Semi-Naive Exploitation of One-Dependence Estimators

Nan Li Yang Yu Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
$\{$*lin, yuy, zhouzh*$\}$*@lamda.nju.edu.cn*

***Abstract*—It is well known that the key of Bayesian classifier learning is to balance the two important issues, that is, the exploration of attribute dependencies in high orders for ensuring a sufficient flexibility in approximating the ground-truth dependencies, and the exploration of low orders for ensuring a stable probability estimate from limited training samples. By allowing one-order attribute dependencies, one-dependence estimators (ODEs) have been shown to be able to approximate the ground-truth attribute dependencies whilst keeping the effectiveness of probability estimation, and therefore leading to excellent performance. In previous studies, however, ODEs were exploited in simple ways, such as by averaging, for classification. In this paper, we propose a semi-naive exploitation of ODEs that fits a function of ODEs to pursue higher-order attribute dependencies. Extensive experiments show that the proposed SNODE approach can achieve better performance than many state-of-the-art Bayesian classifiers.**

***Keywords*-Bayesian classifier; one-dependence estimator; semi-naive Bayes**

I. Introduction

In principle, the optimal Bayesian classifier is the optimal way for supervised classification [9], which is, however, a theoretical model that requires infinite number of samples for true joint probabilities. Hence, in practice, restricted Bayesian models are used to approximate the optimal Bayesian model. The restricted models are ideally designed to be flexible sufficiently for capturing the ground-truth attribute dependencies as well as succinct sufficiently for estimating probabilities effectively. There is, however, a dilemma; that is, the model should only allow lower-order attribute dependencies for obtaining an effective estimate of probabilities from limited samples, yet for capturing the ground-truth attribute dependencies, the model should consider higher-order dependencies. It is noteworthy that learning the optimal dependencies has been proven to be NP-hard [5], [6].

Consequently, a spectrum of Bayesian learning approaches has been developed, which takes different balances between the complexity of attribute dependencies and the effectiveness of probability estimation. The two extremes of the spectrum are the *naive Bayes* [20] and *Bayesian network* [23], [16]; the former totally ignores the attribute dependencies while the latter takes the maximum flexibility for approximating the attribute dependencies. Between the two extremes on the spectrum are *semi-naive Bayesian classifiers* which try to find proper tradeoffs and have achieved successes [26], [25], [12], [11], [17], [4], [33], [36], [29], [32].

Among the numerous Bayesian learning approaches, semi-naive Bayesian classifiers which utilize one-dependence estimators (ODEs) have achieved remarkable performance. By allowing one-order dependencies, the probability estimation in ODEs is effective, while the model still has some flexibility for capturing the attribute dependencies. Representative approaches include TAN (tree augmented naive Bayes) [12], AODE (averaged one-dependence estimators) [29], HNB (hidden navie Bayes) [32], etc. Note that in previous studies, ODEs were exploited directly in a simple way, e.g., simple average of ODEs. It will be interesting to study whether a better performance can be achieved by exploiting the ODEs in other ways such that the resulting model inherits the effectiveness of ODEs while gains more flexibility for modelling higher-order dependencies.

In this paper, we propose a framework of semi-naive exploitation of ODEs (SNODE), where the ODEs are employed to capture high-order attribute dependencies. SNODE approximates the joint probabilities by using functions of ODEs that are fitted according to maximum likelihood estimation. For implementation, generalized additive model (GAM) [15] is employed in this paper, and the function fitting problem is reduced to a constrained concave optimization problem whose global optimal solution can be efficiently obtained. Experiments show that the performance of SNODE is superior to many state-of-the-art Bayesian classifiers. Bias-variance decomposition shows that, the success of SNODE mainly owes to its very low bias with slightly increased variance, which validates our proposal of semi-naive exploitation of ODEs.

The rest of the paper is organized as follows. In Section II, we introduce the background of Bayesian classifier learning and give a brief review on related work. In Section III, we propose the SNODE approach. Then we report on our experiments in Section IV. Finally, we conclude the paper in Section V.

II. Background and Related Work

Denote $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ as an instance, where x_i is the value of the i-th attribute, $c \in \{c_1, c_2, \cdots, c_k\}$ the class where k is the number of classes, and $c(\boldsymbol{x})$ denotes the class of an instance $\boldsymbol{x}$.

The optimal Bayesian classifier provides an optimal way to decide the class of an instance $\boldsymbol{x}$ as

$$c(\boldsymbol{x}) = \arg\max_{c} P(c|\boldsymbol{x}) = \arg\max_{c} P(\boldsymbol{x}, c),$$

which yields the minimum possible classification error [9]. However, it is intractable in practical applications, since real-world training data is usually insufficient for a reliable estimate of the joint distribution $P(\boldsymbol{x}, c)$. Consequently, approximating $P(\boldsymbol{x}, c)$ becomes the central problem in Bayesian classifier learning.

The modification of the optimal Bayesian classifier towards practical applicability forms a spectrum, along which different Bayesian learning approaches utilizing the attribute dependencies in different ways.

One extreme is *naive Bayes* [20]. It simply ignores all the attribute dependencies by taking the conditional independence assumption, i.e., the attributes are independent given the class, thus it follows that

$$P(\boldsymbol{x}, c) = \prod_{i=1}^{n} P(x_i|c).$$

Ignoring the dependencies makes the naive Bayes approach efficient, yet the missing of important dependence information sometimes hampers the classification performance seriously.

Another extreme is *Bayesian network* [23], [16], which has a strong flexibility for representing and exploiting various attribute dependencies. Suppose that the Bayesian network structure of attributes is known, *Bayesian network* can encode the joint probability distribution as

$$P(\boldsymbol{x}, c) = P(c) \prod_{i=1}^{n} P(x_i|\pi_i, c),$$

where π_i denotes the values of the parents of the i-th attribute excluding the class value c for the instance $\boldsymbol{x}$. However, learning the optimal structure, even on restricted structures, is NP-hard [5], [6]. Moreover, the data become overly sparse when the dependency order goes high, which causes both the ineffectiveness of probability estimation and inaccurate evaluations in the search of a good structure.

Between the two extremes of the spectrum are *semi-naive Bayesian classifiers* [33]. Different from totally ignoring the attribute dependencies as naive Bayes or taking maximum flexibility for modelling dependencies as Bayesian network, *semi-naive Bayesian classifiers* try to exploit attribute dependencies in moderate orders. For example, k-DE (k-dependence estimator) [25] allows each attribute to depend on at most k other attributes in addition to the class.

