Journal of Machine Learning Research 1 (2013) 1-30 Submitted 7/13; Revised 10/13; Published 00/00

Scalable learning of Bayesian network classifiers

Ana M. Martínez ANAM.MARTINEZF@GMAIL.COM
Geoffrey I. Webb GEOFF.WEBB@MONASH.EDU
Faculty of Information Technology
Monash University
VIC 3800, Australia

Shenglei Chen TRISTAN_CHEN@126.COM
College of Information Science/Faculty of Information Technology
Nanjing Audit University/Monash University
China/Australia

Nayyar A. Zaidi NAYYAR.ZAIDI@MONASH.EDU
Faculty of Information Technology
Monash University
VIC 3800, Australia

Editor: Kevin Murphy

Abstract

Ever increasing data quantity makes ever more urgent the need for highly scalable learners that have good classification performance. Therefore, an out-of-core learner with excellent time and space complexity, along with high expressivity (i.e., capacity to learn very complex multivariate probability distributions) is extremely desirable. This paper presents such a learner. We propose an extension to the k-dependence Bayesian classifier (KDB) that discriminatively selects a sub-model of a full KDB classifier. It requires only one additional pass through the training data, making it a three-pass learner. Our extensive experimental evaluation on 16 large data sets reveals that this out-of-core algorithm achieves competitive classification performance, and substantially better training and classification time than state-of-the-art in-core learners such as random forest and linear and non-linear logistic regression.

Keywords: scalable Bayesian classification, feature selection, out-of-core learning, big data

1. Introduction

Until very recently most machine learning research has been conducted in the context of relatively small datasets, i.e., no more than $50,000$ instances and a few hundred attributes. It is only in the last five years that larger datasets have started to appear in the literature (Erhan et al., 2010; Sonnenburg and Franc, 2010; Agarwal et al., 2011; Sariyar et al., 2011) on a regular basis. We argue that larger data quantities do not simply call for scaling-up of existing learning algorithms. Rather, we believe, as first argued by Brain and Webb (1999), that the most accurate learners for large data will have much lower bias than the most accurate learners for small data. Thus we need a new generation of computationally efficient low bias learners. To achieve the necessary efficiency, a learning algorithm for very large data should ideally require only a few passes through the training data. Further, to remove memory size as a bottleneck, the algorithm should be able to process data out-of-core.

In this paper, we extend the KDB classifier. KDB is a form of restricted Bayesian network classifier (BNC). It has numerous desirable properties in the context of learning from large quantities of data. These include:

- the capacity to control its bias/variance trade-off with a single parameter, k,
- it does not require data to be held in-core,
- the capacity to learn in just two passes through the training data.

The contributions of this paper are as follows:

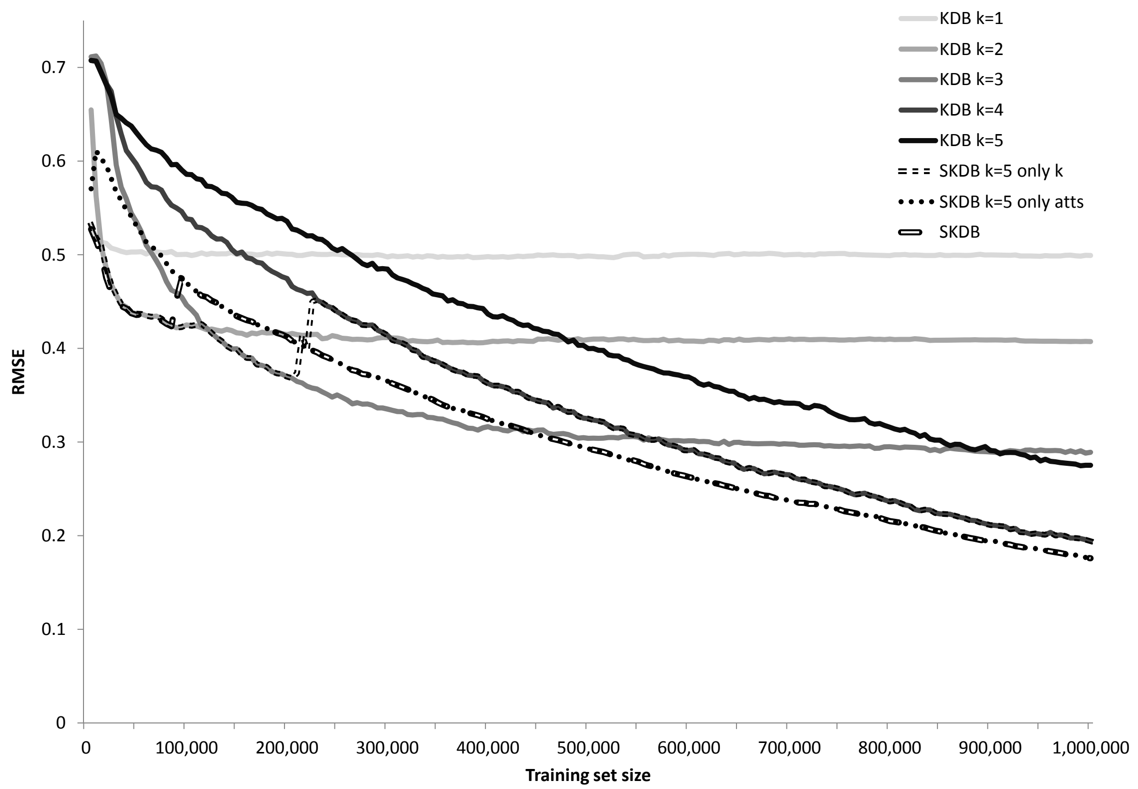
- We extend KDB to perform discriminative model selection in a single additional pass through the training data. In this single pass, our algorithm selects between both attribute subsets and network structures. The resulting highly scalable algorithm combines the computational efficiency of classical generative learning with the low bias of discriminative learning.
- We compare the performance of our algorithm with other state-of-the-art classifiers on several large datasets, ranging in size from 165 thousand to 54 million instances and 5 to 784 attributes. We show that our highly scalable algorithms achieve comparable or lower error on large data than a range of out-of-core and in-core state-of-the-art classification learning algorithms.

To illustrate our motivations, in Figure 1 we present learning curves for the poker-hand dataset (which is described in Table 7 in Appendix A.) As can be seen, for lower quantities of data the lower variance delivered by low values of k results in lower error for KBD, while for larger quantities of data the lower bias of higher values of k results in lower error. While selective KDB sometimes selects a k that overfits this data with smaller training set sizes, when the training set size exceeds $550,000$ it successfully selects the best k and by selecting a subset of the attributes it can substantially reduce error across the entire range of training set sizes relative to the KDB classifier using the same k.

Section 2 reviews the state-of-the-art in out-of-core BNCs and introduces our proposal: selective KDB (SKDB, Section 2.1). In Section 3 we present a set of comparisons for our proposed algorithm on 16 big datasets with *out-of-core* BNCs (Section 3.1), *out-of-core* linear classifiers using Stochastic Gradient Descent (Section 3.2), and with *in-core* (batch) learners BayesNet and Random Forest (Section 3.3). To finalize, Section 4 shows the main conclusions and outlines future work.

2. Scalable Bayesian Networks for Classification

We define the classification learning task as the assignment of a label $y \in \Omega_Y$, from a set of c labels of the class variable Y, to a given example $\mathbf{x} = (x_1, \ldots, x_d)$, with values for the d attributes $\mathcal{A} = \{X_1, \ldots, X_d\}$. Bayesian networks (BN) (Pearl, 1988) provide



The values plotted are averages over 10 runs. For each run 1,000 examples are selected as a test set and samples of successive sizes are selected from the remaining data for training. As can be seen, for lower quantities of data the lower variance delivered by low values of k results in lower error for KBD, while for larger quantities of data the lower bias of higher values of k results in lower error, although there is not sufficient data for $k = 5$ *to asymptote on this dataset. While only selecting a k (SKDB k=5 only k) causes selective KDB to overfit this data with smaller training set sizes (between 20,000 and 550,000), when the training set size is large enough it successfully selects the best k. Only selecting attributes (SKDB k=5 only atts) always reduces error (relative to KDB k=5). By selecting both k and attributes (SKDB k=5) selective KDB substantially reduces error relative to KDB with any value of k once there are more than approximately 450,000 training examples.*

Figure 1: Learning curves for KDB and Selective KDB on the poker-hand dataset.

a framework for computing the joint probability of these d random variables. A BN is a directed acyclic graph where the the joint probability of all attributes can be written as the product of the individual probabilities of all attributes given their parents, that is $p(y, \mathbf{x}) = p(y|\pi_y) \prod_{i=1}^{d} p(x_i|\pi_{x_i})$, where π_{x_i} denotes the parents of attribute X_i. If the structure is known, the likelihood of the BN given the data can be maximized by estimating $p(x_i|\pi_{x_i})$ using the empirical estimates from the training data $\mathcal{T}$, composed of t training instances.

While unrestricted BNs are the least biased, training such a model on even moderate size data sets can be extremely challenging, as the search-space that needs to be explored grows exponentially with the number of attributes. This has led to restricted BNCs, including naive Bayes (NB) (Duda and Hart, 1973; Minsky, 1961; Lewis, 1998), tree-augmented naive Bayes (TAN) (Friedman et al., 1997), averaged n-dependence estimators (ANDE) (Webb et al., 2012) and k-dependence estimators (KDB) (Sahami, 1996). These classifiers have in common either no structural learning or minimal structural learning that requires only one or two passes through the data. They have been referred to as semi-naive BNCs (Zheng and Webb, 2005). Even though highly biased due to their inherent conditional attribute independence assumption, they have been shown to be very effective classification methods.

NB is the simplest of the BNCs, assuming that all attributes are independent given the class. It estimates the joint probability using $\hat{p}(y)\prod_{i=1}^{d}\hat{p}(x_i|y)$, where $\hat{p}(\cdot)$ denotes an estimate of $p(\cdot)$. TAN is a structural augmentation of NB where every attribute has as parents the class and at most one other attribute. The structure is determined by using an extension of the Chow-Liu tree (Chow and Liu, 1968), that utilizes conditional mutual information (CMI) to find a maximum spanning tree. This alleviates some of NB's independence assumption and therefore reduces its bias at the expense of increasing its variance. This results in better performance on larger data sets. TAN, provides an intermediate bias-variance trade-off, standing between NB on one hand and unrestricted BNCs on the other.

There has been a lot of prior work that has explored approaches to alleviate NB's independence assumption. One approach is to add a bounded number of additional interdependencies (Friedman et al., 1997; Zheng and Webb, 2000; Webb et al., 2005; Zhang et al., 2005; Yang et al., 2007; Flores et al., 2009a,b; Zheng et al., 2012; Zaidi et al., 2013). At the other extreme, Su and Zhang (2006) propose the use of a full Bayesian network. Of these algorithms, only two approaches allow the number of interdependencies to be adjusted to best accommodate the best trade-offs between bias and variance for differing data quantities. ANDE (Webb et al., 2012) averages many BNCs, where a parameter n controls the number of interdependencies to be modeled. KDB uses a parameter k to control the number of interdependencies modeled.

KDB relaxes NB's independence assumption by allowing every attribute to be conditioned on the class and, at most, k other attributes. Like TAN, KDB requires two passes over the training data for learning: the first pass collects the statistics to perform calculations of mutual information for structure learning, and the second performs the parameter learning based on the structure created in the former step (Algorithm 1). An advantage over the ANDE model is that KDB does not need to collect all statistics for all combinations of n or k attributes, allowing it to scale to higher order dependencies. It also has the capacity to select a model and hence more closely fit the data. This is a disadvantage for small data, which it may overfit, but KDB will often lead to higher accuracy than ANDE on large data which are more difficult to overfit.

Recently, some techniques have been proposed that address the classification problem with Bayesian networks from a relatively efficient discriminative perspective. Carvalho et al. (2011) propose a decomposable approximation to the conditional log-likelihood scoring criteria. Pernkopf and Bilmes (2010) propose a BNC learning algorithm with some resemblance to KDB but using discriminative evaluation for parent assignment. Neither algorithm is suited to learning a BNC in a limited number of passes through the training data.

Algorithm 1: The KDB algorithm

```
   Input: Training set $\mathcal{T}$ with attributes $\{X_1, \ldots, X_a, Y\}$ and $k$.
   Output: KDB model.
 1 Let $\mathcal{G}$ be a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of
   links.
 2 Let $\Theta$ be a Bayesian network with structure $\mathcal{G}$.
 3 First pass begin
 4     $\mathcal{G}$ =learnStructure($\mathcal{T}$) ;                                  /* Algorithm 2 */
 5 end
 6 Second pass begin
 7     $\Theta$ = learnParameters($\mathcal{T}$, $\mathcal{G}$) ;                           /* Algorithm 3 */
 8 end
 9 Let KDB be a BN with structure $\mathcal{G}$ and conditional probability
   distributions in $\Theta$.
10 return KDB
```

Algorithm 2: learnStructure($\mathcal{T}$)

```
   Input: Training set $\mathcal{T}$
   Output: $\mathcal{G}$, network structure.
 1 Calculate $MI(X_i; Y)$ from $\mathcal{T}$ for all attributes.
 2 Calculate $CMI(X_i, X_j; Y)$ from $\mathcal{T}$ for each pair of attributes $(i \neq j)$.
 3 Let $\mathcal{L}$ be a list of all $X_i$ in decreasing order of $MI(X_i; Y)$.
 4 $\mathcal{V} = \{Y\}$; $\mathcal{E} = \emptyset$;
 5 for $i = 1 \to \mathcal{L}.size$ do
 6     $\mathcal{V} = \mathcal{V} \cup \mathcal{L}_i$;
 7     $\mathcal{E} = \mathcal{E} \cup (Y, \mathcal{L}_i)$;
 8     $vk = min(i-1, k)$;
 9     while $(vk > 0)$ do
10         $m = \text{argmax}_j \{MI(\mathcal{L}_i; \mathcal{L}_j | Y) \mid 1 \leq j < i \wedge (\mathcal{L}_j, \mathcal{L}_i) \notin \mathcal{E}\}$;
11         $\mathcal{E} = \mathcal{E} \cup (\mathcal{L}_j, \mathcal{L}_i)$;
12         $vk = vk - 1$;
13     end
14 end
15 return $\mathcal{G}$
```

Algorithm 3: learnParameters($\mathcal{T}$, $\mathcal{G}$)

```
   Input: Training set $\mathcal{T}$ and $\mathcal{G}$.
   Output: $\Theta$.
 1 Initialize $\Theta$ to structure $\mathcal{G}$.
 2 Compute the CPTs for $\Theta$ from $\mathcal{T}$.
 3 return $\Theta$
```

There have also been some important refinements that improve KDB's performance. Bouckaert (2006) proposes averaging all possible network structures for a fixed value of k (including lower orders). Time complexity is significantly reduced by multiplying the sum of every attribute given all possible parents in the order. For $k = 2$ the authors propose a simplification that identifies one attribute as super parent and consider for every attribute of the class, possibly the super parent and possibly one other lower ordered attribute. The sum of class probabilities is taken as selecting criteria for the superparent. However they agree that this approach is not easy to implement incrementally for large datasets. Rubio and Gámez (2011) present a variant of KDB that employs a hill-climbing search to incrementally build a KDB classifier. This approach requires at least one pass through the data for each attribute.

