# Sample-based Attribute Selective A$n$DE for Large Data

Shenglei Chen, Ana M. Martínez, Geoffrey I. Webb, *Senior Member, IEEE,* and Limin Wang

**Abstract**—More and more applications come with large data sets in the past decade. However, existing algorithms cannot guarantee to scale well on large data. Averaged *n*-Dependence Estimators (A*n*DE) allows for flexible learning from out-of-core data, by varying the value of $n$ (number of super parents). Hence A*n*DE is especially appropriate for large data learning. In this paper, we propose a sample-based attribute selection technique for A*n*DE. It needs one more pass through the training data, in which a multitude of approximate A*n*DE models are built and efficiently assessed by leave-one-out cross validation. The use of a sample reduces the training time. Experiments on 15 large data sets demonstrate that the proposed technique significantly reduces A*n*DE's error at the cost of a modest increase in training time. This efficient and scalable out-of-core approach delivers superior or comparable performance to typical in-core Bayesian network classifiers.

**Index Terms**—Bayesian network classifiers, Large data, Classification learning, Attribute selection, Averaged *n*-Dependence Estimators (A*n*DE), Leave-one-out cross validation

## 1 Introduction

In the past decade, large data sets have attracted a lot of research interest [1], [2], [3]. When learning from very large data, it is infeasible to load the entire data into RAM. One possible way to process large data is to learn out-of-core [4]. As data access would be very expensive in this case, learning algorithms should ideally require only a few passes through the training data.

At the same time, existing algorithms cannot guarantee to scale well on large data. Past research shows that classification error can be usefully decomposed into bias and variance [5], and variance will be lower when learning on large data than when learning from small data sets, and hence have less effect on total error [6]. That is to say, a low variance algorithm will usually have an advantage for small data while a low bias algorithm will usually have an advantage for large data. Accordingly, algorithms with low bias are highly appealing for large data learning.

Averaged *n*-Dependence Estimators (A*n*DE) is one family of Bayesian network classifiers that can learn in a single pass through training examples [7], [8]. The requirement of only one single pass through training data supports out-of-core learning. Since A*n*DE allows every attribute to depend on *n* shared super parent attributes, which is more coincident with the characteristics of real data sets, it has lower bias than Naive Bayes (NB) [9] and Tree Augmented Naive Bayes (TAN) [10]. And as the parameter *n* increases, A*n*DE achieves progressively lower bias at the cost of higher variance [8]. This low bias characteristic, combined with the single pass learning, makes A*n*DE well suited to large data, where variance is generally low.

When A*n*DE is applied to large data, we expect large *n* to obtain low bias. However, the memory requirements in A*n*DE increase combinatorially with the number of attributes and the parameter *n*. Thus, higher *n* means not only lower bias but also higher memory requirements. Given the memory constraint of existing machines, we focus on Averaged One-Dependence Estimators (AODE) and Averaged 2-Dependence Estimators (A2DE) in this paper.

An efficient out-of-core attribute selection technique for AODE [11] has been demonstrated to be more effective at reducing AODE's bias than previous approaches. However, performance has only been demonstrated on small data sets. In order to deal with large data, we exploit the low bias advantage of A2DE by generalizing the attribute selection technique to A2DE. Meanwhile, we propose to use a sample for the attribute selection pass in order to reduce the training time on large data. This paper presents the empirical evidence that the proposed out-of-core technique significantly reduces A*n*DE's error and delivers superior or comparable performance to typical in-core Bayesian network classifiers on 15 large data sets.

The rest of the paper is organized as follows: First, we summarize related work on Bayes network classification from three different viewpoints, structure extension to NB, attribute weighting and attribute selection in Section 2. Specially, we describe the A*n*DE algorithm in Section 3. Then we present our novel attribute selection algorithms for AODE and A2DE in Section 4. This is followed by the description of our experimental setup and results in detail in Section 5. We present conclusions in Section 6.


- *S. Chen is with the Department of E-Commerce, Nanjing Audit University, Nanjing, 211815, China, also with the Faculty of Information Technology, Monash University, VIC 3800, Australia and the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, 130012, China. E-mail: tristan_chen@126.com*
- *A.M. Martinez is with Aalborg University, DK-9220 Aalborg, Denmark, and also with the Faculty of Information Technology, Monash University, VIC 3800, Australia. E-mail: anam.martinezf@gmail.com*
- *G.I. Webb is with the Faculty of Information Technology, Monash University, VIC 3800, Australia. E-mail: geoff.webb@monash.edu*
- *L. Wang is with the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, 130012, China. E-mail: wanglim@jlu.edu.cn*


*Manuscript received November 15, 2014; revised September 17, 2014.*

TABLE 1
Notation

| Symbols | Description |
|---|---|
| $X, X_i, Y$ | discrete random variable |
| $x, x_i, y$ | value of $X$, $X_i$, $Y$ |
| $n$ | number of super parents in A$n$DE |
| $c$ | number of classes |
| $d$ | number of attributes |
| $\mathbf{X} = \langle X_1, \ldots, X_d \rangle$ | $d$-dimensional random vector |
| $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$ | an example |
| $t$ | number of training examples |
| $\mathcal{T}$ | set of training examples |
| $v$ | average number of values per attribute |
| $M$ | number of parent tuples in A2DE |
| $S$ | sample size |

## 2 Related Work

In this section, we first introduce some notations and the idea of NB. Then we review A$n$DE and some other important structure extensions to NB. In the end, we list improvements to A$n$DE from two different viewpoints, attribute weighting and attribute selection.

Let $X$ be a *discrete random variable* taking values in a countable set $\mathcal{X}$. We assume the domain $\mathcal{X}$ is finite. A $d$-dimensional *random vector* is denoted by $\mathbf{X} = \langle X_1, \ldots, X_d \rangle$ where each component $X_i$ is a random variable over $\mathcal{X}_i$. For variable $X$, we denote the value of $X$ by lower letter $x$. Then the value $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$ of $\mathbf{X}$ represents an example in classification context. Let $Y \in \{1, \ldots, c\}$ represent the class variable, where $c$ is the number of classes. Then the classification task could be described as to estimate the probability $P(y \mid \mathbf{x})$ that a new example $\mathbf{x}$ belongs to some class $y$, given a training sample $\mathcal{T}$ of $t$ examples, and predict the class of $\mathbf{x}$ as $\arg\max_y P(y \mid \mathbf{x})$. These and other elements of notation are listed in Table 1.

From the definition of conditional probability, we have $P(y \mid \mathbf{x}) = P(y, \mathbf{x}) / P(\mathbf{x})$ . Since $P(\mathbf{x}) = \sum_{y=1}^{c} P(y, \mathbf{x})$, it is reasonable to consider $P(\mathbf{x})$ as the normalizing constant and estimate only the joint probability $P(y, \mathbf{x})$ in the remainder of this paper.

Since the example $\mathbf{x}$ does not appear frequently enough in the training data, we cannot directly derive an accurate estimate of the joint probability $P(y, \mathbf{x})$ and must extrapolate this estimate from observations of lower-dimensional probabilities in the data [8]. Applying the definition of conditional probabilities again, we have $P(y, \mathbf{x}) = P(y)P(\mathbf{x} \mid y)$ . The first term $P(y)$ on the right side can be sufficiently accurately estimated from the sample frequencies, if the number of classes, $c$, is not too large. Researchers in Bayesian Network classification community have developed different techniques to estimate the second term $P(\mathbf{x} \mid y)$.

NB [9] assumes the attributes are independent of each other given the class, so it calculates the joint probability $\mathrm{P}(y, \mathbf{x})$ according to the following formula,

$$\mathrm{P}_{\mathrm{NB}}(y, \mathbf{x}) = \mathrm{P}(y)\mathrm{P}(\mathbf{x} \mid y) = \mathrm{P}(y) \prod_{i=1}^{d} \mathrm{P}(x_i \mid y). \quad (1)$$

### 2.1 Structure Extension to NB

Because the attributes independence assumption is too strict in NB, many efforts have been done to alleviate the assumption, among which TAN [10] is a popular approach. It approximates the interactions between attributes by a tree structure imposed on the NB structure. That is to say, it requires that the class variable has no parents and each attribute has as parents the class variable and at most one other attribute. They developed an algorithm to learn TAN classifiers in polynomial time, which extends a well-known result by Chow and Liu [12]. TAN is a one pass algorithm, because the probability distributions required for selecting the network structure and parameterizing the conditional probability tables can be obtained in one pass learning through the training examples.

Keogh and Pazzani [13] proposed two methods for adding the set of augmenting arcs, a greedy hill-climbing search, and a novel, more computationally efficient algorithm called SuperParent. The SuperParent algorithm significantly outperforms NB, but it involves relatively high time complexity. Qiu et al. [14] investigated the class probability estimation performance of SuperParent in terms of Conditional log likelihood (CLL).

*k*-Dependence Bayesian classifier (KDB) [15] is another famous improvement to NB. It relaxes NB's independence assumption by allowing each attribute to have a maximum of *k* attributes as parents. In this sense, NB is a 0-dependence Bayesian classifier and TAN is a 1-dependence Bayesian classifier. By increasing the value of *k*, KDB can generalize to higher degrees of attribute dependence than TAN. KDB constructs classifiers at arbitrary values of *k*, while retaining much of the computational efficiency of NB. It is a two pass algorithm. The first pass collects the statistics required for selecting a network structure in which each attribute has at most *k* parents. The second pass computes the conditional probability tables inferred by the structure of *k*-dependence Bayesian network.

Jiang et al. [16] proposed a novel Bayes model: Hidden Naive Bayes (HNB), in which a hidden parent was created for each attribute which combined the influences from all other attributes. The experimental results show that HNB significantly outperforms other improvements of NB.

Another significant improvement to NB is A$n$DE [8], which relaxes the attribute independence assumption and averages over all possible $n$-dependence estimators ($n$DE), with the aim of reducing the inductive bias in the classifier. Martínez et al. [17] argued that the idea of non-disjoint discretization, already justified in NB classifiers, could also be profitably extended to AODE.

### 2.2 Attribute Weighting

Jiang and Zhang [18] first proposed the idea to assign each attribute a different weight in AODE. Jiang et al. [19] further designed four different weighting approaches and created four different versions of weighted AODE (WAODE). These weighting approaches include mutual information, classification accuracy, conditional log likelihood (CLL) and area under the ROC curve (AUC). Wu et al. [20] proposed a self-adaptive SPODEs, namely SODE, which used immunity theory in artificial immune systems to automatically and self-adaptively select the weight for each single SPODE.

### 2.3 Attribute Selection

Zheng et al. [21] proposed attribute selection approaches for AODE, such as backwards sequential elimination (BSE)

and forward sequential selection (FSS). These approaches require multiple passes through the training data, so they are not feasible on large data sets. Yang et al. [22], [23] compared attribute selection and weighting techniques in AODE. Zheng and Webb [24] explored Lazy Elimination (LE), later called subsumption resolution (SR) [25], which eliminated highly related attribute-values at classification time without the computational overheads inherent in wrapper techniques. It can be considered as a kind of attribute selection technique.

Zaidi and Webb [4] generalized weighting and subsumption resolution ideas from AODE to the more general case of A*n*DE.

## 3 Averaged *n*-Dependence Estimators

In this section, we first describe the probabilistic model of A*n*DE. Then we present the algorithm to learn the parameters. Finally, we show how to predict a new example with A*n*DE model.

### 3.1 Probabilistic Model of A*n*DE

A*n*DE can be modelled as a set of Bayesian networks $\mathcal{B}_i (i = 1, \ldots, d)$, each one is corresponding to an *n*DE. Each $\mathcal{B}_i = \langle \mathcal{G}, \Theta \rangle$ consists of a directed acyclic graph $\mathcal{G}$ and a set of parameters $\Theta$. $\mathcal{G} = (\langle \mathbf{X}, C \rangle, \mathbf{E})$ is composed of a set of nodes $\langle \mathbf{X}, C \rangle$ and a set of directed edges $\mathbf{E}$ connecting the nodes, where $C$ is the class node. A*n*DE makes two important assumptions. One is that each node depends not only on the class node $C$, but on $n$ common parent nodes in each network $\mathcal{B}_i$. The other is that all the nodes can be the parent nodes in turn. Specifically, AODE assumes all nodes depend on the class node and one common parent node. Fig. 1 shows an example of a set of networks for AODE with 3 attributes. From left to right, the parent nodes are $X_1$, $X_2$ and $X_3$, respectively.