Among numerous semi-naive Bayesian classifiers, approaches which utilize ODEs have demonstrated remarkable performance. Representative examples include TAN and SP-TAN [12], [18], AODE and its variants [29], [34], [35], [31], HNB [32], etc. TAN restricts that each attribute can only depend on one parent in addition to the class, and thus it follows that

$$P(\boldsymbol{x}, c) = P(c) \prod_{i=1}^{n} P(x_i|pa(x_i), c),$$

where $pa(x_i)$ denotes x_i's dependent attribute, and is determined in the learning process. In AODE, an ensemble of ODEs is learned and the prediction is produced by aggregating the predictions of all qualified ODEs. In HNB, a hidden parent is created for each attribute which combines the influences from all other attributes.

Some other approaches implicitly restrict the dependencies by localizing naive Bayes classifiers, such as NBTree [19] which embeds a naive Bayes classifier in the leaves of a pre-trained decision tree, and LBR (lazy Bayesian rule) [36] which trains a naive Bayes classifier under a local rule.

The success of semi-naive Bayesian classifiers using ODEs suggests that ODEs are on well balance between the ground-truth dependencies approximation and the effectiveness of probability estimation. However, in previous approaches, ODEs were used directly in simple ways for classification. In next section, we will present the SNODE method which exploits ODEs in a semi-naive way.

III. SNODE

A. The Framework

Suppose that the ground-truth dependency structure of the attributes is known. The joint probability distribution can be denoted as

$$P(\boldsymbol{x}, c) = P(c) \prod_{i=1}^{n} P(x_i|\boldsymbol{\pi}_i, c), \tag{1}$$

where $\boldsymbol{\pi}_i$ is the attribute values on which the i-th attribute depends. Based on Eq. 1, the problem of estimating $P(\boldsymbol{x}, c)$ can be reduced to estimating the probabilities $P(c)$ and $P(x_i|\boldsymbol{\pi}_i, c)$'s. Since $P(c)$ can be easily estimated from the data, the key is to estimate $P(x_i|\boldsymbol{\pi}_i, c)$'s. Obviously, $P(x_i|\boldsymbol{\pi}_i, c)$'s also encode the dependencies between attributes.

To maintain the effectiveness of probability estimation, we use only ODEs. Consequently, we want to approximate $P(x_i|\boldsymbol{\pi}_i, c)$'s from $P(x_i|x_j, c)$'s; in other words, we want to model the underlying relationship between $P(x_i|\boldsymbol{\pi}_i, c)$ and $P(x_i|x_j, c)$'s for each different i.

For this purpose, we can search in some function space for a function F_i by optimizing a certain criterion, such that the inputs of F_i are $P(x_i|x_j, c)$'s and the output is $P(x_i|\boldsymbol{\pi}_i, c)$. The log-likelihood and conditional likelihood are commonly used optimization criterions, thus can be used here.

Let $\boldsymbol{p}_i = [P(x_i|x_1, c), \cdots, P(x_i|x_n, c)]^\top$. The joint distribution can be written as

$$P(\boldsymbol{x}, c) = P(c) \prod_{i=1}^{n} F_i(\boldsymbol{p}_i), \tag{2}$$

where F_i's are functions to be found. Given a training data set $\mathcal{D}$, the log-likelihood of the model is

$$\begin{aligned} LL(\boldsymbol{F}) &= \sum_{\boldsymbol{x}\in\mathcal{D}} \log P(c(\boldsymbol{x})) + \sum_{i=1}^{n} \sum_{\boldsymbol{x}\in\mathcal{D}} \log F_i(\boldsymbol{p}_i) \\ &= \sum_{\boldsymbol{x}\in\mathcal{D}} \log P(c(\boldsymbol{x})) + \sum_{i=1}^{n} L_i(F_i), \end{aligned} \tag{3}$$

where

$$L_i(F_i) = \sum_{\boldsymbol{x}\in\mathcal{D}} \log F_i(\boldsymbol{p}_i), \tag{4}$$

$c(\boldsymbol{x})$ is the class of an instance $\boldsymbol{x}$, and $\boldsymbol{F} = \{F_1, \cdots, F_n\}$ is the set of functions to be determined.

Taking *log-likelihood* as the optimization criterion, the function optimization problem can be formalized as

$$\begin{aligned} \max_{\{F_1,\cdots,F_n\}} \quad & \sum_{\boldsymbol{x}\in\mathcal{D}} \log P(c(\boldsymbol{x})) + \sum_{i=1}^{n} L_i(F_i) \\ s.t. \quad & F_i \in \mathcal{F} \qquad i = 1, 2, \cdots, n \end{aligned} \tag{5}$$

where $\mathcal{F}$ is the feasible function space from which the F_i's are chosen.

Obviously, in Eq. 5, $\sum_{\boldsymbol{x}\in\mathcal{D}} \log P(c(\boldsymbol{x}))$ is a constant, and the objective is linear. Thus, if all F_i's are independent, solving Eq. 5 is equivalent to solving a series of problems in Eq. 6 for different i's

$$\max_{F_i\in\mathcal{F}} \quad L_i(F_i) \tag{6}$$

where $LL_i(F_i)$ is defined in Eq. 4.

Once the functions F_i's are obtained by solving the optimization problem in Eq. 6, the joint distribution $P(\boldsymbol{x}, c)$ can be computed according to Eq. 2, and the class of instance $\boldsymbol{x}$ can be decided.

In SNODE, the functions of ODEs are fitted to gain the model flexibility as that by modelling higher-order attribute dependencies. Though maximum likelihood criterion is employed here to fit the functions, other criterions such as minimum square error or conditional likelihood can also be employed.

B. Modelling Higer-Order Dependencies with ODEs

Generalized additive model (GAM) [15] is a statistical model that permits the response probability distribution to be any member of the exponential family of distributions, which is also the probability distribution represented by Bayesian networks [16]. Here, we employ GAM for the implementation of our proposed SNODE framework. Note that it is also possible to use other models, which is an interesting future work.

1) Formulation: GAM relates the variables v_i's to the objective variable o through the link function $g(\cdot)$ and the smoothing functions $f_i(\cdot)$'s as

$$g(o) = \sum_{j=1}^{n} \lambda_{ij} f_j(v_i),$$

where λ_{ij}'s are model parameters.

