MassBayes: A New Generative Classifier with Multi-dimensional Likelihood Estimation

Sunil Aryal and Kai Ming Ting

Gippsland School of Information Technology
Monash University, Australia
{sunil.aryal,kaiming.ting}@monash.edu

Abstract. Existing generative classifiers (e.g., BayesNet and AnDE) make independence assumptions and estimate one-dimensional likelihood. This paper presents a new generative classifier called *MassBayes* that estimates multi-dimensional likelihood without making any explicit assumptions. It aggregates the multi-dimensional likelihoods estimated from random subsets of the training data using varying size random feature subsets. Our empirical evaluations show that MassBayes yields better classification accuracy than the existing generative classifiers in large data sets. As it works with fixed-size subsets of training data, it has constant training time complexity and constant space complexity, and it can easily scale up to very large data sets.

Keywords: Generative classifier, Likelihood estimation, MassBayes.

1 Introduction

The learning task in classification is to learn a model from a labelled training set that maps each instance to one of the predefined classes. The model learned is then used to predict a class label for each unseen test instance. Each instance $\mathbf{x}$ is represented by a d-dimensional vector $\langle x_1, x_2, \cdots, x_d \rangle$ and given a class label $y \in \{y_1, y_2, \cdots, y_c\}$, where c is the total number of classes. The training set D is a collection of labelled instances $\{(\mathbf{x}^{(i)}, y^{(i)})\}$ $(i = 1, 2, \cdots, N)$.

The generative approach of classifier learning models the joint distribution $p(\mathbf{x}, y)$ and predicts the most probable class as:

$$\hat{y} = \arg\max_{y} \; p(\mathbf{x}, y) \tag{1}$$

Using the product rule, the joint probability can be factorised as:

$$p(\mathbf{x}, y) = p(y) \times p(\mathbf{x}|y) \tag{2}$$

Generative classifiers learn either the joint distribution $p(\mathbf{x}, y)$ or the likelihood $p(\mathbf{x}|y)$. However, estimating $p(\mathbf{x}, y)$ or $p(\mathbf{x}|y)$ directly from data using existing data modelling techniques is difficult. Density estimators such as Kernel Density Estimation [1], k-Nearest Neighbour [1] and Density Estimation Trees [2] are impractical in large data sets due to their high time and space complexities. The research has thus focused on learning one-dimensional likelihood to approximate $p(\mathbf{x}, y)$ in different ways.

J. Pei et al. (Eds.): PAKDD 2013, Part I, LNAI 7818, pp. 136–148, 2013.

Existing generative classifiers allow limited probabilistic dependencies among attributes and assume some kind of conditional independence. Different generative classifiers make different assumptions and allow different level of dependencies. They learn a network (or its simplification) of probabilistic relationship between the attributes and estimate the likelihood at each node given its parents from D (i.e., one-dimensional likelihood estimation). The joint distribution $p(\mathbf{x}, y)$ is estimated as the product of likelihood of each attribute given their parents in the network:

$$\hat{p}(\mathbf{x}, y) = p(x_1|\pi_1) \times p(x_2|\pi_2) \times \cdots \times p(x_d|\pi_d) \times p(y|\pi_y) \tag{3}$$

where π_i is $parent(x_i)$ and π_y is $parent(y)$.

Though these one-dimensional likelihood generative classifiers have been shown to perform well [3,4,5,6,7], we hypothesize that a multi-dimensional likelihood generative classifier will produce even better results.

In this paper, we propose an ensemble approach to estimate multi-dimensional likelihood without making any explicit assumption about attribute independence. The idea is to construct an ensemble of t multi-dimensional likelihood estimators using random sub-samples $\mathcal{D}_i \subset D$ $(i = 1, 2, \cdots, t)$. Each estimator estimates the multi-dimensional likelihood using a random subset of d attributes from $\mathcal{D}_i$. The average estimation from t estimators provides a good approximation of $p(\mathbf{x}|y)$. We call the resulting generative classifier *MassBayes*. It has constant space complexity and constant training time complexity because it employs a fixed-size training subset to build each of the t estimators.

The rest of the paper is structured as follows. Section 2 provides a brief overview of well-known generative classifiers. The proposed method is described in Section 3 followed by the implementation details in Section 4. The empirical evaluation results are presented in Section 5. Finally, we provide conclusions and directions for future research in Section 6.