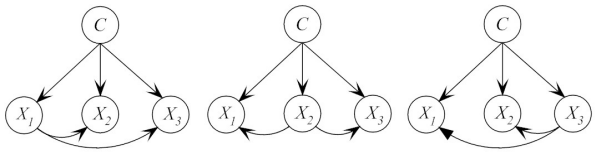



Fig. 1. An example of a set of networks for AODE with 3 attributes

The set of parameters $\Theta$ is used to quantify the network $\mathcal{B}_i$. Each node $X_i$ is represented as a local conditional probability distribution given its parents $\mathbf{\Pi}$. Here the subscript $i$ for $\mathbf{\Pi}$ is omitted because A*n*DE assumes all nodes depend on the same parent nodes in each $\mathcal{B}_i$. A specific conditional probability table entry $\theta^i_{j|h}$ is used to denote the probability that variable $X_i$ takes on its $j^{th}$ value assignment given that its parent take their $h^{th}$ assignment, i.e. $\theta^i_{j|h} = P_\Theta(X_i = j | \mathbf{\Pi} = h)$.

### 3.2 Parameters Learning

There are two paradigms for learning the parameters: Generative and discriminative methods [26]. The goal of generative learning is to find the parameters that best represent the sample distribution [27] . One such approach is to find the parameters that maximize the likelihood of the entire data or (more conveniently) its logarithm.

Given the training data set $\mathcal{T} = \{\langle \mathbf{x}^t, y^t \rangle\}_{t=1}^T$, the joint probability distribution of a sample $\mathbf{x}^t$ is,

$$P_\Theta(\mathbf{X} = \mathbf{x}^t) = \prod_{i=1}^{d} (\theta^i_{j|h}) = \prod_{i=1}^{d} \prod_{j=1}^{|X_i|} \prod_{h} (\theta^i_{j|h})^{\mu^{i,t}_{j|h}}, \quad (2)$$

where the indicator function $\mu^{i,t}_{j|h}$ is 1 for $x^t_i = j$ and $x^t_{\Pi_i} = h$, and is 0 elsewhere. The log likelihood function of a fixed structure of $\mathcal{B}_i$ is

$$LL(\mathcal{B}_i | \mathcal{T}) = \sum_{t=1}^{T} P_\Theta(\mathbf{X} = \mathbf{x}^t) = \sum_{t=1}^{T} \sum_{i=1}^{d} \sum_{j=1}^{|X_i|} \sum_{h} \mu^{i,t}_{j|h} \log(\theta^i_{j|h}), \quad (3)$$

Maximizing $LL(\mathcal{B}_i | \mathcal{T})$ leads to the maximum likelihood (ML) estimate of the parameters,

$$\theta^i_{j|h} = \frac{m^i_{j|h}}{m^i_h}, \quad (4)$$

where $m^i_{j|h} = \sum_{t=1}^{T} \mu^{i,t}_{j|h}$ denotes the number of occurrences in the training set of the $j^{th}$ state of $X_i$ given the $h^{th}$ state of its parent, and $m^i_h = \sum_{t=1}^{T} \sum_{j=1}^{|X_i|} \mu^{i,t}_{j|h}$ denotes the sum of $m^i_{j|h}$ over all $j$.

Based on A*n*DE's two assumptions, $m^i_{j|h}$ is actually the joint frequency of each possible combination of $n+1$ attribute values and the class labels. As a result, we should form an $(n+2)$-dimensional frequency table in the process of training. For example, Algorithm 1 depicts the training process of AODE. Note that here we store only the observed counts of each combination of 2 attributes and the class label. With these data we can easily compute the frequencies of each combination when necessary. This process in A2DE is similar. The only difference lies in that we need to store the counts of each combination of 3 attributes and the class label. The restriction that $X_2$ precedes $X_1$ can help to save memory.

**Algorithm 1** Training process of AODE.

```
Count : vector of observed counts of combination of 2 attribute values and the class label
for instance inst ∈ T do
    y = value of class label in inst
    for X1 ∈ X do
        x1 = value of attribute X1 in inst
        for X2 ∈ X AND X2 precedes X1 do
            x2 = value of attribute X2 in inst
            increase the element in Count with index (X1, x1, X2, x2, y) by 1
        end for
    end for
end for
```

### 3.3 Predicting New Examples

When we have a new example, the class labels are predicted by first computing the maximum a-posteriori (MAP) estimates on each $\mathcal{B}_i$ and then averaging them. To be more spe- [c3.1]

cific, the joint probability $\mathrm{P}(y, \mathbf{x})$ for some $\mathcal{B}_i$ is calculated as follows,

$$\mathrm{P}_{n\mathrm{DE}}(y, \mathbf{x}) = \mathrm{P}(y, \pi) \prod_{i=1}^{d} \mathrm{P}(x_i \mid y, \pi), \tag{5}$$

where $\pi$ is the set of values of attributes in $\mathbf{\Pi}$ corresponding to example $\mathbf{x}$. When trying to select the parent set $\mathbf{\Pi}$ of size $n$ from $d$ attributes, we have $C(d, n) = d!/(n!(d-n)!)$ possible options. For every eligible set of parents, we have one $n$DE model. The average across all eligible models gives a final probability. So the joint probability in A$n$DE is calculated by

$$\mathrm{P}_{\mathrm{A}n\mathrm{DE}}(y, \mathbf{x}) = \frac{\sum_{\mathbf{\Pi}: F(\pi) \geq m} \mathrm{P}(y, \pi) \prod_{i=1}^{d} \mathrm{P}(x_i \mid y, \pi)}{C(d, n) - |\{\mathbf{\Pi} : F(\pi) < m\}|}, \tag{6}$$

where $\mathbf{\Pi}$ ranges over all size-$n$ subsets of attributes, $F(\pi)$ is the frequency of $\pi$, $m$ is the minimum frequency to accept $\pi$ as a parent tuple and $|\cdot|$ denotes the cardinality of a set. The current research uses $m = 1$ [18], [28], [29].

The space complexity of the frequency table is $\mathcal{O}(cC(d, n+1)v^{n+1})$, where $v$ is the average number of values per attribute. The time complexity of compiling it is $\mathcal{O}(tC(d, n+1))$, as we need to update each entry for every combination of the $n+1$ attribute-values for every instance.

It is evident that A$n$DE has linear time complexity with respect to the number of training examples, which allows single pass learning through the training examples and makes out-of-core learning for large data sets possible.

## 4 Attribute Selection

We can see from Eq. (6) that A$n$DE averages across all eligible $n$DE, where each $n$DE assume that all children attributes depend on some parent tuple. These design choices have been made to gain computational efficiency and control variance, but they also serve to increase inductive bias. Previous research has shown that attribute selection can reduce A$n$DE's bias [21], [22]. This inspires us to perform attribute selection in A$n$DE.

Furthermore, we could observe from Eq. (6) that the joint probability is the sum of products of conditional probabilities and the computation process actually contains multiple approximations to $\mathrm{P}_{\mathrm{A}n\mathrm{DE}}(y, \mathbf{x})$. These observations imply that it is possible to nest a large space of alternative approximate models such that each one is a trivial extension to another. Importantly, multiple models that build upon one another in this way can be efficiently evaluated in a single set of computations. Using these observations, we create a space of models that are nested together, and then select the best model using leave-one-out cross validation in a single extra pass through the training data. Consequently, our purpose in attribute selection could be achieved by selecting the best model.

In this section, we first build the model space. Then we rank the attributes by mutual information. Next we select the best model using leave-one-out cross validation error. At the same time, we propose to sample the data in the second pass through the training data to accelerate the training process. Finally, we summarize the algorithm and present the complexity analysis.

### 4.1 Building the Model Space

#### 4.1.1 Model Space for AODE

For AODE, the joint probability is calculated by:

$$\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x}) = \frac{\sum_{j:1 \leq j \leq d \wedge \mathrm{F}(x_j) \geq m} \mathrm{P}(y, x_j) \prod_{i=1}^{d} \mathrm{P}(x_i \mid y, x_j)}{|\{j : 1 \leq j \leq d \wedge \mathrm{F}(x_j) \geq m\}|}. \tag{7}$$

From Eq. (7), we could see that at most $d^2$ nested submodels of attribute subsets will be created when calculating $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})$. To be more specific, suppose we select the former $r$ attributes as parents and the former $s$ attributes as children, where $1 \leq r, s \leq d$, the approximate AODE model would be,

$$\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{r,s} = \frac{\sum_{j:1 \leq j \leq r \wedge \mathrm{F}(x_j) \geq m} \mathrm{P}(y, x_j) \prod_{i=1}^{s} \mathrm{P}(x_i \mid y, x_j)}{|\{j : 1 \leq j \leq r \wedge \mathrm{F}(x_j) \geq m\}|}. \tag{8}$$

This assumes that there is an ordering on the attributes. By default A$n$DE uses the order in which attributes are presented in the data, because without attribute selection the order has no effect. However, it has been shown that orderings that place more predictive attributes first are preferable for attribute selection [30], we will rank the attribute in Section 4.2.

All these approximate AODE models form the model space as depicted in Fig. 2. Because each model is only a minor extension to previous model, for instance, $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{1,2}$ is obtained by adding child attribute $x_2$ to $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{1,1}$, all these models can be applied to a test instance in a single nested computation. Consequently all models can be efficiently evaluated.

| parent | children | | | | |
|---|---|---|---|---|---|
| | $x_1$ | ... | $x_s$ | ... | $x_d$ |
| $x_1$ | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{1,1}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{1,s}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{1,d}$ |
| $x_2$ | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{2,1}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{2,s}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{2,d}$ |
| ... | ... | ... | ... | ... | ... |
| $x_r$ | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{r,1}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{r,s}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{r,d}$ |
| ... | ... | ... | ... | ... | ... |
| $x_d$ | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{d,1}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{d,s}$ | ... | $\mathrm{P}_{\mathrm{AODE}}(y, \mathbf{x})_{d,d}$ |

Fig. 2. Space of approximate models of AODE with $d$ attributes.

Figure 3 gives an example of the model space with 3 attributes. For instance, model $m_{22}$ considers two attributes $\{x_1, x_2\}$ as parents and two attributes $\{x_1, x_2\}$ as children. Then, when attribute $x_3$ is added as a child, we obtain a new model $m_{23}$. When instead attribute $x_3$ is added as a parent, we obtain a new model $m_{32}$. Both of these models are minor extensions to the existing model $m_{22}$ and all three (and all their extensions) can be applied to a test instance in a single nested computation. Consequently all models can be efficiently evaluated in a single set of nested computations.

| | | children | | |
|---|---|---|---|---|
| | | $\{x_1\}$ | $\{x_1, x_2\}$ | $\{x_1, x_2, x_3\}$ |
| parents | $\{x_1\}$ | $m_{11}$ | $m_{12}$ | $m_{13}$ |
| | $\{x_1, x_2\}$ | $m_{21}$ | $m_{22}$ | $m_{23}$ |
| | $\{x_1, x_2, x_3\}$ | $m_{31}$ | $m_{32}$ | $m_{33}$ |

Fig. 3. An example of the model space with 3 attributes.