Employing GAM to approximate $P(x_i|\boldsymbol{\pi}_i, c)$'s from $P(x_i|x_j, c)$'s , it becomes

$$g(P(x_i|\boldsymbol{\pi}_i, c)) = \sum_{j=1}^{n} \lambda_{ij} f_j(P(x_i|x_j, c)), \tag{7}$$

where $P(x_i|\boldsymbol{\pi}_i, c)$ is the objective variable, and λ_{ij}'s are parameters. For simplicity, we use link function $g(\cdot) = \log(\cdot)$ and smoothing function $f_j(\cdot) = \log(\cdot)$ here. As the consequence, Eq. 7 can be rewritten as

$$P(x_i|\boldsymbol{\pi}_i, c) = \exp\big(\sum_{j=1}^{n} \lambda_{ij} \log\big(P(x_i|x_j, c)\big)\big).$$

Considering normalization, the probability distribution can be written as

$$P(x_i|\boldsymbol{\pi}_i, c) = \frac{1}{S_i(\boldsymbol{\lambda}_i)} \exp\big(\boldsymbol{\lambda}_i^\top \cdot \boldsymbol{l}_i\big), \tag{8}$$

where

$$\boldsymbol{\lambda}_i = \big[\lambda_{i1}, \cdots, \lambda_{in}\big]^\top, \tag{9}$$

$$\boldsymbol{l}_i = \big[\log\big(P(x_i|x_1, c)\big), \cdots, \log\big(P(x_i|x_n, c)\big)\big]^\top, \tag{10}$$

$$S_i(\boldsymbol{\lambda}_i) = \sum_{x'_i} \exp\big(\boldsymbol{\lambda}_i^\top \cdot \boldsymbol{l}_i\big). \tag{11}$$

Now, Eq. 8 relates $P(x_i|\boldsymbol{\pi}_i, c)$ and $P(x_i|x_j, c)$'s, and $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_n]$ is the parameter with the constraints

$$\sum_{j=1}^{n} \lambda_{ij} = 1 \quad \text{and} \quad 0 \le \lambda_{ij} \le 1.$$

2) Optimization Problem: Substituting Eq. 8 into the optimization problem in Eq. 6, we get the optimization problem to be solved as

$$\begin{aligned} \max_{\boldsymbol{\lambda}_i} \quad & \sum_{\boldsymbol{x}\in\mathcal{D}} \big(\boldsymbol{\lambda}_i^\top \cdot \boldsymbol{l}_i - \log(S_i(\boldsymbol{\lambda}_i))\big) \\ s.t. \quad & \sum_{j=1}^{n} \lambda_{ij} = 1, \quad 0 \le \lambda_{ij} \le 1 \\ & j = 1, 2, \ldots, n \end{aligned} \tag{12}$$

where $\boldsymbol{l}_i$ and $S_i(\boldsymbol{\lambda}_i)$ are defined in Eq. 10 and 11, respectively.

Proposition 1 *The optimization problem defined in Eq. 12 is a constrained concave optimization problem.*

```
◇ Training Process
  Input:
      $\mathcal{D}$ - Training data set
  Process:
  1.  Estimate $P(c)$ and $P(x_i|x_j, c)$ from $\mathcal{D}$;
  2.  Initialize $\lambda_{ij} = 1/n$,
  3.  for $i = 1, \cdots, n$
  4.      Solve Eq. 12 to get $\boldsymbol{\lambda}_i$
  5.  end for

◇ Testing Process
  Input:
      $\boldsymbol{x} = (x_1, \cdots, x_n)$ - Test instances to be classified
  Process:
  1.  for all class values $c$
  2.      for $i = 1, \cdots, n$
  3.          Calculate $P(x_i|\boldsymbol{\pi}_i, c)$ based on Eq. 8
  4.      end for
  5.  end for
  6.  Return $c(\boldsymbol{x}) = argmax_c P(c) \prod_{i=1}^{n} P(x_i|\boldsymbol{\pi}_i, c)$
```

Figure 1. Pseudo-code of the SNODE algorithm

Proof: With a simple derivation we can get

$$\frac{\partial^2 L_i(\boldsymbol{\lambda}_i)}{\partial \boldsymbol{\lambda}_i^2} \leq 0.$$

Thus, $L_i(\boldsymbol{\lambda}_i)$ is a concave function with respect to $\boldsymbol{\lambda}_i$. Moreover, the feasible region is convex.

So, the problem in Eq. 12 is a constrained concave optimization problem. □

With the above proposition, the problem in Eq. 12 has one unique global optimal solution, and it can be solved efficiently by standard optimization algorithms such as interior-point methods [2].

3) Algorithm: The pseudo-code is shown in Figure 1.

At the training time, SNODE forms the tables of the joint variable values and class frequencies from which the probabilities $P(c)$ and $P(x_i|x_j, c)$ are calculated. The space complexity of the tables is $O\big(kn^2v^2\big)$, where k is the number of classes, v is the average number of values of every attribute, and n is the number of attributes. Derivation of the frequencies required to populate these tables is of time complexity $O(Nn^2)$, where N is the number of training examples. The concave optimization problem is critical to the training process, and its time complexity is $O(n^3)$ [22]. Therefore, the overall training time required is $O(n^2(N + n^2))$.

At the testing time, classifying an instance requires the calculation of Eq. 8, and it is of time complexity $O(kn^2)$.

C. Discussion

It is easy to see that our proposed SNODE is a flexible framework. In the current paper, GAM and log-likelihood are employed as the model and optimization criterion, respectively. Obviously, other models and criterions can also be utilized.

By using logistic functions, we can get a special case of SNODE which is similar to the well-studied log-linear model [7]. This suggests that, it might be possible to derive more insight by connecting the log-linear model to semi-naive Bayesian classifiers.

Discriminative learning of Bayesian classifiers has attached much attention. Instead of optimizing the log-likelihood, discriminative approaches try to maximize the conditional likelihood and may produce better class probability estimates [14], [24], [3]. A discriminant version of SNODE can be developed by maximizing the conditional likelihood; however, the corresponding optimization problem may be more difficult than the current one, since the conditional likelihood does not decompose into separate terms for each function to be learned.

IV. Experiments

A. Settings

Thirty-five UCI data sets are used in our experiments. These data sets span a broad range of real domains, with sizes ranging from 76 to 20,000. Some statistics of the data sets are summarized in Table I, where*#Inst*, *#Attr* and *#Class* mean the number of instances, attributes and classes, respectively.

Considering that many Bayesian classifiers could not handle missing values and real values directly, for simplicity, we filled in the missing values by the mode or mean of the corresponding variable, and discretized all the data using the MDL discretization [10].

We compare SNODE with several Bayesian classifiers, including the simplest one, naive Bayes (NB), a general Bayesian network learning algorithm, K2 [8], and state-of-the-art semi-naive Bayesian methods including TAN, AODE and HNB. SNODE was implemented in Weka [30], using the Matlab optimization toolbox [27] for optimization. The implementations of NB, K2, TAN, AODE and HNB in Weka were used here. *Laplacian correction* was used in probability estimations.

