac{1}{|X|}\mathbf{1}, \alpha_0\right)
\end{aligned}$$

Here $Dir(\phi, \alpha)$ is the Dirichlet distribution parameterized[1] by probability vector ϕ and positive real α, which are the base distribution and the concentration parameter respectively. In our model, the base distribution is the parent's probability vector, and α controls how similar the sampled distribution is to the base distribution. A larger α means they should be more similar; it corresponds to an inverse variance. Below we use a different α per level of the tree, though different configurations are experimented with subsequently.

The probability smoothing formula is defined recursively as follows

$$\hat{\theta}_{X|y,x_1,\ldots,x_n} = \frac{N_{X|y,x_1,\ldots,x_n}}{N_{\cdot|y,x_1,\ldots,x_n} + \alpha_n} + \frac{\alpha}{N_{\cdot|y,x_1,\ldots,x_n} + \alpha_n} \hat{\phi}_{X|y,x_1,\ldots,x_{n-1}}$$
$$\hat{\phi}_{X|y,x_1,\ldots,x_i} = \frac{N_{X|y,x_1,\ldots,x_i}}{N_{\cdot|y,x_1,\ldots,x_i} + \alpha_i} + \frac{\alpha}{N_{\cdot|y,x_1,\ldots,x_i} + \alpha_i} \hat{\phi}_{X|y,x_1,\ldots,x_{i-1}}$$

Here $N_{X|y,x_1,\ldots,x_n}$ are the observed vector of statistics for $\theta_{X|y,x_1,\ldots,x_n}$, and $N_{\cdot|y,x_1,\ldots,x_n}$ their totals. The $N_{X|y,x_1,\ldots,x_i}$ are the vector of sufficient statistics for $\phi_{X|y,x_1,\ldots,x_i}$ generated by the training algorithm.

This formula can be derived from the Chinese Restaurant Process (CRP) (Teh and Jordan, 2010). Each node in the tree can be compared to a restaurant offering $|Y|$ dishes. Each data point in the node is a customer, where $y \in Y$ are the dishes, and $N_{X|y,x_1,\ldots,x_n}$ is a vector of the number of customers who are eating dish y. Then a new customer comes in and can sit in an existing table serving dish y with probability $\frac{N_{x|y,x_1,\ldots,x_n}}{N_{\cdot|y,x_1,\ldots,x_n} + \alpha_n}$, or choose to eat at another table with probability $\frac{\alpha_n}{N_{\cdot|y,x_1,\ldots,x_n} + \alpha_n}$. The value of statistics for $\phi_{X|y,\ldots}$ and the αs are sampled using Gibbs sampling (Du, 2011; Petitjean et al, 2018). Our method, however, does not use hierarchical CRP methods, which have high memory consumption and are slow. Rather, we use an efficient collapsed sampler, roughly doubling the memory requirements of simple M-estimation and only a few times slower.

It has been illustrated in (Petitjean et al, 2018) that the concentration parameter controls how similar the child probability will be to the parent probability. Since the number of concentration parameters is equal to the number of nodes in the tree, there are possibly too many to sample. So rather than using a separate concentration parameter for every node, we tie some together to share the same concentration. There are four different tying strategies experimented with as follows,

- None: no tying, each node has its own concentration parameter.
- Parent: the sibling nodes sharing the same parent are tied together.
- Level: the nodes on the same level are tied together.
- Single: all the nodes are tied together share one single concentration.

3 ESKDB: an ensemble of BN classifiers

This section describes our ESKDB algorithm. As we have discussed in Section 2.3, good ensembles require that the base classifiers be as accurate as possible, and also as diverse as possible. Our base learner is SKDB, which is deterministic. To build an

[1] The more common representation $Dir(\alpha_1, \ldots, \alpha_C)$ is not used here.

ensemble, we inject stochasticity into the learning process of SKDB. We use two types of stochasticity: (1) a stochastic selection of cut-points for continuous attributes, and (2) sampling the attribute order for each SKDB classifier. We detail these two sources of stochasticity and then present our improved smoothing technique (prior and associated sampler for HDP). These ideas are motivated by the approach of RF and the ensemble theory of Section 2.3, but modified for the particular characteristics of Bayesian networks. The two sources of stochasticity are placed at the major design points for KDB.

Algorithm 1 gives the general learning framework for ESKDB: it takes as inputs a training set $\mathscr{T}$ and a user-supplied ensemble size E. It returns an ensemble model $\mathscr{B}$ that consists of E different SKDB classifiers. $\mathscr{B}$, which contains the base learners, is first initialised (line 1). The for-loop learns E different SKDB classifiers one by one (line 2-10). To learn the classifier i, we start by obtaining a randomized discretization (with associated cut-points $cutPoints_i$ – described below in Section 3.1), then learn the structure $\mathscr{G}_i$ of the Bayesian network and associated parameters and the parameters $\mathscr{B}_i$ in sequence. Sampling the cut-points means selecting preferred cut-points stochastically for each continuous attribute in the training data $\mathscr{T}$, as described in Subsection 3.1. The training data is then discretized according to $cutPoints_i$. Another important contribution of learning the structure $\mathscr{G}_i$ is that its attribute order $\mathscr{O}_i$ is randomly sampled (detailed below in Section 3.2). We then use the standard SKDB algorithm (Martınez et al, 2016), with the difference that it is now parameterized by our sampled order $\mathscr{O}_i$ and discretized dataset $\mathscr{T}_i$. We finally add our improved HDP smoothing to learn the parameters of the model (SKDB learns the structure only), by using the method developed in (Petitjean et al, 2018) with our improved prior and sampler (detailed in Section 3.3).

