Comparative analysis of the impact of discretization on the classification with Naive Bayes and semi-Naïve Bayes classifiers

Marcin Mizianty[1,2], Lukasz Kurgan[2], and Marek Ogiela[3]
[1]*Faculty of Physics and Applied Computer Science, AGH Univ. of Science and Technology, Krakow, Poland*
[2] *Department of Electrical and Computer Engineering, Univ. of Alberta, Edmonton, Canada*
[3] *Institute of Automatics, AGH Univ. of Science and Technology, Krakow, Poland*
marcin.mizianty@gmail.com, lkurgan@ece.ualberta.ca, mogiela@agh.edu.pl

Abstract

While data could be discrete and continuous (defined as ordinal numerical features), *some classifiers, like Naïve Bayes (NB), work only with or may perform better with the discrete data. We focus on NB due to its popularity and linear training time. We investigate the impact of eight discretization algorithms (Equal Width, Equal Frequency, Maximum Entropy, IEM, CADD, CAIM, MODL, and CACC) on the classification with NB and two modern semi-NB classifiers, LBR and AODE.*

Our comprehensive empirical study indicates that unsupervised discretization algorithms are the fastest while among the supervised algorithms the fastest is Maximum Entropy, followed by CAIM and IEM. The CAIM and MODL discretizers generate the lowest and the highest number of discrete values, respectively.

We compare the time to build the classification model and classification accuracy when using raw and discretized data. We show that discretization helps to improve the classification with the NB when compared with Flexible NB which models continuous features using Gaussian kernels. The AODE classifier obtains on average the best accuracy, while the best performing setup includes discretization with IEM and classification with AODE. The runner-up setups include CAIM and CACC coupled with AODE and CAIM and IEM coupled with LBR. IEM and CAIM are shown to provide statistically significant improvements across all considered datasets for LBR and AODE classifiers when compared with using NB on the continuous data. We also show that the improved accuracy comes at the trade-off of substantially increased runtime.

1. Introduction

Discretization of continuous features (ordinal numerical features) has been extensively studied in the past two decades [3-5,9,11,13,14,18,20-23,25-29,30-33,36,37]. Discretization algorithms were found useful in developing decision tree methods, in computing conditional probability tables in Bayesian networks [3], and in implementing rule-based classifiers [24]. Some classification algorithms, like AQ [16], CLIP [6,7], CN2 [8], and DataSqueezer [24], work only with discrete data. Some other classifiers that can handle continuous features may perform better with discrete-valued features [5,23,27,30]. The discretization methods have been evaluated mainly when coupled with decision tree classifiers, including ID3 [4], C4.5 [9,10,19,25-28,31], and C5.0 [22,23,31,32]. A few works also investigated the application of discretization methods to classification with AQ [4] and CLIP4 classifiers [6,22,23], and nearest neighbor and logistic regression [1].

We focus on the family of methods based on the Naïve Bayes (NB) algorithm. The two major advantages of NB classifiers are their scalability, i.e., the classification model is learned in linear time with respect to the number of training examples, and no need for parameterizations, i.e., NB is parameterless. The NB is appealing due to its simplicity, elegance, and robustness, which is why it was included in the recent list of the top 10 data mining algorithms [38]. This classifier works only with nominal or discrete features; the continuous features are estimated using a distribution. Applying discretization algorithms as the front-end for the NB outperforms the NB that uses normal distribution to model continuous features [9]. A few recent studies also show the positive impact of several discretizers on the accuracy of the NB models [12,25]. These studies were limited to the classical NB algorithm. Recently several semi-NB algorithms, i.e., algorithms that relax the requirement of conditional independence, which have the same linear complexity and improved accuracy, were proposed [17,34,39]. Although one study explored the impact of discretization on the performance of a few NB variants [1], it was limited to a few medical datasets and investigated only three older discretization methods [11]. We investigate the impact of using a representative set of eight modern discretization algorithms on the quality of models generated by both NB and semi-NB classifiers from seven benchmark datasets.

2. Background

2.1. Discretization

A discretization scheme D on a feature F converts the domain of values of F into d disjoint discrete subintervals, bound by a pair of values (boundary points)

D: { $[d_1;d_2]$, …,$(d_i;d_{i+1}]$, …, $(d_{d-1};d_d]$ }

The discretization algorithms can be categorized as *supervised* versus *unsupervised*, *global* versus *local*, *top-down* (splitting) versus *bottom-up* (merging), and *direct* versus *incremental* [27]. Unsupervised algorithms compute the boundary points given the knowledge of the values of F while supervised algorithms also use the corresponding class labels. Local methods compute/adjust the boundaries using a subset of the training examples at a given time, while global algorithms use all training examples. The search for an "optimal" discretization scheme could start with all potential boundary points (usually assumed as all values of F or all midpoints between neighboring sorted values of F) and successively merge the neighboring intervals (bottom-up/merging algorithms). The top-down/splitting algorithms start from one interval that covers the entire range of values of F and divide it by adding new boundary points. The merging and splitting of intervals is accomplished with the use of a discretization criterion, which estimated whether a given merger/division improves D. The direct algorithms require a user-defined final/initial number of intervals in D, while incremental methods find this number on their own.

2.1.1. Unsupervised discretizers. The two unsupervised algorithms include Equal Width and Equal Frequency. Equal Width finds a minimal and a maximal value of F and divides the corresponding interval into d equally wide intervals. The number of intervals is user-defined or it can be estimated as $d = M / 3k$, where M is the number of distinct values of F and k is the number of classes [36]. Equal frequency sorts the values of F and computes d intervals that contain the same number of values.