#### 4.1.2 Model Space for A2DE

In order to describe the formulation of A2DE conveniently, we define

$$x_{\langle p,q\rangle} = \langle x_p, x_q\rangle \tag{9}$$

So from Eq. (6), we can obtain the joint probability of A2DE,

$$\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x}) = \frac{\sum_{1\le q<p\le d\wedge \mathrm{F}(x_{\langle p,q\rangle})\ge m} \mathrm{P}(y, x_{\langle p,q\rangle}) \prod_{i=1}^{d} \mathrm{P}(x_i \mid y, x_{\langle p,q\rangle})}{C(d,2) - |\{\langle p,q\rangle : F(x_{\langle p,q\rangle}) < m\}|}. \tag{10}$$

From Eq. (10), we could see that we have at most $C(d,2) = d(d-1)/2$ parent tuples. This number of parent tuples is denoted by $M$. For any specific parent tuple $x_{\langle p,q\rangle}(1 \le q < p \le d)$, its index in all parent tuples is $(p-1)(p-2)/2+q$, denoted by $r$. For each parent tuple $x_{\langle p,q\rangle}$, there are $d$ children attributes $x_i$ $(i = 1,\ldots,d)$.

During the process of computation of $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})$, it forms $d^2 \times (d-1)/2$ approximate models to $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})$. To be more specific, for the former $r$ parent attribute tuples and the former $s$ children attributes, where $1 \le r \le d(d-1)/2,\ 1 \le s \le d$, the approximate model would be

$$\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{r,s} = \frac{\sum_{((p-1)(p-2)/2+q)\le r\wedge \mathrm{F}(x_{\langle p,q\rangle})\ge m} \mathrm{P}(y, x_{\langle p,q\rangle}) \prod_{i=1}^{s} \mathrm{P}(x_i \mid y, x_{\langle p,q\rangle})}{r - |\{\langle p,q\rangle : F(x_{\langle p,q\rangle}) < m\}|}. \tag{11}$$

By this means, we obtain a model space as is shown in Fig. 4. Each model corresponds to a certain selection of parent attributes and children attributes. They could also be evaluated efficiently. We could see that $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{M,d}$ is actually $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})$. Consequently, so long as the best model is selected, it is guaranteed that the performance of the selected model is no worse than A2DE. By restricting ourselves to the $d^2 \times (d-1)/2$ nested models we support very efficient simultaneous evaluation of all models, at the cost that we exclude all models that are not in this space of nested models.

### 4.2 Ranking the Attributes

As is demonstrated in Fig. 2 and Fig. 4, models containing attributes that are later in the ordering will be built upon models containing earlier attributes. Therefor this method for nesting models depends on an ordering of the attributes.

The mutual information between an attribute $X$ and the class $Y$ is defined as:

$$\begin{aligned} I(X,Y) &= H(X) - H(X \mid Y) \\ &= \sum_{y\in Y}\sum_{x\in X} \mathrm{P}(x,y) log_2 \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)}, \end{aligned} \tag{12}$$

where $H(X) = -\sum_{x\in X} \mathrm{P}(x) log \mathrm{P}(x)$ is the entropy of $X$, and $H(X \mid Y) = -\sum_{y\in Y} \mathrm{P}(y) \sum_{x\in X} \mathrm{P}(x \mid y) log \mathrm{P}(x \mid y)$ is the conditional entropy. This mutual information measures how informative this attribute is about the class [31], as such it is a suitable metric to rank the attributes.

An advantage of using mutual information is that it can be computed very efficiently after one pass learning through the training data. Although the mutual information between an attribute and the class can help to identify the attributes that are individually most discriminative, it is important to note that it does not directly assess the discriminative power of an attribute in combination with other attributes. Nevertheless, the ranking of attributes based on mutual information with the class will permit the search over a large space of possible models and the deficiencies of this discriminative approach will be mitigated by the richness of the search space that is evaluated in a discriminative fashion.

### 4.3 Selecting the Best Model

To evaluate the discriminative ability of alternative models and avoid over fitting on training data, we use leave-one-out cross validation error as the evaluation criterion [32], [33]. Rather than building new models for every fold, we exploit incremental cross validation [34], in which the contribution of the training example being left out in each fold is simply subtracted from the frequency table, thus producing a model without that training example. This method not only obtains a low-bias estimate of the generalization error, but also allows the models to be evaluated in one pass through the training data.

In addition, the fact that the models are nested together such that each one is a trivial extension to another, as is shown in Eq. 8 and Eq. 11, provides us a way to efficiently evaluate these models. That is to say, for the training example being left out in each fold, these models can be simultaneously evaluated inside the process of construction of them. The process of leave-one-out cross validation has been demonstrated in Algorithm 2 (line 6-14).

There are several loss functions to measure model performance for leave-one-out cross validation, zero-one loss and root mean squared error (RMSE) are among the most common and effective. Zero-one loss simply assigns a loss of '0' to correct classification, and '1' to incorrect classification, treating all misclassifications as equally undesirable. RMSE, however, accumulates for each example the squared error, where the error is the difference between 1.0 and the probability estimated by the algorithm for the true class for the example, and then computes the squared root of the mean of the sum. This could be computed as,

$$E_{rmse} = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(1 - P(y = y_i \mid \mathbf{x}_i))^2}\ , \tag{13}$$

where $y_i$ is the true class for the example $\mathbf{x}_i$. As RMSE gives a finer grained measure of the calibration of the probability estimates compared to zero-one loss, with the error depending not just on which class is predicted, but also on the probabilities estimated for each class, we use RMSE to evaluate the candidate models in this research.

Consequently, selecting the best model can be described as the following optimization problem,

$$\langle r,s\rangle^* = \underset{\langle r,s\rangle}{\mathrm{argmax}} \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(1 - \mathrm{P}^{LOO}_{\mathrm{AnDE}}(y = y_t \mid \mathbf{x}_t)_{r,s}\right)^2} \tag{14}$$

| index | parent tuple | children | | | | |
|---|---|---|---|---|---|---|
| | | $x_1$ | ... | $x_s$ | ... | $x_d$ |
| 1 | $x_{\langle 2,1\rangle}$ | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{1,1}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{1,s}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{1,d}$ |
| 2 | $x_{\langle 3,1\rangle}$ | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{2,1}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{2,s}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{2,d}$ |
| 3 | $x_{\langle 3,2\rangle}$ | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{3,1}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{3,s}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{3,d}$ |
| ... | ... | ... | ... | ... | ... | ... |
| r | $x_{\langle p,q\rangle}$ | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{r,1}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{r,s}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{r,d}$ |
| ... | ... | ... | ... | ... | ... | ... |
| $M^*$ | $x_{\langle d,d-1\rangle}$ | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{M,1}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{M,s}$ | ... | $\mathrm{P}_{\mathrm{A2DE}}(y,\mathbf{x})_{M,d}$ |

$^*$ $M = d(d-1)/2$

Fig. 4. Space of approximate models of A2DE with $d$ attributes.

where $\mathrm{P}^{LOO}_{\mathrm{AnDE}}(y \mid \mathbf{x}_t)_{r,s}$ can be computed by first estimating $\mathrm{P}^{LOO}_{\mathrm{AnDE}}(y,\mathbf{x}_t)_{r,s}$ from training set $(\mathcal{T} - \{\langle y_t, x_t\rangle\})$ as in Eq. 6 or Eq.9, and then normalizing across all possible $y$.

### 4.4 Sampling the Data in the Second Pass

As we need to evaluate $d^2$ alternative models for AODE or $d^2 \times (d-1)/2$ alternative models for A2DE for each example in the second pass through the training data, the computing time is high for this model selection stage when processing high dimension $d$ and large data number $t$ in large data. In order to reduce the time requirement of this method, we use only a sample of size $S$ to select among models. Since we know the number of examples after the first pass, it is straightforward to use uniform sequential fixed-sized sampling without replacement (Algorithm 2, line 7-13).

It is worthwhile to clarify this sampling technique as follows:

1) The sampling process is used to select the best among alternative models, but not to train the parameters of the model. The latter will deteriorate the performance greatly as indicated in Table 5. But the former will save training time greatly without significant performance loss. Consequently, the sampling process is an indispensable part in our framework.
2) Using only a sample to select among models introduces the possibility that the quality of the model selected might not be as good as if all training data were used. However, the leave-one-out cross validation process that we use for model selection has low variance and hence is reasonably accurate even when using only a sample of the data. As we will see from the experiments, the accuracy decrease in practice is acceptable.
3) It might be thought that using a sample is contrary to our objective of extracting as much value from large data as possible by using low-bias algorithms on the full dataset. However, this does not follow. A*n*DE's low bias is obtained from the probability tables constructed in the first pass that are able to extract fine detail about complex high-dimensional interactions in the data. In theory the use of a sample to select between low-bias models each developed from all the data should not affect the quality of those models, and if the sample is sufficiently large should provide effective model selection.

### 4.5 Algorithm

Based on the methodology presented above, we develop the training algorithm for sample-based attribute selective A*n*DE shown in Algorithm 2.

**Algorithm 2** Training algorithm for sample-based attribute selective A*n*DE.

1: $S$: sample size, $t$: number of training examples
2: $selected \leftarrow 0$, $i \leftarrow 0$
3: Form the table of joint frequencies of all combinations of $n$ attribute values and the class label as in Algorithm 1 ▷ first pass through training data
4: Compute the mutual information
5: Rank the attributes
6: **for** instance $inst \in \mathcal{T}$ **do** ▷ second pass through training data
7: **with** probability $(S - selected)/(t - i)$ **do**
8: Build $d^2$ models for AODE or $d^2 \times (d-1)/2$ models for A2DE
9: Predict $inst$ using all models
10: Accumulate the squared error for each model
11: increment $selected$
12: **end with**
13: increment $i$
14: **end for**
15: Compute the root mean squared error for each model
16: Select the model with the lowest RMSE

As in AODE or A2DE, we need to form the table of joint frequencies of attributes values and the class label from which the probability estimates $\mathrm{P}(y, x_j)$, $\mathrm{P}(x_i \mid y, x_j)$, or $\mathrm{P}(y, x_{\langle p,q\rangle})$, $\mathrm{P}(x_i \mid y, x_{\langle p,q\rangle})$, and the mutual information between the attributes and class are derived. This is done in one pass through the training data (line 3). Note that this provides all the information needed to create any selective A*n*DE model with any sets of parent and child attributes.

In the second pass through the training data (line 6-14), the squared error is accumulated for each model using incremental leave-one-out cross validation [34]. Incremental leave-one-out cross validation simply develops a model from the full data and then selects each training example in turn and removes it before classifying and then restoring it. The addition and removal of an example from an A*n*DE model is extremely efficient. In consequence, the computation is dominated by the classification of the holdout examples. After this pass, the RMSE will be computed and used to select the best model.

### 4.6 Complexity Analysis

From the training process in Algorithm 2, we could see that the space complexity of the table of joint frequencies of all

combinations of $n$ attributes values and the class label is $\mathcal{O}(c(dv)^2)$ for AODE or $\mathcal{O}(c(dv)^3)$ for A2DE, where $v$ is the average number of values per attribute. Attribute selection will not require more memory. The time complexity consists of two parts. One is derivation of the frequencies required to populate the table, the time complexity of which is $\mathcal{O}(td^2)$ for AODE or $\mathcal{O}(td^3)$ for A2DE. The other is attribute selection in a second pass through the training data, the time complexity of which for AODE is $\mathcal{O}(tcd^2)$, since for each example we need to compute the joint probability in Eq. (7). The time complexity of attribute selection for A2DE is $\mathcal{O}(tcd^3)$ as the frequency of the base operation is $tcd^2(d-1)/2$ when we compute the joint probability in Eq. (10) for each class. So the overall time complexity is $\mathcal{O}(tcd^2)$ for AODE and $\mathcal{O}(tcd^3)$ for A2DE. If sampling is used for attribute selection, the complexity changes to $O(Scd^2)$ and $\mathcal{O}(Scd^3)$, where $S$ is the sample size and $S < t$.

Classification requires the table of joint frequencies formed at training time of space complexity $\mathcal{O}(c(dv)^2)$ for AODE or $\mathcal{O}(c(dv)^3)$ for A2DE. The time complexity of classifying a single example is $\mathcal{O}(cd^2)$ for AODE and $\mathcal{O}(cd^3)$ for A2DE in the worst-case scenario, because some attributes may be omitted after attribute selection.