Algorithm 1: learnESKDB($\mathscr{T}$,E)

```
   Input  : A training set 𝒯
   Input  : an ensemble size E
   Output: An ESKDB model ℬ
 1 Let ℬ ← ∅.
 2 for i ← 1 to E do
 3 |   cutPoints_i ← learnDiscretizer(𝒯). // Section 3.1
 4 |   𝒯_i ← discretize(𝒯, cutPoints_i)
 5 |   𝒪_i ← randomSampleForOrders(𝒯_i). // Algorithm 3
 6 |   𝒢_i ← learnSKDB(𝒪_i, 𝒯_i). // (Martınez et al, 2016)
 7 |   Θ_i ← learnParameters(𝒢_i, 𝒯_i). /* build probability tables and smooth
   |      estimates (see (Petitjean et al, 2018) and Section 3.3).          */
 8 |   ℬ_i ← (𝒢_i, Θ_i, cutPoints_i).
 9 |   ℬ ← ℬ ∪ ℬ_i.
10 end
11 return ℬ.
```

3.1 Randomized Discretization

The first source of stochasticity to vary the deterministic behaviour of our base classifier SKDB is about discretization of numerical attributes. We achieve this by

proposing a new randomized discretization method for continuous attributes based on the method of MDLP (Fayyad and Irani, 1993). The question we answer in this section is how to generate different discretizations that both good and varied. The discretizations cannot be fully random or else the accuracy of the base learner would degrade. What we propose here is, for each continuous attribute, to calculate the quality of all the different cut-points (similarly to the standard MDLP discretizer), but then instead of using the best cut point, we use all the quality values to build a probability distribution from which we then sample. Sampling allows for variations in the process, but the sampler is guided by the quality of the cut-points, which means that bad cut-points have a very low probability of being selected.

Now that we have given the motivation for the randomized discretization, let us give a few more high-level details before turning to the algorithm itself. The first step is to identify all the candidate cut-points P_A. This is standard: we sort all the values of the attribute and use all the midpoint values for each pair of adjacent values. We then score these candidates using information gain, which is the same criterion as in MDLP. MDLP provides a minimum level of information gain for the attribute to be discretized. We then subtract that minimum level from all the gains, zero-out the negative values (the ones not passing the MDLP criterion), and normalize to 1. We then obtain a categorical probability distribution, which we sample to choose a cut point before calling the discretization recursively on the left and right sides of that cut point. The additional case we add is to ensure we have at least one cut point: if no candidate cut-point could pass the MDLP criterion, we sample a single cut point from information gain.

Algorithm 2 details this process. The input is a set of possible cut-points $\mathscr{P}$ and a flag $firstFlag$. The output is a set of the selected cut-points $\mathscr{P}^s$. The algorithm starts by calculating the entropy vector (information gain) IG and the difference vector IG^t (line 3-14). If the extreme case occurs where no cut point could pass the MDLP criterion, i.e. the $\sum IG^t = 0$ and this is not a recursive call ($firstFlag = true$), we normalize the entropy vector IG into a multinomial distribution, samples a cut point and return it (line 16-26). Otherwise, sample a cut point from the normalized IG^t and recursively selects point for the left subset $\mathscr{P}_l = \{p \in \mathscr{P} | p < p^s\}$ and the right subset $\mathscr{P}_l = \{p \in \mathscr{P} | p > p^s\}$ (line 28-35).

3.2 Randomized Attribute Ordering

The second source of stochasticity we use to vary our base learner SKDB is to randomize its attribute order. As discussed in Section 2.1, the attribute order determines the dependencies between the attributes that SKDB will use to make predictions. Attributes are usually ordered using mutual information and the top selected attribute is more likely to be selected as the parent of another attribute. SKDB calculates the Mutual Information $MI(X_i;Y)$ to measure the correlation between attributes X_i and the class Y. The attribute order is decided by sorting the attributes with the MI in descending order. This is so that the attributes that correlate most with the class are studied in combination with other attributes the most.

Our idea here is to randomize the order: instead of ordering the attributes by sorting them on mutual information, we turn the mutual information to a probability vector which we sample to find the first attribute. We then remove that attribute from the potential choices (make its probability zero), renormalize and repeat until

Algorithm 2: *randomSampleCutPoints*($\mathscr{P}, firstFlag$)

```
   Input  : possible cut-points 𝒫
   Input  : firstFlag ← true if this is the first time this method been called to ensure we
            have at least 1 cut point
   Output: the selected cut-points 𝒫^s
 1 𝒫^s ← ∅
 2 /* we sample cut-points by building a probability distribution over all the
      possible cut-points.                                                    */
 3 Let IG be a vector of the information gain for all the cut-points in 𝒫
 4 Let IG^t be a vector of the information gain minus the MDL threshold.
 5 for each cut point p ∈ 𝒫 do
 6 |   IG_p ← InformationGain(p)
 7 |   threshold_p ← MDL(p)
 8 |   // refer to (Fayyad and Irani, 1993)
 9 |   IG^t_p ← IG_p − threshold_p
10 |   if IG_p < threshold_p then
11 |   |   // if p doesn't meet the MDL criterion, then p will not be selected
12 |   |   IG^t_p ← 0
13 |   end
14 end
15 /* if all the cut-points do not meet the MDL criterion and firstFlag = true ,
      we assume that we get at least 1 cut point                              */
16 if ∑IG^t = 0 then
17 |   if firstFlag then
18 |   |   Normalize IG into a probability distribution.
19 |   |   if ∑IG ≠ 0 then
20 |   |   |   p^s ∼ IG // sampling from a multinomial distribution
21 |   |   |   𝒫^s ← 𝒫^s ∪ {p^s}
22 |   |   |   return 𝒫^s
23 |   |   end
24 |   end
25 |   return ∅
26 end
27 /* Otherwise, at least one cut point meet the MDL criterion                */
28 Normalize IG^t into a probability distribution.
29 p^s ∼ IG^t // sampling from a multinomial distribution
30 𝒫^s ← 𝒫^s ∪ {p^s}
31 /* recursively calling the left and right of p^s                           */
32 𝒫_l ← {p ∈ 𝒫 | p < p^s}
33 𝒫_r ← {p ∈ 𝒫 | p > p^s}
34 𝒫^s ← 𝒫^s ∪ randomSampleCutPoints(𝒫_l, false)
35 𝒫^s ← 𝒫^s ∪ randomSampleCutPoints(𝒫_r, false)
36 return 𝒫^s
```

all attributes have been chosen. This ensures that attributes with high correlation with the class are still likely to be in the top attributes, but makes it possible to vary the base learners and hence lower their covariance. This process is described in Algorithm 3. The method returns $\mathscr{O}$ a sampled attribute order to build an SKDB classifier.