2.1.2. Supervised discretizers. The supervised algorithms use discretization criteria to add, remove, and adjust boundary points. The Maximum Entropy method [36], the Information Entropy Maximization (IEM) algorithm [11], and the Class-Attribute Dependency Discretization (CADD) algorithm [4] and the above two unsupervised methods were used as a benchmark for several modern discretizers [1,3,21-23,27] and thus they are included in our study. We also included several modern methods. The Class-Attribute Interdependence Maximization (CAIM) algorithm is a top-down method that iteratively adds boundary points by accepting a boundary, from among all midpoints, that maximizes a custom designed CAIM criterion [23]. The MODL algorithm is based on Bayesian approach and thus it seems the best suited towards application with NB classifiers [3]. Finally, one of the newest discretization methods, Class-Attribute Contingency Coefficient (CACC), is based on maximizing a value of a novel CACC criterion [32]. The considered discretizers are summarized in Table 1. In the case of both unsupervised algorithms, Maximum Entropy, and CADD algorithms $d = M / 3k$ [36]; the remaining methods are incremental.

Table 1. Summary of the considered discretization methods.

Name	Ref.	Characteristics	Criterion
Equal Width	N/A	unsupervised, splitting, global, direct	N/A
Equal Freq.	N/A	unsupervised, splitting, global, direct	N/A
Max.Entropy	[36]	supervised, splitting, global, direct	Entropy
IEM	[11]	supervised, splitting, local, incremental	MDLP
CADD	[4]	supervised, splitting/merging, global, direct	CAIR
CAIM	[23]	supervised, splitting, global, incremental	CAIM
MODL	[3]	supervised, merging, global, incremental	MODL
CACC	[32]	supervised, splitting, global, incremental	CACC

2.2. Naïve Bayes and semi-Naïve Bayes classifiers

2.2.1. Naïve Bayes. Using a training set of t examples described by n features, we predict the class label $y \in c_1, \ldots, c_k$ of a test example x = $<x_1, \ldots, x_n>$, where x_i is the value of the i^{th} feature and k is the number of class labels. NB assumes that the features are independent given the class label and performs classification using

$$\arg\max_y (P'(y)\prod_{i=1}^{n} P'(x_i \mid y))$$

where P'(y) and P'($x_i \mid y$) are estimates of the respective probabilities derived from the training set. For discrete features the conditional probabilities correspond to the probability that i^{th} feature takes a particular value x_i when the class label y is c_i. Continuous features are modeled using Gaussian distribution. The maximum likelihood estimates of the mean and the standard deviation of the normal distributions are based on the sample average and standard deviation for each c_i. The estimates may lead to classification errors when the continuous features do not obey the Gaussian distribution. Therefore, in the flexible Naïve Bayes (FNB) [15], the distribution is averaged over a set of Gaussian kernels, which helps reducing the errors.

2.2.2. Lazy Bayes Rule. The NB was extended in the framework of lazy learning by relaxing the assumption of feature independence [39]. The Lazy Bayes Rule (LBR) method performs classification using

$$\arg\max_y (P'(y \mid W)\prod_{i=1}^{n} P'(x_i \mid y, W))$$

where W is a subset of features, and when assuming independence among the remaining features given W and y. W is selected using a heuristic wrapper that aims at minimizing error on the training set [39].

2.2.4. Aggregating one-dependence estimators. The aggregating one-dependence estimators (AODE) algorithm [34] uses an ensemble of 1-dependence NB classifiers to perform classification, i.e.

$$\arg\max_y (\sum_{i:1\le i\le n \wedge F(x_i)\ge 30} P'(y \mid x_i)\prod_{j=1}^{n} P'(x_j \mid y, x_i))$$

where $F(x_i)$ is count of the number of training examples having the attribute-value x_i [34].

NB, FNB, and LBR have O(tn), and AODE has O(tn^2) complexity, i.e., the classification model in computed in linear time. Both AODE and LBR require front-end discretization of the continuous features.

Table 2. Summary of the benchmark datasets.

Dataset	# features (real/integer)	# classes	# examples	Avg. # distinct values	ratio # examples / # values
glass	9 (9/0)	7	214	97.244	2.2
ionosphere	34 (34,0)	2	351	216.106	1.6
pendigits	16 (0,16)	10	10990	100.419	109.4
sat	36 (0,36)	7	6433	77.469	83.0
segment	19 (18,0)	7	2310	632.784	3.7
sonar	60 (60,0)	2	208	170.148	1.2
vehicle	18 (0,18)	4	846	77.433	10.9

3. Experimental results

The NB, FNB, LBR, and AODE algorithms are implemented in WEKA [35]. They were used to perform classification on 7 datasets from UCI repository [2], see Table 2. The datasets include only continuous features, i.e., ordinal real numbers and ordinal integers, and cover a wide range of problem sizes. The NB and FNB were used on the raw data (without discretization), and the four classifiers were used on datasets discretized with eight discretizers including Equal Width, Equal Frequency, Maximum Entropy, IEM, CADD, CAIM, MODL, and CACC (we use in-house implementation of these methods). We perform tenfold cross-validation where generation of the discretization and classification models is based on the same training folds.

3.1. Comparison of discretization algorithms

The eight discretizers were compared based on average runtime (over the ten folds) and average (over all features in a given dataset and the ten folds) number of generated intervals, see Table 3. The results show, as expected, that unsupervised discretizers are the fastest. The fastest supervised algorithm is Maximum Entropy, which is followed by CAIM and IEM. The differences in runtime are up to three orders of magnitude, e.g., for the *segment* dataset, Equal Width and Equal Entropy execute in 17 milliseconds while the slowest CADD takes almost 27 thousand milliseconds to converge. The CACC method performs poorly on the *glass* and *sonar* datasets. These datasets have relatively low ratio of # examples to # of distinct values (see Table 2), i.e., they include large number of distinct continuous values. MODL algorithm is the worst on the *vehicle*, *sat*, and *pendigits* datasets, which are among the datasets with the highest abovementioned ratio, i.e., these datasets include large number of redundant continuous values. Finally, CADD is the slowest on the *ionosphere*, *segment*, and *sonar* datasets.