2 Existing Generative Classifiers

Naive Bayes (NB) [3] is the simplest generative approach that estimates $p(\mathbf{x}, y)$ by assuming that the attributes are statistically independent given y:

$$\hat{p}(\mathbf{x}, y)_{NB} = p(y) \prod_{i=1}^{d} p(x_i|y) \tag{4}$$

Despite the strong independence assumption, it has been shown that NB produces impressive results in many application domains [3,4]. Its simplicity and clear probabilistic semantics have motivated researchers to explore different extensions of NB to improve its performance by relaxing the unrealistic assumption.

BayesNet [5] learns a network of probabilistic relationship among the attributes including the class attribute from the training data. Each node in the network is independent of its non-descendants given the state of its parents. At each node, the conditional probabilities with respect to its parents are learned from D. The joint probability $p(\mathbf{x}, y)$ is estimated as:

$$\hat{p}(\mathbf{x}, y)_{BayesNet} = p(y|\pi_y) \prod_{i=1}^{d} p(x_i|\pi_i) \tag{5}$$

Learning an optimal network requires searching over a set of every possible network, which is exponential in d. It is intractable in high-dimensional problems [8]. NB is the simplest form of a Bayesian network, where each attribute is dependent on y only.

In another simplification of BayesNet, AnDE [7] relaxes the independence assumption by allowing dependency between y and a fixed number of privileged attributes or super-parents. The other attributes are assumed to be independent given the n super-parents and y. AnDE with $n = 0$, A0DE, is NB. AnDE avoids the expensive searching in learning probabilistic dependencies by constructing an ensemble of n-dependence estimators. The joint probability $p(\mathbf{x}, y)$ is estimated as:

$$\hat{p}(\mathbf{x}, y)_{AnDE} = \sum_{s \in S^n} p(\mathbf{x}_s, y) \prod_{j \in \{1,2,\cdots,d\} \setminus s} p(x_j|\mathbf{x}_s, y) \tag{6}$$

where S^n is the collection of all subsets of size n of the set of d attributes $\{1, 2, \cdots, d\}$; and $\mathbf{x}_s$ is a n-dimensional vector of values of $\mathbf{x}$ defined by s.

It has been shown that A1DE and A2DE produce better predictive accuracy than the other state-of-the-art generative classifiers [6,7]. However, it only allows dependencies on a fixed number of attributes and y. Because of the high time complexity of $O\left(N\binom{d}{n+1}\right)$ [1] and space complexity of $O\left(c\binom{d}{n+1}v^{n+1}\right)$, where v is the average number of values for an attribute [7], only A2DE or A3DE is feasible even for a moderate number of dimensions. Furthermore, selecting an appropriate value of n for a particular data set requires a search.

AnDE and many other implementations of BayesNet require all the attributes to be discrete. The continuous-valued attributes must be discretised using a discretisation method before building a classifier.

3 MassBayes: A New Generative Classifier

Rather than aggregating an ensemble of n-dependence single-dimensional likelihood estimators, we propose to aggregate an ensemble of t multi-dimensional likelihood estimators where each likelihood is estimated using different random subsets of d attributes from data. The likelihood $p(\mathbf{x}|y)$ is estimated as:

$$\hat{p}(\mathbf{x}|y) = \frac{1}{t} \sum_{g \in G_t} p(\mathbf{x}_g|y) \tag{7}$$

where G_t is a collection of t subsets of varying sizes of d attributes; and $\mathbf{x}_g$ is a $|g|$-dimensional vector of values of $\mathbf{x}$ defined by g; and $1 \leq |g| \leq d$.

Each $p(\mathbf{x}_g|y)$ is estimated using a random subset of training instances $\mathcal{D} \subset D$, where $|\mathcal{D}| = \psi < N$.

$$\hat{p}(\mathbf{x}_g|y) = \frac{|\mathcal{D}_{y,\mathbf{x}_g}|}{|\mathcal{D}_y|} \tag{8}$$

[1] $\binom{d}{n}$ is a binomial coefficient of n out of d.

where $|\mathcal{D}_{y,\mathbf{x}_g}|$ is the number of instances having attribute values $\mathbf{x}_g$ belonging to class y in $\mathcal{D}$ and $|\mathcal{D}_y|$ is the number of instances belonging to class y in $\mathcal{D}$.