3.3 Improved HDP Smoothing

We have now seen the two main sources of stochasticity that we use to vary our base learner and turn to our improved smoothing technique (or probability estimation).

Algorithm 3: randomSampleForOrders($\mathscr{T}$)

```
Input  : A training set 𝒯 with attributes 𝒜
Output: Orders 𝒪
n ← |𝒜|
MI ← {MI_i ← 0, i = 1,...,n}
for each A_i ∈ 𝒜 do
    MI_i ← mutualInformation(A_i; Y)
end
Let 𝒪 ← ∅.
for i ← 1 to n do
    MI ← normalize(MI) // normalize MI into a probability distribution that
        sums to 1
    z ~ MI // sampling from a multinomial distribution
    𝒪 ← 𝒪 + 𝒜_z.
    MI_z ← 0. // make sure A_z will not be selected again
end
return 𝒪
```

A critical issue for HDP estimation shown in (Petitjean et al, 2018) is its computational complexity. Petitjean et al (2018) show that the Gibbs sampler requires 50,000 iterations to obtain the most accurate probability estimates. The training time complexity increases linearly with the number of iterations, and 50,000 iterations make HDP extremely slow.

In this subsection, we show how to improve this significantly. In our new method, 1,000 iterations of the sampler are sufficient to give accurate estimates. We achieve this by adding a more carefully crafted prior to the concentration parameter. Algorithm 4 describes the sampling for concentration parameters in the tree, taken from (Buntine and Mishra, 2014). In the experiments, we use a $Gamma(2,1)$ prior instead of the uniform prior of (Petitjean et al, 2018). This new prior corresponds to a Gamma distribution with shape 2 and rate 1 ($priorShape = 2$ and $priorRate = 1$ in Algorithm 4). This is the default prior reported to work well with advance topic models (Buntine and Mishra, 2014). To interpret this, note that α corresponds to M in the M-estimate, so in estimation one adds $\alpha/|X|$ to the positive count. The prior makes α have a prior mean of 2 and a prior standard deviation of 1.4. Moreover, one is apriori 90% confident that α lies between and 0.36 and 4.7.

4 Experiments

This section aims to show the performance of our ESKDB algorithm in terms of accuracy, training time, and probability calibration. After having described the datasets and setting in Section 4.1, we show that ESKDB is the best out-of-core BNC to date (Section 4.2). We then proceed with a sensitivity analysis (ablation study) of ESKDB about its two sources of stochasticity in ESKDB (Section 4.3), before settling the question of ensemble size (Section 4.4). We then compare our superior HDP smoothing technique to traditional m-estimation (Sections 4.5 and 4.6). We then compare ESKDB to RF and XGBoost in Section 4.7. Finally, we study the running time for all the models mentioned in this paper 4.8.

Algorithm 4: sampleConcentration(α, *nodes*, *priorShape*, *priorRate*)

```
   Input  : α: concentration to sample
   Input  : nodes: nodes sharing this concentration parameter (tying)
   Input  : priorRate: prior on rate
   Input  : priorShape: prior on shape
 1 rate ← priorRate
 2 sumTk ← 0
 3 for each node ∈ nodes do
 4 |   q ~ Beta(α, node.n)
 5 |   rate ← rate − log(q)
 6 |   sumTk ← sumTk + node.t
 7 end
 8 α ~ Gamma(sumTk + priorShape, rate)
 9 for each node ∈ nodes do
10 |   node.α ← α
11 end
```

4.1 Experiment Design and Setting

Design: An extensive set of experiments are conducted on 72 standard datasets, where most of them are from the UCI archive (Lichman, 2013), but some larger datasets are also included from (Martınez et al, 2016). Table 1 summarizes the characteristics of 72 datasets, including the dataset name, size, number of attributes and number of classes. A missing value is treated as a separate attribute value.

Software: To ensure reproducibility of our work and allow other researchers to build on our research easily, we have made our source code for ESKDB available on Github [2].

Evaluation Measure: We use 5 times 2-fold cross-validation for all the methods and only a single run of 2-fold cross-validation for the six largest datasets. The results are assessed by RMSE (Root Mean Squared Error) and error rate. Win-Draw-Loss (WDL) is used when comparing two different models. A one-tail binomial sign test is used to determine the significance of the results, using $p \leq 0.05$.

RMSE, in discrete contexts known as the Brier score, which is used to measure how well-calibrated the probability estimates are. We use RMSE as the most important measure because the main research goal of this paper is to improve the probability estimates, a reasonable proxy for a variety of other decision contexts. RMSE is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}\ , \tag{4}$$

where y_i is the ground truth label and $\hat{y}$ is the predicted label.

The error rate is the proportion of samples that are misclassified. It is defined as

$$error\ rate = \frac{FP + FN}{N}\ , \tag{5}$$

where FP is the number of false-positive predictions and FN is the number of false-negative predictions. N is the size of the test set. The smaller the RMSE and the error rate, the better the performance of the model.