The results concerning the number of generated discretization intervals indicate that CAIM on average performs the best. CACC and CADD also generate relatively low numbers of intervals. The remaining incremental algorithm, MODL, generates on average the largest number of intervals. The two algorithms that require the user defined initial number of intervals (CADD and IEM) reduce this initial number, i.e., their average (across all datasets) ranks are lower than the ranks of Equal Width and Equal Frequency algorithms, which are initialized with the same number of intervals.

3.2. Classification with Naïve Bayes and semi-Naïve Bayes algorithms

The eight discretizers were used on the data inputted into NB and two semi-NB classifiers. Their accuracies and the accuracy of NB and FNB classifiers on continuous data were compared in Table 4. For each setup (each classifier and input raw or discretized data obtained with a given discretization algorithm), we report the average (over the ten folds) accuracy and rank for each dataset. We investigate statistical significance of the differences in the accuracy over the ten folds by comparing a given setup with results obtained with the NB classifier on the raw data. We used paired t-test for normally distributed data and Mann-Whitney u-test for the nonparametric data. The normality was tested using Shapiro-Wilk test. We assume 95% confidence level for all tests.

Table 3. The average runtime (in milliseconds) and number of intervals associated with the discretization performed by the considered 8 discretization algorithms. The values in round brackets indicate standard deviation (over the 10 folds), the values in square brackets show rank of a given discretization algorithm for a given dataset, underlined values and shaded cells indicate best results.

	Algorithms	Datasets							Avg. rank
		glass	ionosphere	pendigits	sat	segment	sonar	vehicle	
Avg. runtime	Equal Width	0.6 (0.06)[2]	<u>3.8</u> (0.05)[1]	91.3 (6.22)[2]	<u>101.2</u> (0.42)[1]	<u>17.3</u> (0.15)[1]	5.3 (4.86)[2]	<u>5.1</u> (0.07)[1]	<u>1.4</u>
	Equal Frequency	<u>0.6</u> (0.02)[1]	3.8 (0.05)[2]	<u>89.8</u> (5.63)[1]	101.2 (0.48)[2]	17.3 (0.15)[2]	<u>3.7</u> (0.01)[1]	6.4 (4.01)[2]	1.6
	Maximum Entropy	8.2 (0.32)[3]	203.4 (7.52)[5]	916.4 (19.61)[3]	739.8 (9.05)[3]	2163.0 (128.88)[3]	220.4 (15.6)[5]	61.3 (4.32)[3]	3.6
	IEM	20.4 (0.50)[4]	141.5 (8.04)[4]	4397.3 (23.17)[7]	3490.2 (29.98)[7]	3750.9 (22.37)[4]	101.8 (4.55)[3]	177.8 (4.14)[6]	5.0
	CADD	23.7 (1.37)[5]	3726.3 (185)[8]	1507.5 (43.77)[4]	1343.5 (20.95)[4]	26997.3 (748.47)[8]	2606.6 (30.69)[8]	343.5 (35)[7]	6.3
	CAIM	32.1 (0.33)[6]	127.5 (5.56)[3]	4314.4 (22.07)[6]	2993.7 (19.03)[5]	4891.7 (15.51)[5]	110.1 (6.84)[4]	119.1 (0.77)[4]	4.7
	MODL	71.0 (3.83)[7]	344.7 (6.56)[7]	7001.8 (144.9)[8]	6372.6 (58.39)[8]	5266.3 (193.78)[7]	447.2 (8.99)[6]	525.9 (26.36)[8]	7.3
	CACC	153.9 (17.0)[8]	209.0 (9.87)[6]	4306.0 (15.25)[5]	3025.9 (17.59)[6]	4904.9 (26.71)[6]	784.6 (115.73)[7]	135.3 (5.88)[5]	6.1
Avg. # of intervals	**Algorithms**	**glass**	**ionosphere**	**pendigits**	**sat**	**segment**	**sonar**	**vehicle**	**Avg. rank**
	Equal Width	7.1 (0.04)[5]	36.4 (0.42)[6]	10.0 (0.0)[3]	7.0 (0.0)[3]	31.3 (0.11)[7]	28.7 (0.05)[6]	7.6 (0.05)[7]	5.3
	Equal Frequency	7.1 (0.04)[5]	36.4 (0.42)[6]	10.0 (0.0)[3]	7.0 (0.0)[3]	30.9 (0.11)[6]	28.7 (0.05)[6]	7.6 (0.05)[7]	5.1
	Maximum Entropy	5.2 (0.08)[3]	31.7 (0.64)[5]	4.0 (0.0)[2]	4.0 (0.0)[2]	30.7 (0.1)[5]	28.7 (0.05)[5]	6.9 (0.07)[6]	4.0
	IEM	<u>2.5</u> (0.06)[1]	3.7 (0.12)[3]	10.2 (0.15)[7]	12.0 (0.13)[7]	8.9 (0.13)[3]	<u>1.4</u> (0.02)[1]	<u>3.9</u> (0.1)[1]	3.3
	CADD	5.0 (0.11)[2]	24.1 (0.58)[4]	<u>3.8</u> (0.03)[1]	<u>3.8</u> (0.02)[1]	25.4 (0.13)[4]	28.0 (0.1)[4]	6.5 (0.1)[5]	3.0
	CAIM	7.0 (0.0)[4]	<u>1.9</u> (0.0)[1]	10.0 (0.03)[6]	7.0 (0.0)[3]	<u>6.2</u> (0.02)[1]	2.0 (0.0)[2]	4.0 (0.0)[2]	<u>2.7</u>
	MODL	43.9 (5.37)[8]	48.2 (2.63)[8]	11.8 (0.22)[8]	13.1 (0.21)[8]	50.8 (2.85)[8]	49.7 (3.56)[8]	6.1 (0.91)[4]	7.4
	CACC	24.7 (2.95)[7]	3.2 (0.14)[2]	10.0 (0.0)[3]	7.0 (0.0)[3]	<u>6.2</u> (0.02)[1]	16.7 (2.81)[3]	4.2 (0.07)[3]	3.1

Table 4. The average accuracy for a given classification setup (a given classifier executed on raw or discretized data). The values in round brackets indicate standard deviation (over the 10 folds), square brackets show rank of a given setup for a given dataset, underlined values and shaded cells show best results. Second line shows p-values inside round brackets; we assume 95% confidence level; ++/--/~ indicates that a given setup is significantly better/ worse/indifferent when compared with NB on the raw data.