Bias-variance decomposition [13] is helpful for analyzing performance of learning algorithms. It breaks the error into *bias* and *variance*. Bias describes the error of the learner in expectation, while variance reflects the sensitivity of the learner to variations in the training samples. We performed bias-variance decomposition using the repeated cross-validation approach [28], which is the default method in Weka.

Table I
DATA SETS USED IN THE EXPERIMENTS.

Data set	# Inst	# Attr	# Class	Data set	# Inst	# Attr	# Class
artificial	5,109	8	10	*iris*	150	5	3
auto-mpg	398	8	4	*kr-vs-kp*	3,196	37	2
balance-scale	625	5	3	*led24*	3,200	25	10
balloons	76	5	2	*led7*	3,200	8	10
clean1	476	133	2	*machine*	209	8	8
cmc	1,473	10	3	*mfeat-mor*	2,000	7	10
coding	20,000	16	2	*nursery*	12,960	9	5
colic	368	17	2	*page-blocks*	5,473	11	5
cylinder-bands	540	26	2	*segment*	2,310	19	7
ecoli	336	7	8	*sick*	3,772	26	2
glass	214	8	7	*solar-flare-2*	1,066	12	6
haberman	306	3	2	*tic-tac-toe*	958	10	2
heart-c	303	12	5	*titanic*	2,201	4	2
heart-statlog	270	10	2	*vehicle*	846	19	4
hepatitis	155	17	2	*waveform-5000*	5,000	20	3
house-votes-84	435	17	2	*yeast*	1,484	8	10
hypothyroid	3,772	26	4	*zoo*	101	17	7
ionosphere	351	34	2				

In our experiments, fifty runs of two-fold cross-validation were executed and divided into 5 groups each contained 10 runs. Bias-variance decomposition was performed on each group, and thus the averaged predictive error and a pair of bias/variance were obtained for each group. The mean and standard deviations were recorded, and pairwise t-tests with 95% significance level were conducted.

B. Results

The error, bias and variance averaged across all data sets for the compared classifiers are presented in Table II. The results of pairwise t-tests are summarized in Table III, where *win/tie/loss* means that SNODE wins, ties and loses on *#win*, *#tie* and *#loss* number of data sets against the corresponding comparison algorithm, according to pairwise t-tests. Detailed information over all data sets can be found in Tables IV, V and VI, respectively.

Figures 2, 3 and 4 depict the error, bias and variance relative to NB, respectively, where the x-axis shows error (bias, variance) of SNODE divided by that of NB, while the y-axis shows that of the compared classifier. Each point in the figures corresponds to one data set. The vertical and horizonal lines (i.e., $x = 1$ and $y = 1$) are used to highlight the performance of NB, and the diagonal lines (i.e., $y = x$) indicate equal performance of the compared approaches. If the error (bias, variance) of SNODE is better than the compared method on a data set, the corresponding point will appear above the diagonal line.

1) Comparison with State-of-the-art Methods : It can be observed from Tables II and IV that SNODE achieved the lowest error on much more data sets than the compared algorithms, and it achieved the lowest average error. Concerning pairwise t-tests results in Table III, SNODE always has larger counts of wins than losses.

Table II
AVERAGED ERROR, BIAS AND VARIANCE OVER ALL DATA SETS, WHERE THE BEST AND WORST PERFORMANCE IN EACH ROW ARE BOLDED AND UNDERLINED, RESPECTIVELY.

	SNODE	AODE	HNB	TAN	K2	NB
Error	**0.1868**	0.1913	0.1879	0.2081	0.1927	<u>0.2095</u>
Bias	**0.1505**	0.1605	0.1524	0.1805	0.1511	<u>0.1833</u>
Variance	0.0363	0.0308	0.0355	0.0275	<u>0.0416</u>	**0.0273**

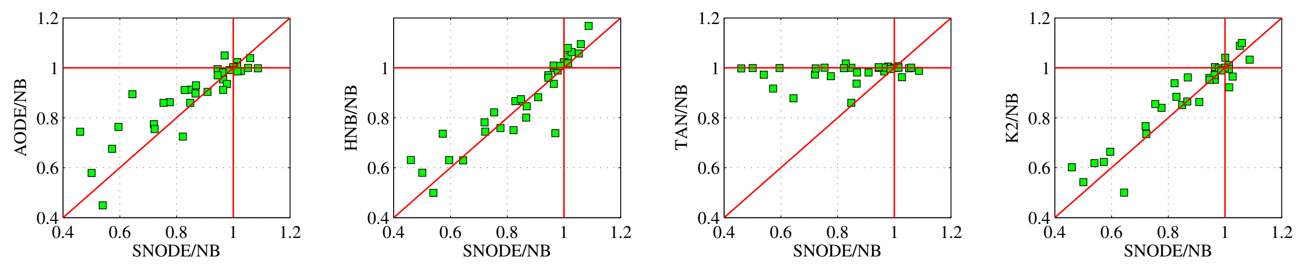
Table III
WIN/TIE/LOSS COUNTS, WHICH SUMMARIZES THE COMPARISON OF SNODE AGAINST THE CORRESPONDING ALGORITHM ACCORDING TO PAIRWISE t-TESTS WITH 95% SIGNIFICANCE LEVEL

	AODE	HNB	TAN	K2	NB
Error	16/10/9	10/19/6	19/9/7	14/16/5	23/5/7
Bias	25/5/5	16/12/7	30/4/1	10/16/9	30/4/1
Variance	7/12/16	8/19/8	5/8/22	24/5/6	5/11/19

In Figure 2, most points appear to the left of the vertical line $x = 1$, indicating that SNODE performs better than NB. Also, the number of points appear above the diagonal line are apparently more than that below the diagonal line, indicating that the error of SNODE error is lower than that of the compared algorithms.

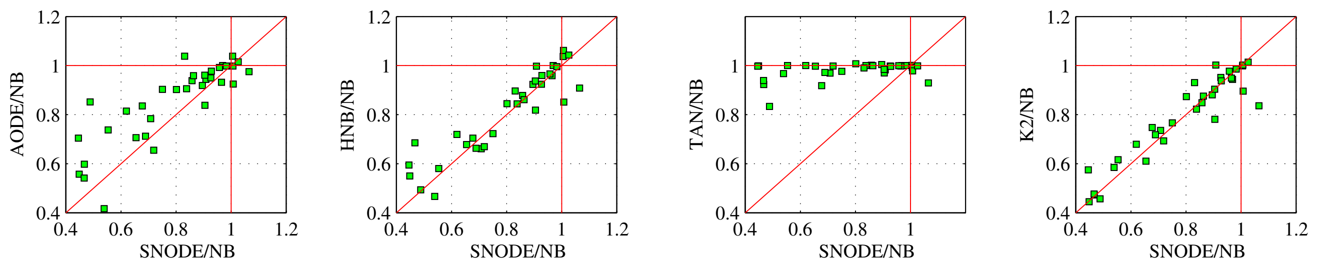
If we divide these algorithms into two groups, i.e., *extreme* group consisting of NB and K2, and *moderate* group consisting of SNODE, AODE, HNB and TAN, it can be found that classifiers in the moderate group performs better than those in the extreme group, which supports the motivation of developing Bayesian classifiers with the exploitation of moderate attribute dependencies.