[2] https://github.com/icesky0125/ESKDB-on-numerical-data

Table 1 Datasets

Domain	Case	Att	Class	Domain	Case	Att	Class
Donation	5,749,132	12	2	Vowel	990	14	11
Poker-hand	1,025,010	11	10	Tic-Tac-Toe	958	10	2
Census-income	299,285	42	2	Anneal	898	39	6
Skin-Segment	245,057	4	2	Vehicle	846	19	4
Localization	164,860	6	11	PIndiansDiabetes	768	9	2
Diabetes	101,766	47	3	BreastCancer-w	699	10	2
Connect-4	67,557	43	3	CreditScreening	690	16	2
Shuttle	58,000	10	7	BalanceScale	625	5	3
Adult	48,842	15	2	Syncon	600	61	6
LetterRecognition	20,000	17	26	Chess	551	40	2
Magic	19,020	11	2	Cylinder	540	40	2
Nursery	12,960	9	5	Musk1	476	167	2
Sign	12,546	9	3	HouseVotes84	435	17	2
PenDigits	10,992	17	10	HorseColic	368	22	2
Thyroid	9,169	30	20	Dermatology	366	35	6
Mushrooms	8,124	23	2	Ionosphere	351	35	2
Musk2	6,598	167	2	PrimaryTumor	339	18	22
Satellite	6,435	37	6	Heart Disease-c	303	14	2
Optical Digits	5,620	49	10	Hungarian	294	14	2
Texture	5500	41	11	Audiology	226	70	24
Page Blocks	5,473	11	5	New-Thyroid	215	6	3
Wall-following	5,456	25	4	Glass-id	214	10	3
Nettalk(Phoneme)	5,438	8	52	Sonar	208	61	2
Waveform-5000	5,000	41	3	Autos	205	26	7
Spambase	4,601	58	2	Wine	178	14	3
Abalone	4,177	9	3	Hepatitis	155	20	2
Hypothyroid	3,772	30	4	Teaching Assistant	151	6	3
Sick	3,772	30	2	Iris	150	5	3
Kr vs. kp	3,196	37	2	Lymphography	148	19	4
Splice-C4.5	3,190	62	3	Echocardiogram	131	7	2
Segment	2,310	20	7	Promoters	106	58	2
Car	1,728	8	4	Zoo	101	17	7
Yeast	1,484	9	10	Post-operative	90	9	3
Contraceptive-mc	1,473	10	3	Labor	57	17	2
German	1,000	21	2	LungCancer	32	57	3
LED	1,000	8	10	Contact-lenses	24	5	3

Compared models and parameters: In our experiment we use BNCs including NB, TAN, KDB, SKDB, AODE, KDF and ESKDB, and tree-based ensemble models including RF and XGBoost. We compare HDP and M-estimation as the smoothing techniques for them. Here M-estimation is used with a backoff strategy, which means when the leaf node has no data, back off the probability estimates to its nearest non-empty ancestor. The list of parameter settings is shown in Table 2.

4.2 ESKDB is better than existing BNCs

In this section, we compare several single BNCs, including NB, TAN, KDB and SKDB with our proposed ensemble model ESKDB using M-estimation. This is to show the benefit of ensembling of our model. Besides, we also compare ESKDB with two other existing ensemble models of BNCs, including AODE and KDF. This is to show our ensembling technique is better than others. The ensemble size of ESKDB is set to be 10 in this part.

Table 2 List of parameters

Methods	Parameter	Description
BNCs	$maxK = 5$	Maximum number of parents allowed for SKDB
	$E = 10$	Ensemble size
HDP	$Iteration = 1000$	Gibbs sampling iteration
	$BurnIn = 100$	Collect counts after sampling for 500 times
	$Tying = LEVEL$	Same LEVEL nodes share the same concentration
M-estimation	$M = 1/C$	The value for calculating M-estimation, C is the number of class labels
RF	$F = 100$	Number of trees
	$Atts = log_2(n) + 1$	Numbers of attributes for each splitting nodes
XGBoost	$objective = multi : softprob$	Softmax objective returns predicted probability
	$num\_rounds = 10, 50, 100$	The number of rounds for boosting
	$max\_depth = 2, 4, 6, 8$	Maximum tree depth

Table 3 Win-Draw-Loss for the ESKDB compared with existing BNCs. The value in boldface is statistically significant better tested by a one-tailed binomial sign test. A difference is considered to be significant if $p \leq 0.05$.

ESKDB vs.	RMSE	Error rate
NB	63-0-9	60-1-11
TAN	64-0-8	57-2-13
KDB	64-0-8	61-2-9
KDF	71-0-1	67-2-3
AODE	62-1-9	60-0-12
SKDB	60-0-12	59-3-10

Figure 1 shows the averaged RMSE and error rate over all the datasets for each model mentioned above, which are represented by blue curve and red curve respectively. It clearly shows that our ESKDB with only 10 classifiers gets much better performance than other models both on RMSE and error rate. Table 3 is the WDL result of comparing ESKDB with the existing BNC models. Values in boldface are statistically significant better tested by a one-tailed binomial sign test. A different is considered to be significant if $p \leq 0.05$. It can be seen from this table that ESKDB is significant better than all of the existing BNCs both on RMSE and error rate.

4.3 The benefits of the two stochasticities in ESKDB

In this section, we show the benefit of the two stochasticities in our ESKDB model. The first one is that the cut-points are randomly selected for each SKDB classifier in the ensemble. The second one is that the attribute order of each SKDB is randomly selected.

First, to show the benefits of the first stochasticity, we compare ESKDB with ESKDB_randomO, which represents ESKDB with stochasticity on attribute orders

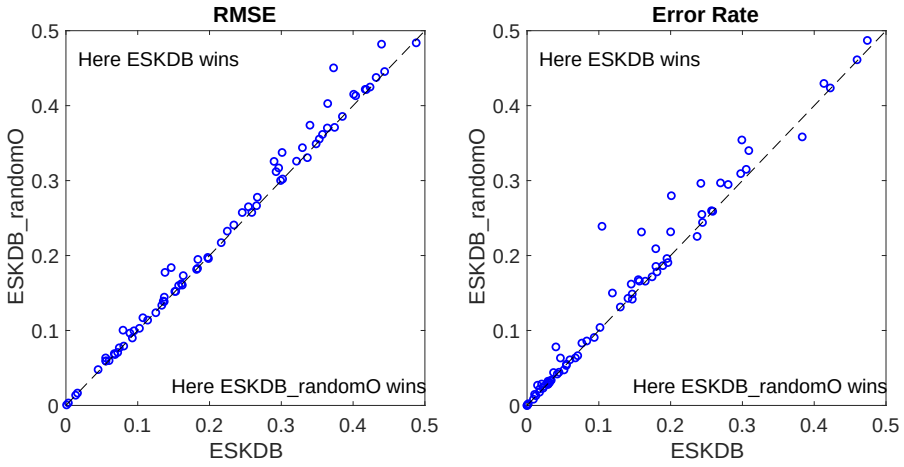


Fig. 3 Scatter plot of ESKDB with ESKDB_randomO

only (no random discretization, but just straight MDLP discretization). Figure 3 is the scatter plot ESKDB with ESKDB_randomO on both RMSE and error rate. It can be seen from this figure that the random selection of the cut-points makes ESKDB perform better both on RMSE and error rate compared with same cut-points in ESKDB_randomO. The averaged RMSE value of ESKDB and ESKDB_randomO are 0.2248 ± 0.016 and 0.2326 ± 0.017, respectively.