Rather than relying on a specific discretisation method in the preprocessing step, we propose to build a model directly from data, akin to an adaptive multi-dimensional histogram, to determine $\mathbf{x}_g$ which adapts to the local data distribution. The feature space partitioning we employed (to be discussed in Section 4) produces large regions in sparse area and small regions in the dense area of the data distribution.

Let $T(\cdot)$ be a function that divides the feature space into non-overlapping regions and $T(\mathbf{x})$ be the region where $\mathbf{x}$ falls. In a multi-dimensional space, each instance in $\mathcal{D}$ can be isolated by splitting only on few dimensions i.e., only a subset of d attributes ($g \subset \{1, 2, \cdots, d\}$) is used to define $T(\mathbf{x})$. Hence, $|\mathcal{D}_{y,\mathbf{x}_g}|$ is the number of instances belonging to class y in the region $T(\mathbf{x})$. Let $p(T(\mathbf{x})|y)$ be the probability of region $T(\mathbf{x})$ when only class y instances in $\mathcal{D}$ are considered.

$$p(T(\mathbf{x})|y) = \hat{p}(\mathbf{x}_g|y) = \frac{|\mathcal{D}_{y,\mathbf{x}_g}|}{|\mathcal{D}_y|} \tag{9}$$

The new generative classifier, called *MassBayes*, estimates the joint distribution as:

$$\hat{p}(\mathbf{x}, y)_{MassBayes} = p(y)\ \frac{1}{t} \sum_{g \in G_t} p(\mathbf{x}_g|y) = p(y)\ \frac{1}{t} \sum_{i=1}^{t} p(T_i(\mathbf{x})|y) \tag{10}$$

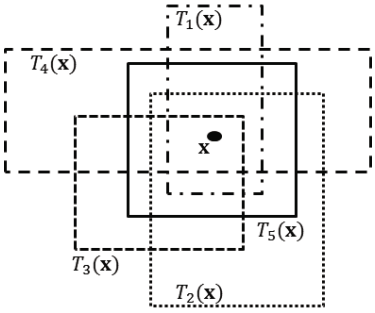


Fig. 1. Different regions from different $T_i(\cdot)$ ($i = 1, 2, \cdots, 5$) that cover $\mathbf{x}$

The average probability of t different regions $T_i(\mathbf{x})$ ($i = 1, 2, \cdots, t$), constructed using $\mathcal{D}_i \subset D$, provides a good estimate for $p(\mathbf{x}|y)$ as it estimates the multi-dimensional likelihood by considering the distribution in different local neighbourhood of $\mathbf{x}$ in the data space. An illustrative example is provided in Figure 1. Note that, the estimator employed in MassBayes is not a true density estimator as it does not integrate to 1.

MassBayes has the following characteristics in comparison with AnDE:

1. In each estimator, AnDE estimates one-dimensional likelihood given a fixed number of super-parents and y, whereas MassBayes estimates multi-dimensional likelihood using varying number of dimensions.
2. In AnDE, the ensemble size is fixed to $\binom{d}{n}$. But, MassBayes allows the flexibility for users to set the ensemble size.

3. AnDE requires continuous-valued attributes to be discretised before building the model. The performance of AnDE is affected by the discretisation technique used. In contrast, MassBayes builds models directly from data. It can be viewed as a dynamic multi-dimensional discretisation where the information loss is minimised by averaging over multiple models.
4. Each model in MassBayes is built with training subset of size $\psi < N$ which gives rise to the constant training time. In contrast, each model in AnDE is trained using the entire training set.
5. AnDE is a deterministic algorithm whereas MassBayes is a randomised algorithm.
6. Like AnDE, MassBayes is a generative classifier without search.

4 Implementation

In order to partition the feature space to define the regions $T_i(\cdot)$, we use the implementation described by Ting and Wells (2010) using a binary tree called *h:d-tree* [9]. A parameter h defines the maximum level of sub-division. The maximum height of a tree is $h \times d$.

Let the data space that covers the instances in $\mathcal{D}$ be Δ. The data space Δ is adjusted to become δ using a random perturbation conducted as follows. For each dimension j, a split point v_j is chosen randomly within the range $max_j(\Delta) - min_j(\Delta)$. Then, the new range δ_j along dimension j is defined as $[v_j - r, v_j + r]$, where $r = max(v_j - min_j(\Delta), max_j(\Delta) - v_j)$. The new range on all dimensions defines the adjusted work space for the tree building process.