Within the moderate group, it can be found that TAN sometimes has relatively poor performance, such as on *house-votes-84*, *nursery* and *kr-vs-kp*, while SNODE, AODE



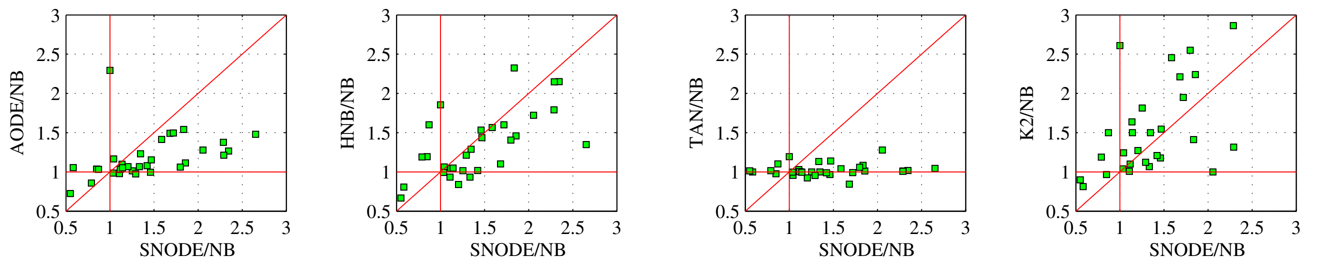
(a) SNODE vs. AODE (b) SNODE vs. HNB (c) SNODE vs. TAN (d) SNODE vs. K2

Figure 2. Comparison of relative ERROR. The x-axis presents error of SNODE divided by that of NB, y-axis presents error of the compared classifier divided by that of NB, and each point corresponds to one data set.



(a) SNODE vs. AODE (b) SNODE vs. HNB (c) SNODE vs. TAN (d) SNODE vs. K2

Figure 3. Comparison of relative BIAS. The x-axis presents bias of SNODE divided by that of NB, the y-axis presents bias of the compared classifier divided by that of NB, and each point corresponds to one data set.



(a) SNODE vs. AODE (b) SNODE vs. HNB (c) SNODE vs. TAN (d) SNODE vs. K2

Figure 4. Comparison of relative VARIANCE. The x-axis presents variance of SNODE divided by that of NB, the y-axis presents variance of the compared classifier divided by that of NB, and each point corresponds to one data set.

and HNB, which utilize ODEs, have relatively close performance on most data sets.

2) Bias-Variance Analysis: It has been shown that the maximum dependency determines the upper bound of representation ability of a Bayesian classifier [21]; this implies that, from the viewpoint of version space, the maximum dependency determines the upper bound of the range of the function space. It is known that the larger the version space, the lower the bias, and the larger the variance; this suggests a connection between bias/variance and the flexibility for dependencies in Bayesian classifiers.

From the comparison of average bias and variance shown in Table II, it can be found that NB was with the highest bias but the lowest variance, K2 exhibited the lowest bias but the highest variance, and TAN, AODE and HNB achieved a tradeoff with bias and variance. Similar behaviors can also be observed in detailed results shown in Tables V and VI. This observation can be explained from the fact that NB ignores all dependencies, K2 tries to exploit all possible dependencies, while TAN, AODE and HNB try to utilize dependencies only in moderate orders.

In Figures 3 and 4 it can be found that, compared with K2, SNODE achieved comparable bias yet much lower variance; compared with TAN, AODE and HNB, SNODE exhibited lower bias yet higher variance. This is also verified by pairwise t-tests shown in Table III. Hence, we can conclude that the success of SNODE owes much to its low bias.

In Table II it can be seen that SNODE achieved the lowest bias while its variance is still lower than K2. This may

Table IV
COMPARISON OF PREDICTIVE ERRORS (MEAN±STD.), WHERE THE BEST AND WORST PERFORMANCE IN EACH ROW ARE BOLDED AND UNDERLINED, RESPECTIVELY. THE *average* ROW PRESENTS THE RESULTS AVERAGED OVER ALL THE 35 DATA SETS; *win/tie/loss* ROW SUMMARIZES THE COMPARISON OF SNODE AGAINST THE CORRESPONDING ALGORITHM ACCORDING TO PAIRWISE *t*-TESTS WITH 95% SIGNIFICANCE LEVEL.