Second, to show the benefits of the random selection of the attribute order for each SKDB classifier, we compare ESKDB with ESKDB_randomCP, which represents ESKDB but where the attribute orders are not sampled. Figure 4 is the scatter plot of ESKDB and ESKDB_randomCP. It can be seen from this figure that ESKDB performs slightly better than ESKDB_randomCP, which indicates that the second stochasticity of the attribute orders can also make our ESKDB model have more diversity and get better results.

To compare the importance of these two randomnesses, we show the scatter plot of ESKDB_randomCP with ESKDB_randomO as shown in Figure 5. It can be seen from this figure that the two versions of ESKDB have a quite similar effect on ESKDB. The averaged RMSE value of ESKDB_randomO and ESKDB_randomCP are 0.2326 ± 0.017 and 0.2312 ± 0.017, respectively. This indicates that the randomness of the cut-points makes ESKDB slightly better than the randomness of the attribute orders.

4.4 The ensemble size of ESKDB

In this part of the experiment results, we show how ESKDB performs with different ensemble sizes, including 1, 10, 20, and 30. Figure 6 shows that both RMSE and error rate has a big improvement when ensembling on 10 SKDB classifiers compared with 1 classifier. However, even we increase the ensemble to 20 and 30, further improvement

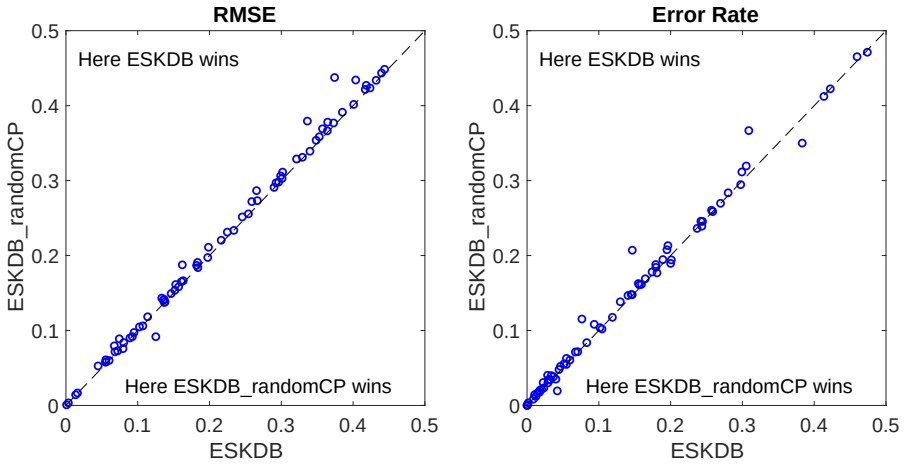


Fig. 4 Scatter plot of ESKDB with ESKDB_randomCP

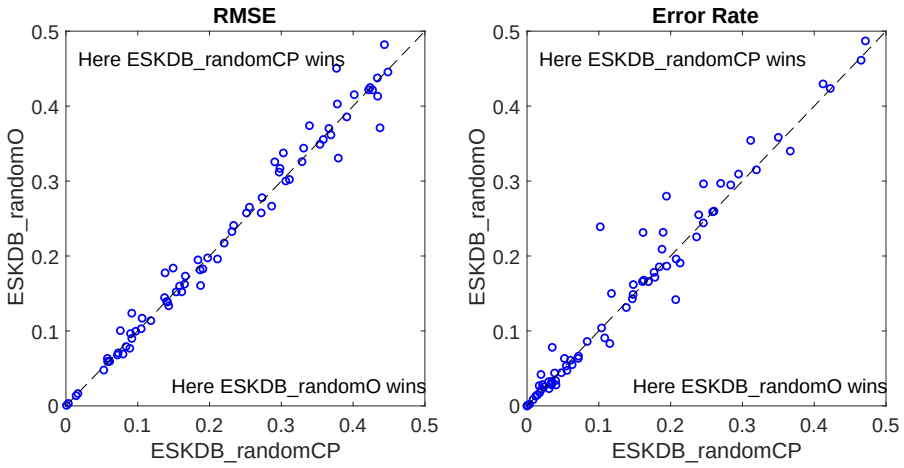


Fig. 5 Scatter plot of ESKDB_sameO compared with ESKDB_sameP

is not so big. This indicates that our ESKDB with only 10 classifiers already gives good results. In the following experiments, 10 is used for ESKDB.

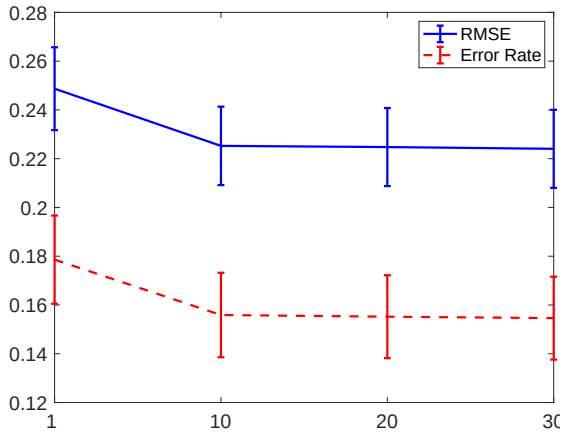


Fig. 6 The RMSE and error rate changes as the increasing of the ensemble size.