Classifier	Discretization Algorithm	Datasets							Avg. rank
		glass	ionosphere	pendigits	sat	segment	sonar	vehicle	
NB	Original Data	0.47 (0.09)[26]	0.83 (0.1)[26]	0.86 (0.01)[24]	0.80 (0.02)[25]	0.80 (0.02)[26]	0.69 (0.09)[25]	0.46 (0.05)[26]	25.4
FNB	Original Data	0.52 (0.10)[25] ~ (0.292)	0.92 (0.05)[3] ++ (0.023)	0.88 (0.005)[17] ++ (<0.001)	0.82 (0.02)[17] ++ (0.008)	0.86 (0.02)[25] ++ (<0.001)	0.72 (0.07)[18] ~ (0.524)	0.61 (0.07)[19] ++ (<0.001)	17.7
NB / FNB	Equal Width	0.60 (0.10)[23] ++ (0.007)	0.89 (0.05)[15] ~ (0.099)	0.87 (0.006)[23] ++ (0.015)	0.80 (0.01)[24] ~ (0.774)	0.90 (0.02)[21] ++ (<0.001)	0.75 (0.07)[10] ~ (0.113)	0.60 (0.07)[23] ++ (<0.001)	19.9
	Equal Freq.	0.71 (0.09)[11] ++ (<0.001)	0.88 (0.05)[24] ~ (0.174)	0.87 (0.005)[22] ++ (0.013)	0.80 (0.02)[23] ~ (0.350)	0.89 (0.01)[23] ++ (<0.001)	0.74 (0.04)[15] ~ (0.116)	0.60 (0.07)[22] ++ (<0.001)	20.0
	Max. Entropy	0.63 (0.10)[22] ++ (0.001)	0.89 (0.06)[22] ~ (0.120)	0.84 (0.01)[26] -- (0.016)	0.77 (0.02)[26] -- (0.001)	0.88 (0.02)[24] ++ (<0.001)	0.75 (0.07)[9] ~ (0.108)	0.57 (0.07)[25] ++ (<0.001)	22.0
	IEM	0.72 (0.11)[7] ++ (<0.001)	0.90 (0.04)[6] ~ (0.052)	0.87 (0.006)[18] ++ (0.001)	0.82 (0.02)[20] ++ (0.005)	0.91 (0.01)[17] ++ (<0.001)	0.78 (0.08)[6] ++ (0.030)	0.61 (0.08)[20] ++ (<0.001)	13.4
	CADD	0.66 (0.11)[20] ++ (<0.001)	0.89 (0.05)[15] ~ (0.099)	0.85 (0.01)[25] ~ (0.574)	0.81 (0.01)[22] ++ (0.034)	0.90 (0.02)[22] ++ (<0.001)	0.70 (0.08)[20] ~ (0.798)	0.58 (0.09)[24] ++ (<0.001)	21.1
	CAIM	0.68 (0.11)[17] ++ (<0.001)	0.89 (0.07)[20] ~ (0.138)	0.87 (0.004)[21] ++ (0.009)	0.82 (0.02)[21] ++ (0.014)	0.90 (0.01)[20] ++ (<0.001)	0.76 (0.07)[7] ~ (0.070)	0.62 (0.07)[17] ++ (<0.001)	17.6
	MODL	0.76 (0.06)[1] ++ (<0.001)	0.89 (0.04)[18] ~ (0.91)	0.87 (0.004)[19] ++ (0.003)	0.82 (0.02)[19] ++ (0.007)	0.91 (0.01)[19] ++ (<0.001)	0.70 (0.10)[23] ~ (0.908)	0.60 (0.07)[21] ++ (<0.001)	17.1
	CACC	0.71 (0.12)[8] ++ (<0.001)	0.89 (0.05)[13] ~ (0.090)	0.87 (0.003)[20] ++ (0.008)	0.82 (0.02)[18] ++ (0.011)	0.91 (0.01)[16] ++ (<0.001)	0.75 (0.10)[13] ~ (0.193)	0.61 (0.08)[18] ++ (<0.001)	15.1
LBR	Equal Width	0.60 (0.11)[24] ++ (0.009)	0.89 (0.05)[23] ~ (0.114)	0.96 (0.006)[9] ++ (<0.001)	0.87 (0.01)[12] ++ (<0.001)	0.92 (0.02)[14] ++ (<0.001)	0.74 (0.07)[16] ~ (0.207)	0.66 (0.06)[16] ++ (<0.001)	16.3
	Equal Freq.	0.70 (0.09)[14] ++ (<0.001)	0.88 (0.05)[24] ~ (0.187)	0.97 (0.01)[7] ++ (<0.001)	0.87 (0.01)[11] ++ (<0.001)	0.93 (0.02)[13] ++ (<0.001)	0.71 (0.07)[19] ~ (0.373)	0.67 (0.05)[15] ++ (<0.001)	14.7
	Max. Entropy	0.70 (0.10)[15] ++ (<0.001)	0.89 (0.06)[13] ~ (0.093)	0.96 (0.005)[16] ++ (<0.001)	0.84 (0.02)[16] ++ (<0.001)	0.92 (0.02)[15] ++ (<0.001)	0.68 (0.10)[26] ~ (0.730)	0.68 (0.08)[10] ++ (<0.001)	15.9
