International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 27, No. 1 (2013) 1350002 (26 pages)
© World Scientific Publishing Company
DOI: 10.1142/S021800141350002X

A DISCRETIZATION METHOD BASED ON MAXIMIZING THE AREA UNDER RECEIVER OPERATING CHARACTERISTIC CURVE

MURAT KURTCEPHE* and H. ALTAY GÜVENIR†

Computer Engineering, Bilkent University
Ankara, 06800, Turkey
**kurtcephe@gmail.com*
†guvenir@cs.bilkent.edu.tr

Received 27 March 2012
Accepted 26 November 2012
Published 21 February 2013

Many machine learning algorithms require the features to be categorical. Hence, they require all numeric-valued data to be discretized into intervals. In this paper, we present a new discretization method based on the receiver operating characteristics (ROC) Curve (AUC) measure. Maximum area under ROC curve-based discretization (MAD) is a global, static and supervised discretization method. MAD uses the sorted order of the continuous values of a feature and discretizes the feature in such a way that the AUC based on that feature is to be maximized. The proposed method is compared with alternative discretization methods such as ChiMerge, Entropy-Minimum Description Length Principle (MDLP), Fixed Frequency Discretization (FFD), and Proportional Discretization (PD). FFD and PD have been recently proposed and are designed for Naïve Bayes learning. ChiMerge is a merging discretization method as the MAD method. Evaluations are performed in terms of M-Measure, an AUC-based metric for multi-class classification, and accuracy values obtained from Naïve Bayes and Aggregating One-Dependence Estimators (AODE) algorithms by using real-world datasets. Empirical results show that MAD is a strong candidate to be a good alternative to other discretization methods.

Keywords: Data mining; discretization; area under ROC curve.

1. Introduction

Many machine learning algorithms require all features to be categorical. In order to apply such algorithms to a dataset containing continuous features, the dataset needs to be preprocessed so that such continuous features are converted into categorical ones. This conversion is done by discretizing the range of the continuous feature into intervals, and mapping these intervals to unique categorical values. The discretization methods aim to find the proper cut-points that separate the intervals.

*Corresponding author. EECS, Case Western Reserve University, Cleveland, OH, 44106, USA.

Discretization can also be viewed as mechanism for generalization. Humans usually use discretization to name range of continuous values. For example, rather than referring to particular values for age, one can use terms such as child, adolescent, adult, middle aged, and so on.

Discretization methods have received great attention from researchers and different kinds of discretization methods based on different metrics have been proposed.[33] Recently, Wong[36] proposed a new hybrid discretization method and Yang and Webb[37] proposed two new discretization methods for the Naïve Bayes classifier. There are important reasons for this attention such as the inability of many machine-learning algorithms to work with continuous values. For example, Aggregating One-Dependence Estimators (AODE) is one of the algorithms used in this research that cannot process continuous features.[35] In addition, it has been shown by Dougherty *et al.*[7] that discretization methods improve the predictive performance and running time of the machine learning algorithms.

Liu *et al.*[21] categorized discretization algorithms on four axes. These dimensions include *supervised versus unsupervised*, *splitting versus merging*, *global versus local* and *dynamic versus static*. Simple methods such as equal-width or equal-frequency binning algorithms do not use class labels of instances during the discretization process.[16] These methods are called *unsupervised discretization methods*. In order to improve the quality of the discretization, methods that use the class labels are proposed and they are referred to as *supervised discretization methods*. Splitting methods try to divide a continuous space into a set of small intervals by finding proper cut-points, whereas merging methods handle each distinct point on the continuous space as an individual candidate for a cut-point and merge similar neighbors to form larger intervals. Some discretization methods process localized parts of the instance space during discretization. As an example, the C4.5 algorithm handles numerical features by using a discretization (binarization) method that is applied to subsets of the instance space.[29,30] These methods are called local methods. Methods that use the whole instance space of the attribute to be discretized are called global methods. Dynamic discretization methods use the whole attribute space during discretization and perform better on data with interrelations between attributes. Conversely, static discretization methods discretize attributes one by one and assume that there are no interrelations between attributes.

Splitting discretization methods usually aim to optimize measures such as entropy,[4,5,9,28,31,32] which aims to obtain pure intervals, dependency[15] or accuracy[6] of values placed into the bins. On the other hand, the merging algorithms proposed so far use the χ^2 statistic.[17,20,34] To the best of our knowledge, the receiver operating characteristics (ROC) curve has never been applied in the discretization domain.

In this paper, we propose a discretization method called maximum area under ROC curve-based discretization (MAD). According to the dimensions defined above, MAD is a supervised, merging, global and static discretization method. The next section provides a brief introduction to the receiver operating characteristics, ROC space and area under ROC curve as measure of performance. Section 3 presents the MAD method, in detail. Section 4 compares the MAD method with four other discretization algorithms on real-world datasets. The last section concludes with some future directions for improvement.