4.5 Improved HDP

In this part of the experiment results, we test the four different tying strategies of HDP smoothing on ESKDB and the benefit of the new Gamma prior, as was discussed in Section 3.3. The four different tying strategies are explained in Section 2.5. Figure 7 shows the RMSE of HDP smoothing using four different tying strategies and two different priors. It can be seen from this figure that no matter which tying strategy HDP used, the result of the new Gamma(2,1) is always better than the uniform prior. Besides, the *Level* strategy gets the best performance among all the four tying strategies. The *Level* tying strategy and the $Gamma(2,1)$ prior are used in the following experiments.

4.6 HDP compared with M-estimation for ESKDB

HDP has been shown for single BNCs to achieve more accurate parameter estimates compared with M-estimation. Here we aim to verify that HDP could also improve the performance of ensemble models. Note that for decision trees, conventional wisdom is that no smoothing leads to superior results with ensembles (Bostrom, 2007), but we need to check the case for BNCs.

Table 4 is the WDL of HDP compared with M-estimation applied to ESKDB. We can see from this table that HDP outperforms M-estimation for ESKDB on both RMSE and error rate. This indicates that HDP smoothing is helpful both to single and ensemble models. The average performance for HDP and M-estimation are shown in Table 5. HDP makes the RMSE improve from 0.2252 to 0.2227 with a cost of 7 times longer training time.

Both M-estimation and HDP can get good estimates, but they have advantages and shortcomings. M-estimation on ESKDB gets good results compared with the state-of-the-art classifiers with only a limited learning time. HDP achieves better

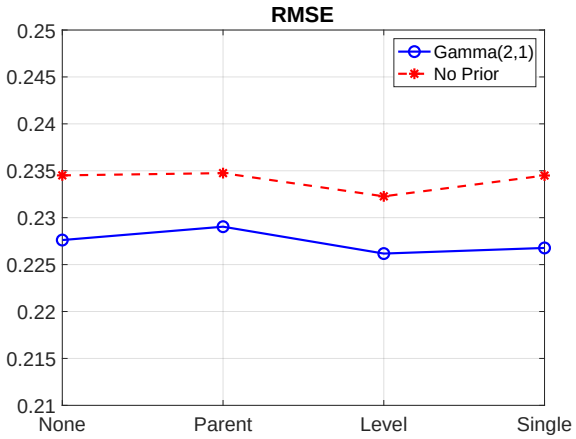


Fig. 7 The HDP smoothing using different tying strategies and the new prior.

Table 4 Win-Draw-Loss for the ESKDB using HDP and M-estimation. The value in boldface is statistically significant better tested by a one-tailed binomial sign test. A difference is considered to be significant if $p \leq 0.05$.

Smoothing methods	RMSE	error rate
HDP vs. M-estimation	48-1-23	38-7-27

Table 5 Averaged performance of ESKDB using HDP and M-estimation

Smoothing	RMSE	error rate	Time (s)
M-estimation	0.2252 ± 0.016	0.1559 ± 0.017	25
HDP	0.2227 ± 0.016	0.1536 ± 0.017	190

performance but with a cost of computation. We recommend the use of HDP for ESKDB and use that configuration in the remainder of this paper.

4.7 ESKDB compared to Random Forest and XGBoost

In this section, we compare our ESKDB model with the two state-of-the-art ensemble classifiers: Random Forest (RF) and XGBoost. RF is built with 100 trees, to pure leaves, and with the number of randomly selected features to be $atts = log_2(|A|)+1$ where $|A|$ is the total number of attributes. XGboost is an advanced implementation of gradient boosting algorithm. We present two versions of XGBoost: XGBoost$_{default}$, which uses default parameters ($max\_depth = 6$ and $num\_rounds = 100$); and XGBoost$_{tuned}$, for which we tune max_depth and num_rounds using 10-fold cross-validation (we use values $\{10, 50, 100\}$ for num_rounds and $\{2, 4, 6, 8\}$ for max_depth).

Table 6 ESKDB compared with RF and XGBoost

Classifier	RMSE	error rate	Time (s)
RF	0.2314 ± 0.016	0.1563 ± 0.018	6.8
XGBoost$_{default}$	0.2288 ± 0.018	0.1565 ± 0.018	2.7
ESKDB$_{M}$	0.2252 ± 0.016	0.1559 ± 0.017	25
ESKDB$_{HDP}$	0.2227 ± 0.016	0.1536 ± 0.017	190
XGBoost$_{tuned}$	0.2179 ± 0.017	0.1496 ± 0.017	60

Our ESKDB method runs basically parameter-free, with the only parameter controlling the maximum depth of our trees – $maxK$ – which we set to 5. Note that the higher $maxK$, the higher the accuracy as the actual value of K is cross-validated internally with a fast leave-one-out cross-validation in SKDB.

Tables 6 gives the average classification performance over all the datasets listed in Table 1. We can observe from Table 6 that, compared to RF, ESKDB has similar error rate but much better-calibrated probabilities. This result also holds when compared to XGBoost ran with default parameters. Performing a grid-search for the best XGBoost parameters allows it to turn ahead of the competition. It is important to remember here that the aim of this paper is not to show that ESKDB should now replace any other algorithm 'on the market', but rather that it is possible to obtain very accurate classifiers based on Bayesian Network classifiers. Also, note that our algorithm runs completely out-of-core while XGBoost does require large amounts of data to be stored in memory and on-disk. Table 7 gives a similar story with ESKDB finishing just behind a tuned version of XGBoost.