	IEM	0.74 (0.10)[5] ++ (<0.001)	0.90 (0.03)[12] ++ (0.0494)	0.96 (0.01)[14] ++ (<0.001)	0.86 (0.02)[13] ++ (<0.001)	0.94 (0.01)[7] ++ (<0.001)	0.79 (0.08)[5] ++ (0.023)	0.70 (0.06)[4] ++ (<0.001)	8.6
	CADD	0.67 (0.11)[19] ++ (<0.001)	0.89 (0.05)[15] ~ (0.099)	0.96 (0.005)[13] ++ (<0.001)	0.87 (0.01)[9] ++ (<0.001)	0.93 (0.01)[11] ++ (<0.001)	0.70 (0.08)[20] ~ (0.798)	0.68 (0.05)[11] ++ (<0.001)	14.0
	CAIM	0.68 (0.11)[17] ++ (<0.001)	0.93 (0.04)[1] ++ (0.008)	0.96 (0.005)[12] ++ (<0.001)	0.88 (0.02)[8] ++ (<0.001)	0.94 (0.01)[4] ++ (<0.001)	0.80 (0.06)[3] ++ (0.008)	0.70 (0.06)[6] ++ (<0.001)	7.3
	MODL	0.75 (0.06)[2] ++ (<0.001)	0.89 (0.04)[18] ~ (0.091)	0.96 (0.004)[10] ++ (<0.001)	0.86 (0.02)[14] ++ (<0.001)	0.94 (0.01)[8] ++ (<0.001)	0.70 (0.10)[23] ~ (0.908)	0.69 (0.05)[8] ++ (<0.001)	11.9
	CACC	0.71 (0.12)[8] ++ (<0.001)	0.89 (0.05)[21] ~ (0.111)	0.96 (0.005)[11] ++ (<0.001)	0.87 (0.02)[10] ++ (<0.001)	0.94 (0.02)[6] ++ (<0.001)	0.75 (0.10)[13] ~ (0.193)	0.70 (0.09)[5] ++ (<0.001)	10.6
AODE	Equal Width	0.65 (0.13)[21] ++ (0.002)	0.90 (0.04)[10] ++ (0.049)	0.97 (0.005)[5] ++ (<0.001)	0.88 (0.01)[7] ++ (<0.001)	0.91 (0.02)[18] ++ (<0.001)	0.74 (0.07)[17] ~ (0.265)	0.67 (0.05)[14] ++ (<0.001)	13.1
	Equal Freq.	0.75 (0.14)[3] ++ (<0.001)	0.91 (0.04)[5] ++ (0.024)	0.97 (0.005)[4] ++ (<0.001)	0.88 (0.02)[6] ++ (<0.001)	0.93 (0.01)[11] ++ (<0.001)	0.76 (0.08)[8] ~ (0.087)	0.69 (0.05)[9] ++ (<0.001)	6.6
	Max. Entropy	0.71 (0.10)[10] ++ (<0.001)	0.90 (0.04)[9] ~ (0.083)	0.96 (0.005)[15] ++ (<0.001)	0.85 (0.02)[15] ++ (<0.001)	0.93 (0.01)[10] ++ (<0.001)	0.80 (0.08)[2] ++ (0.012)	0.68 (0.07)[13] ++ (<0.001)	10.6
	IEM	0.74 (0.12)[5] ++ (<0.001)	0.90 (0.04)[6] ++ (0.036)	0.98 (0.005)[2] ++ (<0.001)	0.89 (0.02)[3] ++ (<0.001)	0.95 (0.01)[2] ++ (<0.001)	0.79 (0.07)[4] ++ (0.019)	0.70 (0.04)[7] ++ (<0.001)	4.1
	CADD	0.71 (0.11)[12] ++ (<0.001)	0.90 (0.04)[10] ++ (0.045)	0.97 (0.006)[8] ++ (<0.001)	0.88 (0.02)[5] ++ (<0.001)	0.94 (0.01)[9] ++ (<0.001)	0.75 (0.05)[10] ~ (0.798)	0.68 (0.05)[12] ++ (<0.001)	9.4
	CAIM	0.69 (0.14)[16] ++ (<0.001)	0.92 (0.04)[2] ++ (0.010)	0.97 (0.005)[6] ++ (<0.001)	0.88 (0.01)[4] ++ (<0.001)	0.95 (0.01)[3] ++ (<0.001)	0.80 (0.07)[1] ++ (0.008)	0.71 (0.04)[2] ++ (<0.001)	4.9
	MODL	0.75 (0.09)[4] ++ (<0.001)	0.90 (0.03)[6] ++ (0.028)	0.98 (0.004)[1] ++ (<0.001)	0.89 (0.02)[2] ++ (<0.001)	0.94 (0.01)[5] ++ (<0.001)	0.70 (0.10)[21] ~ (0.908)	0.71 (0.06)[1] ++ (<0.001)	5.9
	CACC	0.70 (0.13)[13] ++ (<0.001)	0.91 (0.04)[4] ++ (0.023)	0.97 (0.004)[3] ++ (<0.001)	0.89 (0.02)[1] ++ (<0.001)	0.96 (0.01)[1] ++ (<0.001)	0.75 (0.09)[10] ~ (0.193)	0.71 (0.06)[3] ++ (<0.001)	5.0