A subset $\mathcal{D}$ is constructed from D by sampling ψ instances without replacement. The sampling process is restarted with D when all the instances are used. The random adjustment of the work space and random sub-sampling, as described earlier, ensure that no two trees are identical.

The dimension to split is selected from a randomised set of d dimensions in a round-robin manner at each level of a tree. A tree is constructed by splitting the work space into two equal-volume half spaces at each level. The process is then repeated recursively on each non-empty half-space. The tree building process stops when there is only one instance in a node or the maximum height is reached.

At the leaf node, the number of instances in the node belonging to each class is stored. Figure 2 shows a typical example of an implementation of $T(\cdot)$ as an *h:d-tree* for $h = 2$ and $d = 2$. The dotted lines enclosed the instances in $\mathcal{D}$ and the solid lines enclosed the adjusted work space which has ranges δ_1 and δ_2 on x_1 and x_2 dimensions. $R1, R2, R3, R4$ and $R5$ represent different regions in $T(\cdot)$ depending on the data distribution in $\mathcal{D}$. Region $R1$ is defined by splitting the work space in x_1 dimension only, $g = \{1\}$, whereas the other four regions use dimensions x_1 and x_2, i.e., $g = \{1, 2\}$.

In the original implementation by Ting and Wells (2010) for mass estimation, each tree is built to the maximum height of $h \times d$ resulting in equal-size regions regardless of the data distribution [9]. In our implementation, in order to adapt

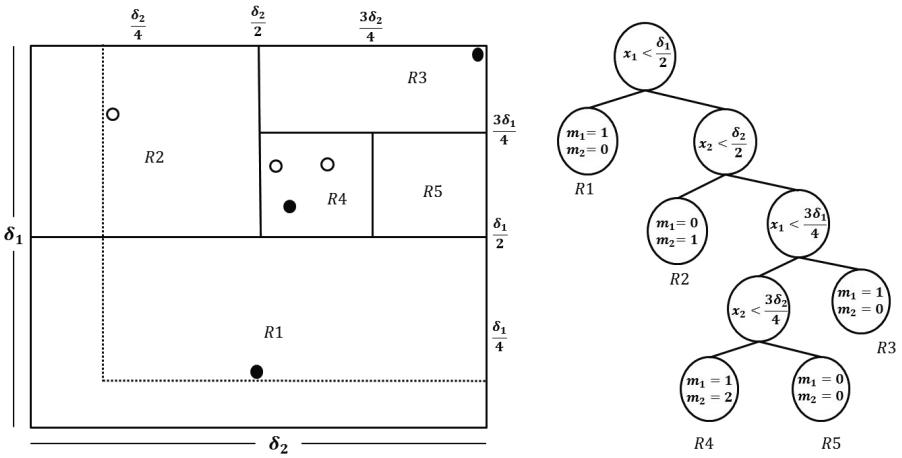


Fig. 2. An example of an h:d-*tree* for $h = 2$ and $d = 2$

to the data distribution, the tree building stops early once the instances are separated. We use the same algorithm as used by Ting and Wells (2010) to generate h:*d-trees* to represent $T_i(\cdot)$ in [9] with the required modification.

The procedures to generate t trees from a given data set D are provided in Algorithms 1 and 2.

The maximum height of each tree is hd, and ψ instances have to be assigned to either of the two child nodes at each level of a tree. Hence, the total training time complexity to construct t trees is $O(thd\psi)$. There are a maximum of ψ (as $\psi < 2^{hd}$ in general) leaf nodes in each tree. The total space complexity is $O(t(d + c)\psi)$.

The time and space complexities of two variants of NB (NB-KDE that estimates $p(x_i|y)$ through kernel density estimation [4]; and NB-Disc that estimates $p(x_i|y)$ through discretisation [10]), AnDE and MassBayes are presented in Table 1. Both training time complexity and space complexity of MassBayes are

Table 1. Time and space complexities of different generative classifiers

Classifiers	Training time	Testing time	Space
NB-KDE [4]	$O(Nd)$	$O(cmd)$	$O(cmd)$
NB-Disc [6]	$O(Nd)$	$O(cd)$	$O(cdv)$
AnDE [7]	$O\left(N\binom{d}{n+1}\right)$	$O\left(cd\binom{d}{n}\right)$	$O\left(c\binom{d}{n+1}v^{n+1}\right)$
MassBayes	$O(thd\psi)$	$O(thd)$	$O\left(t(d + c)\psi\right)$

N: total number of training instances, m: average number of training instances in a class, d: number of dimensions, c: number of classes, v: average number of discrete values of an attribute, n: number of super-parents, t: number of trees, h: level of divisions, and ψ: sample size.

independent of N. Note that the complexities for NB-Disc and AnDE do not include the additional discretisation needed in the preprocessing.