2.1 Selective KDB

The parameter k controls KDB's bias-variance trade-off. Higher k results in higher variance and lower bias. Unfortunately there is no apriori means to preselect a value of k for which this trade-off will result in the lowest error for a given training set as this is a complex interplay between the data quantity and the complexity and strength of the interactions between the attributes. A further factor that can increase KDB's error is that attributes that carry no useful information about the class must be treated as if they do, which invariably introduces some noise into the estimates. Discarding these attributes can both reduce error and classification time. Our proposed algorithm, Selective KDB (SKDB), extends KDB to select between attribute subsets and values of k in a single additional pass through the training data.

In the general case, attribute selection is a complex combinatorial task. For d attributes there are 2^d alternative attribute subsets that could be explored. Many attribute selection algorithms utilize either forward selection or backwards elimination hill-climbing strategies (Langley and Sage, 1994; Koller and Sahami, 1996). These start with either no attributes or all attributes and then iteratively add or remove one attribute at a time. This requires that all attribute subsets resulting from adding/removing any one attribute are considered at each step. There are at most d such steps resulting in $\mathcal{O}(d^2)$ attribute subset evaluations. If all candidate modifications to a current subset are considered simultaneously in a single pass through the data, this implies up to d passes for the attribute selection stage.

We seek to perform both attribute and best-k selection in a single pass. For attribute selection, we take advantage of the attribute ordering selected by KDB based on mutual information with the class. Given that the attributes are sorted on this order, we consider only the attribute sets $\{x_1, \dots x_i\}, 1 \leq i \leq d$, that is attribute sets that each contain the i attributes that have the greatest mutual information with the class. These d attribute subsets are evaluated simultaneously in a single pass through the data using leave-one-out cross validation (LOOCV). For big data, LOOCV can be expected to provide an unbiased low-variance estimate of the out-of-sample error. It is possible to efficiently evaluate all d subsets simultaneously, because a KDB classifier using subset $\{x_1, \dots x_i\}$ is a minor addition to a KDB classifier using subset $\{x_1, \dots x_{i-1}\}$. Calculating a KDB classifier for all attributes already incorporates all the calculations for all the KDB classifiers using subsets that we consider.

We use incremental cross validation (Kohavi, 1995), which performs LOOCV on BNCs very efficiently. A naive approach to LOOCV will for each holdout example recalculate from scratch all of the joint frequency counts required by a BNC. Incremental cross-validation first gathers the counts for all training examples in a single pass through the data. Then, when classifying a holdout example, its counts are removed from the table. SKDB collects the complete counts in its second pass through the training data and performs LOOCV in a third pass.

Using the given order, the number of attributes with the best leave-one-out error is selected. In case of a draw, preference is given to the smallest number of attributes. Any loss function can be used that is a function, $\{\langle y^{\mathbf{x}}, \hat{p}(Y, \mathbf{x})\rangle \mid \mathbf{x} \in \mathcal{T}\} \rightarrow \mathbb{R}$ over all objects $\mathbf{x} \in \mathcal{T}$ of a predicted class distribution $\hat{p}(Y, \mathbf{x})$ and the true class for that object, $y^{\mathbf{x}}$. Such loss functions include 0-1 loss, root mean square error, log-loss, Matthews correlation coefficient (Matthews, 1975) or Brier score (Brier, 1950) to name a few. Because we believe it to be an effective measure of the calibration of a classifier's class probability estimates, we use root mean squared error (RMSE) as follows:

$$RMSE = \sqrt{\frac{1}{t}\sum_{\mathbf{x}\in\mathcal{T}} (1 - \hat{p}(y^{\mathbf{x}}|\mathbf{x}))^2} \qquad (1)$$

where $\hat{p}(y^{\mathbf{x}}|\mathbf{x})$ is the estimated posterior probability of the true class given $\mathbf{x}$, $y^{\mathbf{x}}$ the class label for the example $\mathbf{x}$.

In order to make full use of available computational resources while reducing the risk of overfitting, we propose to select algorithmically the best value of k for a particular dataset. Just as the KDB classifiers built on successive attribute subsets are embedded one inside the other, a KDB classifier with $k = i$, can be evaluated with little additional computational effort during the process of evaluating a KDB classifier with $k = i + 1$. We leverage this capacity to simultaneously evaluate KDB classifiers with all attribute subsets $\{x_1, \ldots x_{i-1}\}$ and values of k up to the maximum capacity available, taking advantage of the above mentioned LOOCV. The combination of attribute subset and k with the best LOOCV error is selected. In case of a draw, the smaller attribute subset and smaller value of k are selected. The resulting SKDB algorithm is presented in Algorithm 4.

In the first pass our implementation of SKDB generates a three-dimensional table of co-occurrence counts for each pair of attribute values and each class value. This is required to calculate the mutual information of each attribute with the class and the conditional mutual information of every pair of attributes given the class. The resulting space complexity is $\mathcal{O}(c(dv)^2)$. The time complexity of forming the three dimensional probability table is $\mathcal{O}(td^2)$, as an entry must be updated for every training case and every combination of two attribute-values for that case. To calculate the conditional mutual informations, SKDB must consider every pairwise combination of their respective values in conjunction with each class value $\mathcal{O}(c(dv)^2)$. Attribute ordering and parent assignment are $\mathcal{O}(d \log d)$ and $\mathcal{O}(d^2 \log d)$ respectively.

The second pass requires d tables of $k+2$ dimensions (one for each of k parents, one for the child and one for the class), with $\mathcal{O}(cdv^{k+1})$. Updating the probability tables is $\mathcal{O}(tdk)$.

The algorithm to this stage is simply KDB, with time complexity $\mathcal{O}(td^2 + c(dv)^2 + tdk)$, where v is the maximum number of values per attribute, and space complexity $\mathcal{O}(cdv^{k+1})$.

Algorithm 4: The SKDB algorithm

```
Input: Training set $\mathcal{T}$ with attributes $\{X_1, \dots, X_a, Y\}$ and $k^{\max}$.
Output: SKDB model.
1  First pass begin
2      $\mathcal{G}$ =learnStructure($\mathcal{T}$) ;                    /* Algorithm 2 */
3  end
4  Second pass begin
5      $\Theta$ = learnParameters($\mathcal{T}$, $\mathcal{G}$) ;          /* Algorithm 3 */
6  end
7  Let $\mathcal{L}$ be a list of all $X_i$ in decreasing order of $MI(X_i;Y)$.
8  Let $\mathcal{P}$ be a $k \,\mathrm{x}\, a$ matrix of posterior probabilities;
9  Let $\mathcal{LF}$ be a matrix of LOOCV results (of length $k \,\mathrm{x}\, a$); Initialize it with
   zeros;
10 Let $\Theta^{\downarrow \mathrm{x}}$ be the BN $\Theta$ with example x *discounted* from its CPTs;
11 Third pass ($\forall \mathrm{x} \in \mathcal{T}$) begin
12     for $k' = 1$ to $k^{\max}$ do
13         $\mathcal{P}[k'][y^*] = \hat{p}_{\Theta^{\downarrow \mathrm{x}}}(y^*|\mathbf{x})$, $\forall y^* \in Y$;
14         for $l = 1 \to l = \mathcal{L}.size$ do
15             $X_{\max} = \mathcal{L}.nextElement$;
16             $\mathcal{P}[k'][y^*] = \mathcal{P}[k'][y^*] \cdot \hat{p}_{\Theta^{\downarrow \mathrm{x}}}(x_{\max}|pa^{k'}_{x_{\max}}, y^*)$, $\forall y^* \in \Omega_Y$; where $pa^{k'}_{x_{\max}}$ are
               the $k'$ first parent-values of $X_{\max}$ in x.
17             $\mathcal{LF}[k'][l] = \mathcal{LF}[k'][l] + LossFunction(\mathcal{P}[k'], y^{\mathrm{x}})$; where $\mathcal{P}[k']$ is the
               vector of posterior probabilities considering the top $l$ attributes
               by MI.
18         end
19     end
20 end
21 Select $b$ and $k$ indexes with best $\mathcal{LF}$;
22 Truncate $\Theta$ to attribute subset $\{1 \dots b\}$ and maximum number of parents $k$;
23 return $\Theta$;
```

SKDB then extends KDB with an extra pass through the training data to perform leave-one-out cross validation for the different (ordered) attributes $\mathcal{O}(tcdk)$.

At classification time, SKDB need only store the conditional probability distribution for the d_b selected attributes and the k_b selected, $\mathcal{O}(cd_b v_b^k)$. The time complexity of classifying a single example is $\mathcal{O}(cd_b k_b)$.

In summary, SKDB requires an extra pass over the data compared with KDB, and the complexity of this extra pass is linearly dependent on the number of training examples, classes, attributes and k so the order of the time complexity increases from $\mathcal{O}(td^2 + c(dv)^2 + tdk)$ to $\mathcal{O}(td^2 + c(dv)^2 + tcdk)$. This is not a great increase in practice, since the number of classes is generally small, and indeed for high dimensional data the complexity of the first pass can dominate that of the subsequent passes. Classification time and space complexity are reduced compared to KDB if the number of selected attributes $d_b < d$ and/or $k_b < k_{\max}$.

For the cost of this modest increase in computation, KDB is extended so as to discriminatively choose between a large class of alternative BNC models. Note that while it is possible to use cross validation to select between any collection of alternative models, it is only possible to do so in the extremely efficient manner we have developed for a very restricted class of learners, those that can be evaluated using incremental cross-validation and can be decomposed into nested models. The only types of learners of which we are aware that belong to this class are KDB and ANDE. We leave applying the approach to ANDE and variants of KDB as a topic for future research.

It may be possible to gain further efficiencies through sampling. As discussed in Hulten and Domingos (2002), it is sometimes possible to obtain sufficiently accurate estimates of the parameters for a learning program from a subsample of the full data. For very large datasets we could potentially utilize a random sample for the first pass of our algorithm, which only needs to estimate the parameters for three-way interactions between each pair of attributes and the class. We believe that the best value of k will usually be the one for which there is only just enough data in the dataset to obtain sufficiently accurate estimates of the required parameters. Thus we do not believe that sampling will usually be useful for the second pass. A technique such as Racing (Maron and Moore, 1994) could be used to select the model in the final pass. However, we suspect that it will usually require extremely large samples to confidently distinguish between the large number of very similar models that are assessed in the third pass. Given that the computational overheads of scanning a large dataset in random order are considerable, we do not believe that sampling will often be useful. However, this also remains a potential direction for future research.

3. Experimental Methodology

We undertook an extensive online search to gather a group of large datasets, all of which have more than 100K instances. These datasets are described in Table 7 in Appendix A, in ascending order of number of samples. Those below the double line have more than 1 million instances. All datasets except for `poker-hand`, `uscensus1990` and `splice` contain one or more numeric attributes. 7 datasets contain only numeric attributes: `MITFaceSetA`, `MITFaceSetB`, `MITFaceSetC`, `Mnist`, `USPSExtended`, `MSDYearPrediction` and `satellite`. For the Bayesian network classifiers we discretize numeric attributes using 5-bin equal frequency discretization. To avoid loading the whole data into memory, only a random sample of 100K points is used to define the bins for discretization.

As described in Section 2.1, we use SKBD with RMSE as the objective function for the third pass that selects between structures because we believe it to be a good measure of the calibration of a classifier's class probability predictions. However, this raises the issue that we might be reporting the measure that best favors our approach. To guard against this, we also present in the Appendices 0-1 loss results of the relative performance of all algorithms, and discuss these in the following sections where they are revealing. Table 9 shows that when evaluated on RMSE, SKDB trained to optimize RMSE and SKDB trained to optimize 0-1 loss (SKDB$^{0\text{-}1}$) attain identical 0-1 loss on 12 out of 16 datasets, suggesting that SKDB's performance is similar when selection uses different but closely related loss functions.

Table 1: Orders of complexity for the different semi-naive BNCs where: t is the number of training examples, d is the number of attributes, v is the maximum number of values per attribute, c the number of classes, d_b (the best d) is the remaining attributes after applying SKDB, and k_b (the best k) is the k value selected in SKDB.

	Train		Test	
	time	space	time	space
NB	$\mathcal{O}(td)$	$\mathcal{O}(cdv)$	$\mathcal{O}(cd)$	$\mathcal{O}(cdv)$
TAN	$\mathcal{O}(td^2 + c(dv)^2 + d^2 \log a)$	$\mathcal{O}(c(dv)^2)$	$\mathcal{O}(cd)$	$\mathcal{O}(cdv^2)$
AODE	$\mathcal{O}(td^2)$	$\mathcal{O}(c(dv)^2)$	$\mathcal{O}(cd^2)$	$\mathcal{O}(c(dv)^2)$
KDB	$\mathcal{O}(td^2 + c(dv)^2 + tdk)$	$\mathcal{O}(c(dv)^2 + cdv^{k+1})$	$\mathcal{O}(cdk)$	$\mathcal{O}(cdv^{k+1})$
SKDB$_k$	$\mathcal{O}(td^2 + c(dv)^2 + tcdk)$	$\mathcal{O}(c(dv)^2 + cdv^{k+1})$	$\mathcal{O}(cd_bk)$	$\mathcal{O}(cd_bv^{k+1})$
SKDB	$\mathcal{O}(td^2 + c(dv)^2 + tcdk)$	$\mathcal{O}(c(dv)^2 + cdv^{k+1})$	$\mathcal{O}(cd_bk_b)$	$\mathcal{O}(cd_bv^{k_b+1})$

The structure of the experimental Section is as follows: Section 3.1 compares our proposed SKDB algorithm's results with different out-of-core semi-naive BNCs. Section 3.2 compares our SKDB algorithm with several *online* (out-of-core) linear classifiers. Section 3.3 includes comparisons with two state-of-the-art in-core machine learning algorithms, BayesNet (Augmented Bayesian network) and Random Forest. Section 3.4 presents a global comparison of all learners considered, along with empirical time comparisons for the different out-of-core classifiers. Each of these sections provides a summary of the relevant results. Detailed RMSE error outcomes are presented in Table 8 in Appendix A and a more detailed analysis of results is provided in in Appendix C.