Data set	SNODE	AODE	HNB	TAN	K2	NB
artificial	.3709±.0028	.4119±.0017	**.3554±.0021**	.4619±.0016	.4013±.0015	.4779±.0012
auto-mpg	**.2252±.0065**	.2422±.0061	.2395±.0086	.3042±.0080	.2395±.0084	.3127±.0066
balance-scale	.2636±.0027	.2659±.0014	.2664±.0018	**.2600±.0019**	.2627±.0014	**.2600±.0013**
balloons	.2622±.0171	**.2592±.0129**	.2663±.0206	.2708±.0143	.2727±.0226	.2719±.0210
clean1	**.0847±.0045**	.0999±.0043	.1067±.0039	.1356±.0041	.0921±.0018	.1479±.0010
cmc	**.4560±.0028**	.4799±.0021	.4590±.0026	.4827±.0028	.4631±.0020	.4826±.0020
coding	**.2444±.0004**	.2628±.0003	.2450±.0002	.2880±.0005	.2450±.0003	.2880±.0003
colic	.1889±.0048	**.1807±.0014**	.1873±.0040	.1941±.0027	.1927±.0032	.1932±.0028
cylinder-bands	.2443±.0054	.2348±.0039	.2480±.0061	**.2290±.0031**	.2296±.0029	.2379±.0028
ecoli	.1667±.0101	.1531±.0068	.1756±.0074	**.1515±.0056**	.1584±.0045	.1534±.0068
glass	.2763±.0068	**.2612±.0032**	.2629±.0057	.2825±.0080	.2730±.0037	.2866±.0014
haberman	.2645±.0052	.2625±.0028	.2762±.0063	.2627±.0029	**.2597±.0035**	.2610±.0020
heart-c	.1762±.0046	.1674±.0030	.1736±.0026	**.1672±.0072**	.1822±.0021	.1675±.0022
heart-statlog	.1749±.0055	.1715±.0031	.1772±.0043	.1652±.0057	.1815±.0027	**.1651±.0025**
hepatitis	.1323±.0086	.1416±.0047	**.1263±.0069**	.1494±.0118	.1464±.0031	.1523±.0027
house-votes-84	**.0494±.0016**	.0571±.0010	.0561±.0016	.0986±.0009	.0535±.0006	.0987±.0007
hypothyroid	.0116±.0004	.0161±.0002	.0111±.0004	.0158±.0002	**.0090±.0002**	.0180±.0001
ionosphere	.0829±.0018	.0824±.0033	.0789±.0024	.0895±.0013	**.0787±.0005**	.0912±.0012
iris	.0680±.0033	.0571±.0018	.0663±.0052	.0544±.0027	**.0542±.0017**	.0543±.0015
kr-vs-kp	**.0586±.0007**	.0947±.0006	.0787±.0007	.1270±.0011	.0766±.0012	.1273±.0011
led24	**.2755±.0014**	.2757±.0011	.2758±.0010	.2753±.0016	.2865±.0007	.2753±.0008
led7	**.2649±.0004**	.2656±.0006	.2655±.0005	.2650±.0006	.2651±.0002	.2650±.0002
machine	.1297±.0070	.1342±.0046	**.1174±.0043**	.1401±.0035	.1294±.0071	.1496±.0069
mfeat-mor	.3086±.0034	.2997±.0023	.3034±.0036	.3043±.0023	**.2804±.0011**	.3041±.0013
nursery	**.0583±.0003**	.0747±.0005	.0605±.0004	.0978±.0010	.0650±.0002	.0979±.0002
page-blocks	.0352±.0004	**.0293±.0006**	.0319±.0006	.0634±.0007	.0403±.0004	.0652±.0003
segment	.0509±.0008	.0551±.0007	**.0380±.0005**	.0846±.0013	.0903±.0009	.0525±.0009
sick	**.0242±.0002**	.0266±.0002	.0248±.0001	.0297±.0002	.0258±.0002	.0292±.0002
solar-flare-2	**.2508±.0011**	.2554±.0016	.2575±.0018	.2605±.0016	.2528±.0013	.2603±.0012
tic-tac-toe	**.2196±.0072**	.2504±.0029	.2346±.0044	.2915±.0062	.2491±.0011	.2913±.0010
titanic	.2113±.0007	.2169±.0004	**.2104±.0005**	.2242±.0013	.2128±.0013	.2236±.0015
vehicle	**.2715±.0043**	.2835±.0054	.2742±.0021	.3747±.0018	.2760±.0016	.3757±.0018
waveform-5000	.1585±.0004	**.1397±.0009**	.1418±.0007	.1925±.0013	.1809±.0004	.1928±.0004
yeast	**.4100±.0028**	.4114±.0024	.4103±.0024	.4133±.0022	.4111±.0028	.4154±.0030
zoo	**.0743±.0044**	.0753±.0066	.0751±.0049	.0753±.0123	.1090±.0067	.0876±.0064
average	**0.1868**	0.1913	0.1879	0.2081	0.1927	0.2095
win/tie/loss	—	16/10/9	10/19/6	19/9/7	14/16/5	23/5/7

suggest that, the utilization of many ODEs is sufficient to obtain a bias as low as that can be obtained by exploiting a full Bayesian structure. Furthermore, it is usually feasible to estimate one-dependent estimators accurately in real applications. This validates our purpose of gaining flexibility as that by modelling higher-order probabilities based on one-dependent estimators.

V. CONCLUSION

In contrast to many previous Bayesian learning methods which utilize ODEs directly for classification in a simple way, in this paper, we present the SNODE framework, a semi-naive exploration of ODEs. In SNODE, functions of ODEs are employed to gain the flexibility by modelling higher-order attribute dependencies. As a special case, generalized additive model is employed for the implementation, and the function optimization problem is then reduced to an optimization problem whose global optimum can be solved efficiently.

Experiment results show that the proposed SNODE achieves better performance than many state-of-the-art Bayeian classifiers. Bias-variance decomposition discloses that, SNODE achieves very low bias; this validates our proposal of gaining the flexibility as that by modelling higher-order attribute dependencies based on ODEs. Moreover, the bias-variance decomposition suggests that there is a notable space for reducing the variance of SNODE. It is an interesting future work to improve SNODE by exploring regularization in the maximum likelihood estimation [1].

ACKNOWLEDGMENT

This work was supported by the National Science Foundation of China (60635030, 60721002), the Jiangsu Science Foundation (BK2008018) and the Jiangsu 333 High-Level Talent Cultivation Program.

Table V
COMPARISON OF BIAS (MEAN±STD.), WHERE THE BEST AND WORST PERFORMANCE IN EACH ROW ARE BOLDED AND UNDERLINED, RESPECTIVELY. THE *average* ROW PRESENTS THE RESULTS AVERAGED OVER ALL THE 35 DATA SETS; *win/tie/loss* ROW SUMMARIZES THE COMPARISON OF SNODE AGAINST THE CORRESPONDING ALGORITHM ACCORDING TO PAIRWISE*t*-TESTS WITH 95% SIGNIFICANCE LEVEL.