## 5 Experiments

In this section, we first describe the experiments setting. Then we evaluate the impact of sample size in sample-based attribute selective AODE (SASAODE) and compare different sampling strategies in SASAODE. Next, we present the RMSE, zero-one loss, and negative conditional log likelihood comparisons of sample-based attribute selective AODE and A2DE. Finally we compare attribute selective AODE (ASAODE) and sample-based selective A2DE with typical in-core Bayesian network classifiers and state-of-the-art Random Forest [35].

### 5.1 Experiments Setting

As the algorithms are proposed for large data, we undertake an extensive online search to gather a group of large datasets, all of which have more than 100k instances. These are all the publicly available datasets we could find. The detailed source of each data set has been indicated in Table 2. From left to right, we present the following characteristics of each data set: name, number of instances, number of attributes, number of classes, source, description. Note that the data sets have been ranked in ascending order of number of instances.

All datasets except `poker-hand`, `uscensus1990` and `splice` contain one or more numeric attributes. 6 datasets contain only numeric attributes: `MITFaceSetA`, `MITFaceSetB`, `MITFaceSetC`, `USPSExtended`, `MSDYearPrediction` and `satellite`. We discretize these numeric attributes using 5-bin equal frequency discretization (EF5). We have observed that EF5 and Minimal Description Length (MDL) [42] discretization provide the best results in approximately half of the datasets each. In fact, the discretization method does not matter if the group of data sets is large enough [43]. EF5 has been chosen because it is faster than MDL, and also because it is not supervised and hence does not potentially provide the classifier with class information from the holdout data when used for pre-discretization. Using a pre-fixed number of bins gives us another advantage of not having to deal with a huge number of values per attribute as in MDL discretization in some cases. To avoid loading the whole data into memory, only a sample of 100k examples is used to define the bins for discretization.

We run the experiments on a C++ system which is specially designed for out-of-core learning[1]. It has the following characteristics:

1) It supports out-of-core learning, which means it can fetch an instance one time from the disk. This could address the problem that large data sets could not be loaded into memory entirely.
2) It provides the ability to flexibly set the number of learning passes through training data.
3) It supports 10-fold cross validation and other running modes.

The base probabilities are estimated using $m$-estimation ($m = 1$) [18], [28], [29]. Missing values have been considered as a distinct value. Note that the root mean square error is calculated exclusively on the true class label. This is different from Weka's implementation [44], where all class labels are considered. All the experiments have been done by 10-fold cross validation.

We present detailed RMSE, zero-one loss (ZOL) and negative conditional log likelihood (nCLL) [27] results of comparing algorithms in Table 9, Table 11 and Table 12 in the Appendix. In order to give the results a more intuitionistic explanation, we present also summaries of win/draw/loss records of alternative algorithms, for example in Table 3, which indicates the number of data sets on which one algorithm has lower, equal or higher outcome relative to the other. Each entry compares the algorithm in the row against the algorithm in the column. We perform two-tailed binomial sign test to assess the probability of observing the given number of wins and losses if each were equally likely. We consider a difference to be significant if the probability is less than or equal to 0.05. All such entries have been changed to boldface in the table.

### 5.2 Impact of the Sample Size in SASAODE

Attribute selective AODE (ASAODE) has been tested on small data sets in [11]. In order to obtain an understanding of the performance of ASAODE and sample-based ASAODE (SASAODE) on large data sets and evaluate the impact of sample size $S$, we run ASAODE and SASAODE on 15 large data sets.

We set the sample size $S$ to 1k, 5k, 10k, 20k, 50k, 100k and 200k respectively, so we get 7 different SASAODE classifiers. We evaluate the impact of the sample size with respect to only RMSE. Table 3 presents win/draw/loss records of these algorithms.

We could see from Table 3 that ASAODE reduces RMSE significantly often relative to AODE. As for sample-based ASAODE, all SASAODE classifiers achieve lower RMSE

1. http://i.giwebb.com/software/sasande.zip

TABLE 2
Data sets used for experiments[1]

| No. | Name | ♯Inst | ♯Att | ♯Class | Source | Description |
|---|---|---|---|---|---|---|
| 1 | localization | 164860 | 5 | 11 | UCI [36] | Recordings of 5 people performing different activities. Each person wore 4 sensors while performing the same scenario 5 times. |
| 2 | census-income | 299285 | 41 | 2 | UCI [37] | Weighted census data extracted from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau. |
| 3 | USPSExtended | 341462 | 676 | 2 | CVM [38] | 0/1 digit classification (extended version of the USPS data set). |
| 4 | MITFaceSetA | 474101 | 361 | 2 | CVM [38] | Face detection using an extended version of the MIT face database[c]. By adding nonfaces to the original training set. |
| 5 | MITFaceSetB | 489410 | 361 | 2 | CVM [38] | Each training face is blurred and added to set A. They are then flipped laterally. |
| 6 | MSDYearPrediction | 515345 | 90 | 90 | UCI [37] | 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. |
| 7 | covertype | 581012 | 54 | 7 | UCI [37] | Predicting forest cover type from cartographic attributes only (no remotely sensed data). |
| 8 | MITFaceSetC | 839330 | 361 | 2 | CVM [38] | Each face in set B is rotated. |
| 9 | poker-hand | 1025010 | 10 | 10 | UCI [37] | 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. |
| 10 | uscensus1990 | 2458285 | 67 | 4 | UCI [37] | Discretized version of the USCensus1990raw dataset, a 1% sample from the full 1990 census. 'Temp. Absence From Work' has been selected as class. |
| 11 | PAMAP2 | 3850505 | 54 | 19 | UCI [39] | 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. |
| 12 | kddcup | 5209460 | 41 | 40 | UCI [37] | 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. |
| 13 | linkage | 5749132 | 11 | 2 | [40] | 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. |
| 14 | satellite | 8705159 | 138 | 24 | [41] | Satellite image time series to predict land cover. |
| 15 | splice | 54627840 | 141 | 2 | [1] | Recognising a human acceptor splice site (largest public data for which subsampling is not an effective learning strategy). |

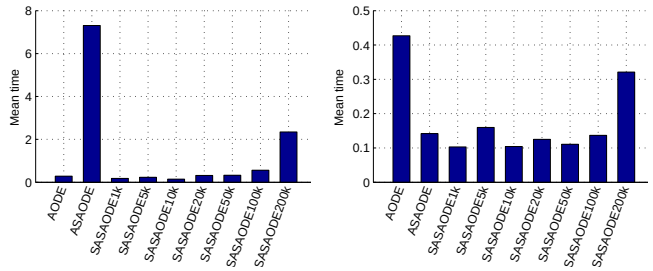
[1] The data sets are ranked in ascending order of number of instances and the appendix gives the results for individual datasets for those who wish to consider the effects of different factors on the outcomes.

TABLE 3
Win/Draw/Loss of RMSE for SASAODE

| | AODE | ASAODE | SASAODE1k | SASAODE5k | SASAODE10k | SASAODE20k | SASAODE50k | SASAODE100k |
|---|---|---|---|---|---|---|---|---|
| ASAODE | **14/1/0** | | | | | | | |
| SASAODE1k | 11/0/4 | **0/0/15** | | | | | | |
| SASAODE5k | **13/1/1** | **0/1/14** | **14/0/1** | | | | | |
| SASAODE10k | **14/0/1** | **0/1/14** | **15/0/0** | **10/4/1** | | | | |
| SASAODE20k | **14/0/1** | **0/4/11** | **14/0/1** | **12/3/0** | 10/2/3 | | | |
| SASAODE50k | **14/0/1** | **1/6/8** | **15/0/0** | **13/2/0** | **11/3/1** | **8/7/0** | | |
| SASAODE100k | **14/0/1** | **0/6/9** | **15/0/0** | **13/2/0** | **12/3/0** | **9/6/0** | 4/10/1 | |
| SASAODE200k | **14/1/0** | 1/10/4 | **15/0/0** | **13/2/0** | **13/2/0** | **10/5/0** | 6/8/1 | 6/8/1 |

more often than AODE, not significantly so only when comparing SASAODE1k to AODE. As expected, all SASAODE classifiers deliver higher RMSE significantly more often than ASAODE. This illustrates the weakness of the sample technique. When comparing among SASAODE classifiers, we see that increasing $S$ from 1k to 200k consistently decreases RMSE. That is to say, with the number of the instances increasing, the performance will be enhanced. This indicates that we should sample more data to select the model.

Nevertheless, we should also consider the computation time. From Fig. 5, We could see that ASAODE needs much more training time than AODE. This is because ASAODE needs one more pass through the training data and it assesses many models during this pass. SASAODE saves much training time compared to ASAODE due to the sample technique. The training time of SASAODE consistently increases as the sample size $S$ increases. Both ASAODE and SASAODE reduce the classification time as they need less attributes.




(a) Training time (b) Classification time

Fig. 5. Computation time comparison of SASAODE (hours).

TABLE 4
Average percentages of parent and children selected across 15 data sets(%)

| | parent | children |
|---|---|---|
| AODE | 100 | 100 |
| ASAODE | 28.98 | 60.66 |
| SASAODE1k | 26.66 | 56.46 |
| SASAODE5k | 28.44 | 59.29 |
| SASAODE10k | 27.44 | 57.79 |
| SASAODE20k | 26.32 | 56.53 |
| SASAODE50k | 27.77 | 56.95 |
| SASAODE100k | 28.32 | 58.64 |
| SASAODE200k | 28.88 | 60.08 |

In Table 10, we also present the details of the number of parents and children selected for each algorithm. In order to give an overall idea, we compile the average percentage of parents and children selected across 15 data sets in Table 4. We could see that ASAODE selects only $28.98\%$ attributes as parents and 60.66% attributes as children. SASAODE selects attributes as parents and children almost the same as ASAODE. These data illustrate the effectiveness of attribute selection in ASAODE and the reason why ASAODE and SASAODE reduce the classification time.

Given the trade off between accuracy and computation time, we set the sample size to 20k and 50k in the next comparisons of attribute selective AODE and A2DE.

### 5.3 Sampling for Selection versus Sampling for Parameterization of the Model

It is interesting that SASAODE200k could achieve comparable performance to ASAODE while it saves much training time. As the sampling technique could also be used in the first pass through the training data for parameterization of the model, we implement an algorithm called SASAODEp200k in which a sample of 200k examples is used both for attribute selection in the second pass and for parameterization of the model in the first pass. We present the comparison result of SASAODEp200k with AODE, ASAODE and SASAODE200k in Table 5 .

TABLE 5
Win/Draw/Loss of RMSE for SASAODEp200k

| | AODE | ASAODE | SASAODE200k |
|---|---|---|---|
| ASAODE | **14/1/0** | | |
| SASAODE200k | **14/1/0** | 1/10/4 | |
| SASAODEp200k | **3/0/12** | **0/1/14** | **0/1/14** |

We could see that SASAODEp200k achieves higher RMSE significantly more often than the other three algorithms. We believe that this is because the accuracy of the probability estimates in the conditional probability tables is critical to accurate posterior probability estimates, while the selection of good network structures requires less fine-grained information. This is the reason why a sample is used for selection but not for parameterization of the model.

### 5.4 Comparison of SASA*n*DE

#### 5.4.1 Comparison of SASAODE

As ASAODE and SASAODE have been demonstrated to be significant improvements to AODE, we would compare them with other improvements to AODE. Just as indicated in [11], AODE with BSE requires much more training time than AODE. So it is not feasible to run AODE with BSE on large data sets. Consequently, we compare ASAODE and SASAODE with weighted AODE (WAODE) and AODE with subsumption resolution (AODESR).

Table 6 presents win/draw/loss records of the above algorithms. We can see both WAODE and AODESR achieve lower RMSE, ZOL and nCLL more often than AODE. But the differences are not significant, except that between AODESR and AODE. Both ASAODE and SASAODE reduce RMSE, ZOL and nCLL significantly often relative to AODE. Compared to WAODE and AODESR, ASAODE and SASAODE reduce RMSE significantly often. ZOL and nCLL measures reveal similar results, although the differences are not significant except those between SASAODE and AODESR with respect to nCLL.