All the experiments for the Bayesian out-of-core algorithms use C++ software specially designed to deal with out-of-core classification methods[1]. Appendix B specifies the implementation details.

Note that both RMSE and 0-1 Loss are assessed using 10-fold cross-validation. This should not be confused with the leave-one-out cross-validation used to select the number of attributes and parents in SKDB. For each fold of the 10-fold cross-validation, SKDB performs leave-one-out cross-validation on the training set to select the parameters of the model that is then tested on the holdout test fold.

3.1 SKDB vs Bayesian out-of-core algorithms

The first set of experiments compare SKDB with the out-of-core semi-naive BNCs described in Section 2: NB, TAN, AODE and KDB. We also consider a version of SKDB which fixes the value of k and just selects among the attributes, referred to as SKDB$_k$. Table 1 shows a summary of the orders of complexity for all these BNCs.

Tables 2 and 3 show win-draw-loss records summarizing the relative RMSE and 0-1 loss of the different approaches. Cell $[i, j]$ of each table contains the number of wins/draw/losses

1. A minimum functional part of the software containing SKDB can be found in the (two first) authors' academic web pages, e.g. `http://www.csse.monash.edu.au/~webb/`.

Table 2: Win-draw-loss records when comparing the RMSE of different out-of-core BNCs.

	KDB				
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
TAN	6-3-7	0-0-16	-	-	-
AODE	3-0-13	0-0-16	-	-	-
SKDB$_k$	8-8-0	6-10-0	6-10-0	7-9-0	6-9-1
SKDB	16-0-0	16-0-0	14-2-0	14-2-0	8-8-0
SKDB $k_{\max} = 5$	6-10-0 (vs best k)*				

* The result is 6-9-1 (shift in uscensus) if SKDB is optimized with 0-1 loss.

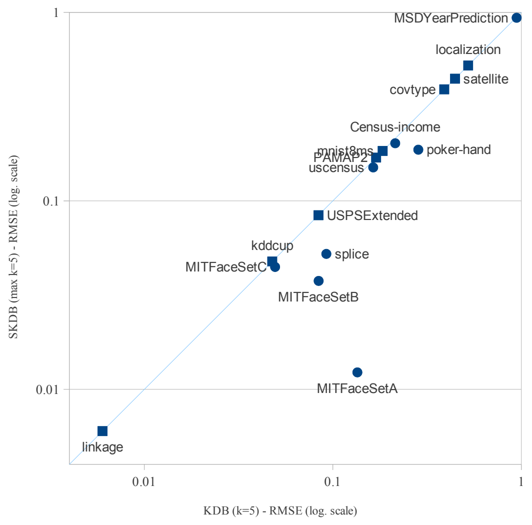
Table 3: Win-draw-loss records when comparing the 0-1 loss of different out-of-core BNCs.

	KDB				
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
TAN	7-2-7	0-0-16	-	-	-
AODE	4-1-11	0-0-16	-	-	-
SKDB$_k$	5-10-1	5-11-0	6-10-0	6-9-1	6-9-1
SKDB	16-0-0	14-2-0	13-3-0	10-5-1	6-9-1
SKDB $k_{\max} = 5$	5-11-0 (vs best k)*				

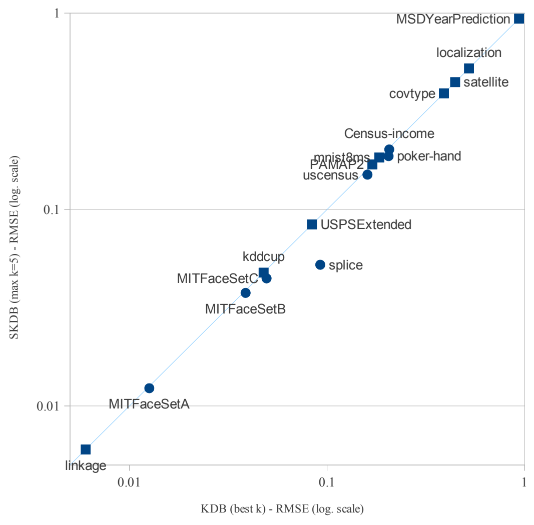
* The result is 6-10-0 if SKDB is optimized with 0-1 loss.

for the classifier on row i against the classifier on column j. A win indicates that the algorithm has significantly lower error than the comparator. A draw indicates that the differences in error are not significant. Statistical significance has been assessed using Wilcoxon signed-rank tests to compare the 10-fold outputs. SKDB$_k$ and SKDB are compared against standard KDB with the same value of k. Additionally, SKDB with $k = 5$ is compared against the lowest error of any standard KDB with k between 1 and 5 (which is a theoretical result, since the value of k that will minimize out of sample error can only be determined a posteriori)..

The results indicate that TAN is competitive with KDB $k = 1$, whereas AODE only achieves lower error on three or four of the datasets, depending on the loss function. We believe the reason AODE performs so poorly is that the advantage it gets in reduced variance from ensembling the d models is less useful for larger data. Both TAN and AODE consistently show higher error than higher orders of KDB on these large datasets. SKDB$_k$ only increases the RMSE of KDB relative to one value of k and only on one dataset, MITFaceSetB, and then only from 0.0841 to 0.0844. For many datasets it does not affect error (but nonetheless speeds up classification time). For many datasets it substantially reduces error, such as the reduction from 0.2048 to 0.1868 for poker-hand.



(a) KDB with $k = 5$ and SKDB ($k_{\max} = 5$).



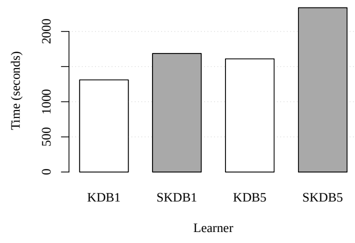
(b) KDB with the best k for each dataset and SKDB ($k_{\max} = 5$).

Figure 2: Scatter plot of RMSE comparisons for KDB and SKDB.

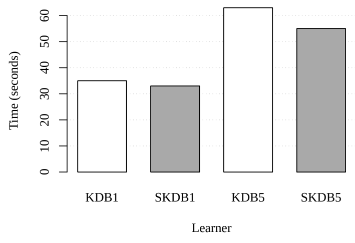
The average best k for SKDB across the 16 datasets is 4.08. On average, $78.46 \pm 29\%$ of the attributes are selected, which explains some of the ties, although in other cases SKDB obtains the same results as KDB with the best value of k, but reducing the number of attributes. Indeed, reducing the number of attributes without affecting error is the worst case encountered by SKDB, because it never loses in terms of RMSE compared to the best KDB).

One could think that KDB with $k = 5$ would be the fairest comparison with SKDB ($k_{\max} = 5$). This comparison is shown in Figure 2a, where the X-axis represents the RMSE results with KDB with $k = 5$ for the 16 datasets and the Y-axis the RMSE with SKDB. We have used a logarithmic scale because almost half of the points fall below the 0.1 range. Note that there is not a single point above the diagonal line, which indicates that SKDB is never worse than KDB $k = 5$ or even KDB with the best k value considered, as shown in Figure 2b.

If only the best number of parents is selected by SKDB (best k) and no feature selection is performed, then the RMSE is always equivalent to the best RMSE obtained by KDB with any value of $k \in [1, 5]$. That is, for all datasets SKDB without attribute selection is successful at selecting the best value of k with respect to the specified loss function. However, for five datasets the value of k that attains the best RMSE is different to that which attains the best 0-1 loss, demonstrating the value of optimizing with respect to the relevant loss function. These results are summarized on row SKDB$_{only_k}$ in Tables 8 and 9. The non-shaded cells on these rows correspond to those datasets for which feature selection contributes to further reduce the error of SKDB. There are 9 datasets in total for which the feature selection part of the algorithm does not reduce RMSE and 10 for which it does



(a) Training Times



(b) Classification Times

Figure 3: Training and classification time comparisons for KDB and SKDB$_k$.

not reduce 0-1 loss. Note, however, that the extra computational complexity added by the feature selection evaluation is only related to the complexity of the loss function utilized. In the case of considering RMSE-like functions, this extra time is negligible, but if more complex evaluation functions were to be used in a highly time-constrained scenario, then only a random sample of instances could be considered at this step.

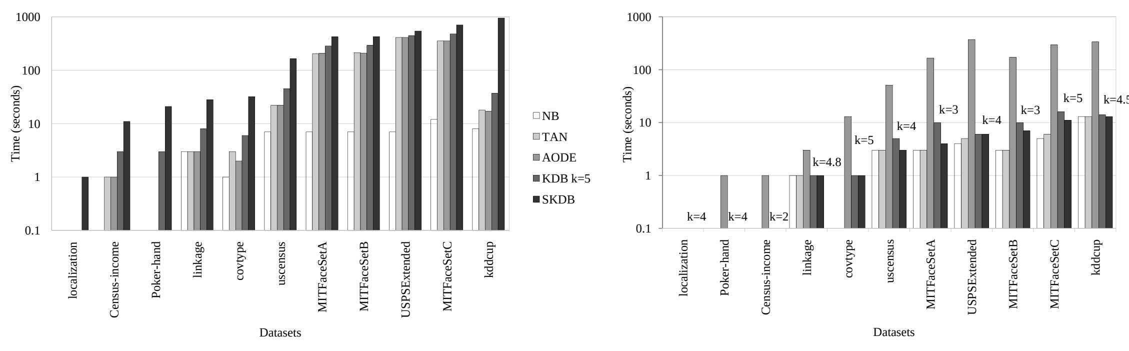
Figures 3a and 3b show the training and classification time comparisons for KDB and SKDB$_k$. Only the 11 smallest out of the 16 datasets have been considered for time measurement, since these experiments have been conducted on a desktop computer with an Intel(R) Core(TM) i5-2400 CPU @ 3.10 GHz, 3101 MHz, 64 bits and 7866 MiB of memory, whereas the remaining experiments have been conducted in a heterogeneous grid environment for which CPU times are not commensurable. Each bar represents the sum of all 11 datasets[2] in a 10-fold cross validation experiment. These graphs reinforce what the orders of complexity for the two algorithms indicated, that is, SKDB$_k$ requires a bit more time for training (extra pass on the data), while the classification time is less in practice, since the number of attributes selected is generally smaller than d.

Figures 4a and 4b show the training and classification empirical time comparisons of the different out-of-core BNCs relative to SKDB (with $k_{\text{max}=5}$). The datasets are ordered at increasing time for SKDB. Note that SKDB always takes a bit more time for training, taking the largest time for `kddcup`, due to its large number of classes. Nevertheless, for the datasets with many attributes: `MITFaceSetA`, `MITFaceSetB`, `USPSExtended` and `MITFaceSetC`; the time differences with other BNCs become smaller. At classification time some gain can be appreciated for SKDB compared to KDB $k = 5$ if the selected k is less than 5. AODE's classification time is quadratic in the number of attributes, and hence much higher for all datasets.

3.2 SKDB vs non-Bayesian out-of-core algorithms

In this section we compare SKDB with the state-of-the-art in online learning: Logistic Regression with Stochastic Gradient Descent (LRSGD). We use LRSGD's implementation in

2. census-income, covtype, donation, kddcup, localization, MITFaceSetA, MITFaceSetB, MITFaceSetC, USPSExtended, poker-hand, uscensus1990.



(a) Training Times (b) Classification Times

Figure 4: Time comparisons per dataset for NB, TAN, AODE, KDB against SKDB.

Vowpal Wabbit (VW) (Agarwal et al., 2011), an open source out-of-core linear learning system. VW provides a scalable implementation of LRSGD, which can be used in combination with several loss functions, different parameters for the learning rate and regularization, and quadratic or cubic combination of features.

We have performed experiments with varying numbers of passes for LRSGD, varying loss functions and combinations of features. We use the default advanced (step size) updates provided by VW, which are:

1. Importance weight invariance (Karampatziakis and Langford, 2011): this update is meant to be useful specially for cost-sensitive learning, that is, when the misclassification penalty varies for the different class labels. However, it is also claimed to reduce classification error even when no matrix of weights is provided.

$$w_i \leftarrow w_i - s(\eta I)\frac{\partial L(\hat{y}_w(x), y)}{\partial w_i}, \tag{2}$$

   where y is the true class; $\hat{y}_w(x)$ is the predicted value for example $x \in \mathbb{R}^d$ with parameters $w \in \mathbb{R}^d$; $L(\hat{y}_w(x), y)$ is the loss function; η is the learning rate; $I \in \mathbb{N}$ the number of times an example is more important than a typical example, in our case $I = 1$; and $s(\eta I)$ is a scaling factor, for which a closed form solution can be found for each loss function (Karampatziakis and Langford, 2011) .

2. Adaptive updates (Duchi et al., 2011; McMahan and Streeter, 2010): the learning rate must decay to converge. Instead of using $\eta_t = \frac{1}{\sqrt{t}}$ or $\eta_t = \frac{1}{t}$ at iteration t for all features, we use a per-feature learning rate decay:

$$w_i \leftarrow w_i - \eta\frac{g_{it}}{\sqrt{\sum_{t'=1}^{t} g_{it'}^2}}, g_{it} = \frac{\partial L(\hat{y}_w(x_t), y_t)}{\partial w_i}, \tag{3}$$

3. Normalized updates (Ross et al., 2013): instead of using Gaussian sphering standard solution, which would imply in-core processing and an increase in the size of the

Table 4: Win-draw-loss records for VW and SKDB in terms of RMSE and 0-1 Loss

		VWLF	VWSF
SKDB	RMSE	$7^{(+2)}$-0-7	14-0-2
	0-1 Loss	$8^{(+1)}$-1-$5^{(+1)}$	8-1-7

dataset, a scale-free update is used that renormalizes w_i at each timestep and keeps track of a global scale.

A fair comparison of LRSGD with SKDB should perform the same number of passes on the data. Since LRSGD requires data to be pre-shuffled, an n-pass LRSGD learner actually results in $n+1$ passes through the data. This is also true for SKDB on numeric data, as an extra pass is required for discretization. We have run experiments with 3, 10 and 20 passes for LRSGD. Each 10-fold cross validation experiment is repeated 10 times. Both the quadratic and logistic loss functions are considered. The one-against-all reduction from multiclass classification to binary classification has been used for the datasets with multiple class labels, which creates one binary problem for each of the classes. Table 10 in Appendix A shows the results for all of the different combinations for which we have conducted comprehensive experiments. The overall best results for LRSGD are obtained with 3 passes and using quadratic features. When the squared loss function is optimized we refer to it as VWSF, and VWLF if the logistic function is used. In all cases, SKDB is highly competitive compared to VW.