Data set	SNODE	AODE	HNB	TAN	K2	NB
artificial	.2685±.0021	.2973±.0033	**.2509±.0014**	.3679±.0040	.2791±.0013	.3795±.0026
auto-mpg	.1736±.0069	.1793±.0042	**.1670±.0089**	.2448±.0045	.1809±.0058	.2519±.0065
balance-scale	.1848±.0032	.1909±.0031	.1907±.0029	.1841±.0036	**.1837±.0011**	.1839±.0014
balloons	**.1776±.0157**	.1845±.0062	.1950±.0110	.1921±.0097	.1959±.0182	.1955±.0173
clean1	**.0631±.0032**	.0808±.0036	.0926±.0056	.1247±.0030	.0638±.0018	.1351±.0014
cmc	**.3721±.0042**	.4069±.0041	.3809±.0024	.4342±.0020	.3680±.0042	.4337±.0043
coding	.2328±.0003	.2516±.0004	.2344±.0002	.2778±.0009	**.2285±.0004**	.2778±.0004
colic	.1580±.0021	.1622±.0027	.1630±.0033	.1772±.0026	**.1554±.0022**	.1765±.0019
cylinder-bands	.2041±.0027	.1870±.0043	.1741±.0063	.1781±.0042	**.1603±.0057**	.1917±.0047
ecoli	.1271±.0057	.1261±.0029	.1341±.0045	**.1253±.0016**	.1266±.0015	.1263±.0007
glass	.2318±.0047	.2128±.0033	**.1958±.0031**	.2252±.0060	.2061±.0050	.2301±.0094
haberman	.2185±.0053	.2240±.0038	**.2180±.0054**	.2355±.0041	.2245±.0060	.2359±.0057
heart-c	**.1469±.0033**	.1547±.0039	.1519±.0037	.1578±.0056	.1485±.0027	.1583±.0030
heart-statlog	.1591±.0044	.1575±.0042	.1617±.0037	**.1549±.0066**	.1573±.0025	.1551±.0026
hepatitis	.1058±.0059	.1273±.0084	**.1019±.0116**	.1377±.0077	.1081±.0058	.1410±.0058
house-votes-84	.0432±.0017	.0537±.0013	.0530±.0004	.0963±.0011	**.0429±.0008**	.0964±.0007
hypothyroid	.0079±.0004	.0138±.0002	.0080±.0002	.0135±.0004	**.0074±.0003**	.0162±.0003
ionosphere	.0788±.0025	.0730±.0047	.0713±.0037	.0845±.0014	**.0680±.0022**	.0871±.0022
iris	**.0433±.0041**	.0541±.0016	.0467±.0031	.0516±.0044	.0485±.0020	.0521±.0013
kr-vs-kp	**.0503±.0006**	.0795±.0008	.0671±.0009	.1126±.0021	.0649±.0010	.1129±.0011
led24	.2370±.0011	.2446±.0010	.2446±.0010	.2447±.0015	**.2308±.0011**	.2446±.0012
led7	**.2257±.0019**	.2294±.0014	.2288±.0022	.2298±.0017	.2265±.0009	.2298±.0008
machine	**.0754±.0065**	.0930±.0048	.0784±.0038	.1022±.0047	.0832±.0067	.1113±.0039
mfeat-mor	.2346±.0017	.2606±.0026	**.2341±.0025**	.2716±.0024	.2379±.0024	.2718±.0023
nursery	**.0513±.0001**	.0684±.0006	.0538±.0004	.0927±.0010	.0571±.0004	.0927±.0005
page-blocks	.0314±.0006	**.0243±.0003**	.0272±.0009	.0564±.0004	.0341±.0009	.0583±.0008
segment	.0366±.0012	.0425±.0009	**.0280±.0002**	.0737±.0017	.0374±.0007	.0785±.0009
sick	**.0222±.0003**	.0250±.0002	.0234±.0001	.0279±.0003	.0242±.0005	.0277±.0003
solar-flare-2	.2106±.0024	.2239±.0017	.2183±.0013	.2294±.0022	**.2106±.0017**	.2330±.0012
tic-tac-toe	**.1660±.0038**	.2182±.0033	.1927±.0034	.2680±.0036	.1820±.0028	.2679±.0030
titanic	.2105±.0002	**.2034±.0011**	.2092±.0004	.2178±.0010	.2070±.0012	.2182±.0010
vehicle	.2116±.0058	.2282±.0051	.2191±.0017	.3224±.0038	**.1973±.0043**	.3232±.0041
waveform-5000	.1340±.0010	**.1222±.0008**	.1248±.0010	.1862±.0013	.1292±.0008	.1865±.0009
yeast	**.3621±.0008**	.3753±.0018	.3654±.0041	.3780±.0018	.3696±.0027	.3783±.0015
zoo	**.0158±.0072**	.0408±.0048	.0293±.0067	.0409±.0069	.0260±.0080	.0550±.0061
average	**0.1505**	0.1605	0.1524	0.1805	0.1511	0.1833
win / tie / loss	—-	25 / 5 / 5	16 / 12 / 7	30 / 4 / 1	10 / 16 / 9	30 / 4 / 1

REFERENCES

[1] G. Andrew and J. Gao. Scalable training of L^1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine learning*, pages 33–40, Corvalis, OR, 2007.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[3] J. Burge and T. Lane. Learning class-discriminative dynamic bayesian networks. In *Proceedings of the 22nd International Conference on Machine learning*, pages 97–104, Bonn, Germany, 2005.

[4] J. Cerquides and R. L. D. Mántaras. Robust bayesian linear classifier ensembles. In *Proceedings of the 16th European Conference on Machine learning*, pages 70–81, Porto, Portugal, 2005.

[5] D. M. Chickering. Learning Bayesian networks is NP-Complete. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pages 121–130, Fort Lauderdale, FL, 1996.

[6] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[7] R. Christensen. *Log-Linear Models and Logistic Regression*. Springer, New York, NY, 1997.

[8] G. Cooper and E. Herskovits. Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[9] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, NY, 1973.

[10] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In

Table VI
COMPARISON OF VARIANCE (MEAN±STD.), WHERE THE BEST AND WORST PERFORMANCE IN EACH ROW ARE BOLDED AND UNDERLINED, RESPECTIVELY. THE *average* ROW PRESENTS THE RESULTS AVERAGED OVER ALL THE 35 DATA SETS; *win/tie/loss* ROW SUMMARIZES THE COMPARISON OF SNODE AGAINST THE CORRESPONDING ALGORITHM ACCORDING TO PAIRWISE *t*-TESTS WITH 95% SIGNIFICANCE LEVEL.