Algorithm 1. BuildTrees(D, t, ψ, h)

Inputs: D - input data, t - number of trees, ψ - sub-sampling size, h - number of times an attribute is employed in a path.
Output: F - a set of t h:d-*trees*

$H \leftarrow h \times d$ {Maximum height of a tree}
Initialize F
for $i = 1$ to t **do**
  $\mathcal{D} \leftarrow sample(D, \psi)$ {strictly without replacement}
  (min,max) $\leftarrow$ InitialiseWorkSpace($\mathcal{D}$)
  $A \leftarrow$ {Randomised list of d attributes.}
  $F \leftarrow F \cup$ SingleTree($\mathcal{D}, min, max, 0, A$)
end for
return F

Algorithm 2. SingleTree$(\mathcal{D}, min, max, \ell, A)$

Inputs: $\mathcal{D}$ - input data, min & max - arrays of minimum and maximum values for each attribute that define a work space, A - a randomised list of d attributes, ℓ - current height level.
Output: an h:d-*tree*

Initialize $Node(\cdot)$
while ($\ell < H$ and $|\mathcal{D}| > 1$) **do**
  $q \leftarrow nextAttribute(A, \ell)$ {Retrieve an attribute from A based on height level.}
  $mid_q \leftarrow (max_q + min_q)/2$
  $\mathcal{D}_l \leftarrow filter(\mathcal{D}, q < mid_p)$
  $\mathcal{D}_r \leftarrow filter(\mathcal{D}, q \geq mid_q)$
  if ($|\mathcal{D}_l| = 0$) or ($|\mathcal{D}_r| = 0$) **then** {Reduce range for single-branch node.}
    if ($|\mathcal{D}_l| > 0$) **then** $max_q \leftarrow mid_q$
    else $min_q \leftarrow mid_q$
    end if
    $\ell \leftarrow \ell + 1$
    continue at the start of while loop
  end if
  {Build two nodes: $Left$ and $Right$ as a result of a split into two half-spaces.}
  $temp \leftarrow max_q$; $max_q \leftarrow mid_q$
  $Left \leftarrow$ SingleTree($\mathcal{D}_l, min, max, \ell + 1, A$)
  $max_q \leftarrow temp$; $min_q \leftarrow mid_q$
  $Right \leftarrow$ SingleTree($\mathcal{D}_r, min, max, \ell + 1, A$)
  terminate while loop
end while
$classCount \leftarrow updateClassCount(\mathcal{D})$
return $Node(Left, Right, SplitAtt \leftarrow q, SplitValue \leftarrow mid_q, classCount)$

5 Empirical Evaluation

This section presents the results of the experiments conducted to evaluate the performance of MassBayes against seven well known contenders: two variants of NB (NB-KDE and NB-Disc), BayesNet, three variants of AnDE (A1DE, A2DE, A3DE) and decision tree J48 (i.e., the WEKA [11] version of C4.5 [12]).

MassBayes was implemented in Java using the WEKA platform [11] which also has implementations of NB, BayesNet, A1DE and J48. For A2DE and A3DE, we used the WEKA implementations provided by the authors of AnDE.

All the experiments were conducted using a 10-fold cross validation in a Linux machine with 2.27 GHz processor and 100 GB memory. The average accuracy (%) and the average runtime (seconds) over a 10-fold cross validation were reported. A two-standard-error significance test was conducted to check whether the difference in accuracies of two classifiers was significant. A win or loss was counted if the difference was significant; otherwise, it was a draw.

Ten data sets with $N > 10000$ were used. All the attributes in the data sets are numeric. The properties of the data sets are provided in Table 2. The RingCurve, Wave and OneBig data sets were three synthetic data sets and the rest were real-world data sets from UCI Machine Learning Repository [13]. RingCurve and Wave are subsets of the RingCurve-Wave-TriGaussian data set used in [9] and OneBig is the data set used in [14].