The average rank given in the last column in Table 4 (over all datasets) shows that the classification with NB on the continuous data results in the poorest accuracy. The rank shows that AODE classifier obtains on average (over all datasets and discretization methods) the best results. The best performing setup includes discretization with the IEM algorithm and subsequent classification with the AODE classifier, which is closely followed by results when using CAIM and CACC discretizers and the AODE classifier. The best results for the LBR classifier are obtained when discretizing the data with CAIM and IEM algorithms. Among the unsupervised discretization methods, Equal Frequency performs better for both semi-NB classifiers and comparably well for the NB classifier.

Comparison of results obtained with NB and FNB classifiers on the raw data and results obtained by these classifiers on the discretized data (they are the same when working with the discrete data) reveals that FNB classifiers applied on the raw data in some cases performs better than when used with the discretized data, i.e., the modeling of continuous features using a set of Gaussian kernels works on average better than when using more sophisticated discretization methods such as CADD and Maximum Entropy. At the same time, discretization with IEM, MODL, and CACC algorithms helps to improve classification with NB.

The statistical tests show that setups that include IEM and CAIM discretizers provide statistically significant improvements over all considered datasets for both semi-NB classifiers when compared with using NB on the raw data. The two results that are statistically significantly worse than the results of NB on raw data concern classification with NB on the data discretized with Maximum Entropy method for the *pendigits* and *sat* datasets. Overall, the Maximum Entropy seems to perform the poorest among the considered supervised discretization algorithms. The largest number of setups fails to provide significant improvements on the *sonar* and the *ionosphere* datasets. These datasets have the highest relative numbers of distinct continuous values (smallest ratio shown in Table 2). This result could be due to the relatively good performance of NB on these datasets. We hypothesize that the large number of distinct values allows establishing a robust estimate of the Gaussian distribution that is used to model the continuous features.

Table 5. The average runtime (in milliseconds) for a given classification setup (a given classifier executed on raw or discretized data). The values in round brackets indicate standard deviation (over the 10 folds), the values in square brackets show rank of a given setup for a given dataset, underlined values and shaded cells indicate best results.

Classifier	**Discretization algorithm**	**Datasets**							**Avg. rank**
		Glass	**ionosphere**	**pendigits**	**sat**	**segment**	**sonar**	**vehicle**	
NB	Original Data	0.9 (0.06)[15]	5.5 (0.04)[20]	119.0 (11.33)[19]	140.0 (0.89)[22]	25.3 (1.26)[17]	5.6 (0.09)[19]	7.7 (0.51)[25]	19.6
FNB	Original Data	1.1 (0.13)[22]	7.3 (0.05)[21]	133.0 (0.60)[24]	155.0 (2.78)[24]	31.0 (0.56)[20]	7.9 (0.26)[20]	8.8 (0.07)[26]	22.4
NB / FNB	Equal Width	0.2 (0.01)[5]	0.7 (0.002)[6]	10.6 (0.06)[5]	12.4 (0.10)[3]	2.6 (0.09)[5]	0.7 (0.14)[8]	0.9 (0.03)[8]	5.7
	Equal Freq.	0.2 (0.002)[1]	0.7 (0.001)[5]	10.6 (0.05)[3]	12.4 (0.15)[5]	3.3 (2.37)[8]	0.7 (0.002)[4]	0.9 (0.004)[4]	4.3
	Max. Entropy	0.2 (0.001)[3]	0.7 (0.002)[4]	10.6 (0.06)[2]	12.4 (0.10)[4]	2.6 (0.04)[6]	0.7 (0.03)[6]	0.9 (0.003)[1]	3.7
	IEM	0.2 (0.005)[4]	1.0 (1.15)[8]	10.7 (0.10)[8]	12.6 (0.18)[7]	2.5 (0.01)[2]	0.7 (0.02)[1]	0.9 (0.03)[7]	5.3
	CADD	0.2 (0.001)[2]	0.7 (0.01)[3]	10.6 (0.14)[6]	12.3 (0.07)[1]	2.6 (0.02)[4]	0.7 (0.002)[5]	0.9 (0.003)[3]	3.4
	CAIM	0.2 (0.01)[7]	0.7 (0.002)[1]	10.6 (0.06)[4]	12.4 (0.05)[2]	2.5 (0.01)[1]	0.7 (0.04)[2]	0.9 (0.001)[2]	2.7
	MODL	0.2 (0.002)[8]	0.7 (0.002)[7]	10.6 (0.07)[7]	12.6 (0.31)[8]	2.6 (0.06)[7]	0.7 (0.01)[7]	0.9 (0.004)[6]	7.1
	CACC	0.2 (0.002)[6]	0.7 (0.004)[2]	10.6 (0.04)[1]	12.4 (0.10)[6]	2.5 (0.02)[3]	0.7 (0.03)[3]	0.9 (0.005)[5]	3.7
LBR	Equal Width	1.0 (0.03)[18]	3.7 (0.08)[17]	120.0 (0.51)[21]	101.0 (1.04)[17]	38.9 (4.02)[25]	4.9 (4.94)[18]	4.9 (0.02)[22]	19.7
	Equal Freq.	1.0 (0.04)[19]	4.0 (1.17)[18]	120.0 (0.23)[20]	101.0 (4.04)[18]	37.5 (0.38)[24]	3.4 (0.03)[14]	5.0 (0.11)[23]	19.4
	Max. Entropy	1.0 (0.02)[21]	3.4 (0.04)[16]	99.5 (1.25)[17]	101.0 (3.85)[16]	37.3 (0.32)[23]	3.4 (0.02)[13]	6.3 (4.01)[24]	18.6
	IEM	0.9 (0.02)[16]	1.9 (0.03)[11]	146.0 (2.92)[25]	130.0 (1.38)[22]	24.5 (0.26)[16]	1.8 (0.03)[10]	4.4 (0.06)[19]	16.8
	CADD	1.0 (0.01)[20]	3.1 (0.06)[15]	101.0 (1.03)[18]	98.0 (0.97)[15]	35.1 (3.95)[21]	3.4 (0.02)[12]	4.9 (0.09)[21]	17.4
	CAIM	1.0 (0.06)[17]	1.6 (0.005)[9]	122.0 (2.70)[22]	104.0 (6.03)[20]	21.4 (0.47)[14]	1.7 (0.01)[9]	3.9 (0.04)[17]	15.4
	MODL	2.1 (0.16)[25]	4.2 (0.12)[19]	148.0 (6.21)[26]	141.0 (7.75)[23]	50.2 (1.68)[26]	4.6 (0.18)[17]	4.7 (0.23)[20]	22.3
	CACC	1.6 (0.08)[24]	1.9 (0.04)[10]	122.0 (4.95)[23]	102.0 (2.71)[19]	21.2 (0.73)[13]	2.7 (0.16)[11]	4.1 (0.07)[18]	16.9
AODE	Equal Width	0.4 (0.003)[13]	21.6 (4.25)[24]	33.0 (0.19)[13]	88.9 (4.53)[12]	19.9 (0.28)[12]	44.5 (0.78)[22]	2.5 (0.16)[16]	16.0
	Equal Freq.	0.4 (0.003)[14]	22.1 (1.05)[25]	33.1 (0.34)[14]	88.2 (0.72)[11]	26.3 (0.31)[18]	45.2 (0.11)[23]	2.5 (0.02)[14]	17.0
	Max. Entropy	0.3 (0.015)[11]	19.8 (1.47)[23]	23.8 (0.08)[10]	58.8 (4.72)[10]	27.5 (0.21)[19]	45.3 (0.17)[24]	2.5 (0.28)[15]	16.0
	IEM	0.3 (0.004)[9]	2.8 (0.03)[14]	32.7 (1.55)[12]	183.0 (3.31)[25]	7.9 (0.07)[11]	4.2 (0.02)[15]	2.3 (0.02)[9]	13.6
	CADD	0.3 (0.003)[10]	14.6 (0.37)[22]	23.7 (0.13)[9]	55.3 (0.21)[9]	22.0 (0.22)[15]	45.9 (5.01)[25]	2.4 (0.04)[13]	14.7
	CAIM	0.4 (0.002)[12]	2.6 (0.03)[12]	32.1 (0.86)[11]	93.0 (4.70)[13]	7.0 (0.05)[9]	4.3 (0.02)[16]	2.3 (0.09)[10]	11.9
	MODL	3.8 (0.81)[26]	26.7 (2.84)[26]	37.3 (1.08)[16]	191.0 (2.94)[26]	36.8 (2.36)[22]	147.0 (51.49)[26]	2.4 (0.05)[12]	22.0
	CACC	1.4 (0.31)[23]	2.7 (0.03)[13]	34.0 (0.25)[15]	93.5 (0.78)[14]	7.1 (0.04)[10]	18.0 (3.73)[21]	2.3 (0.03)[11]	15.3