Note that while LRSGD has a number of parameters that could potentially be tuned using cross validation, doing so does not make a reasonable comparator to SKBD as such tuning would require multiple passes through the data for each cross-validation fold.

A win-draw-loss record summary is displayed on Table 4. The $^{(+1)}$ and $^{(+2)}$ for SKDB vs VWLF aim at distinguishing the results for `satellite` and `splice`, for which more than 400 hours are required to get the 10cv results using VWLF, and hence VWSF is considered instead. Irrespective of the fact that the SKDB model is optimized with respect to RMSE, the win-draw-loss records only vary for three datasets between evaluation with respect to RMSE or 0-1 loss. The win-draw-loss records remain exactly the same whichever loss function is optimized in VW (see Table 10 for more details).

Since the output of the VW algorithms is not probabilistic, we are using the class prediction, and hence computing 0-1 loss for comparisons with SKDB as well. By using, for example, the following sigmoid function $\frac{1}{1+e^{-\hat{y}w(x)}}$, probability estimates can be estimated from VW. Using this approach, the RMSE results for VWSF are only better than SKDB for two of the datasets (MSDYearPrediction and USPSExtended), both of them originally sparse and containing exclusively numeric attributes. These records improve when using a logistic function for optimization, that is VWLF (see Table 10 in Appendix A for more details). Nevertheless, computation time for VWLF is much larger than for VWSF as shown below.

Figures 5a and 5b show the comparisons per dataset with SKDB in terms of RMSE, whereas Figures 5c and 5d show comparisons per dataset for 0-1 Loss. Figures on the left (5a and 5c) show VW optimizing a logistic function, while the two figures on the right (5b and 5d) show VW optimizing RMSE. A diamond symbol is used for those points above the diagonal line, indicating better results for SKDB, as opposed to the bullets. A squared symbol is used to show the ties. We can see that VW achieves lower error using a logistic function (VWLF), with the same number of datasets above and below the diagonal line, as opposed to only 2 wins for VW with the squared function (VWSF). Figures 5c and 5d shows the comparisons of VW with SKDB in terms of 0-1 Loss. In this case, VWSF performs slightly better than VWLF compared to SKDB.

Figure 5 shows the relative training and test times for VW and SKDB across all datasets. SKDB has substantially lower training and test time on most datasets. One exception is the sparse numeric USPSExtended dataset. Another is covtype, which has a large number of sparse boolean attributes.

While neither SKDB nor VW dominates the other in terms of error, SKDB is substantially more efficient, as shown in Figure 6a, where again training and classification times per dataset for VWLF, VWSF and SKDB are displayed. There is only one dataset for which SKDB takes more than the alternatives, which is `USPSExtended`. We believe that for this dataset VW is able to exploit the sparseness of the data while SKDB cannot. Some of the reasons why VW is taking in general more time than SKDB both for training and classifying is the consideration of quadratic features, the need to binarize discrete attributes (creating many binary attributes) and the need also to use one-against-all for multiclass classification.

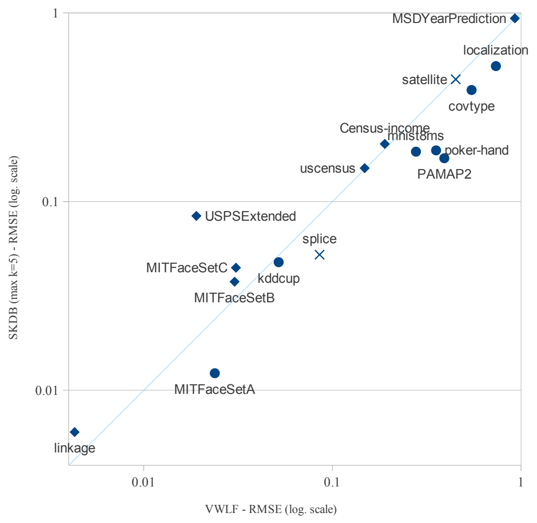
3.3 SKDB vs in-core algorithms

In order to understand how much predictive capacity, if any, is sacrificed by using our out-of-core classifiers, it is useful to compare them to the state-of-the-art in in-core classification. To this end, we have replicated our set of experiments with two in-core algorithms: a more general BNC and Random Forest, a powerful exemplar of the state-of-the-art. For some datasets it was infeasible to get results, even with high-performance computers, due either to time or memory limitations (indicated in Table 8 in Appendix A with $>$ 600h, when more than 600 CPU hours are required; or by >138G, when more than 138GB of RAM memory are required).

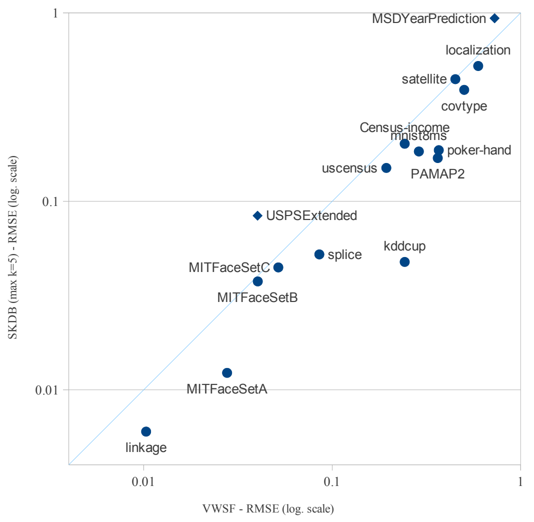
Note that while each of these algorithms has a number of parameters that could potentially be tuned using cross validation, doing so would not make reasonable comparators to SKBD. First, SKDB uses cross-validation to choose between multiple models rather than to choose between parameterizations. Cross validation could also be used to tune SKBD parameters such as the smoothing technique. Second, SKDB uses cross-validation in a constrained manner that requires only a single additional pass through the data while parameter tuning of these algorithms would increase compute time proportionally to the number of folds employed.

3.3.1 Augmented Bayesian Network

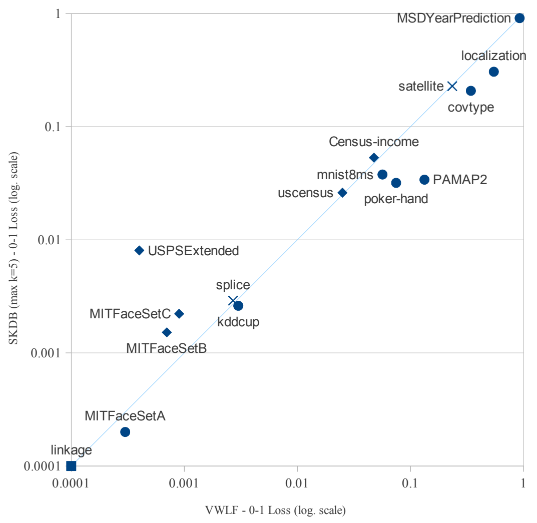
We compare performance against a state-of-the-art generic in-core Bayesian network classifier, that uses a hill-climbing algorithm for adding, deleting and reversing arcs (starting with a NB structure, this hill climber also considers arrows as part of the NB structure for deletion, that is, arcs from the class to the attributes can be removed). It optimizes the



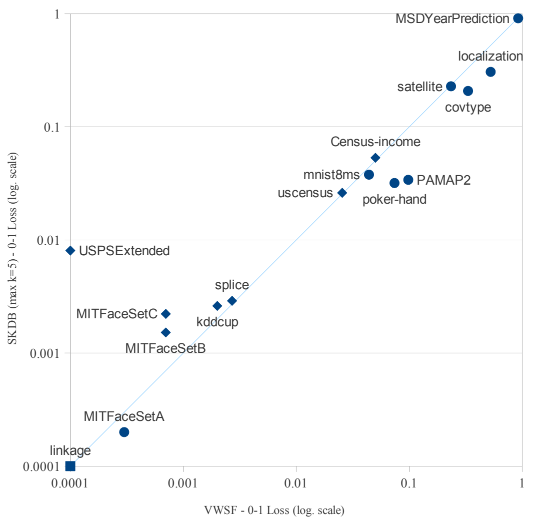
(a) RMSE for VWLF and SKDB.



(b) RMSE for VWSF and SKDB.

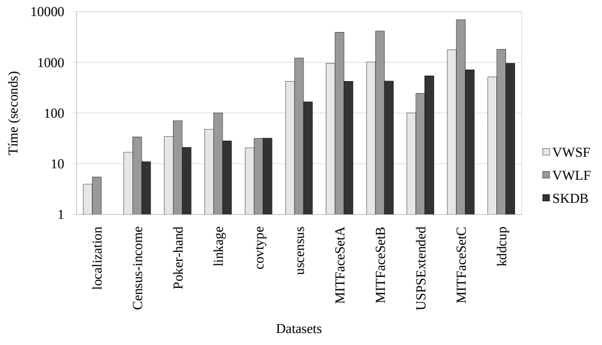


(c) 0-1 loss for VWLF and SKDB.

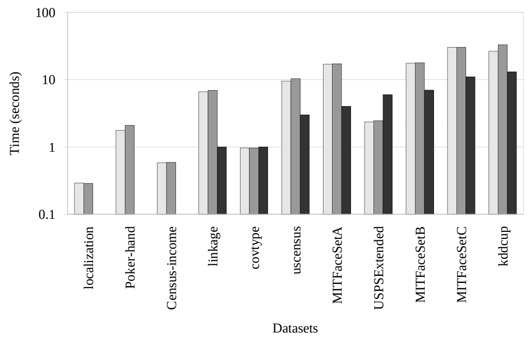


(d) 0-1 loss for VWSF and SKDB.

Figure 5: Scatter plot of RMSE and 0-1 loss comparisons for VW and SKDB ($k_{\max} = 5$).



(a) Training Times



(b) Classification Times

Figure 6: Time comparisons per dataset for VW with squared and logistic functions against SKDB.

Table 5: Win-draw-loss records for BayesNet and SKDB in terms of RMSE

	BayesNet
SKDB	$6^{(+4)}$-3-3

K2 metric (Cooper and Herskovits, 1992). The maximum number of parents allowed for a node in the Bayes net is set to 5. The win-draw-loss comparison is presented in Table 5.

The detailed results can be found in Table 8 (Appendix A), row BayesNet. The darker grey background corresponds to those cases where BayesNet significantly outperforms SKDB. Note that there are only records for 12 out of the 16 datasets, because there are four cases for which results could not be computed for BayesNet due to memory constraints, since the network cannot be learned incrementally.

Even though BayesNet performs a more exhaustive search than SKDB of the model space of Bayesian networks with up to 5 parents, SKDB obtains lower error than BayesNet more often than the reverse. We hypothesize that this is due to SKDB's use of both generative and discriminative measures for model selection in contrast to BayesNet's use exclusively of generative measures.

3.3.2 Random Forest

Random Forest (RF) is a state-of-the-art learning algorithm introduced in Breiman (2001). It uses bagging (Breiman, 1996) to aggregate an ensemble of trees that are each grown using a process that involves a stochastic element to increase diversity.

We have conducted experiments with RF selecting 100 trees. We present results both with pre-discretized data, to show relative performance on categorical data, and with undiscretized data. The detailed results can be found on the fifth and sixth from last rows in Table 8 in Appendix A. Table 6 shows the win-draw-loss records when compared with SKDB. The superscripts compute those datasets for which results cannot be obtained for

Table 6: Win-draw-loss records for RF and SKDB in terms of RMSE and 0-1 Loss

		RF (5EF)	RF (Num)
RMSE	SKDB	$5^{(+4)}$-1-6	$5^{(+5)}$-0-6
0-1 Loss	SKDB	$2^{(+4)}$-3-7	$2^{(+5)}$-2-7
	SKDB^{0-1}	$3^{(+4)}$-2-7	$3^{(+5)}$-1-7
	SKDB^{MDL}	$8^{(+4)}$-2-2	$3^{(+5)}$-2-6

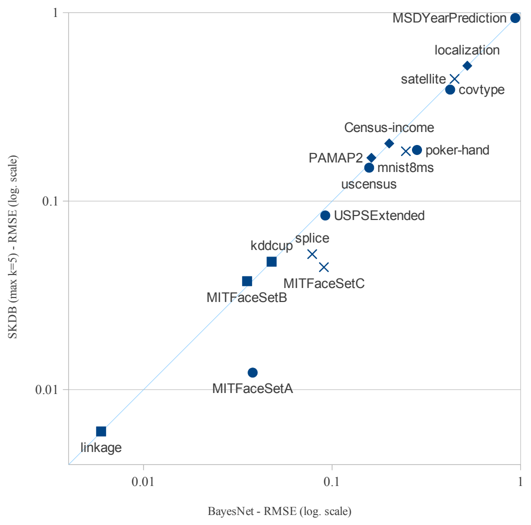
RF due to main memory constrains ($>$ 138G are required). The results show that SKDB is competitive with RF in terms of RMSE, even when RF is performing its own discretization (5 wins for SKDB and 6 wins for RF), for which the computational cost in terms of memory and time is much higher than SKDB's. When 0-1 Loss is considered, out-of-core RF wins more frequently than SKDB, even when 0-1 Loss is used as the objective function during structure selection. When MDL discretization is used instead of 5EF for SKDB, then the results improve slightly for SKDB compared to RF performing its own discretization. When interpreting these results it is important to keep in mind that SKDB is a 3-pass out-of-core algorithm (4-pass with discretization) in contrast to the inherently computationally intensive in-core processing of RF.

Note that the sum of some cells is not exactly 16 but smaller, due to computational constraints for RF. RF is an in-core algorithm, and hence we have not been able to learn classification models for the largest datasets. Interestingly enough, RF is more negatively affected (in terms of computation) by a large number of classes than the BNCs above studied, e.g. no 100 tree model can be learnt for `MSDYearPrediction`, which has 90 class labels. Precisely, there are three cases where RF with 10 trees cannot be run, and four in the case of RF with 100 trees. These four datasets are `MSDYearPrediction`, `satellite`, `mnist8ms` and `splice`, so the largest possible samples able to be computed have been considered.