Data set	SNODE	AODE	HNB	TAN	K2	NB
artificial	.1024±.0024	.1146±.0024	.1045±.0033	.0941±.0044	.1223±.0027	**.0984±.0024**
auto-mpg	**.0515±.0029**	.0629±.0030	.0725±.0027	.0593±.0044	.0587±.0060	.0607±.0061
balance-scale	.0787±.0019	**.0750±.0026**	.0756±.0032	.0759±.0028	.0790±.0011	.0761±.0014
balloons	.0842±.0074	.0745±.0089	**.0711±.0104**	.0784±.0053	.0766±.0149	.0761±.0158
clean1	.0215±.0032	.0191±.0022	.0141±.0025	.0108±.0028	.0283±.0012	**.0128±.0017**
cmc	.0839±.0023	.0730±.0033	.0781±.0014	**.0484±.0032**	.0951±.0037	.0488±.0042
coding	.0116±.0002	.0112±.0004	.0106±.0002	**.0102±.0009**	.0167±.0002	.0102±.0002
colic	.0308±.0036	.0185±.0023	.0242±.0027	.0168±.0035	.0372±.0011	**.0166±.0010**
cylinder-bands	**.0402±.0042**	.0477±.0011	.0739±.0046	.0509±.0023	.0693±.0038	.0462±.0027
ecoli	.0394±.0096	.0269±.0055	.0414±.0062	**.0261±.0040**	.0318±.0051	.0270±.0072
glass	**.0444±.0085**	.0483±.0028	.0670±.0068	.0572±.0101	.0669±.0033	.0563±.0103
haberman	.0459±.0076	.0385±.0022	.0581±.0062	.0271±.0025	.0353±.0034	**.0250±.0050**
heart-c	.0294±.0045	.0127±.0017	.0216±.0025	.0094±.0026	.0339±.0022	**.0091±.0018**
heart-statlog	.0157±.0040	.0140±.0043	.0155±.0024	.0103±.0030	.0243±.0022	**.0099±.0018**
hepatitis	.0265±.0043	.0143±.0057	.0243±.0052	.0115±.0083	.0383±.0033	**.0113±.0041**
house-votes-84	.0061±.0024	.0034±.0008	.0031±.0013	.0024±.0011	.0105±.0010	**.0023±.0006**
hypothyroid	.0037±.0003	.0023±.0002	.0031±.0003	.0023±.0005	**.0018±.0003**	.0018±.0004
ionosphere	**.0041±.0011**	.0094±.0024	.0076±.0015	.0049±.0011	.0107±.0017	.0041±.0014
iris	.0247±.0047	.0030±.0009	.0195±.0062	.0028±.0027	.0056±.0010	**.0021±.0015**
kr-vs-kp	**.0084±.0007**	.0152±.0008	.0116±.0009	.0144±.0010	.0117±.0006	.0144±.0006
led24	.0385±.0019	.0311±.0006	.0312±.0014	**.0307±.0004**	.0557±.0012	.0307±.0013
led7	.0392±.0019	.0362±.0012	.0366±.0022	.0352±.0016	.0386±.0007	**.0351±.0007**
machine	.0543±.0090	.0412±.0044	.0389±.0028	**.0378±.0013**	.0462±.0035	.0382±.0061
mfeat-mor	.0740±.0032	.0392±.0013	.0694±.0036	.0327±.0009	.0425±.0033	**.0323±.0034**
nursery	.0070±.0003	.0064±.0002	.0067±.0003	**.0052±.0004**	.0078±.0005	.0052±.0005
page-blocks	**.0038±.0005**	.0050±.0006	.0046±.0004	.0070±.0003	.0062±.0009	.0069±.0009
segment	.0142±.0006	.0126±.0007	**.0099±.0005**	.0109±.0010	.0150±.0003	.0118±.0004
sick	.0020±.0002	.0016±.0003	**.0014±.0002**	.0017±.0002	.0016±.0005	.0015±.0003
solar-flare-2	.0401±.0019	.0315±.0006	.0392±.0015	.0311±.0029	.0422±.0012	**.0273±.0007**
tic-tac-toe	.0535±.0041	.0322±.0028	.0419±.0031	.0236±.0041	.0670±.0026	**.0234±.0028**
titanic	**.0008±.0007**	.0136±.0009	.0012±.0008	.0065±.0022	.0058±.0023	.0053±.0023
vehicle	.0599±.0044	.0553±.0026	.0551±.0024	**.0523±.0034**	.0787±.0038	.0525±.0038
waveform-5000	.0245±.0009	.0174±.0009	.0170±.0009	**.0063±.0013**	.0517±.0006	.0063±.0007
yeast	.0478±.0036	.0361±.0037	.0449±.0043	**.0353±.0024**	.0415±.0030	.0370±.0039
zoo	.0584±.0038	.0345±.0050	.0457±.0068	.0344±.0150	.0829±.0043	**.0325±.0016**
average	0.0363	0.0308	0.0355	0.0275	0.0416	**0.0273**
win / tie / loss	—-	7 / 12 / 16	8 / 19 / 8	5 / 8 / 22	24 / 5 / 6	5 / 11 / 19

Proceedings of the 13th International Joint Conference on Artificial Intelligence, pages 1022–1029, Chambery, France, 1993.

[11] E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 249–256, Acapulco, Mexico, 2003.

[12] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[13] S. German, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[14] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the 21st International Conference on Machine Leanring*, pages 361–368, Banff, Canada, 2004.

[15] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, New York, NY, 1990.

[16] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[17] Y. Jing, V. Pavlović, and J. M. Rehg. Efficient discriminative learning of bayesian network classifier via boosted augmented naive bayes. In *Proceedings of the 22nd International Conference on Machine learning*, pages 369–376, Bonn, Germany, 2005.

[18] E. Keogh and M. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the 15th International Workshop on Artificial Intelligence and Statistics*, pages 225–230, Stockholm, Sweden, 1999.

[19] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the 2nd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 202–207, Portland, OR, 1996.

[20] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 223–228, San Jose, CA, 1992.

[21] C. X. Ling and H. Zhang. The representational power of discrete Bayesian networks. *Journal of Machine Learning Research*, 3:709–721, 2003.

[22] S. G. Nash and A. Sofer. On the complexity of a practical interior-point method. *SIAM Journal on Optimization*, 8(3):833–849, 1998.

[23] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[24] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative bayesian network classifiers and logistic regression. *Machine Learning*, 59(3):267–296, 2005.

[25] M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 335–338, Portland, OR, 1996.

[26] M. Singh and G. M. Provan. Efficient learning of selective bayesian network classifiers. In *Proceedings of the 13th International Conference on Machine Learning*, pages 453–461, Bari, Italy, 1996.

[27] P. Venkataraman. *Applied Optimization with MATLAB Programming*. John Wiley & Sons, New York, NY, 2002.

[28] G. I. Webb. Multiboosting: A technique for combining Boosting and Wagging. *Machine Learning*, 40(2):159–196, 2000.

[29] G. I. Webb, J. Boughton, and Z. Wang. Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

[30] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005.

[31] Y. Yang, G. I. Webb, J. Cerquidesz, K. Korb, J. Boughton, and K-M. Ting. To select or to weigh: A comparative study of linear combination schemes for superparent-one-dependence estimators. *IEEE Transactions on Knowledge and Data Engineering*, 9(12):1652–1665, 2007.

[32] H. Zhang, L. Jiang, and J. Su. Hidden naive Bayes. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 919–924, Pittsburgh, PA, 2005.

[33] F. Zheng and G. I. Webb. A comparative study of semi-naive Bayes methods in classification learning. In *Proceedings of the 4th Australasian Data Mining Conference*, pages 141–156, Sydney, Australia, 2005.

[34] F. Zheng and G. I. Webb. Efficient lazy elimination for averaged one-dependence estimators. In *Proceedings of the 23rd European Conference on Machine Learning*, pages 1113–1120, Pittsburgh, PA, 2007.

[35] F. Zheng and G. I. Webb. Finding the right family: Parent and child selection for averaged one-dependence estimators. In *Proceedings of the 18th European Conference on Machine Learning*, pages 490–501, Warsaw, Poland, 2007.

[36] Z. Zheng and G. I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 41(1):53–84, 2000.