TABLE 6
Win/Draw/Loss of RMSE, ZOL and nCLL for SASAODE

| | | AODE | WAODE | AODESR | ASAODE | SASAODE20k |
|---|---|---|---|---|---|---|
| RMSE | WAODE | 9/0/6 | | | | |
| | AODESR | 6/8/1 | 8/1/6 | | | |
| | ASAODE | **14/1/0** | **14/0/1** | **13/0/2** | | |
| | SASAODE20k | **14/0/1** | **14/0/1** | **13/0/2** | **0/4/11** | |
| | SASAODE50k | **14/0/1** | **14/0/1** | **13/0/2** | **1/6/8** | **8/7/0** |
| ZOL | WAODE | 8/1/6 | | | | |
| | AODESR | 6/8/1 | 8/2/5 | | | |
| | ASAODE | **11/2/2** | 11/1/3 | 10/1/4 | | |
| | SASAODE20k | **12/1/2** | 10/1/4 | 10/1/4 | 1/7/7 | |
| | SASAODE50k | **12/1/2** | 10/1/4 | 10/1/4 | 1/9/5 | 6/8/1 |
| nCLL | WAODE | 11/1/3 | | | | |
| | AODESR | **7/8/0** | 8/1/6 | | | |
| | ASAODE | **13/1/1** | 10/0/5 | 11/0/4 | | |
| | SASAODE20k | **14/0/1** | 11/0/4 | **12/0/3** | 5/2/8 | |
| | SASAODE50k | **14/0/1** | 11/0/4 | **12/0/3** | 6/4/5 | 5/5/5 |

#### 5.4.2 Comparison of SASA2DE

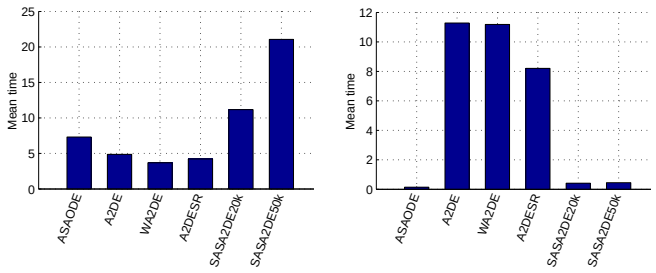
Table 7 presents win/draw/loss records of sample-based attribute selective A2DE (SASA2DE) to ASAODE, A2DE, WA2DE and A2DESR. Note that the sum of win/draw/loss records of A2DE with respect to alternative algorithms is only 13. The reason is that the maximum wall time available for the experiments is 120 hours, but it is not enough for each fold on `satellite` and `splice`. So we get the RMSE results for A2DE on only 13 data sets. We can only compare A2DE with alternative algorithms on 13 data sets. That is why the sum of win/draw/loss records of A2DE with respect to comparing algorithms is 13. That of SASA2DE is 12 accordingly.

TABLE 7
Win/Draw/Loss of RMSE, ZOL and nCLL for SASA2DE

| | | SASAODE | A2DE | WA2DE | A2DESR | SASA2DE20k |
|---|---|---|---|---|---|---|
| RMSE | A2DE | 6/0/7 | | | | |
| | WA2DE | 7/0/6 | **11/0/2** | | | |
| | A2DESR | 6/0/7 | 6/6/1 | 5/0/8 | | |
| | SASA2DE20k | **12/0/0** | **12/0/0** | **10/1/1** | **11/0/1** | |
| | SASA2DE50k | **12/0/0** | **12/0/0** | **11/0/1** | **11/0/1** | **9/2/1** |
| ZOL | A2DE | 6/1/6 | | | | |
| | WA2DE | 6/1/6 | 7/4/2 | | | |
| | A2DESR | 6/1/6 | 5/7/1 | 4/4/5 | | |
| | SASA2DE20k | **12/0/0** | **11/0/1** | 9/0/3 | **10/0/2** | |
| | SASA2DE50k | **12/0/0** | **11/0/1** | 9/0/3 | **10/0/2** | 6/5/1 |
| nCLL | A2DE | 6/0/7 | | | | |
| | WA2DE | 7/1/5 | **11/0/2** | | | |
| | A2DESR | 7/0/6 | **6/7/0** | 6/0/7 | | |
| | SASA2DE20k | **11/0/1** | **11/0/1** | **11/0/1** | **10/0/2** | |
| | SASA2DE50k | **11/0/1** | **11/0/1** | **11/0/1** | **10/0/2** | 6/2/4 |

We can see that SASA2DE obtains lower RMSE, ZOL and nCLL significantly more often than ASAODE, A2DE, WA2DE and A2DESR, except the ZOL between SASA2DE and WA2DE. SASA2DE50k also reduces RMSE, ZOL and nCLL often relative to SASA2DE20k, not significantly so when comparing with respect to ZOL and nCLL. It is also worthwhile to note that ASAODE achieves lower RMSE , ZOL and nCLL almost often as higher than A2DE.

Fig. 6 presents the computation time comparison of SASA2DE. We can see that SASA2DE needs more training time but less classification time than A2DE, WA2DE and A2DESR. ASAODE needs a little more training time, but much less classification time than A2DE. What is more important is that ASAODE requires much less memory than A2DE. This demonstrates that the sample-based attribute selection method by leave-one-out cross validation is a powerful technique.




(a) Training time (b) Classification time

Fig. 6. Computation time comparison of SASA2DE (hours).

### 5.5 Comparison with Typical In-Core Bayesian Network Classifiers

In order to present a thorough understanding of comparison with typical in-core Bayesian network classifiers, we compare ASAODE and SASA2DE ($S$=50k) with NB, TAN and KDB in this section. The proposed technique has also been applied to NB, so we get attribute selective NB (ASNB). The results of ASNB have been included. Table 8 gives the win/draw/loss records of alternative algorithms. Notably, KDB5 denotes KDB when $k$ is set to 5.

TABLE 8
Win/Draw/Loss of RMSE, ZOL and nCLL for Out-of-Core Bayesian Network Classifiers

| | | NB | ASNB | TAN | KDB5 | ASAODE |
|---|---|---|---|---|---|---|
| RMSE | ASNB | **11/4/0** | | | | |
| | TAN | **14/0/1** | **14/0/1** | | | |
| | KDB5 | **14/0/1** | **13/0/2** | **13/0/2** | | |
| | ASAODE | **15/0/0** | **15/0/0** | 6/0/9 | 4/0/11 | |
| | SASA2DE50k | **12/0/0** | **12/0/0** | **10/0/2** | 5/0/7 | **12/0/0** |
| ZOL | ASNB | **11/4/0** | | | | |
| | TAN | **14/0/1** | **12/0/3** | | | |
| | KDB5 | **14/0/1** | **13/0/2** | **14/0/1** | | |
| | ASAODE | **14/1/0** | **12/2/1** | 6/0/9 | **3/0/12** | |
| | SASA2DE50k | **12/0/0** | **12/0/0** | 9/1/2 | 4/0/8 | **12/0/0** |
| nCLL | ASNB | **11/4/0** | | | | |
| | TAN | **14/0/1** | **13/0/2** | | | |
| | KDB5 | **12/0/3** | 11/0/4 | 11/0/4 | | |
| | ASAODE | **15/0/0** | **14/0/1** | 7/0/8 | 6/0/9 | |
| | SASA2DE50k | **12/0/0** | **11/0/1** | **10/0/2** | 5/0/7 | **11/0/1** |

We can see that ASNB, along with TAN, KDB5, ASAODE and SASA2DE, reduces RMSE, ZOL and nCLL significantly often relative to NB. ASAODE achieves lower RMSE, ZOL and nCLL less often than TAN and KDB5, but only the difference of ZOL between ASAODE and KDB5 is significant. SASA2DE50k achieves lower RMSE significantly more often than ASNB and TAN, not significantly so when comparing SASA2DE50k and TAN with respect to ZOL. And SASA2DE50k obtains lower RMSE, ZOL and nCLL almost as often as higher than KDB5, although it needs less parents than KDB5.

### 5.6 Comparison with In-Core Random Forest

Random Forest (RF) [35] is a powerful in-core learning algorithm of the state-of-the-art. Here we compare SASA2DE50k with Random Forest which uses 100 trees (RF100) with respect to RMSE.

From Table 9 , we can see that the win/draw/loss of SASA2DE50k against RF100 is 1/0/10. That is to say, SASA2DE50k has lower RMSE than RF100 on only 1 data set. In order to know how much RF100 wins by, we present the scatter plot in Fig. 7, where the X-axis represents the RMSE results of SASA2DE50k and the Y-axis the RMSE results of RF100.

Note that we do not get the results for both the algorithms on such 4 data sets as `MSDYearPrediction`, `USPSExtended`, `satellite` and `splice`, so we remove these 4 points in the plot. We can see that the dataset uscensus1990 is above the diagonal line, which means SASA2DE50k could beat Random Forest on uscensus1990. Except localization, covertype, pokerhand and PAMP2, the other data sets fall close to the diagonal line. That means the performance of SASA2DE50k is close to RF100 on most data sets. It is worthwhile to keep in mind that SASA2DE50k is a 2-pass out-of-core algorithm in contrast to typical in-core processing of RF.

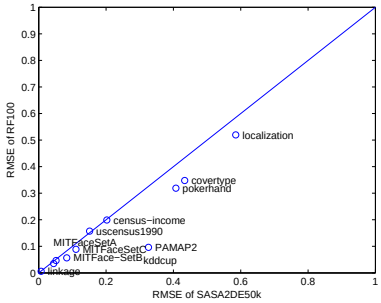



Fig. 7. SASA2DE50k and Random Forest.

### 5.7 Global Comparison

In order to explore the statistical significance of the results, we perform a nonparametric Friedman test followed by Nemenyi post test with respect to RMSE, as advised by Demšar [45] and García and Herrerato [46] to statistically compare multiple algorithms on multiple data sets.

The Friedman test needs to rank the algorithms on different data sets. Since we do not obtain the results of all algorithms on such data sets as `MSDYearPrediction`, `USPSExtended`, `satellite` and `splice`, we remove these data sets. So we rank the algorithms on 11 large data sets. The null hypothesis of the Friedman test is that all algorithms are equivalent. The test result is $p = 0$. This means that at $\alpha = 0.05$, there is an evidence to reject the null hypothesis. So the eight algorithms are not equivalent in terms of RMSE results.

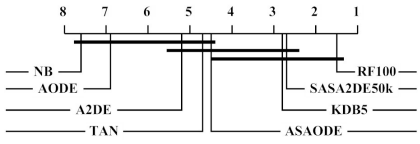



Fig. 8. RMSE comparison of eight algorithms against each other with Nemenyi test.

In order to further explore which algorithm is significantly different to others, we perform a Nemenyi test shown in Fig. 8. The analysis reveals that the RMSE of RF100 is significantly lower than those of NB, AODE, A2DE and TAN, which seem to have equivalent performance. RMSE of SASA2DE50k is significantly lower than those of NB and AODE. There is no consistent evidence to indicate statistical

differences between ASAODE, KDB5, SASA2DE50k and RF100, respectively.

## 6 Conclusion

In this paper, we propose a sample-based attribute selection technique for AODE and A2DE. It performs attribute selection by selecting the most accurate approximate model in one extra pass through the training data in terms of leave-one-out cross validation error. The idea in this framework is different from the greedy strategy search in BSE or FSS in AODE [24]. We create a series of nested submodels of attribute subsets, each being only a minor extension to the previous one. All these models can be efficiently evaluated in one pass learning through the training data. We can not guarantee the selected model is the best in the entire space, but it is acceptable given the accuracy and the training time.

The experiments on 15 large data sets demonstrate that the proposed technique significantly reduces A$n$DE's error at the cost of a modest increase in training time. It delivers lower RMSE, ZOL and nCLL than typical in-core Bayesian classifiers such as NB and TAN, and has comparable error to KDB when $k$ is set to 5. The performance of SASA2DE50k is very close to RF100 on most data sets.

It worthwhile to note that the technique proposed in this paper can also be applied to other Bayesian network classifiers. We leave to future research application of the technique to TAN.