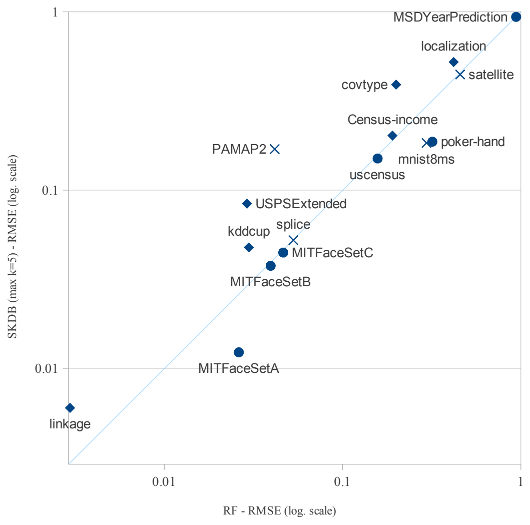
RMSE comparisons for all datasets for BayesNet and RF when compared with SKDB are shown in Figures 7a and 7b, where the points with an X symbol indicate those datasets for which sampling is required. While BayesNet is not competitive with SKDB, RF provides competitive results for some datasets. `PAMAP2` provides a notable case for which only a sample of 2 millions out of 3.5 suffices for RF to provide better classification performance than SKDB using all the training data. This is however the only dataset for which RF learning from a sample can achieve lower RMSE than SKDB learning from all the data, the reverse being true for `splice`, `mnist` and `satellite`.

Figures 8a and 8b display the training and classification times per dataset for these classifiers compared to SKDB. For most datasets the in-core learners require substantially more time for learning and classifying[3].

3. RF requires more than 8GB of main memory to complete for poker-hand and uscensus1990, since these experiments to measure time has been conducted in a desktop computer no time measurements can be provided to compare with the rest of experiments. Weka's implementation has been considered.

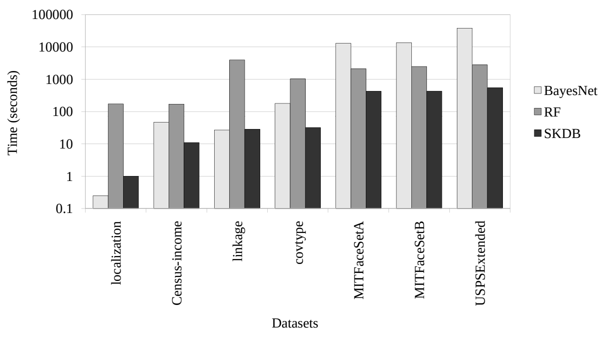


(a) BayesNet and SKDB.

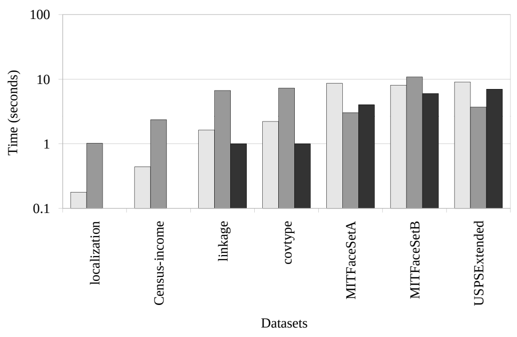


(b) RF and SKDB.

Figure 7: Scatter plot of RMSE comparisons for BayesNet, RF and SKDB.



(a) Training Times

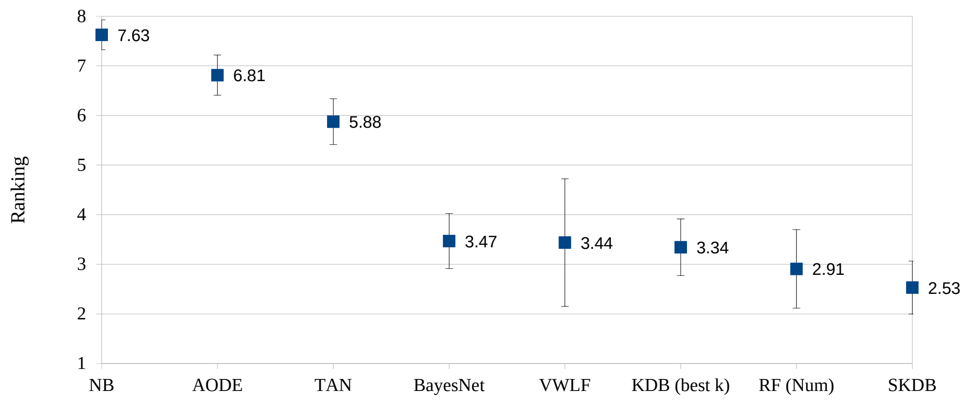


(b) Classification Times

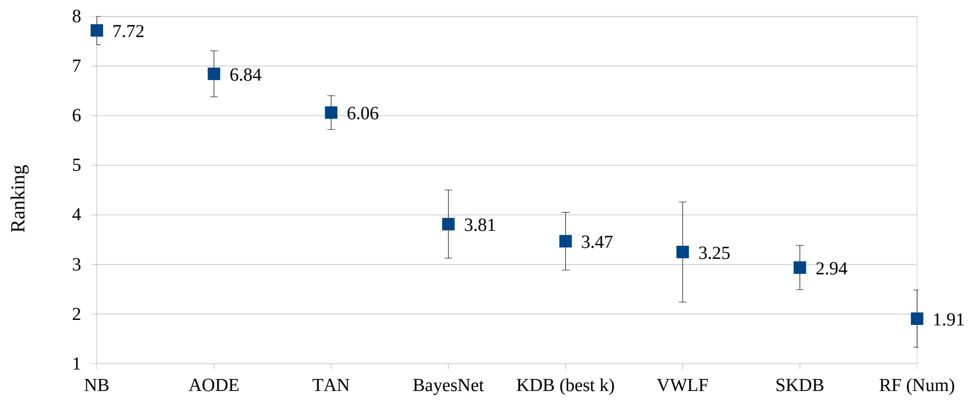
Figure 8: Time comparisons per dataset for BayesNet and RF against SKDB.

3.4 Global comparison of all classifiers

In this section we analyze how all the classifiers explored in this paper perform when they are compared as a set. Figure 9 plots the average rankings across all datasets, along with the standard deviation for each learner. This ranking corresponds to the ranking obtained when a Friedman test is conducted. Note that in this analysis we present the 0-1 Loss performance of SKDB when it is optimized for RMSE, its average rank on 0-1 Loss improves marginally from 2.94 to 2.84 when it is instead optimized for 0-1 Loss.



(a) RMSE



(b) 0-1 Loss

Figure 9: Ranking in terms of RMSE and 0-1 Loss for NB, TAN, AODE, KDB (with best k), SKDB, RF using numeric attributes, BayesNet and VW with a logistic function.

When assessing the calibration of the probability estimates using RMSE, SKDB obtains the lowest average rank of 2.53, followed by RF with 2.91 and KDB with 3.34 (very close to those for VW and BayesNet). When assessing performance using 0-1 Loss, RF using its own internal discretization of numeric attributes enjoys an average advantage of 1 in rank over SKDB, and SKDB enjoys an average advantage of almost one third over VWLF.

We find NB at the other extreme on both measures, with average ranks 7.63 and 7.72 out of a total of 8 classifiers.

SKDB obtains in the worst case the 4.5th position (this half is due to a tie with KDB-best-k) for the *linkage* dataset and the 4th position for *census-income* when ranked on RMSE and the 4th position for *MITFaceSetB* and *kddcup* when ranked on 0-1 Loss.

On both measures the largest standard deviation is observed for VW, which usually ranks either very well or rather poorly among the datasets. Still, it is VW which obtains the largest number of top 1 datasets for RMSE, with a total number of 7, against 5 for RF and 4 for SKDB. On error RF ranks first 8 times, VW 4 times and SKDB twice (one of those times being a tie with KDB-best-k) for lowest 0-1 Loss.

A Friedman test (Demšar, 2006) was performed for the set of 8 classifiers, yielding statistical difference for both measures. To identify where exactly these differences are found, we ran a set of posterior Nemenyi tests. These tests confirm the existence of two sets of algorithms in terms of performance on both RMSE and 0-1 Loss for these large datasets: on one hand NB, AODE and TAN; and on the other hand BayesNet, VW, KDB, RF and SKDB. The Nemenyi tests[4] showed significant differences for the second sets of algorithms over NB and AODE, and only RF and SKDB over TAN.

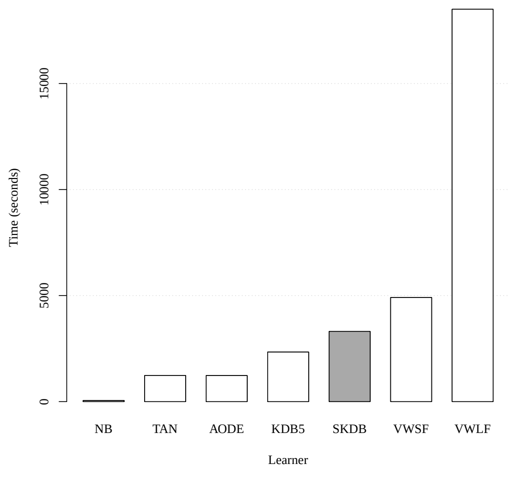
Even though the small number of datasets only allow tentative conclusions to be drawn, the following dataset characteristics appear to indicate a preference for one or another learning scheme.

- RF performs better on datasets with a small number of attributes.
- RF derives an advantage with respect to 0-1 Loss due to its internal discretization of numeric values. It does not achieve the lowest RMSE or 0-1 Loss on any of the three categorical only datasets, *poker-hand*, *PAMAP2* or *splice*.
- SKDB seems to cope well with high-dimensional datasets and datasets with more than 1 million points. However its complexity is quadratic with respect to the number of attributes so, as most existing BNCs, it is not feasible for *very* high-dimensionality.
- VW appears to have an advantage for sparse numeric datasets.

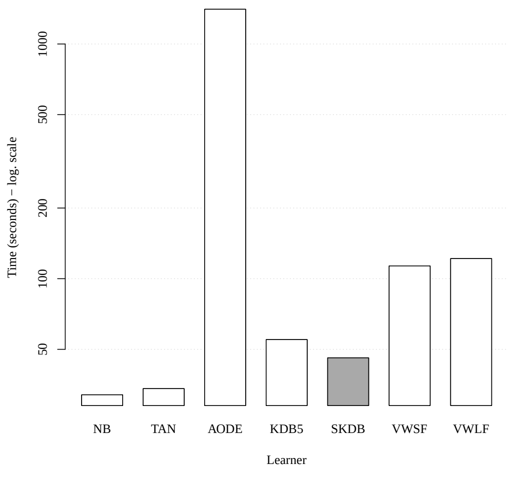
A more detailed study of the domain of competence of these classifiers remains an important area for future work.

We can also analyze the performance of these classifiers with respect to the complexity of the decision function they provide. NB and linear regression, represented in this work by VW with linear features, are classifiers that learn linear decision boundaries. They have low variance and are particularly useful for small data quantity and sparse data. An extra level of expressiveness is offered by AODE, KDB ($k = 1$), TAN and VWSF/VWLF (with quadratic features); which produce quadratic decision functions. A2DE, KDB ($k = 2$) and VWSF/VWLF (with cubic features) produce cubic decision functions. More generally, AnDE and KDB can separate order-k(n) polynomially separable concepts (Jaeger,

4. Similarly for Holms and Shaffer post-hoc tests.



(a) Training Times



(b) Classification Times

Figure 10: Training and classification time comparisons for the out-of-core classifiers.

2003). These two families of classifiers provide a set of a spectrum of models of increasing complexity, allowing a fine-tuned trade-off between expressivity on the one hand, and efficiency of learning and guarding against overfitting on the other hand, i.e., the well-known bias-variance trade off (Brain and Webb, 1999).

Figures 10a and 10b show the time comparison for the out-of-core classifiers considered: namely NB, TAN, AODE, KDB with $k = 5$, SKDB and LRSGD with quadratic features in VW (using either a squared and a logistic function). The time required to pre-randomize the data for VW has not been included. No parallelization techniques have been used in any case, although most algorithms could be parallelized. The average of the 10-fold CV is shown. It is important to note that logarithmic scale has been used for classification times in order to appreciate the smaller differences in the faster learners. Again, only the 11 smallest out of the 16 datasets have been considered for time measurement, since these experiments have been conducted in a personal computer for a controlled environment.

Note that training SKDB with $k = 5$ (maximum) takes only a bit more time than training KDB with $k = 5$. The classification time for KDB is slightly higher than SKDB's (since all attributes are considered). Training time for LRSGD with quadratic features and 3 passes (VWSF and VWLF), which are those that obtain comparable results with SKDB, are also higher, presumably because they have to compute all pairs of quadratic features as indicated above. Similar reasoning holds for classification time. We believe that the extra time for VW is also due to the need to binarize discrete features, that is, since VW cannot deal with discrete attributes directly, v binary attributes are created for each originally discrete attribute with v values; and also because of the one-against-all approach required to handle classes with multiple labels. This overhead may be noticeable since we are using quadratic features. The time required for VWLF is much larger than VWSF, and in any case much larger both for training and testing than that for SKDB (specially note the logarithmic scale in Figure 10b).

4. Summary

We have developed an extension to KDB which performs discriminative attribute and best-k selection using leave-one-out cross validation.

SKDB only requires a single extra pass over the training data compared to KDB. In return for this extra pass during training, it often increases prediction accuracy relative to KDB with the best possible k, and always improves classification time (and space) by reducing the number of attributes considered, which can be quite significant for some datasets (e.g. for `MITFaceSetC` 60%, 215.8 out of 361, of the attributes are selected improving the classification performance at the same time). It is embarrassingly parallelizable. Furthermore, it can optimize any loss function that is a function over each training example $\mathbf{x}$ of $\hat{p}(Y, \mathbf{x})$ and $y^{\mathbf{x}}$, making it extremely flexible.

Due to its low bias, when datasets are large enough to avoid overfitting, SKDB attains lower error than the other in and out-of-core Bayesian network algorithms considered. It even provides competitive error compared to Random Forest, one of the most powerful in-core classification methods; and online linear and non-linear classifiers with stochastic gradient descent (i.e., linear regression with quadratic and cubic features). Compared to the latter, it is in most cases substantially more efficient both at training and classification time. Further, it provides well-calibrated posterior class probability estimates.

Future directions for research include:

1. using only a sample of the data in the leave-one-out cross validation, in order to use more complex loss functions efficiently;
2. tackling SKDB's tendency to overfit small training sets;
3. studying a customized selection of the different links and attributes in a discriminative manner, that is, different numbers of parents for different attributes and also discarded in an order that does not follow the conditional mutual information rank;
4. consideration of other means of generating alternative classifiers to be selected in the final LOOCV pass; and
5. study of more appropriate loss functions for imbalanced datasets.