Table 2. Properties of the data sets used

Data sets	#N	#d	#c	Data sets	#N	#d	#c
CoverType	581012	10	7	RingCurve	20000	2	2
MiniBooNE	129596	50	2	Letters	20000	16	26
OneBig	68000	20	10	Magic04	19020	10	2
Shuttle	58000	8	7	Mamograph	11183	6	2
Wave	20000	2	2	Pendigits	10992	16	10

For AnDE, BayesNet and NB-Disc, data sets were discretised by a supervised discretisation technique based on minimum entropy [15] as suggested by the authors of AnDE before building the classification models.

Two variants of MassBayes were used: MassBayes with ($\psi = 5000$) and MassBayes′ ($\psi = N$). The other two parameters t and h were set as default to 100 and 10, respectively.

For BayesNet, the parameter *'maximum number of parents'* was set to 100 to examine whether a large number of parents produces better results; and the parameter *'initialise as Naive Bayes'* was set to *'false'* to initialise an empty network structure. The default values were used for the rest of the parameters. All the other classifiers were executed with the default parameter settings.

Table 3. Average classification accuracies (%) over a 10-fold cross validation

Data sets	Mass Bayes′	Mass Bayes	A3 DE	A2 DE	A1 DE	Bayes Net	NB-KDE	NB-Disc	J48
CoverType	94.00	78.21	88.16	80.81	72.89	87.79	66.72	66.61	92.39
MiniBooNE	92.68	91.11	N/A*	91.48	89.58	90.25	86.07	86.29	90.47
OneBig	100.00	100.00	N/A*	99.81	99.69	99.99	99.98	99.97	99.84
Shuttle	99.89	99.89	99.94	99.94	99.85	99.93	92.68	94.36	99.97
Letters	96.63	95.63	95.11	94.31	88.81	86.97	74.21	73.94	87.92
RingCurve	100.00	100.00	99.99	99.99	99.99	99.99	99.27	99.48	99.91
Wave	100.00	100.00	78.51	78.51	78.51	78.51	77.91	78.51	99.79
Magic04	85.72	85.53	85.08	84.57	83.00	83.46	76.13	78.27	85.01
Mamograph	98.69	98.71	98.51	98.37	98.42	98.54	97.86	97.62	98.57
Pendigits	99.45	99.28	98.80	98.82	97.84	96.81	88.64	87.9	96.56

* Did not complete because of integer overflow error.

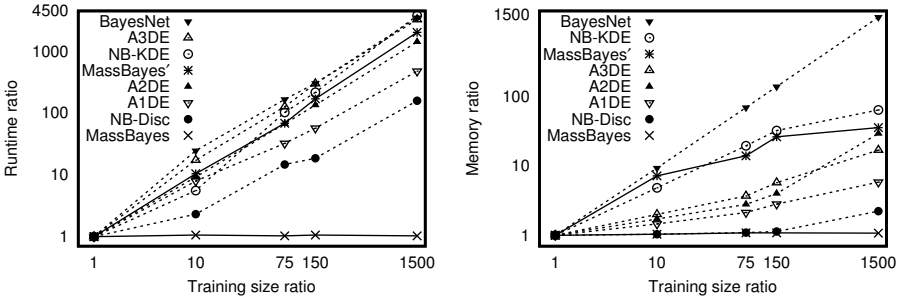
Table 4. Win:Loss:Draw counts of MassBayes over the other contenders in terms of classification accuracy based on the two-standard-error significance test

	A3DE	A2DE	A1DE	BayesNet	NB-KDE	NB-Disc	J48
MassBayes′	4:1:3	7:1:2	7:0:3	7:1:2	10:0:0	10:0:0	7:1:2
MassBayes	3:2:3	6:3:1	7:0:3	6:2:2	10:0:0	10:0:0	6:2:2

Table 5. Average runtime (seconds) over a 10-fold cross validation

Data sets	Mass Bayes′	Mass Bayes	A3 DE	A2 DE	A1 DE	Bayes Net	NB-KDE	NB-Disc	J48
CoverType	1075.8	45.7	45.6	13.9	4.9	387.9	96.3	3.2	3690.7
MiniBooNE	431.1	33.7	N/A	231.3	5.9	308.9	831.6	2.1	323.8
OneBig	113.9	10.5	N/A	11.6	3.9	432.5	253.0	0.8	15.1
Shuttle	48.5	8.0	1.8	0.7	0.5	6.8	1.5	0.4	4.2
Letters	18.9	5.5	11.5	2.6	0.8	4.9	2.5	0.4	7.3
RingCurve	4.4	2.3	0.2	0.2	0.2	0.3	2.4	0.2	0.4
Wave	4.9	2.1	0.2	0.2	0.2	0.2	2.5	0.1	0.6
Magic04	10.9	3.9	0.7	0.5	0.2	0.7	8.8	0.2	3.4
Mamograph	4.7	3.1	0.3	0.2	0.2	0.3	0.5	0.2	0.4
Pendigits	7.3	3.6	5.7	0.9	0.4	1.8	1.6	0.2	1.2