## Appendix

Detailed results of RMSE, ZOL and nCLL are presented in Table 9, Table 11 and Table 12. Average numbers of parents and children selected in ASAODE and SASAODE are in Table 10.

## Acknowledgments

This research has been supported by the Australian Research Council under grant DP140100087, Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under contract FA2386-15-1-4007, National Natural Science Foundation of China under grant 61202135, 61473157, Natural Science Foundation of Jiangsu, China under grant BK20130735, Natural Science Foundation of Jiangsu Higher Education Institutions of China under grant 14KJB520019, 13KJB520011, 13KJB520013, the open project program of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Priority Academic Program Development of Jiangsu Higher Education Institutions.

This research 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. We also want to thank the anonymous reviewers for their insightful comments which have helped to greatly improve the revised version of the paper.

## References

[1] S. Sonnenburg and V. Franc, "COFFIN : A computational framework for linear SVMs," in *Proc. ICML 2010*, 2010.

[2] A. Agarwal, O. Chapelle, M. Dudí, and J. Langford, "A reliable effective terascale linear learning system," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1111–1133, 2013.

[3] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010.

[4] N. A. Zaidi and G. I. Webb, "Fast and effective single pass Bayesian learning," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2013, pp. 149–160.

[5] R. Kohavi and D. H. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufman Publishers, Inc., 1996, pp. 275–283.

[6] D. Brain and G. I. Webb, "The need for low bias algorithms in classification learning from large data sets," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 62–73.

[7] G. I. Webb, J. R. Boughton, and Z. Wang, "Not so naive Bayes: aggregating one-dependence estimators," *Machine Learning*, vol. 58, no. 1, pp. 5–24, 2005.

[8] G. I. Webb, J. R. Boughton, F. Zheng, K. M. Ting, and H. Salem, "Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification," *Machine learning*, vol. 86, no. 2, pp. 233–272, 2012.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, 1st ed. John Wiley & Sons Inc, 1973.

[10] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[11] S. Chen, A. M. Martínez, and G. I. Webb, "Highly scalable attribute selection for averaged one-dependence estimators," in *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 86–97.

[12] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *Information Theory, IEEE Transactions on*, vol. 14, no. 3, pp. 462–467, 1968.

[13] E. J. Keogh and M. J. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches," in *Proceedings of the 7th international workshop on artificial intelligence and statistics*, 1999, pp. 225–230.

[14] C. Qiu, L. Jiang, and C. Li, "Not always simple classification: Learning superparent for class probability estimation," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5433–5440, 2015.

[15] M. Sahami, "Learning limited dependence Bayesian classifiers," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 335–338.

[16] L. Jiang, H. Zhang, and Z. Cai, "A novel Bayes model: Hidden naive Bayes," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 10, pp. 1361–1371, 2009.

[17] A. Martínez, G. I. Webb, M. Flores, and J. Gámez, "Non-disjoint discretization for aggregating one-dependence estimator classifiers," in *Proceedings of the 7th International Conference on Hybrid Artificial Intelligent Systems*. Berlin / Heidelberg: Springer, 2012, pp. 151–162.

[18] L. Jiang and H. Zhang, "Weightily averaged one-dependence estimators," in *PRICAI 2006: trends in artificial intelligence*. Springer, 2006, pp. 970–974.

[19] L. Jiang, H. Zhang, Z. Cai, and D. Wang, "Weighted average of one-dependence estimators," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 24, no. 2, pp. 219–230, 2012.

[20] J. Wu, S. Pan, X. Zhu, P. Zhang, and C. Zhang, "Sode: Self-adaptive one-dependence estimators for classification," *Pattern Recognition*, vol. 51, pp. 358–377, 2016.

[21] F. Zheng and G. I. Webb, "Finding the right family: parent and child selection for averaged one-dependence estimators," in *Machine Learning: ECML 2007*. Springer, 2007, pp. 490–501.

[22] Y. Yang, G. I. Webb, J. Cerquides, K. B. Korb, J. Boughton, and K. M. Ting, "To select or to weigh: a comparative study of linear combination schemes for superparent-one-dependence estimators," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 12, pp. 1652–1665, 2007.

[23] Y. Yang, K. Korb, K.-M. Ting, and G. Webb, "Ensemble selection for superparent-one-dependence estimators," in *Lecture Notes in Computer Science 3809: Advances in Artificial Intelligence, Proceedings of the 18th Australian Joint Conference on Artificial Intelligence (AI*

*2005)*, S. Zhang and R. Jarvis, Eds. Berlin/Heidelberg: Springer, 2005, pp. 102–111.

[24] F. Zheng and G. I. Webb, "Efficient lazy elimination for averaged one-dependence estimators," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 1113–1120.

[25] F. Zheng, G. I. Webb, P. Suraweera, and L. Zhu, "Subsumption resolution: an efficient and effective technique for semi-naive Bayesian learning," *Machine learning*, vol. 87, no. 1, pp. 93–125, 2012.

[26] F. Pernkopf and J. Bilmes, "Discriminative versus generative parameter and structure learning of Bayesian network classifiers," in *Proceedings of the 22Nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 657–664. [Online]. Available: http://doi.acm.org/10.1145/1102351.1102434

[27] D. Grossman and P. Domingos, "Learning Bayesian network classifiers by maximizing conditional likelihood," in *Proceedings of the 21st International Conference on Machine Learning*. ACM, 2004, p. 46.

[28] B. Cestnik, "Estimating probabilities: a crucial task in machine learning," in *ECAI*, vol. 90, 1990, pp. 147–149.

[29] J. Cerquides and R. L. De Mántaras, "Robust Bayesian linear classifier ensembles," in *Machine Learning: ECML 2005*. Springer, 2005, pp. 72–83.

[30] S. Chen, A. M. Martínez, G. I. Webb, and L. Wang, "Selective ande for large data learning: a low-bias memory constrained approach," *Knowledge and Information Systems*, pp. 1–29, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10115-016-0937-9

[31] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[32] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[33] P. Langley and S. Sage, "Induction of selective Bayesian classifiers," in *Tenth International Conference on Uncertainty in Artificial Intelligence*, 1994, p. 399406.

[34] R. Kohavi, "The power of decision tables," in *ECML*, N. Lavrac and S. Wrobel, Eds. Springer, 1995, pp. 174–189.

[35] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[36] B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams, "An agent-based approach to care in independent living," in *Proceedings of the First international joint conference on Ambient intelligence*, ser. AmI'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 177–186.

[37] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[38] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Dec. 2005.

[39] A. Reiss and D. 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*, ser. PETRA '12. New York, USA: ACM, 2012, pp. 40:1–40:8.

[40] S. I, H. G, S. M, and G.-A. A., "Evaluation des krebsregisters nrw - schwerpunkt record linkage - abschlussbericht," Institut fr medizinische Biometrie, Epidemiologie und Informatik, Universittsmedizin Mainz, Tech. Rep., 2009.

[41] F. Petitjean, J. Inglada, and P. Gançarski, "Satellite Image Time Series Analysis under Time Warping," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 8, pp. 3081–3095, Aug. 2012.

[42] U. M. Fayyad and K. B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993, pp. 1022–1027.

[43] M. Flores, J. Gmez, A. Martínez, and J. Puerta, "Handling numeric attributes when comparing Bayesian network classifiers: does the discretization method matter?" *Applied Intelligence*, vol. 34, no. 3, pp. 372–385, 2011.

[44] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

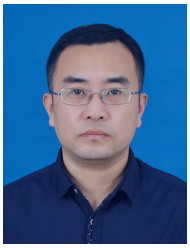
[45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2015.

[46] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

**Shenglei Chen** received the PhD degree in computer science from Nanjing University of Science and Technology in 2007. He is currently an associate professor in the Department of E-Commerce, Nanjing Audit University, China. He also serves as an adjunct research fellow in the Faculty of Information Technology, Monash University, Australia. His research interests include data mining and machine learning. He has published innovative works in journals and conference proceedings such as Journal of Machine Learning Research, Knowledge-Based Systems and PAKDD.

**Ana M. Martínez** received her M.S. and Ph.D. degrees in Computer Science from the University of Castilla-La Mancha (Spain) in 2007 and 2012 respectively. She was afterwards working as a research fellow at Monash University (Australia), in a project to classify large amounts of data with semi-naïve Bayesian classifiers, where she currently holds an adjunct research fellow position. As for July 2014, she is a postdoc in Aalborg University (Denmark), as part of the AMIDST (Analysis of MassIve Data STreams) European project (http://amidst.eu).

**Geoffrey I. Webb** is Director of the Monash University Center for Data Science. He was editor in chief of Data Mining and Knowledge Discovery from 2005 to 2014. He has been Program Committee Chair of both ACM SIGKDD and IEEE ICDM, as well as General Chair of ICDM. He is a Technical Advisor to BigML Inc, who incorporate his best of class association discovery software, Magnum Opus, into their cloud based Machine Learning service. He developed many of the key mechanisms of support-confidence association discovery in the 1980s. His OPUS search algorithm remains the state-of-the-art in rule search. He pioneered multiple research areas as diverse as black-box user modelling, interactive data analytics and statistically-sound pattern discovery. He has developed many useful machine learning algorithms that are widely deployed. He received the 2013 IEEE Outstanding Service Award, a 2014 Australian Research Council Discovery Outstanding Researcher Award and is an IEEE Fellow.



**Limin Wang** received the PhD degree in computer science from JiLin University in 2005. He is currently a professor in the college of computer science and technology in JiLin University, China. His research interests include probabilistic logic inference and Bayesian network. He has published innovative papers in journals such as Knowledge-Based Systems, Expert System with Applications and Progress in Natural Science.