SKDB provides an efficient and effective solution to the problem of scalable low-bias learning. Its low bias allows it to capture and take advantage of the additional fine-detail that is inherent in very large data while its efficiency makes it feasible to deploy. SKDB thus sets new standards for classification learning from big data.

Acknowledgments

This research has been supported by the Australian Research Council grant DP140100087 and Asian Office of Aerospace Research and Development, Air Force Office of Scientific

Research contract FA23861214030. It has also been supported in part by the Monash e-Research Center and eSolutions-Research Support Services through the use of the Monash Campus HPC Cluster and the LIEF grant. This research was also undertaken on the NCI National Facility in Canberra, Australia, which is supported by the Australian Commonwealth Government. The authors would also like to thank the researchers from CESBIO (D. Ducrot, C. Marais-Sicre, O. Hagolle and M. Huc) for providing the land cover maps and the geometrically and radiometrically corrected FORMOSAT-2 images (satellite dataset). We also want to thank the anonymous reviewers for their insightful comments which have helped to greatly improve the revised version of the paper.

Appendix A. Tables of the Experimental Section

id	name	t	a	c	? (max) (average±sd)	Source	Relevant papers	description	size
1	localization	0.165	5	11	N	UCI Kaluža et al. (2010)		Recordings of 5 people performing different activities. Each person wore 3 sensors while performing the same scenario 5 times.	11MB
2	census-income	0.299	41	2	Y (50%) (4.9 ± 14.6%)	UCI Bache and Lichman (2013)	Oza and Russell (2001)	Weighted census data extracted from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau.	136MB
3	USPSExtended	0.341	676	2	N	CVM Tsang et al. (2005)		0/1 digit classification (extended version of the USPS data set[a]).	603MB[s]
4	MITFaceSetA	0.474	361	2	N	CVM Tsang et al. (2005)		Face detection using an extended version of the MIT face database[c]. By adding nonfaces to the original training set.	584MB
5	MITFaceSetB	0.489	361	2	N	CVM Tsang et al. (2005)		Each training face is blurred and added to set A. They are then flipped laterally.	603MB
6	MSDYear-Prediction	0.515	90	90	N	UCI Bertin-Mahieux et al. (2011)		Prediction of the release year of a song from audio features. Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s[c].	601MB[s]
7	covtype	0.581	54	7	N	UCI Blackard (1998)	Gama et al. (2003); Oza and Russell (2001)	Predicting forest cover type from cartographic attributes only (no remotely sensed data).	72MB
8	MITFaceSetC	0.839	361	2	N	CVM Tsang et al. (2005)		Each face in set B is rotated.	1.1GB
9	poker-hand	1.025	10	10	N	UCI	Cattral et al. (2002)	Each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. The class label describes the "Poker Hand". The order of cards is important.	24MB
10	uscensus1990	2.458	67	4	N	UCI		Discretized version of the USCensus1990raw dataset, a 1% sample from the full 1990 census. 'Temp. Absence From Work' has been selected as class.	325MB
11	PAMAP2	3.850	54	19	Y (91%) (1.6 ± 12.2%)	UCI Reiss and Stricker (2012)		Data of 18 different physical activities (such as walking, cycling, playing soccer, etc., the 19th label is transient activities), performed by 9 subjects wearing 3 inertial measurement units and a heart rate monitor.	1.7GB
12	kddcup	5.209	41	40	N	UCI (KDD Cup 1999)		Contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment: "bad" connections, called intrusions or attacks, and "good" normal connections.	754MB
13	linkage	5.749	11	2	Y (100%) (16.5 ± 36.9%)	Schmidtmann et al. (2009)	Sariyar et al. (2011)	Element-wise comparison of records with personal data from a record linkage setting. The task is to decide from a comparison pattern whether the underlying records belong to one person.	251MB
14	mnist8ms	8.100	784	10	N	CVM Loosli et al. (2007)	Erhan et al. (2010); Bottou et al. (1994)	Handwritten digit classification. Extended version by performing random elastic deformations of the original mnist digits.	19GB[s]
15	satellite	8.705	138	24	Y (93%) (26.4 ± 33%)	Petitjean et al. (2012)		Satellite image time series to predict land cover.	3.6GB
16	splice	54.627	141	2	N	Sonnenburg and Franc (2010)	Agarwal et al. (2011)	Recognising a human acceptor splice site (largest public data for which subsampling is not an effective learning strategy).	7.3GB

Table 7: Very large datasets for supervised classification. From left to right showing the identification number; name of the dataset; number of instances, attributes and classes; presence of missing values (and if positive, then maximum percentage per attribute and the average), source paper of the dataset; relevant papers and description.

[a] http://c2inet.sce.ntu.edu.sg/ivor/cvm.html.

[b] http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

[c] Subset of the Million Song Dataset (http://labrosa.ee.columbia.edu/millionsong/). Songs from a given artist may end up in both the train and test set.

[s] The superscript s stands for sparse format.

Table 8: Results in terms of RMSE for NB, TAN, AODE, KDB, BayesNet, RF, SKDB$_k$ and SKDB classifiers.

learner	args	localiz.	Census-income	USPS	MITA	MITB	MSDY	covtype	MITC	poker	uscensus	PAMAP	kddcup	linkage	mnist8	satellite	splice
NB		0.7106± 0.0007	0.4660± 0.0018	0.2256± 0.0026	0.0982± 0.0029	0.1394± 0.0014	0.9600± 0.0002	0.5103± 0.0013	0.2367± 0.0021	0.5801± 0.0006	0.2911± 0.0006	0.4647± 0.0006	0.1849± 0.0008	0.0125± 0.0006	0.4957± 0.0005	0.6540± 0.0002	0.0971± 0.0002
TAN		0.6321± 0.0014	0.2247± 0.0025	0.1153± 0.0022	0.0202± 0.0025	0.1213± 0.0020	0.9436± 0.0002	0.4862± 0.0008	0.2068± 0.0017	0.4987± 0.0006	0.1845± 0.0009	0.3232± 0.0007	0.0572± 0.0006	0.0081± 0.0009	0.3817± 0.0006	0.5424± 0.0003	0.1020± 0.0003
AODE		0.6520± 0.0010	0.2932± 0.0020	0.1538± 0.0028	0.1001± 0.0036	0.1682± 0.0025	0.9459± 0.0002	0.4587± 0.0013	0.1564± 0.0014	0.5392± 0.0006	0.2154± 0.0010±	0.3881± 0.0008	0.0979± 0.0007	0.0120± 0.0005	0.3497± 0.0005	0.5783± 0.0003	0.1034± 0.0003
KDB	$k = 1$	0.6501± 0.0012	0.2219± 0.0024	0.1125± 0.0025	0.0209± 0.0021	0.1291± 0.0019	0.9447± 0.0003	0.4709± 0.0026	0.1940± 0.0021	0.4987± 0.0005	0.1814± 0.0008	0.3276± 0.0007	0.0606± 0.0005	0.0082± 0.0009	0.3751± 0.0005	0.5506± 0.0003	0.0940± 0.0003
	$k = 2$	0.5840± 0.0020	0.2064± 0.0021	0.0990± 0.0029	0.0126± 0.0019	0.0633± 0.0018	0.9393± 0.0002	0.4576± 0.0013	0.0906± 0.0012	0.4055± 0.0007	0.1677± 0.0008	0.2721± 0.0005	0.0497± 0.0008	0.0062± 0.0007	0.2922± 0.0004	0.5072± 0.0006	0.0953± 0.0002
	$k = 3$	0.5485± 0.0020	0.2102± 0.0015	0.0928± 0.0025	0.0142± 0.0018	0.0468± 0.0018	0.9361± 0.0004	0.4399± 0.0015	0.0705± 0.0018	0.2859± 0.0011	0.1664± 0.0008	0.2290± 0.0004	0.0485± 0.0008	0.0060± 0.0006	0.2500± 0.0005	0.4769± 0.0004	0.1008± 0.0005
	$k = 4$	0.5225± 0.0023	0.2107± 0.0012	0.0877± 0.0026	0.0514± 0.0031	0.0387± 0.0021	0.9383± 0.0006	0.4126± 0.0019	0.0616± 0.0016	0.2048± 0.0012	0.1601± 0.0007	0.1937± 0.0004	0.0478± 0.0008	0.0060± 0.0006	0.2169± 0.0005	0.4583± 0.0005	0.1084± 0.0003
	$k = 5$	0.5225± 0.0023	0.2143± 0.0022	0.0839± 0.0034	0.1350± 0.0035	0.0841± 0.0023	0.9447± 0.0003	0.3903± 0.0020	0.0494± 0.0015	0.2842± 0.0011	0.1638± 0.0006	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1839± 0.0005	0.4448± 0.0004	0.0924± 0.0002
SKDB$_{only_k}$	$k_{\max} = 5$	0.5225± 0.0023	0.2064± 0.0021	0.0839± 0.0034	0.0126± 0.0019	0.0387± 0.0021	0.9361± 0.0004	0.3903± 0.0020	0.0494± 0.0015	0.2048± 0.0012	0.1601± 0.0007	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1839± 0.0005	0.4448± 0.0004	0.0924± 0.0002
	bestk	4	2	5	2	4	3	5	5	4	4	5	5	3.8	5	5	5
SKDB$_k$	$k = 1$	0.6501± 0.0012	0.2054± 0.0014	0.1125± 0.0025	0.0207± 0.0020	0.0746± 0.0012	0.9446± 0.0003	0.4704± 0.0016	0.1227± 0.0014	0.4987± 0.0005	0.1540± 0.0007	0.3276± 0.0007	0.0606± 0.0005	0.0082± 0.0009	0.3751± 0.0005	0.5485± 0.0002	0.0527± 0.0002
	$k = 2$	0.5840± 0.0020	0.2021± 0.0021	0.0990± 0.0029	0.0123± 0.0018	0.0574± 0.0011	0.9393± 0.0002	0.4576± 0.0013	0.0782± 0.0017	0.4055± 0.0007	0.1510± 0.0007	0.2721± 0.0005	0.0497± 0.0008	0.0062± 0.0007	0.2922± 0.0004	0.5065± 0.0004	0.0523± 0.0002
	$k = 3$	0.5485± 0.0020	0.2056± 0.0016	0.0928± 0.0026	0.0144± 0.0021	0.0439± 0.0021	0.9361± 0.0004	0.4399± 0.0015	0.0596± 0.0020	0.2847± 0.0011	0.1508± 0.0007	0.2290± 0.0004	0.0485± 0.0008	0.0060± 0.0006	0.2500± 0.0005	0.4769± 0.0004	0.0523± 0.0002
	$k = 4$	0.5225± 0.0023	0.2041± 0.0015	0.0877± 0.0026	0.0386± 0.0019	0.0376± 0.0026	0.9382± 0.0005	0.4126± 0.0019	0.0517± 0.0020	0.1868± 0.0012	0.1504± 0.0007	0.1937± 0.0004	0.0478± 0.0008	0.0060± 0.0006	0.2169± 0.0005	0.4583± 0.0005	0.0524± 0.0002
	$k = 5$	0.5225± 0.0023	0.2070± 0.0015	0.0839± 0.0033	0.0552± 0.0032	0.0844± 0.0025	0.9447± 0.0004	0.3903± 0.0020	0.0446± 0.0020	0.1868± 0.0012	0.1505± 0.0007	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1839± 0.0005	0.4448± 0.0004	0.0526± 0.0002
SKDB	$k_{\max} = 5$	0.5225± 0.0023	0.2021± 0.0021	0.0839± 0.0033	0.0123± 0.0018	0.0376± 0.0026	0.9361± 0.0004	0.3903± 0.0020	0.0446± 0.0020	0.1868± 0.0012	0.1504± 0.0007	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1839± 0.0005	0.4448± 0.0004	0.0523± 0.0002
	bestk	4	2	5	2	4	3	5	5	4	4	5	5	3.8	5	5	3
	# of atts.	5	19.2	647.5	337.9	275.6	90	54	215.8	5	22	54	37	11	773.6	138	16
SKDB$_k$ (# of attributes selected)	$k = 1$	5	7	584.9	354.9	17	80.5	45.8	59	10	11	54	38.2	11	772.2	75	13
	$k = 2$	5	19.2	646.5	337.9	87	90	54	206.7	10	12	54	38.1	11	773.1	125.8	14.9
	$k = 3$	5	18	648.1	356.1	271.3	90	54	209.6	5	22	54	37.6	11	774.2	138	16
	$k = 4$	5	18	640.9	21.3	275.6	78.8	54	214.3	5	22	54	37.9	11	774.2	138	14
	$k = 5$	5	18	647.5	18	359.3	81.2	54	215.8	5	22	54	37	11	773.6	138	13
n		5	41	676	361	361	90	54	361	10	67	54	41	11	784	138	141
BayesNet	$k = 5$	0.5217± 0.0023	0.2009± 0.0016	0.0920± 0.0023	0.0379± 0.0136	0.0354± 0.0063	0.9369± 0.0004	0.4231± 0.0016	>138G	0.2821± 0.0011	0.1573± 0.0008	0.1615± 0.0006	0.0477± 0.0007	0.0059± 0.0005	>138G	>138G	>138G
Random Forest	100 trees	0.5194± 0.0024	0.1992± 0.0013	0.0338± 0.0010	0.0353± 0.0017	0.0566± 0.0012	>138G	0.3478± 0.0018	0.0888± 0.0006	0.3190± 0.0028	0.1573± 0.0008	0.0962± 0.0012	0.0466± 0.0008	0.0060± 0.0007	>138G	>138G	>138G
Random Forest (Num)	100 trees	0.4199± 0.0017	0.1906± 0.0014	0.0291± 0.0007	0.0262± 0.0016	0.0395± 0.0010	>138G	0.1998± 0.0011	0.0465± 0.0004	0.3190± 0.0028	0.1573± 0.0008	>600h	0.0298± 0.0003	0.0029± 0.0006	>138G	>138G	>138G
SKDB$^{0\text{-}1}$	$k_{\max} = 5$	0.5225± 0.0023	0.2045± 0.0015	0.0839± 0.0033	0.0123± 0.0018	0.0376± 0.0026	0.9447± 0.0003	0.3903± 0.0020	0.0446± 0.0020	0.1868± 0.0012	0.1509± 0.0007	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1839± 0.0005	0.4448± 0.0004	0.0523± 0.0002
SKDB	MDL disc.	0.5252± 0.0036	0.1923± 0.0025	0.0837± 0.0023	0.0147± 0.0020	0.0370± 0.0026	0.9430± 0.0005	0.2595± 0.0029	0.1810± 0.0048	0.1868± 0.0012	0.1504± 0.0007	0.0377± 0.0029	0.0326± 0.0005	0.0037± 0.0009	0.1449± 0.0007	0.4460± 0.0006	0.0523± 0.0002
m (millions)		0.1649	0.2993	0.3415	0.4741	0.4894	0.5153	0.5811	0.8393	1.025	2.4583	3.8505	5.2095	5.7491	8.1	8.7052	50
c		11	2	9	2	2	90	7	2	10	4	19	40	2	10	24	2