Table 3 shows the average classification accuracies of MassBayes′ and MassBayes in comparison to the other contenders. The results of the two-standard-error significance test in Table 4 show that both MassBayes′ and MassBayes produced better classification accuracy than the other contenders in most data sets.



(a) Runtime (b) Space

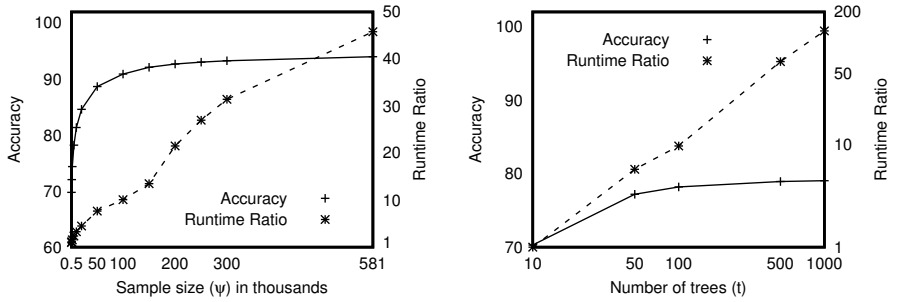
Fig. 3. Scale-up test: MassBayes versus existing generative classifiers. The base for training size ratio is 7000 instances and the bases for runtime ratio and memory ratio are the training time and memory required to save a classification model for 7000 instances. Axes are on logarithmic scales of base 10.

MassBayes produced slightly poorer results than A2DE, A3DE, BayesNet and J48 in CoverType. This was because the default sample size was not enough to yield a good estimate. The accuracy was increased up to 84.62% with $\psi = 20000$ and 88.66% with $\psi = 50000$. More samples are required to grow the trees further to model the distributions well if the class distributions in the feature space are complex. Figure 4(a) shows the improvement in accuracy of MassBayes in CoverType when the sample size was increased.

Table 5 presents the average runtime. In terms of runtime, MassBayes was an order of magnitude faster than A2DE in MiniBooNE; BayesNet in CoverType, MiniBooNE and OneBig; NB-KDE in MiniBooNE and OneBig; and J48 in CoverType and MiniBooNE. It was of the same order of magnitude as A3DE, A2DE, BayesNet, NB-KDE and J48 in many cases and an order of magnitude slower than NB-Disc and A1DE. MassBayes′ was an order of magnitude slower than the other contenders in many data sets. However, it was of the same order of magnitude as A3DE in Letters; A2DE in MiniBooNE; BayesNet and NB-KDE in MiniBooNE and OneBig; and J48 in CoverType and MiniBooNE.

Note that the reported runtime results for AnDE, BayesNet and NB-Disc did not include the discretisation time that must be done as a preprocessing step, which give the existing generative classifiers (except NB-KDE) an unfair advantage over MassBayes. The discretisation time can be substantially large in large data sets. For example, the discretisation took 52 seconds in the largest data set, CoverType. This discretisation time alone was more than the total runtime of MassBayes. Thus, MassBayes in effect runs faster than all existing generative classifiers on equal footing.

In order to examine the scalability of the classifiers in terms of training time and space requirements with the increase in training size N, we used the 48-dimensional (42 irrelevant attributes with constant values) RingCurve-Wave-Tri-Gaussian data set previously employed by Ting and Wells (2010) in [9]. The training data size was increased from 7000 to 70000, half-a-million, 1 million and



(a) Sample size (ψ) (b) Trees (t)

Fig. 4. Effect of parameters ψ and t on the classification accuracy and runtime of MassBayes in the CoverType data set. The base for the runtime ratio while varying ψ and t is the total runtime (training and testing over a 10-fold cross validation) for $\psi = 500$ and $t = 10$, respectively. The horizontal axis of t and the vertical axis of runtime ratio in (b) are on logarithmic scales of base 10.