TABLE 9
RMSE

| Algo | localization | census-income | USPS-Extended | MITFace-SetA | MITFace-SetB | MSDYear-Prediction | covertype | MITFace-SetC | poker-hand | uscensus-1990 | PAMAP2 | kddcup | linkage | satellite | splice |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0.5783± 0.0003 | 0.1034± 0.0003 |
| WAODE | 0.6497± 0.0011 | 0.2739± 0.0021 | 0.1547± 0.0027 | 0.0999± 0.0036 | 0.1693± 0.0025 | 0.9461± 0.0002 | 0.4567± 0.0014 | 0.1569± 0.0014 | 0.5068± 0.0007 | 0.2016± 0.0011 | 0.3873± 0.0008 | 0.0735± 0.0006 | 0.0118± 0.0005 | 0.5795± 0.0003 | 0.1111± 0.0004 |
| AODESR | 0.6520± 0.0010 | 0.2510± 0.0021 | 0.1427± 0.0024 | 0.1001± 0.0036 | 0.1682± 0.0025 | 0.9459± 0.0002 | 0.4551± 0.0013 | 0.1564± 0.0014 | 0.5392± 0.0006 | 0.1933± 0.0010 | 0.3873± 0.0007 | 0.0803± 0.0006 | 0.0120± 0.0005 | 0.5834± 0.0003 | 0.1034± 0.0003 |
| ASAODE | 0.6444± 0.0011 | 0.2057± 0.0013 | 0.1508± 0.0027 | 0.0509± 0.0020 | 0.1005± 0.0019 | 0.9455± 0.0002 | 0.4582± 0.0014 | 0.1419± 0.0017 | 0.5004± 0.0010 | 0.1538± 0.0008 | 0.3819± 0.0006 | 0.0611± 0.0007 | 0.0110± 0.0005 | 0.5783± 0.0003 | 0.0532± 0.0002 |
| SASAODE1k | 0.6452+/-0.0021 | 0.2096± 0.0032 | 0.1546± 0.0048 | 0.0606± 0.0047 | 0.1083± 0.0070 | 0.9462± 0.0005 | 0.4589± 0.0016 | 0.1466± 0.0038 | 0.5013± 0.0016 | 0.1547± 0.0014 | 0.3844± 0.0021 | 0.0636± 0.0030 | 0.0114± 0.0007 | 0.5823± 0.0046 | 0.0559± 0.0036 |
| SASAODE5k | 0.6444± 0.0011 | 0.2059± 0.0013 | 0.1519± 0.0033 | 0.0539± 0.0044 | 0.1024± 0.0020 | 0.9459± 0.0004 | 0.4583± 0.0014 | 0.1432± 0.0028 | 0.5007± 0.0011 | 0.1539± 0.0007 | 0.3833± 0.0013 | 0.0628± 0.0025 | 0.0115± 0.0007 | 0.5788± 0.0009 | 0.0537± 0.0004 |
| SASAODE10k | 0.6444± 0.0011 | 0.2059± 0.0014 | 0.1513± 0.0028 | 0.0522± 0.0034 | 0.1009± 0.0019 | 0.9457± 0.0003 | 0.4583± 0.0014 | 0.1425± 0.0023 | 0.5008± 0.0009 | 0.1539± 0.0007 | 0.3826± 0.0012 | 0.0615± 0.0014 | 0.0113± 0.0005 | 0.5785± 0.0006 | 0.0535± 0.0004 |
| SASAODE20k | 0.6444± 0.0011 | 0.2057± 0.0013 | 0.1510± 0.0026 | 0.0512± 0.0022 | 0.1013± 0.0020 | 0.9456± 0.0002 | 0.4582± 0.0013 | 0.1421± 0.0021 | 0.5005± 0.0010 | 0.1539± 0.0007 | 0.3824± 0.0010 | 0.0611± 0.0007 | 0.0115± 0.0006 | 0.5787± 0.0007 | 0.0534± 0.0002 |
| SASAODE50k | 0.6444± 0.0011 | 0.2057± 0.0013 | 0.1510± 0.0027 | 0.0511± 0.0021 | 0.1004± 0.0017 | 0.9456± 0.0002 | 0.4582± 0.0013 | 0.1419± 0.0018 | 0.5004± 0.0010 | 0.1539± 0.0008 | 0.3821± 0.0006 | 0.0611± 0.0007 | 0.0113± 0.0006 | 0.5786± 0.0007 | 0.0533± 0.0002 |
| SASAODE100k | 0.6444± 0.0011 | 0.2057± 0.0013 | 0.1509± 0.0027 | 0.0511± 0.0021 | 0.1006± 0.0020 | 0.9456± 0.0002 | 0.4582± 0.0013 | 0.1419± 0.0020 | 0.5004± 0.0010 | 0.1539± 0.0007 | 0.3820± 0.0006 | 0.0611± 0.0007 | 0.0111± 0.0006 | 0.5785± 0.0007 | 0.0533± 0.0002 |
| SASAODE200k | 0.6444± 0.0011 | 0.2057± 0.0013 | 0.1508± 0.0027 | 0.0511± 0.0021 | 0.1004± 0.0018 | 0.9455± 0.0002 | 0.4582± 0.0013 | 0.1420± 0.0019 | 0.5004± 0.0010 | 0.1539± 0.0008 | 0.3819± 0.0006 | 0.0611± 0.0007 | 0.0111± 0.0005 | 0.5783± 0.0003 | 0.0532± 0.0002 |
| SASAODE p200k | 0.8589± 0.1113 | 0.3279± 0.1380 | 0.1509± 0.0027 | 0.5763± 0.3313 | 0.5443± 0.2837 | 0.9744± 0.0157 | 0.4582± 0.0014 | 0.5956± 0.2344 | 0.7658± 0.2063 | 0.4374± 0.2682 | 0.9068± 0.1789 | 0.4391± 0.3853 | 0.5000± 0.0000 | 0.9338± 0.0309 | 0.0535± 0.0003 |
| A2DE | 0.5865± 0.0018 | 0.2403± 0.0026 | 0.1485± 0.0028 | 0.0737± 0.0027 | 0.1479± 0.0020 | 0.9368± 0.0003 | 0.4373± 0.0014 | 0.1489± 0.0017 | 0.4956± 0.0007 | 0.1753± 0.0010 | 0.3307± 0.0004 | 0.0833± 0.0007 | 0.0112± 0.0005 | oot[1] | oot[1] |
| WA2DE | 0.5884± 0.0018 | 0.2364± 0.0026 | 0.1497± 0.0027 | 0.0731± 0.0027 | 0.1478± 0.0020 | 0.9367± 0.0003 | 0.4304± 0.0016 | 0.1485± 0.0017 | 0.4201± 0.0008 | 0.1688± 0.0009 | 0.3291± 0.0004 | 0.0684± 0.0006 | 0.0109± 0.0005 | oot[1] | oot[1] |
| A2DESR | 0.5865± 0.0018 | 0.2135± 0.0020 | 0.1501± 0.0025 | 0.0737± 0.0027 | 0.1479± 0.0020 | 0.9368± 0.0003 | 0.4223± 0.0015 | 0.1489± 0.0017 | 0.4956± 0.0007 | 0.1678± 0.0009 | 0.3306± 0.0004 | 0.0643± 0.0006 | 0.0111± 0.0005 | oot[1] | oot[1] |
| SASA2DE20k | 0.5853± 0.0017 | 0.2033± 0.0011 | oot[1] | 0.0446± 0.0031 | 0.0840± 0.0020 | 0.9367± 0.0003 | 0.4337± 0.0015 | 0.1110± 0.0018 | 0.4073± 0.0007 | 0.1513± 0.0007 | 0.3264± 0.0007 | 0.0516± 0.0008 | 0.0078± 0.0010 | oot[1] | oot[1] |
| SASA2DE50k | 0.5853± 0.0017 | 0.2025± 0.0016 | oot[1] | 0.0444± 0.0032 | 0.0834± 0.0018 | 0.9366± 0.0003 | 0.4337± 0.0015 | 0.1107± 0.0015 | 0.4075± 0.0009 | 0.1512± 0.0007 | 0.3262± 0.0005 | 0.0515± 0.0007 | 0.0073± 0.0010 | oot[1] | oot[1] |
| 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 | 0.4953± 0.0012 | 0.2367± 0.0021 | 0.5801± 0.0006 | 0.2911± 0.0006 | 0.4647± 0.0006 | 0.1849± 0.0008 | 0.0125± 0.0006 | 0.6540± 0.0002 | 0.0971± 0.0002 |
| ASNB | 0.7106± 0.0007 | 0.2330± 0.0015 | 0.2254± 0.0026 | 0.0677± 0.0016 | 0.1382± 0.0018 | 0.9530± 0.0001 | 0.4845± 0.0010 | 0.2367± 0.0021 | 0.5801± 0.0006 | 0.2579± 0.0008 | 0.4573± 0.0007 | 0.1046± 0.0005 | 0.0125± 0.0006 | 0.6520± 0.0002 | 0.0537± 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 | 0.4721± 0.0013 | 0.2068± 0.0017 | 0.4987± 0.0006 | 0.1845± 0.0009 | 0.3232± 0.0007 | 0.0572± 0.0006 | 0.0081± 0.0009 | 0.5424± 0.0003 | 0.1020± 0.0003 |
| KDB5 | 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 | 0.4448± 0.0004 | 0.0924± 0.0002 |
| RF100 | 0.5194± 0.0024 | 0.1992± 0.0013 | 0.0338± 0.0010 | 0.0353± 0.0017 | 0.0566± 0.0012 | oom[2] | 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 | oom[2] | oom[2] |

[1] Out of time when the wall time is set to 120 hours for each fold.
[2] Out of memory when the available memory is 138G.

TABLE 10
Average numbers of parents and children selected in ASAODE and SASAODE

| Algo | localization | census-income | USPS-Extended | MITFace-SetA | MITFace-SetB | MSDYear-Prediction | covertype | MITFace-SetC | poker-hand | uscensus-1990 | PAMAP2 | kddcup | linkage | satellite | splice |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AODE | 5,5 | 41,41 | 676,676 | 361,361 | 361,361 | 90,90 | 54,54 | 361,361 | 10,10 | 67,67 | 54,54 | 41,41 | 11,11 | 138,138 | 141,141 |
| ASAODE | 1,5 | 3,5 | 652,469 | 1,25 | 1,85 | 18,80 | 42,54 | 42,361 | 1,5 | 2,4 | 15,54 | 2,20 | 5,11 | 138,138 | 14,6 |
| SASAODE1k | 1,5 | 8,3 | 559,491 | 2,33 | 2,149 | 23,59 | 41,52 | 29,336 | 1,8 | 1,11 | 21,47 | 2,20 | 4,7 | 68,79 | 37,12 |
| SASAODE5k | 1,5 | 3,5 | 629,485 | 1,92 | 1,156 | 41,69 | 44,51 | 34,353 | 1,7 | 1,8 | 18,45 | 2,20 | 4,8 | 104,105 | 11,7 |
| SASAODE10k | 1,5 | 2,5 | 645,474 | 1,27 | 1,89 | 31,78 | 44,52 | 36,354 | 1,8 | 1,6 | 16,48 | 2,20 | 4,7 | 105,105 | 9,7 |
| SASAODE20k | 1,5 | 2,5 | 648,466 | 1,26 | 1,105 | 24,79 | 44,53 | 41,357 | 1,7 | 1,7 | 15,47 | 2,20 | 5,7 | 82,84 | 7,7 |
| SASAODE50k | 1,5 | 3,5 | 650,469 | 1,26 | 1,87 | 23,77 | 46,53 | 40,359 | 1,5 | 1,8 | 15,51 | 2,20 | 5,8 | 104,105 | 8,6 |
| SASAODE100k | 1,5 | 3,5 | 650,469 | 1,26 | 1,85 | 21,79 | 46,53 | 39,361 | 1,5 | 1,6 | 15,51 | 2,20 | 5,10 | 116,116 | 11,6 |
| SASAODE200k | 1,5 | 3,5 | 652,469 | 1,26 | 1,86 | 18,79 | 43,53 | 40,360 | 1,5 | 1,6 | 15,54 | 2,20 | 5,10 | 138,138 | 12,6 |