Table 9: Results in terms of 0-1 Loss for NB, TAN, AODE, KDB, BayesNet, RF, SKDB$_k$ and SKDB classifiers.

learner	args	localiz.	Census-income	USPS	MITA	MITB	MSDY	covtype	MITC	poker	uscensus	PAMAP	kddcup	linkage	mnist8	satellite	splice
NB		0.5449± 0.0026	0.2410± 0.0017	0.0532± 0.0012	0.0100± 0.0006	0.0199± 0.0004	0.9514± 0.0005	0.3536± 0.0025	0.0582± 0.0010	0.4988± 0.0018	0.0896± 0.0003	0.2365± 0.0007	0.0361± 0.0005	0.0002± 0.0000	0.2521± 0.0005	0.4425± 0.0002	0.0121± 0.0001
TAN		0.4367± 0.0033	0.0675± 0.0016	0.0149± 0.0006	0.0008± 0.0001	0.0161± 0.0005	0.9268± 0.0010	0.3151± 0.0016	0.0455± 0.0008	0.3295± 0.0015	0.0390± 0.0005	0.1171± 0.0005	0.0034± 0.0001	0.0001± 0.0000	0.1327± 0.0007	0.3240± 0.0004	0.0133± 0.0001
AODE		0.433± 0.0027	0.1106± 0.0015	0.0244± 0.0008	0.0104± 0.0007	0.0294± 0.0008	0.9281± 0.0013	0.2859± 0.0016	0.0254± 0.0005	0.4812± 0.0028	0.0532± 0.0004±	0.1654± 0.0007	0.0154± 0.0002	0.0002± 0.0000	0.1271± 0.0004	0.3537± 0.0004	0.0134± 0.0001
KDB	$k=1$	0.4642± 0.0040	0.0667± 0.0014	0.0142± 0.0006	0.0005± 0.0001	0.0183± 0.0009	0.9279± 0.0008	0.2968± 0.0034	0.0406± 0.0009	0.3291± 0.0012	0.0383± 0.0004	0.1209± 0.0006	0.0048± 0.0001	0.0001± 0.0000	0.1500± 0.0004	0.3321± 0.0005	0.0114± 0.0001
	$k=2$	0.3710± 0.0037	0.0562± 0.0013	0.0110± 0.0006	0.0002± 0.0001	0.0043± 0.0002	0.9239± 0.0008	0.2799± 0.0025	0.0088± 0.0003	0.1961± 0.0009	0.0333± 0.0004	0.0850± 0.0003	0.0028± 0.0001	0.0000± 0.0000	0.0919± 0.0001	0.2903± 0.0006	0.0116± 0.0001
	$k=3$	0.3338± 0.0023	0.0556± 0.0010	0.0097± 0.0005	0.0002± 0.0001	0.0023± 0.0002	0.9187± 0.0009	0.2641± 0.0020	0.0053± 0.0003	0.0838± 0.0007	0.0331± 0.0004	0.0609± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0682± 0.0002	0.2613± 0.0003	0.0127± 0.0001
	$k=4$	0.3064± 0.0036	0.0547± 0.0008	0.0087± 0.0005	0.0028± 0.0003	0.0016± 0.0002	0.9131± 0.0012	0.2337± 0.0023	0.0040± 0.0002	0.0486± 0.0007	0.0297± 0.0003	0.0438± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0519± 0.0002	0.2426± 0.0005	0.0145± 0.0001
	$k=5$	0.3064± 0.0036	0.0547± 0.0010	0.0080± 0.0006	0.0188± 0.0010	0.0073± 0.0004	0.9124± 0.0006	0.2077± 0.0023	0.0026± 0.0002	0.0877± 0.0008	0.0313± 0.0002	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0104± 0.0001
SKDB$_{only_k}$	$k_{\max}=5$	0.3064± 0.0036	0.0562± 0.0013	0.0080± 0.0006	0.0028± 0.0003	0.0199± 0.0004	0.9187± 0.0009	0.2077± 0.0023	0.0582± 0.0010	0.0486± 0.0007	0.0297± 0.0003	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
SKDB$_k$	$k=1$	0.4642± 0.0040	0.0553± 0.0009	0.0142± 0.0007	0.0005± 0.0001	0.0071± 0.0002	0.9276± 0.0010	0.2961± 0.0021	0.0185± 0.0004	0.3291± 0.0012	0.0293± 0.0004	0.1209± 0.0006	0.0048± 0.0001	0.0001± 0.0000	0.1500± 0.0004	0.3391± 0.0003	0.0029± 0.0000
	$k=2$	0.3710± 0.0037	0.0542± 0.0013	0.0110± 0.0006	0.0002± 0.0000	0.0040± 0.0002	0.9239± 0.0008	0.2799± 0.0025	0.0069± 0.0003	0.1961± 0.0009	0.0271± 0.0003	0.0850± 0.0003	0.0028± 0.0001	0.0000± 0.0000	0.0919± 0.0002	0.2907± 0.0022	0.0029± 0.0000
	$k=3$	0.3338± 0.0023	0.0547± 0.0009	0.0097± 0.0005	0.0002± 0.0001	0.0021± 0.0002	0.9187± 0.0009	0.2641± 0.0020	0.0040± 0.0003	0.0835± 0.0008	0.0268± 0.0003	0.0609± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0682± 0.0002	0.2613± 0.0003	0.0029± 0.0000
	$k=4$	0.3064± 0.0036	0.0532± 0.0008	0.0087± 0.0005	0.0018± 0.0002	0.0015± 0.0002	0.9144± 0.0011	0.2337± 0.0023	0.0030± 0.0002	0.0318± 0.0008	0.0266± 0.0003	0.0438± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0519± 0.0002	0.2426± 0.0005	0.0029± 0.0000
	$k=5$	0.3064± 0.0036	0.0536± 0.0009	0.0080± 0.0006	0.0038± 0.0005	0.0074± 0.0004	0.9135± 0.0008	0.2077± 0.0023	0.0022± 0.0002	0.0318± 0.0008	0.0266± 0.0003	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
SKDB	$k_{\max}=5$	0.3064± 0.0036	0.0542± 0.0013	0.0080± 0.0006	0.0002± 0.0000	0.0015± 0.0002	0.9187± 0.0009	0.2077± 0.0023	0.0022± 0.0002	0.0318± 0.0008	0.0266± 0.0003	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
SKDB$^{0\text{-}1}$	$k_{\max}=5$	0.3064± 0.0036	0.0533± 0.0011	0.0080± 0.0006	0.0002± 0.0000	0.0015± 0.0002	0.9124± 0.0006	0.2077± 0.0023	0.0022± 0.0002	0.0318± 0.0008	0.0261± 0.0002	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
BayesNet	$k=5$	0.3053± 0.0039	0.0533± 0.0012	0.0097± 0.0006	0.0017± 0.0006	0.0014± 0.0004	0.9204± 0.0015	0.2442± 0.0020	>138G	0.0831± 0.0010	0.0288± 0.0003	0.0308± 0.0003	0.0026± 0.0001	0.0000± 0.0000	>138G	>138G	>138G
Random Forest	100 trees	0.3061± 0.0036	0.0527± 0.0009	0.0009± 0.0001	0.0018± 0.0002	0.0031± 0.0003	>138G	0.1157± 0.0013	0.0015± 0.0002	0.0527± 0.0013	0.0266± 0.0003	0.0046± 0.0001	0.0025± 0.0001	0.0000± 0.0000	>138G	>138G	>138G
Random Forest (Num)	100 trees	0.2030± 0.0014	0.0481± 0.0011	0.0005± 0.0001	0.0009± 0.0002	0.0013± 0.0001	>138G	0.0405± 0.0006	0.0004± 0.0000	0.0527± 0.0013	0.0266± 0.0003	>138G	0.0014± 0.0000	0.0000± 0.0000	>138G	>138G	>138G
SKDB	MDL disc.	0.3013± 0.0048	0.0486± 0.0015	0.0078± 0.0005	0.0002± 0.0000	0.0015± 0.0002	0.9129± 0.0006	0.0824± 0.0019	0.0035± 0.0002	0.0318± 0.0008	0.0266± 0.0003	0.0016± 0.0001	0.0017± 0.0001	0.0000± 0.0000	0.0229± 0.0002	0.2175± 0.0005	0.0029± 0.0000
n		5	41	676	361	361	90	54	361	10	67	54	41	11	784	138	141
m (millions)		0.1649	0.2993	0.3415	0.4741	0.4894	0.5153	0.5811	0.8393	1.025	2.4583	3.8505	5.2095	5.7491	8.1	8.7052	50
c		11	2	9	2	2	90	7	2	10	4	19	40	2	10	24	2

Appendix B. Implementation details

All the experiments for the out-of-core algorithms have been carried out in a C++ software specially designed to deal with out-of-core classification methods. The following details in terms of implementation and configuration should be considered:

- 10-fold cross validation has been used.
- Equal frequency pre-discretization with 5 bins (EF5) has been utilized to discretize all numeric attributes (except if otherwise specified[5]). We have observed that EF5 and MDL (Minimum Description Length) (Fayyad and Irani, 1993) discretization provide the best results in approximately half of the datasets each, EF5 has been chosen because it is faster to perform than MDL, and also because it is not supervised and hence does not potentially provide the classifier with class information from the hold-out data when used for pre-discretization. Using a pre-fixed number of bins gives us the added advantage of not having to deal with a huge number of values per attribute created by MDL discretization in some cases.
- Missing values have been considered as a distinct value in all cases.
- The root mean square error is calculated exclusively on the true class label (different from Weka's implementation (Hall et al., 2009), where all class labels are considered).
- KDB's implementation benefits from a tree structure that resembles an ADTtree (Moore and Lee, 1998), sparseness is achieved by storing a NULL instead of a node for any query that matches zero records. In our case, to save the probability distribution of the different attributes once the structure has been determined, we build a (distribution) tree for each attribute. Given, for example, attribute X_4 that has X_0 and X_3 as parents (in this order, and apart from the class), the counts (X_4, Y) are stored in the root node. The following level expands according to the number of values of X_0, which is the first parent, and store the counts for (X_4, Y, X_0). The following level expands as for the number of values of X_3, which is the second parent, and store the counts for (X_4, Y, X_0, X_2), and so forth if there were more parents. Note that as opposed to the ADTtree, all the parameters are stored, that is, more than strictly required, since a branch is created for each attribute value, when strictly one less branch would be sufficient to represent the model parameters. The reason to do that is not to overload classification time by having to explore and combine branches.
- Two combined techniques are considered for smoothing. In the first place, we use m-estimates[6] (Mitchell, 1997) as follows:

$$\hat{p}(x_i|\pi_{x_i}) = \frac{counts(x_i, \pi_{x_i}) + \frac{m}{|X_i|}}{counts(\pi_{x_i}) + m} \tag{4}$$

5. Only for the experiments on the antepenultimate row on Table 8 as specified.
6. 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.

Table 10: Results in terms of RMSE and 0-1 loss for SKDB and VW with different options: -p indicates the number of passes (3 or 20), -q indicates quadratic features (all possible combinations), -c refers to cubic features (all possible combinations), and -l the use of a logistic function for optimization. SKDB$^{0\text{-}1}$ is SKDB optimized for 0-1 loss.