Figure 8 shows the comparison between ESKDB$_{HDP}$ and XGBoost$_{default}$ and XGBoost$_{tuned}$ in detail (ESKDB$_{HDP}$ wins above the diagonal line). This plot is interesting in that it shows the diversity of results obtained by XGBoost and ESKDB, with the results quite spread on either side of the diagonal line. This tends to indicate that there is some important benefit in having ESKDB available because even if the average RMSE is better for a tuned version of XGBoost, there are still many datasets for which ESKDB obtains a significant improvement over it. One point stands out particularly on the right scatter plot with XGBoost$_{tuned}$ obtaining almost a 0.2 improvement over ESKDB$_{HDP}$, which corresponds to the "tic-tac-toe" datasets. This toy dataset is very particular as it contains all the end-game boards of a 'tic-tac-toe' game, and only a single instance for each one (9 attributes, with 3 possible values – empty, cross or circle). It requires deep structures to represent it, with many combinations of 3 attributes to represent all combinations of aligned crosses. Unfortunately, for this dataset, SKDB chooses to use $k = 2$, making it impossible to obtain accurate estimates of the probabilities. If we remove that particular synthetic data, ESKDB$_{HDP}$ gets extremely close to XGBoost$_{tuned}$ – see Table 8 – which shows the high relevance of our proposed approach.

4.8 Running Time

In this section, we compare the training times for classifiers mentioned in this paper. Table 9 lists the averaged results over all the datasets. As can be seen from this table, it takes less than 5 seconds for NB, KDB, AODE and the default version of

Table 7 WDL of ESKDB compared with XGBoost and RF. The value in boldface is statistically significant better.

Classifier	RMSE	error rate
ESKDB$_M$ vs. RF	40-0-32	30-2-40
ESKDB$_M$ vs. XGBoost$_{default}$	38-1-33	30-3-39
ESKDB$_M$ vs. XGBoost$_{tuned}$	27-0-45	26-2-44
ESKDB$_{HDP}$ vs. RF	42-0-30	36-2-34
ESKDB$_{HDP}$ vs. XGBoost$_{default}$	44-0-28	36-1-35
ESKDB$_{HDP}$ vs. XGBoost$_{tuned}$	33-1-38	29-2-41

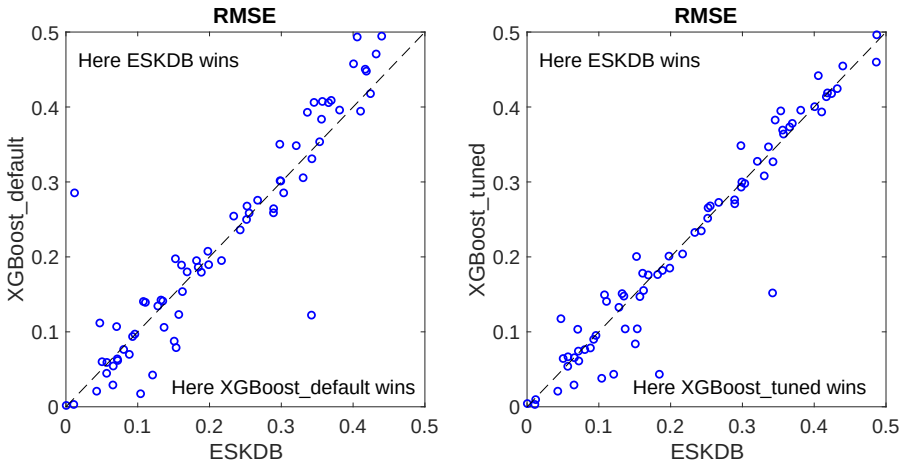


Fig. 8 Scatter plot of ESKDB$_{HDP}$ with XGBoost$_{default}$ and XGBoost$_{tuned}$ on RMSE

Table 8 ESKDB compared with RF and XGBoost without "tic-tac-toe" dataset

Classifier	RMSE	error rate
RF	0.2304 ± 0.016	0.1573 ± 0.018
XGBoost$_{default}$	0.2303 ± 0.018	0.1584 ± 0.018
ESKDB$_{HDP}$	0.2210 ± 0.016	0.1535 ± 0.017
XGBoost$_{tuned}$	0.2191 ± 0.017	0.1517 ± 0.017

XGBoost to be learned. Both of TAN and KDF need 5.7 seconds, and RF needs 6.8 seconds to be learned. The parameter-tuned XGBoost needs 60 seconds to be learned, which is around 22 times longer than XGBoost$_{default}$. This indicates that the parameter tuning for XGBoost is time-consuming. ESKDB$_M$ with 10 SKDBs needs 25 seconds, which is 1.8 times longer than SKDB. HDP makes ESKDB 7.6 times longer than M-estimation because of the 1,000 iterations of Gibbs sampling used. It can be seen from this table that ESKDB_M is 2.4 times faster than XGBoost_tuned and ESKDB_HDP is 3x slower than XGBoost_tuned, which we regard as a good

Table 9 Averaged training time (seconds) for all the datasets

Classifier	RMSE	error rate	Time (s)
$\text{XGBoost}_{default}$	0.2288	0.1565	2.7
NB	0.2753	0.2074	3
KDB	0.2557	0.1851	3.6
AODE	0.2498	0.1821	5
TAN	0.2568	0.1858	5.7
KDF	0.2512	0.1858	5.7
RF	0.2314	0.1563	6.8
SKDB	0.2418	0.1706	13.7
ESKDB_{M}	0.2252	0.1565	25
XGBoost_{tuned}	0.2179	0.1496	60
ESKDB_{HDP}	0.2227	0.1536	190

result given the amount of engineering that went into that classifier. However, like RF, ESKDB scales linearly with the ensemble size (10 used here).

4.9 Discussion

Experimental results show that ESKDB clearly outperforms all existing BNC algorithms (including ensembles such as KDF) both in terms of accuracy, probability calibration and functionally, as being able to handle numerical attributes. We then show that ESKDB obtains better-calibrated probabilities than Random Forest, which is critical for situations where the confidence in the predictions is important. We finish by showing that ESKDB performs competitively to XGBoost, depending on how much time is used to tune XGBoost's hyper-parameters.

Our ESKDB algorithm can be used either with simple Laplace-type smoothing (such as Laplace and M-estimation) or with our improved HDP estimator. This choice mostly trades off running time vs quality of the estimates: ESKDB_{HDP} runs approximately 8x slower than ESKDB_{M} but is able to have significantly better RMSE than an untuned XGBoost. It is also important here to underline that both versions of ESKDB can run without having to load the whole dataset in memory, in contrast to RF and XGBoost.