TABLE 11
Zero-one loss

| Algo | localization | census-income | USPS-Extended | MITFace-SetA | MITFace-SetB | MSDYear-Prediction | cover-type | MITFace-SetC | poker-hand | uscensus-1990 | PAMAP2 | kddcup | linkage | satellite | splice |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AODE | 0.4333± 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 | 0.3537± 0.0004 | 0.0134± 0.0001 |
| WAODE | 0.4314± 0.0036 | 0.0990± 0.0018 | 0.0247± 0.0009 | 0.0103± 0.0007 | 0.0297± 0.0009 | 0.9288± 0.0012 | 0.2812± 0.0017 | 0.0257± 0.0005 | 0.1758± 0.0079 | 0.0474± 0.0005 | 0.1647± 0.0006 | 0.0109± 0.0002 | 0.0002± 0.0000 | 0.3552± 0.0005 | 0.0154± 0.0001 |
| AODESR | 0.4333± 0.0027 | 0.0844± 0.0019 | 0.0210± 0.0007 | 0.0104± 0.0007 | 0.0294± 0.0008 | 0.9281± 0.0013 | 0.2825± 0.0015 | 0.0254± 0.0005 | 0.4812± 0.0028 | 0.0434± 0.0005 | 0.1647± 0.0007 | 0.0108± 0.0002 | 0.0002± 0.0000 | 0.3608± 0.0004 | 0.0134± 0.0001 |
| ASAODE | 0.4556± 0.0033 | 0.0555± 0.0009 | 0.0235± 0.0008 | 0.0030± 0.0002 | 0.0108± 0.0004 | 0.9286± 0.0009 | 0.2852± 0.0018 | 0.0211± 0.0005 | 0.3302± 0.0022 | 0.0274± 0.0003 | 0.1611± 0.0005 | 0.0040± 0.0001 | 0.0002± 0.0000 | 0.3537± 0.0004 | 0.0029± 0 |
| SASAODE20k | 0.4556± 0.0033 | 0.0555± 0.0010 | 0.0236± 0.0008 | 0.0030± 0.0003 | 0.0110± 0.0003 | 0.9280± 0.0017 | 0.2853± 0.0016 | 0.0212± 0.0006 | 0.3302± 0.0022 | 0.0279± 0.0008 | 0.1629± 0.0026 | 0.0040± 0.0001 | 0.0002± 0.0000 | 0.3691± 0.0165 | 0.0029± 0.0001 |
| SASAODE50k | 0.4556± 0.0033 | 0.0555± 0.0009 | 0.0235± 0.0008 | 0.0030± 0.0003 | 0.0109± 0.0004 | 0.9279± 0.0010 | 0.2853± 0.0016 | 0.0211± 0.0005 | 0.3302± 0.0022 | 0.0281± 0.0011 | 0.1617± 0.0016 | 0.0040± 0.0001 | 0.0002± 0.0000 | 0.3630± 0.0151 | 0.0029± 0.0000 |
| A2DE | 0.3598± 0.0047 | 0.0777± 0.0014 | 0.0227± 0.0008 | 0.0057± 0.0004 | 0.0227± 0.0006 | 0.9095± 0.0012 | 0.2609± 0.0019 | 0.0232± 0.0005 | 0.1185± 0.0019 | 0.0366± 0.0005 | 0.1207± 0.0003 | 0.0108± 0.0002 | 0.0002± 0.0000 | oot[1] | oot[1] |
| WA2DE | 0.3602± 0.0044 | 0.0752± 0.0016 | 0.0231± 0.0008 | 0.0056± 0.0004 | 0.0227± 0.0006 | 0.9095± 0.0013 | 0.2505± 0.0025 | 0.0232± 0.0005 | 0.0763± 0.0011 | 0.0342± 0.0004 | 0.1195± 0.0003 | 0.0095± 0.0002 | 0.0002± 0.0000 | oot[1] | oot[1] |
| A2DESR | 0.3598± 0.0047 | 0.0618± 0.0014 | 0.0233± 0.0008 | 0.0057± 0.0004 | 0.0227± 0.0006 | 0.9095± 0.0012 | 0.2433± 0.0020 | 0.0232± 0.0005 | 0.1185± 0.0019 | 0.0345± 0.0004 | 0.1206± 0.0003 | 0.0094± 0.0001 | 0.0002± 0.0000 | oot[1] | oot[1] |
| SASA2DE20k | 0.3594± 0.0047 | 0.0546± 0.0008 | oot[1] | 0.0023± 0.0004 | 0.0078± 0.0005 | 0.9103± 0.0014 | 0.2567± 0.0026 | 0.0132± 0.0004 | 0.0883± 0.0378 | 0.0273± 0.0003 | 0.1184± 0.0007 | 0.0027± 0.0001 | 0.0001± 0.0000 | oot[1] | oot[1] |
| SASA2DE50k | 0.3594± 0.0047 | 0.0546± 0.0011 | oot[1] | 0.0022± 0.0004 | 0.0077± 0.0005 | 0.9101± 0.0010 | 0.2566± 0.0025 | 0.0131± 0.0003 | 0.1005± 0.0509 | 0.0273± 0.0003 | 0.1182± 0.0005 | 0.0027± 0.0001 | 0.0001± 0.0000 | oot[1] | oot[1] |
| 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 | 0.3321± 0.0024 | 0.0582± 0.0010 | 0.4988± 0.0018 | 0.0896± 0.0003 | 0.2365± 0.0007 | 0.0361± 0.0005 | 0.0002± 0.0000 | 0.4425± 0.0002 | 0.0121± 0.0001 |
| ASNB | 0.5449± 0.0026 | 0.0620± 0.0010 | 0.0531± 0.0012 | 0.0046± 0.0002 | 0.0196± 0.0005 | 0.9247± 0.0015 | 0.3094± 0.0017 | 0.0582± 0.0010 | 0.4988± 0.0018 | 0.0759± 0.0005 | 0.2302± 0.0007 | 0.0090± 0.0001 | 0.0002± 0.0000 | 0.4410± 0.0003 | 0.0029± 0.0000 |
| TAN | 0.4367± 0.0033 | 0.0675± 0.0016 | 0.0149± 0.0006 | 0.0005± 0.0001 | 0.0158± 0.0005 | 0.9268± 0.0010 | 0.3005± 0.0023 | 0.0455± 0.0008 | 0.3295± 0.0015 | 0.0390± 0.0005 | 0.1171± 0.0005 | 0.0034± 0.0001 | 0.0001± 0.0000 | 0.3240± 0.0004 | 0.0133± 0.0001 |
| KDB5 | 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 | 0.2284± 0.0004 | 0.0104± 0.0001 |

[1] Out of time when the wall time is set to 120 hours for each fold.

TABLE 12
Negative conditional log likelihood

| Algo | localization | census-income | USPS-Extended | MITFace-SetA | MITFace-SetB | MSDYear-Prediction | cover-type | MITFace-SetC | poker-hand | uscensus-1990 | PAMAP2 | kddcup | linkage | satellite | splice |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AODE | 1.7891± 0.0083 | 0.4898± 0.0062 | 0.9115± 0.0474 | 0.3789± 0.0325 | 1.1192± 0.0438 | 5.7865± 0.0129 | 0.9467± 0.0058 | 0.8297± 0.0156 | 1.2089± 0.0034 | 0.4122± 0.0053 | 1.7758± 0.0104 | 0.0400± 0.0006 | 0.0007± 0.0001 | 6.1527± 0.0119 | 0.0590± 0.0008 |
| WAODE | 1.7824± 0.0094 | 0.4086± 0.0050 | 0.9211± 0.0475 | 0.3774± 0.0325 | 1.1242± 0.0443 | 5.7601± 0.0124 | 0.9233± 0.0055 | 0.8218± 0.0156 | 1.0865± 0.0031 | 0.3186± 0.0044 | 1.7677± 0.0104 | 0.0240± 0.0004 | 0.0007± 0.0001 | 6.1337± 0.0118 | 0.0724± 0.0010 |
| AODESR | 1.7891± 0.0083 | 0.3204± 0.0053 | 0.7169± 0.0360 | 0.3789± 0.0325 | 1.1192± 0.0438 | 5.7865± 0.0129 | 0.9232± 0.0054 | 0.8297± 0.0156 | 1.2089± 0.0034 | 0.2796± 0.0039 | 1.7650± 0.0100 | 0.0273± 0.0005 | 0.0007± 0.0001 | 5.9648± 0.0120 | 0.0590± 0.0008 |
| ASAODE | 1.8528± 0.0098 | 0.2132± 0.0027 | 0.8413± 0.0421 | 0.0198± 0.0027 | 0.1764± 0.0079 | 5.6117± 0.0091 | 0.9435± 0.0058 | 0.5785± 0.0148 | 1.0977± 0.0048 | 0.1278± 0.0011 | 1.6265± 0.0085 | 0.0248± 0.0006 | 0.0005± 0.0001 | 6.1527± 0.0119 | 0.0230± 0.0002 |
| SASAODE20k | 1.8528± 0.0098 | 0.2133± 0.0028 | 0.8347± 0.0424 | 0.0201± 0.0027 | 0.1956± 0.0490 | 5.6020± 0.0732 | 0.9440± 0.0057 | 0.5842± 0.0211 | 1.0981± 0.0046 | 0.1283± 0.0013 | 1.5053± 0.1873 | 0.0248± 0.0006 | 0.0006± 0.0002 | 4.2352± 2.0162 | 0.0229± 0.0010 |
| SASAODE50k | 1.8528± 0.0098 | 0.2132± 0.0027 | 0.8372± 0.0452 | 0.0201± 0.0027 | 0.1783± 0.0089 | 5.5829± 0.0615 | 0.9441± 0.0058 | 0.5782± 0.0159 | 1.0977± 0.0048 | 0.1284± 0.0015 | 1.5873± 0.1286 | 0.0248± 0.0006 | 0.0006± 0.0001 | 5.0036± 1.8474 | 0.0229± 0.0004 |
| A2DE | 1.4750± 0.0105 | 0.2986± 0.0072 | 0.7473± 0.0460 | 0.1650± 0.0145 | 0.7929± 0.0363 | 5.4457± 0.0131 | 0.8488± 0.0057 | 0.6298± 0.0175 | 1.0441± 0.0030 | 0.2046± 0.0026 | 1.3610± 0.0054 | 0.0286± 0.0005 | 0.0006± 0.0001 | oot[1] | oot[1] |
| WA2DE | 1.4839± 0.0108 | 0.2882± 0.0067 | 0.7619± 0.0466 | 0.1614± 0.0144 | 0.7835± 0.0357 | 5.4173± 0.0129 | 0.8174± 0.0058 | 0.6187± 0.0174 | 0.8125± 0.0026 | 0.1796± 0.0023 | 1.3453± 0.0054 | 0.0209± 0.0004 | 0.0005± 0.0001 | oot[1] | oot[1] |
| A2DESR | 1.4750± 0.0105 | 0.2288± 0.0040 | 0.6961± 0.0410 | 0.1650± 0.0145 | 0.7929± 0.0363 | 5.4457± 0.0131 | 0.7865± 0.0052 | 0.6298± 0.0175 | 1.0441± 0.0030 | 0.1759± 0.0022 | 1.3600± 0.0053 | 0.0190± 0.0004 | 0.0006± 0.0001 | oot[1] | oot[1] |
| SASA2DE20k | 1.4752± 0.0101 | 0.2087± 0.0023 | oot[1] | 0.0259± 0.0122 | 0.0970± 0.0262 | 5.3964± 0.0295 | 0.8333± 0.0059 | 0.2487± 0.0099 | 0.7834± 0.0043 | 0.1241± 0.0011 | 1.2770± 0.0467 | 0.0147± 0.0006 | 0.0004± 0.0002 | oot[1] | oot[1] |
| SASA2DE50k | 1.4752± 0.0101 | 0.2070± 0.0036 | oot[1] | 0.0307± 0.0099 | 0.0934± 0.0248 | 5.4081± 0.0142 | 0.8333± 0.0057 | 0.2481± 0.0107 | 0.7848± 0.0052 | 0.1240± 0.0010 | 1.2935± 0.0328 | 0.0146± 0.0004 | 0.0003± 0.0001 | oot[1] | oot[1] |
| NB | 2.1440± 0.0054 | 1.9789± 0.0172 | 1.1939± 0.0268 | 0.3484± 0.0249 | 0.8865± 0.0396 | 8.1275± 0.0344 | 1.1997± 0.0074 | 2.0562± 0.0578 | 1.4158± 0.0048 | 1.7613± 0.0133 | 2.8094± 0.0103 | 0.2249± 0.0022 | 0.0009± 0.0001 | 19.8610± 0.0349 | 0.0509± 0.0003 |
| ASNB | 2.1440± 0.0054 | 0.2884± 0.0035 | 1.1906± 0.0265 | 0.0402± 0.0017 | 0.8071± 0.0374 | 4.8479± 0.0067 | 1.1119± 0.0054 | 2.0562± 0.0578 | 1.4158± 0.0048 | 0.4008± 0.0033 | 2.6305± 0.0107 | 0.0759± 0.0010 | 0.0009± 0.0001 | 17.9436± 0.0335 | 0.0285± 0.0002 |
| TAN | 1.7726± 0.0126 | 0.2571± 0.0056 | 0.1948± 0.0117 | 0.0053± 0.0020 | 0.2335± 0.0097 | 5.6951± 0.0127 | 1.0798± 0.0089 | 0.9739± 0.0112 | 1.0822± 0.0029 | 0.2595± 0.0028 | 1.1518± 0.0068 | 0.0207± 0.0005 | 0.0004± 0.0001 | 3.7185± 0.0091 | 0.0572± 0.0004 |
| KDB5 | 1.3241± 0.0143 | 0.4681± 0.0128 | 0.0912± 0.0088 | 0.7473± 0.0457 | 0.2307± 0.0173 | 16.9637± 0.0732 | 0.7217± 0.0080 | 0.0603± 0.0048 | 0.4014± 0.0030 | 0.1620± 0.0020 | 0.2241± 0.0020 | 0.0134± 0.0005 | 0.0002± 0.0001 | 1.6850± 0.0052 | 0.0592± 0.0006 |

[1] Out of time when the wall time is set to 120 hours for each fold.