	learner	args	localiz.	Census-income	USPS	MITA	MITB	MSDY	covtype	MITC	poker	uscensus	PAMAP	kddcup	linkage	mnist8	satellite	splice
RMSE	SKDB	$k_{\max}=5$	0.5225± 0.0023	0.2021± 0.0021	0.0839± 0.0033	0.0123± 0.0018	0.0376± 0.0026	0.9361± 0.0004	0.3903± 0.0020	0.0446± 0.0020	0.1868± 0.0012	0.1504± 0.0007	0.1697± 0.0004	0.0477± 0.0008	0.0060± 0.0006	0.1449± 0.0007	0.4448± 0.0004	0.0523± 0.0002
	VW	-p3	0.6249± 0.0006	0.2709± 0.0048	0.2156± 0.0011	0.0326± 0.0076	0.0696± 0.0069	0.7101± 0.0001	0.5227± 0.0010	0.1361± 0.0013	0.5355± 0.0006	0.2038± 0.0014	0.5309± 0.0006	0.1240± 0.0020	0.0184± 0.0038	0.4176± 0.0009	0.5107± 0.0006	0.2085± 0.0045
		-p20	0.6243± 0.0006	0.2662± 0.0028	0.2157± 0.0008	0.0266± 0.0035	0.0714± 0.0047	0.7093± 0.0001	0.5116± 0.0009	0.1366± 0.0008	0.5355± 0.0005	0.2018± 0.0011	0.5217± 0.0006	0.1144± 0.0025	0.0151± 0.0032	0.4173± 0.0008	0.5024± 0.0006	0.2115± 0.0038
		-p3 -q	0.5955± 0.0009	0.2426± 0.0080	0.0402± 0.0006	0.0277± 0.0072)	0.0403± 0.0058	0.7276± 0.0005	0.5018± 0.0018	0.0518± 0.0042	0.3679± 0.0019	0.1938± 0.0040	0.3633± 0.0010	0.2426± 0.0080	0.0103± 0.0029	0.2884± 0.0063	0.4505± 0.0015	0.0855± 0.0082
		-p3 -c	0.6217± 0.0006	0.2681± 0.0047	0.1894± 0.0014	0.0319± 0.0058	0.0645± 0.0065	0.7088± 0.0001	0.5177± 0.0012	0.1232± 0.0022	0.5342± 0.0006	0.2023± 0.0015	0.4689± 0.0007	0.1123± 0.0026	0.0079± 0.0017	0.4173± 0.0009	0.5031± 0.0007	0.1964± 0.0045
		-p3 -q -l	0.7357± 0.0021	0.1895± 0.0020	0.0190± 0.0074	0.0238± 0.0021*	0.0303± 0.0016*	0.9273± 0 **	0.5467± 0.0020	0.0308± 0.0016*	0.3545± 0.0018	0.1482± 0.0009*	0.3920± 0.0017*	0.0519± 0.0007	0.0043± 0.0006	0.2773± 0.0039	>400h	>400h
0-1 Loss	SKDB$^{0\text{-}1}$	$k_{\max}=5$	0.3064± 0.0036	0.0533± 0.0011	0.0080± 0.0006	0.0002± 0.0000	0.0015± 0.0002	0.9124± 0.0006	0.2077± 0.0023	0.0022± 0.0002	0.0318± 0.0008	0.0261± 0.0002	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
	SKDB	$k_{\max}=5$	0.3064± 0.0036	0.0542± 0.0013	0.0080± 0.0006	0.0002± 0.0000	0.0015± 0.0002	0.9187± 0.0009	0.2077± 0.0023	0.0022± 0.0002	0.0318± 0.0008	0.0266± 0.0003	0.0340± 0.0002	0.0026± 0.0001	0.0000± 0.0000	0.0378± 0.0002	0.2284± 0.0004	0.0029± 0.0000
	VW	-p3	0.6036± 0.0041	0.0495± 0.0012	0.0311± 0.0011	0.0006± 0.0003	0.0031± 0.0005	0.9192± 0.0012	0.3688± 0.0029	0.0117± 0.0006	0.4989± 0.0015	0.0290± 0.0003	0.3488± 0.0013	0.0040± 0.0002	0.0000± 0.0000	0.1344± 0.0006	0.3053± 0.0013	0.0029± 0.0000
		-p20	0.6025± 0.0036	0.0490± 0.0011	0.0307± 0.0009	0.0005± 0.0002	0.0028± 0.0003	0.9173± 0.0013	0.3530± 0.0023	0.0115± 0.0004	0.4988± 0.0015	0.0283± 0.0003	0.3316± 0.0009	0.0035± 0.0001	0.0000± 0.0000	0.1341± 0.0006	0.2970± 0.0010	0.0029± 0
		-p3 -q	0.5254± 0.0050	0.0502± 0.0020	0.0000± 0.0000	0.0003± 0.0001	0.0007± 0.0004	0.9198± 0.0012	0.3313± 0.0032	0.0007± 0.0003	0.0740± 0.0007	0.0255± 0.0004	0.0980± 0.0006	0.0020± 0.0001	0.0000± 0.0000	0.0440± 0.0010	0.2341± 0.0015	0.0027± 0.0001
		-p3 -c	0.5932± 0.0039	0.0493± 0.0011	0.0193± 0.0009	0.0005± 0.0002	0.0028± 0.0005	0.9162± 0.0013	0.3619± 0.0029	0.0091± 0.0005	0.5015± 0.0026	0.0285± 0.0003	0.2286± 0.0011	0.0024± 0.0001	0.0000± 0.0000	0.1342± 0.0006	0.2947± 0.0010	0.0029± 0.0000
		-p3 -q -l	0.5452± 0.0051	0.0475± 0.0011	0.0004± 0.0005	0.0003± 0.0001*	0.0007± 0.0002*	0.9205± 0 **	0.3418± 0.0030	0.0009± 0.0001*	0.0746± 0.0008	0.0250± 0.0003*	0.1328± 0.0005*	0.0030± 0.0001	0.0000± 0.0000	0.0565± 0.0005	>400h	>400h
	m		0.1649	0.2993	0.3415	0.4741	0.4894	0.5153	0.5811	0.8393	1.025	2.4583	3.8505	5.2095	5.7491	8.1	8.7052	50
	c		11	2	9	2	2	90	7	2	10	4	19	40	2	10	24	2

* Results shown for 1 experiment only due to time limitations when using the logistic function in VW.

** Results shown for 1 fold only due to time limitations when using the logistic function in VW.

where π_{x_i} are the parent-values of X_i and $m = 1$. Secondly, if zero counts are found, we back off as many levels in the tree as necessary to find at least one count, that is, if $counts(x_4, x_0, x_3)$ is equal to zero, then $\hat{p}(x_4|x_0)$ is considered instead of $\hat{p}(x_4|x_0, x_3)$.

- In principle, any loss function that can be computed incrementally is suitable for SKDB. We have chosen RMSE (as defined above) for our first set of experiments.

- We use Averaged One Dependence Estimators (AODE) as a representative of the ANDE family of algorithms where $n = 1$.

  In order to avoid an extensive number of passes on each of the datasets, no dataset-tailored parameter tuning has been carried out for any of the algorithms considered. However as detailed in Sections 3.2, for logistic regression we have pre-conducted experiments with varying numbers of passes for stochastic gradient descent, varying loss functions and combinations of features; and used the default advanced (step size) updates provided by vowpal wabbit. In the case of random forest, we have made experiments with and without pre-discretising continuous attributes, as well as learning a variable number of trees (see Section 3.3.2 for more details).

Appendix C. Detailed analysis of results

The upper part of Table 8 (in the Appendix) shows the results in terms of RMSE ± standard deviation on the 16 datasets described above. We have decided to stop at $k = 5$, since some of the datasets already start to show worse results for $k = 5$ (due to variance increase), so increasing the complexity further does not compensate. The results for NB, TAN and AODE are displayed as baseline orientations. The dark gray background on certain outputs for SKDB$_k$, corresponds to experiments that outperform KDB for the same value of k; the light grey background on others indicate a tie; whereas the absence of the two indicates that plain KDB outperforms SKDB$_k$. As for SKDB, the background coloring has the same meaning, but each entry is compared with the best KDB value for all possible k. Note that only the average of the 10 folds is shown for each experiment, and the background colors indicate statistical difference (Wilcoxon signed-rank tests), dark grey or white (in favor or against the algorithm in the row respectively); or lack of statistical difference, in light grey.

Table 8 also shows the number of selected attributes by SKDB$_k$. There are a few cases for which the number of selected attributes for all values of k is the total number of attributes in the datasets, such as `localization`, `PAMAP` and `linkage`. For some other datasets, e.g. `MSDYearPrediction`, `USPSExtended` or `mnist8ms`, the performance between selected KDB and KDB is similar, but fewer attributes are considered, so classification time (and space) will be reduced. In most of the cases the number of selected attributes is smaller than the original, sometimes even less than half, reducing the RMSE compared to KDB, e.g. `MITFaceSetC` with 215.8 attributes selected (out of 361) results in 0.0446 vs 0.0494 for KDB ($k = 5$), or `uscensus` with 22 attributes selected (out of 67) results in 0.1504 vs 0.1601 for KDB ($k = 4$), that in both cases results in statistical improvement.

The results for SKDB using MDL discretization are also shown in Table 8 (antepenultimate row). Note that there are some datasets, such as `linkage`, for which the discretization method used, in this case EF5, is not an adequate one; whereas a technique based on entropy minimization provide better results. Just the opposite situation occurs for datasets such as MITFaceSetC. Apriori selection of the most appropriate discretization method is an open question, even more for very large datasets where we should consider the same bias/variance trade-off in terms of discretization bias and variance (Yang and Webb, 2009), that is, on the one hand we want that the number of intervals is sufficiently large to provide low discretization bias, but not too large to avoid an increase in discretization variance. The latter should not be a problem in large datasets, but considering a large number of intervals implies bigger datasets, which will have an impact on the classifier complexity.

References

Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. arXiv preprint arXiv:1110.4198, 2011.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Jock A. Blackard. *Comparison of neural networks and discriminant analysis in predicting forest cover types.* PhD thesis, Fort Collins, CO, USA, 1998. AAI9921979.

Léon Bottou, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle Guyon, Lawrence D. Jackel, Yann Le Cun, Urs A. Muller, Eduard Säckinger, Patrice Simard, and Vladimir Vapnik. Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition, Los Alamitos, CA.*, volume 2, pages 77–82, 1994.

Remco R. Bouckaert. Voting massive collections of Bayesian network classifiers for data streams. In *Proceedings of the 19th Australian joint conference on Artificial Intelligence: advances in Artificial Intelligence*, AI'06, pages 243–252, Berlin, Heidelberg, 2006. Springer-Verlag.

Damien Brain and Geoffrey I. Webb. On the effect of data set size on bias and variance in classification learning. In D. Richards, G. Beydoun, A. Hoffmann, and P. Compton, editors, *Proceedings of the Fourth Australian Knowledge Acquisition Workshop (AKAW '99)*, pages 117–128, Sydney, 1999. The University of New South Wales.

Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 75:1–3, 1950.

Alexandra M. Carvalho, Teemu Roos, Arlindo L. Oliveira, and Petri Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *J. Mach. Learn. Res.*, 12:2181–2210, 2011.

Robert Cattral, Franz Oppacher, and Dwight Deugo. Evolutionary data mining with automatic rule generalization. In *Recent Advances in Computers, Computing and Communications*, pages 296–300, 2002.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, 1992.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons Inc, 1 edition, 1973.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010.

Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In Ruzena Bajcsy, editor, *IJCAI*, pages 1022–1029. Morgan Kaufmann, 1993.

M. Julia Flores, José A. Gámez, Ana M. Martínez, and José M. Puerta. GAODE and HAODE: Two proposals based on AODE to deal with continuous variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 313–320. ACM, 2009a.

M. Julia Flores, Josi A. Gamez, Ana M. Martmnez, and Jose Miguel Puerta. Hode: Hidden one-dependence estimator. In Claudio Sossai and Gaetano Chemello, editors, *ECSQARU*, volume 5590 of *Lecture Notes in Computer Science*, pages 481–492. Springer, 2009b. ISBN 978-3-642-02905-9.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, 1997.

João Gama, Ricardo Rocha, and Pedro Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 523–528, New York, NY, USA, 2003. ACM.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

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

Geoff Hulten and Pedro Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 525–531, New York, NY, USA, 2002. ACM.

Manfred Jaeger. Probabilistic classifiers and the concepts they recognize. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 266–273. AAAI Press, 2003.

Boštjan Kaluža, Violeta Mirchevska, Erik Dovgan, Mitja Luštrek, and Matjaž Gams. An agent-based approach to care in independent living. In *Proceedings of the First international joint conference on Ambient intelligence*, AmI'10, pages 177–186, Berlin, Heidelberg, 2010. Springer-Verlag.

Nikos Karampatziakis and John Langford. Online importance weight aware updates. In Fabio Gagliardi Cozman and Avi Pfeffer, editors, *UAI*, pages 392–399. AUAI Press, 2011.

Ron Kohavi. The power of decision tables. In Nada Lavrac and Stefan Wrobel, editors, *Proceedings of the European Conference on Machine Learning ECML-95*, volume 912 of *Lecture Notes in Computer Science*, pages 174–189. Springer Berlin Heidelberg, 1995.

Daphne Koller and Mehran Sahami. Toward optimal feature selection. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 284–292. Morgan Kaufmann Publishers, 1996.

Pat Langley and Stephanie Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 399–406, San Francisco, CA, 1994. Morgan Kaufmann.

David D. Lewis. Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 4–15, London, UK, UK, 1998. Springer-Verlag.

George James Lidstone. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty Actuaries*, 8:182–192, 1920.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training Invariant Support Vector Machines using Selective Sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

Oded Maron and Andrew Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In Jack D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 59–66, San Francisco, CA, 1994. Morgan Kaufmann.

B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451, 1975.

H. Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30, 1961.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

Andrew Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *J. Artif. Int. Res.*, 8(1):67–91, 1998.

Nikunj C. Oza and Stuart Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 359–364, New York, NY, USA, 2001. ACM.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

Franz Pernkopf and Jeff A. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *J. Mach. Learn. Res.*, 11:2323–2360, 2010.

François Petitjean, Jordi Inglada, and Pierre Gançarski. Satellite Image Time Series Analysis under Time Warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8): 3081–3095, 2012.

Attila Reiss and Didier Stricker. Creating and benchmarking a new dataset for physical activity monitoring. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '12, pages 40:1–40:8, New York, NY, USA, 2012. ACM.

Stéphane Ross, Paul Mineiro, and John Langford. Normalized online learning. arXiv preprint arXiv:1305.6646, 2013.

Arcadio Rubio and José Antonio Gámez. Flexible learning of k-dependence Bayesian network classifiers. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1219–1226, New York, NY, USA, 2011. ACM.

Mehran Sahami. Learning limited dependence Bayesian classifiers. In *In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338. AAAI Press, 1996.

Murat Sariyar, Andreas Borg, and Klaus Pommerening. Controlling false match rates in record linkage using extreme value theory. *J. of Biomedical Informatics*, 44(4):648–654, 2011.

I Schmidtmann, G Hammer, M Sariyar, and A. Gerhold-Ay. Evaluation des krebsregisters nrw - schwerpunkt record linkage - abschlussbericht. Technical report, Institut für medizinische Biometrie, Epidemiologie und Informatik, Universitätsmedizin Mainz, 2009.

Sören Sonnenburg and Vojtech Franc. COFFIN : A computational framework for linear SVMs. In *Proc. ICML 2010*, 2010.

Jiang Su and Harry Zhang. Full Bayesian network classifiers. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 897–904, New York, NY, USA, 2006. ACM.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res.*, 6:363–392, 2005.

Geoffrey I. Webb, Janice R. Boughton, and Zhihai Wang. Not so naive Bayes: Aggregating one-dependence estimators. *Mach. Learn.*, 58(1):5–24, 2005.

Geoffrey I. Webb, Janice R. Boughton, Fei Zheng, Kai Ming Ting, and Houssam Salem. Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. *Mach. Learn.*, 86(2):233–272, 2012.

Y. Yang, G.I. Webb, J. Cerquides, K. Korb, J. Boughton, and K-M. Ting. To select or to weigh: A comparative study of linear combination schemes for superparent-one-dependence estimators. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(12):1652–1665, 2007.

Ying Yang and Geoffrey I. Webb. Discretization for naive-Bayes learning: managing discretization bias and variance. *Mach. Learn.*, 74(1):39–74, 2009.

Nayyar A. Zaidi, Jesús Cerquides, Mark James Carman, and Geoffrey I. Webb. Alleviating naive Bayes attribute independence assumption by attribute weighting. *J. Mach. Learn. Res.*, 14(1):1947–1988, 2013.

Harry Zhang, Liangxiao Jiang, and Jiang Su. Hidden naive Bayes. In *Proceedings of the 20th National Conference on Artificial intelligence - Volume 2*, pages 919–924. AAAI Press, 2005.

F. Zheng, G.I. Webb, P. Suraweera, and L. Zhu. Subsumption resolution: An efficient and effective technique for semi-naive Bayesian learning. *Mach. Learn.*, 87(1):93–125, 2012.

Fei Zheng and Geoffrey I Webb. A comparative study of semi-naive Bayes methods in classification learning. In *Proceedings of the Fourth Australasian Data Mining Conference (AusDM05)*, pages 141–156, 2005.

Zijian Zheng and Geoffrey I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 41 (1):53–84, 2000.