10 million by a factor of 1, 10, 75, 150 and 1500, respectively. Figure 3 shows the increase in classification model building time and memory space required to store the classification model for different generative classifiers. Note that the discretisation time was not included in the presented results. The discretisation time increases linearly with the increase in training data size. This additional time for discretisation will increase the training time of AnDE, BayesNet and NB-Disc. MassBayes had constant training time and constant space requirements.

In order to examine the sensitivity of the parameters ψ, t and h in classification accuracy and runtime of MassBayes, we conducted a set of experiments by varying one parameter and fixing the other two to the default values. The result of the experiment varying ψ and t in the largest data set (CoverType) is shown in Figure 4. The increase in runtime was plotted as a ratio to show the factor of runtime increased when the parameters were increased.

In general, accuracy increased up to a certain point and remained flat when each of the three parameters was increased. This indicates that the parameters of MassBayes are not too sensitive in terms of classification accuracy if they are set to sufficiently high values. The runtime increased linearly with t and sublinearly with ψ. With fixed sample size ($\psi = 5000$), increase in h after a certain point did not affect the runtime because the tree building process stopped before reaching the maximum level h once the instances are separated.

6 Conclusions and Future Work

In this paper, we presented a new generative classifier called *MassBayes* that approximates $p(\mathbf{x}|y)$ by aggregating multi-dimensional likelihoods estimated using varying size subsets of features from random subsets of training data. In contrast, existing generative classifiers make assumptions about attribute independence and estimate single-dimensional likelihood only. Our empirical results

show that MassBayes produced better classification accuracy than the existing generative classifiers in large data sets.

In terms of runtime, it scales better than the existing generative classifiers in large data sets as it builds models in an ensemble using fixed-size data subsets. The constant training time and space complexities make it an ideal classifier for large data sets and data streams.

Future work includes applying the proposed method in data sets with discrete and mixed attributes and investigating the effectiveness of MassBayes in the data stream context. In this paper, we have rigorously assessed MassBayes with the state-of-the-art Bayesian classifiers. In the near future, we will assess its performance against some well-known discriminative classifiers and their ensembles. The feature space partitioning can be implemented in various ways. It would be interesting to investigate a more intelligent way of feature space partitioning rather than dividing at mid-point of a randomly selected dimension.

Acknowledgement. This work is supported by the Air Force Research Laboratory, under agreement# FA2386-11-1-4112. Sunil Aryal is partially supported by Monash University Postgraduate Publications Award to write this paper. We would like to thank Geoff Webb and the anonymous reviewers for their helpful comments.

References

1. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapmal & Hall/CRC (1986)
2. Ram, P., Gray, A.G.: Density Estimation Trees. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 627–635. ACM, New York (2011)
3. Langley, P., Iba, W., Thompson, K.: An Analysis of Bayesian Classifiers. In: Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 399–406 (1992)
4. Langley, P., John, G.H.: Estimating continuous distribution in Bayesian classifiers. In: Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence (1995)
5. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning 29, 131–163 (1997)
6. Webb, G.I., Boughton, J.R., Wang, Z.: Not So Naive Bayes: Aggregating one-dependence estimators. Machine Learning 58, 5–24 (2005)
7. Webb, G., Boughton, J., Zheng, F., Ting, K., Salem, H.: Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. Machine Learning 86, 233–272 (2012)
8. Chickering, D.M.: Learning Bayesian Networks is NP-Complete. In: Fisher, D., Lenz, H.J. (eds.) Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer, Heidelberg (1996)
9. Ting, K.M., Wells, J.R.: Multi-Dimensional Mass Estimation and Mass-Based Clustering. In: Proceedings of IEEE ICDM, pp. 511–520 (2010)

10. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features. In: Proceedings of the 12th International Conference on Machine Learning, pp. 194–202. Morgan Kaufmann (1995)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
12. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
13. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2010), http://archive.ics.uci.edu/ml
14. Nanopoulos, A., Theodoridis, Y., Manolopoulos, Y.: Indexed-based density biased sampling for clustering applications. IEEE Transaction on Data and Knowledge Engineering 57(1), 37–63 (2006)
15. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous valued attributes for classification learning. In: Proceedings of 14th International Joint Conference on Artificial Intelligence, pp. 1034–1040 (1995)