Table 5 shows a comparison of the runtime to build classification models. As expected, the runtime of NB and FNB with continuous data is worse than when using the discretized data. This is true except when using MODL discretizer and LBR and AODE classifiers, which is likely due to the large number of intervals produced by this discretization algorithm, see Table 3. Overall, the runtime is positively correlated with the number of discretization intervals, i.e., the smaller the number of intervals the shorter the runtime. The results reveal, as expected, that NB is the fastest, with the AODE and LBR taking the second and third spots, respectively. Although AODE is faster for the datasets with relatively low number of features (*glass*, *pendigits*, *segment*, and *vehicle*), LBR is either similarly fast or faster for the datasets with large number of features (*ionosphere*, *sonar*, and *sat*), which corroborates with the computational complexity of these two methods. LBR has substantially higher complexity of classification of the test examples, O(tkn^2), when compared with O(kn) and O(kn^2) achieved by NB and AODE, respectively [3], i.e. usually $t >> k$ and $t >> n$. Since this time is not included in Table 5, we anticipate that when considering the combined time (including the time to generate the model and to classify test examples) the difference between LBR and AODE would be larger.

We observe that the runtime of supervised discretizers is substantially longer than the time to compute the NB models. The differences are up to two orders of magnitude, and they indicate that the improved classification accuracy comes as the trade-off of significantly increased runtime. The long runtime of discretization algorithms is due to their poor (when compared with NB classifier) complexity, e.g. CAIM, CACC, and MODL have O($t\log(t)$) complexity.

4. Summary and conclusions

The comparison of the eight considered discretizers indicates that the unsupervised algorithms are faster than the supervised methods. The fastest supervised algorithm is Maximum Entropy, followed by CAIM and IEM algorithms. We show that CAIM and MODL discretizers generate the lowest and the highest number of discretization intervals, respectively.

Our empirical comparison shows that discretization helps to improve the subsequent classification with the NB when compared with applying Flexible NB on the continuous features. The classification performed with NB on continuous data is characterized by the poorest accuracy, while AODE classifier obtains on average the most accurate results. The best performing setup includes discretization with IEM algorithm and subsequent classification with AODE classifier, which is closely followed by CAIM and CACC discretizers used with AODE classifier. The highest accuracies for the LBR classifier are obtained when using CAIM and IEM discretizers. IEM and CAIM coupled with the LBR and AODE classifiers provide statistically significantly higher accuracies across all 7 datasets when compared with accuracies obtained with NB on the continuous data. We also show that the improved accuracy when using data discretized by supervised discretizers comes at the expense of substantially increased runtime.

References

[1] R. Abraham, J.B. Simha, S. Iyengar, "A comparative analysis of discretization methods for medical datamining with Naïve Bayesian classifier", *Proc. 9th Conf. Information Technology*, pp.235-236, 2007

[2] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, Irvine, CA: Univ. of California, School of Information and Computer Science, 2007 http://www.ics.uci.edu/~mlearn/MLRepository.html

[3] M. Boulle, "MODL: A Bayes optimal discretization method for continuous attributes", *Machine Learning*, 65(1), pp. 131-165, 2006

[4] J.Y. Ching, A.K. Wong, K.C. Chan, "Class-dependent discretization for inductive learning from continuous and mixed mode data", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7), pp. 641-651, 1995

[5] J. Catlett, "On changing continuous attributes into ordered discrete attributes", *Proc. European Working Session on Learning*, pp. 164-178, 1991

[6] K.J. Cios, L. Kurgan, "Hybrid inductive machine learning: an overview of clip algorithms", *New Learning Paradigms in Soft Computing*, L.C. Jain, J. Kacprzyk, ed., pp. 276-322, Physica-Verlag (Springer), 2001