5 Conclusion and Future Work

This paper introduced ESKDB: a novel ensemble method for Bayesian Network classifiers that can deal with both categorical and numerical features. ESKDB combines two sources of stochasticity to modify SKDB to build diverse ensembles. Our ablation study clearly demonstrates that both sources of stochasticity are important to ESKDB and that ensembling alone produces better class probability estimates than existing BNCs both on RMSE and on error rate using M-estimation for smoothing. We also show that using our improved HDP smoothing gives an important gain to ESKDB with a cost of computational complexity compared with M-estimation.

We believe this work calls for interesting studies in how BNCs, with their interesting functional properties, could be further improved. We saw that an ensemble of

size 10 was sufficient to obtain most of the gains from ESKDB. Although this is a good property for running time, it seems to suggest that more work in diversifying the base classifiers could provide important gains in prediction. We will consider three directions to this aim: discretizing the attributes depending on the level of the trees, having SKDB be less conservative in the choice of $maxK$, and varying the level of smoothing (which tends to make trees more similar to each other. Another avenue for research is algorithm optimization to reduce training and testing time.

References

Bostrom H (2007) Estimating class probabilities in random forests. In: Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on, IEEE, pp 211–216

Breiman L (1996) Bagging predictors. Machine learning 24(2):123–140

Breiman L (2001) Random forests. Machine Learning 45(1):5–32

Buntine W (1991) Theory refinement of Bayesian networks. In: Seventh Conference on Uncertainty in Artificial Intelligence, Anaheim, CA

Buntine W (1993) Learning classification trees. In: Artificial Intelligence frontiers in statistics, Springer US, pp 182–201

Buntine W, Mishra S (2014) Experiments with non-parametric topic models. In: Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 881–890

Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SigKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 785–794

Chipman HA, George EI, McCulloch RE (1998) Bayesian CART model search. Journal of the American Statistical Association 93(443):935–948

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

Dash D, Cooper GF (2004) Model averaging for prediction with discrete Bayesian networks. Journal of Machine Learning Research 5(Sep):1177–1203

Du L (2011) Non-parametric Bayesian methods for structured topic models. PhD thesis, Australian National University

Duan Z, Wang L (2017) K-dependence Bayesian classifier ensemble. Entropy 19(12):651

Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning

Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory, Springer, pp 23–37

Friedman J, Hastie T, Tibshirani R, et al (2000) Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics 28(2):337–407

Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Machine learning 29(2-3):131–163

Hearst MA (1998) Support vector machines. IEEE Intelligent Systems 13(4):18–28

Hoeting JA, Madigan D, Raftery AE, Volinsky CT (1999) Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and E. I. George, and a rejoinder by the authors). Statistical Science 14(4):382–417
Koivisto M, Sood K (2004) Exact Bayesian structure discovery in Bayesian networks. Journal of Machine Learning Research 5(May):549–573
Lewis DD (1998) Naive Bayes at forty: The independence assumption in information retrieval, Springer, pp 4–15
Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
Madigan D, York J, Allard D (1995) Bayesian graphical models for discrete data. International Statistical Review/Revue Internationale de Statistique pp 215–232
Martınez AM, Webb GI, Chen S, Zaidi NA (2016) Scalable learning of Bayesian network classifiers. Journal of Machine Learning Research 17(44):1–35
Petitjean F, Buntine W, Webb GI, Zaidi N (2018) Accurate parameter estimation for Bayesian network classifiers using hierarchical Dirichlet processes. Machine Learning 107(8):1303–1331
Provost F, Domingos P (2003) Tree induction for probability-based ranking. Machine learning 52(3):199–215
Sahami M (1996) Learning limited dependence Bayesian classifiers. In: KDD, vol 96, pp 335–338
Shareghi E, Haffari G, Cohn T (2017) Compressed nonparametric language modelling. In: Proc. of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp 2701–2707
Teh YW, Jordan MI (2010) Hierarchical Bayesian nonparametric models with applications. Bayesian Nonparametrics 1
Tian J, He R, Ram L (2010) Bayesian model averaging using the k-best Bayesian network structures. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, AUAI Press, UAI'10, pp 589–597
Webb GI, Boughton JR, Wang Z (2005) Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning 58(1):5–24
Zadrozny B, Elkan C (2001) Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: ICML, Citeseer, vol 1, pp 609–616
Zhou ZH (2012) Ensemble Methods: Foundations and Algorithms, 1st edn. Chapman & Hall/CRC
Zhou ZH (2015) Ensemble learning. Encyclopedia of biometrics pp 411–416

Author Biographies

He Zhang is currently a PhD candidate in machine learning at Monash University, Australia. After receiving her masters' degree in data mining from China in 2015, she moved to Australia to begin her PhD study under the supervision of Prof. Wray Buntine and Dr François Petitjean. She has extensive research interests in machine learning and data mining, including classification, probability smoothing, ensemble learning, Positive Unlabelled learning and Cost-sensitive learning. She is also interested in research in interdisciplinary fields such as bioinformatics.

François Petitjean is a Senior Researcher (tenure) at Monash University in Melbourne, Australia. He moved to Australia in 2012 after finishing his PhD with the French Space Agency (CNES), for which he received several awards. He currently leads a team of 7 PhD candidates and 3 postdocs working in time series analysis, Earth observation and graphical models. He has published 50+ papers, received multiple 'Best paper' awards, and is the fortunate recipient of the prestigious DECRA fellowship funded by the Australian Government.

Wray Buntine is a full professor at Monash University in February 2014 after 7 years at NICTA in Canberra Australia. He was previously at Helsinki Institute for Information Technology where he ran a semantic search project, NASA Ames Research Center, University of California, Berkeley, and Google. In the '90s he was involved in a string of startups in Silicon Valley. He is known for his theoretical and applied work and in probabilistic methods for document and text analysis, social networks, data mining and machine learning. He has over 150 academic publications, several software products and two patents from his Silicon Valley days.