[7] K.J. Cios, L. Kurgan, "CLIP4: hybrid inductive machine learning algorithm that generates inequality rules", *Information Sciences*, 163(1-3), pp.37-83, 2004

[8] P. Clark, T. Niblett, "The CN2 algorithm", *Machine Learning*, 3, pp. 261-283, 1989

[9] J. Dougherty, R. Kohavi, M. Sahami, "Supervised and unsupervised discretization of continuous features", *Proc. 12th Int. Conf. Machine Learning*, pp.194-202, 1995

[10] U. Fayyad, K. Irani, "On the handling of continuous-valued attributes in decision tree generation", *Machine Learning*, 8, pp. 87-102, 1992

[11] U. Fayyad, K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning", *Proc. 13th Int. Joint Conf. Artificial Intelligence*, pp. 1022-1027, 1993

[12] J.L. Flores, I. Inza, P. Larrañaga, "Wrapper discretization by means of estimation of distribution algorithms", *Intelligent Data Analysis*, 11(5), pp. 525-545, 2007

[13] X. Guan, X. Yi, Y. He, "Discretization of continuous interval-valued attributes in rough set theory and its application", *Proc. 6th Int. Conf. Machine Learning and Cybernetics*, pp. 3682-3686, 2007

[14] W. Huang, "Discretization of continuous attributes for inductive machine learning", M.Sc. thesis, Dept. Computer Science, Univ. of Toledo, Ohio, 1996

[15] G.H. John, P. Langley, "Estimating continuous distributions in Bayesian classifiers", *Proc. 11th Conf. Uncertainty in Artificial Intelligence*, pp. 338-345, 1995

[16] K.A. Kaufman, R.S. Michalski, "Learning from inconsistent and noisy data: the AQ18 approach," *Proc. 11th Int. Symp. Methodologies for Intelligent Systems*, 1999

[17] E. Keogh, M. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches", *Proc. Int. Workshop Artificial Intelligence and Statistics*, pp. 225-230, 1999

[18] R. Kerber, "ChiMerge: discretization of numeric attributes", *Proc. 9th Int. Conf. Artificial Intelligence*, pp. 123-128, 1992

[19] R. Kohavi, M. Sahami, "Error-based and entropy-based discretization of continuous features", *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, pp. 114-119, 1996

[20] J. Kujala, T. Elomaa, "Improved algorithms for univariate discretization of continuous features", *Proc Principles and Practice of Knowledge Discovery in Databases*, LNCS 4702, pp. 188-199, 2007

[21] L. Kurgan, K.J. Cios, "Discretization algorithm that uses class-attribute interdependence maximization," *Proc. 2001 Int. Conf. Artif. Intelligence*, pp. 980-987, 2001

[22] L. Kurgan, K.J. Cios, "Fast class-attribute interdependence maximization (CAIM) discretization algorithm", *Proc. Int. Conf. on Machine Learning and Applications*, pp. 30-36, 2003

[23] L. Kurgan, K.J. Cios, "CAIM discretization algorithm", *IEEE Trans. on Knowledge and Data Engineering*, 16(2), pp.145-153, 2004

[24] L. Kurgan, K.J. Cios, S. Dick, "Highly scalable and robust rule learner: performance evaluation and comparison", *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 36(1), pp. 32-53, 2006

[25] C.-H. Lee, "A Hellinger-based discretization method for numeric attributes in classification learning", *Knowledge-Based Systems*, 20(4), pp. 419-425, 2007

[26] H. Liu, R. Setiono, "Feature selection via discretization", *IEEE Trans. on Knowledge and Data Engineering*, 9(4), pp. 642-645, 1997

[27] H. Liu, F. Hussain, C. Tan, M. Dash, "Discretization: an enabling technique". *J. Data Mining and Knowledge Discovery*, 6(4), pp. 393-423. 2002

[28] X. Liu, H. Wang, "A discretization algorithm based on a heterogeneity criterion", *IEEE Trans. on Data and Knowledge Engineering*, 17(9), pp.1166-1173, 2005

[29] S. Mehta, S. Parthasarathy, H. Yang, "Toward unsupervised correlation preserving discretization", *IEEE Trans. on Knowledge and Data Engineering*, 17(9), pp. 1174-1185, 2005

[30] C-T. Su, J-H. Hsu, "An Extended Chi2 algorithm for discretization of real value attributes", *IEEE Trans. on Knowledge and Data Engineering*, 17(3), pp. 437-441, 2005

[31] F.E.H. Tay, J. Shen, "A modified Chi2 algorithm for discretization", *IEEE Trans. on Knowledge and Data Engineering*, 14(3), pp. 666-670, 2002

[32] C.-J. Tsai, C.-I. Lee, W.-P. Yang, "A discretization algorithm based on Class-Attribute Contingency Coefficient", *Information Sciences*, 178, pp.714-731, 2008

[33] K. Wang, B. Liu, "Concurrent discretization of multiple attributes", *Pacific-Rim Conf Artificial Intelligence*, pp.250-259, 1998

[34] G.I. Webb, J. Boughton, Z. Wang, "Not so naive Bayes: aggregating one-dependence estimators", *Machine Learning*, 58(1), pp. 5-24, 2005

[35] Witten I.H., Frank E., *Data Mining: Practical machine learning tools and techniques*, 2nd edition, Morgan Kaufmann, 2005

[36] A.K.C. Wong, D.K.Y. Chiu, "Synthesizing statistical knowledge from incomplete mixed-mode data", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9, pp. 796-805, 1987

[37] X. Wu, "A Bayesian discretizer for real-valued attributes," *Computer Journal*, 39(8), pp. 688-691, 1996

[38] X. Wu, V. Kumar, J.R. Quinlan, et al., "Top 10 algorithms in data mining", *Knowledge and Information Systems*, 14, pp. 1-37, 2008

[39] Z. Zheng, G.I. Webb, "Lazy learning of Bayesian rules", *Machine Learning*, 41(1), pp. 53-84, 2000