2. Receiver Operating Characteristics

The first application of ROC was in the analysis of radar signals during World War II.[18] Since then, it has been used in different areas such as signal detection theory and medicine.[11,22,39] It was applied to machine learning by Spackman[31] for the first time. According to Fawcett,[8] the ROC graph is a tool that can be used to visualize, organize and select classifiers based on their performance. It became a popular performance measure in the machine learning community after it has been realized that accuracy is often a poor metric to evaluate classifier performance.[19,25,27]

The ROC literature mostly deals with binary classification (two classes) problems. In binary classification, each instance I has one of the two different class labels, as **p** (positive) or **n** (negative). At the end of the classification process, some classifiers simply map each instance to a class label (discrete output). There are also classifiers that are able to estimate the probability of an instance belonging to a specific class (continuous valued output, also called *score* or *confidence*). Classifiers produce a discrete output represented by only one point in the ROC space since only one confusion matrix is produced from their classification output. However, continuous-output-producing classifiers can have more than one confusion matrix by applying some thresholds to predict class membership. Using such a classifier, all instances with a score which is greater than the threshold are predicted as **p** class and all others are predicted as **n** class. Therefore, for each threshold value one confusion matrix is obtained, and each confusion matrix corresponds to a ROC point in an ROC graph.

2.1. *ROC space*

An ROC space is a two-dimensional space that has a range between [0.0, 1.0] on both axes. In an ROC space, the y-axis represents the true positive rate (TPR) of a classification output and the x-axis represents the false positive rate (FPR) of an output.

In order to calculate TPR and FPR values, definitions of the elements in the confusion matrix should be given. The structure of a confusion matrix is shown in Fig. 1. The number of true positives (TP) and false positives (FP) are the most important elements of the confusion matrix in ROC graphs. TP is the number of correctly classified positive instances and FP is the number of negative instances which are classified as positive, falsely. The TPR and FPR values are calculated by using Eq. (1). In this equation N is the number of total negative instances and P is the number of total positive instances.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P}, \quad \mathrm{FPR} = \frac{\mathrm{FP}}{N}. \tag{1}$$

		Actual Class	
		p	**n**
Predicted Class:	**p**	TP	FP
	n	FN	TN
Column Totals:		**P**	**N**

Fig. 1. Structure of a confusion matrix.

2.2. *Obtaining the ROC curve*

As mentioned above, the classifiers producing continuous output can form a curve in the ROC graph as they are represented by more than one point in the graph. In order to calculate the ROC graph, different threshold values are selected and different confusion matrices are formed. By varying the threshold between $-\infty$ and $+\infty$ an infinite number of ROC points can be produced for a given classification output. Although this operation is computationally costly, it is possible to form the ROC curve more efficiently with other approaches. As proposed by Fawcett,[8] in order to calculate ROC curve more efficiently, classification scores are first sorted in increasing order. Starting from $-\infty$, each distinct score element is taken as a threshold, and TPR and FPR values are calculated using Eq. (1).

As an example, take the score values for test instances and actual class labels for a toy dataset given in Table 1. The ROC curve for this toy dataset is shown in Fig. 2. In this figure, each ROC point is given with the threshold value used to calculate it. The same ROC curve is obtained for threshold values when they are selected from the intervals shown in the figure, next to each ROC point. Starting from $-\infty$, nine different thresholds are used; the total number of threshold values is equal to $S+1$ where S is the number of distinct classifier scores in the dataset. With this simple method, it is possible to obtain the ROC curve in linear time.

2.3. *Area under ROC curve (AUC)*

ROC graphs are useful for visualizing the performance of a classifier but a scalar value is preferred to compare classifiers. In the literature, the area under the ROC curve (AUC) is proposed as a performance measure by Bradley.[3] According to the AUC measure, a classifier with higher AUC value performs better classification in general.

The ROC graph space is a one-unit square. The highest possible AUC value is 1.0 which represents the perfect classification. In ROC graphs a 0.5 AUC value means

Table 1. Toy dataset given with hypothetical scores.

Class Label	**n**	**n**	**n**	**p**	**p**	**n**	**p**	**p**	**p**
Score	−7	−3	0	0	4	7	8	10	11

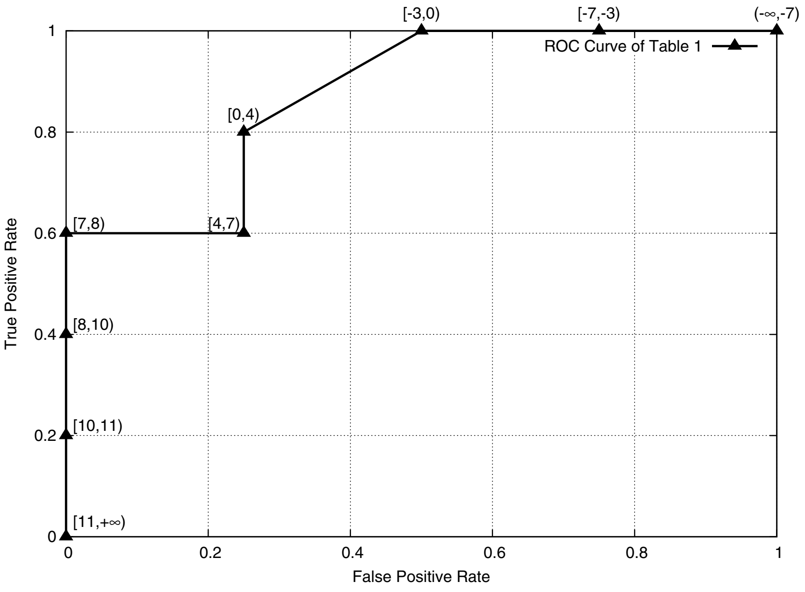

Fig. 2. ROC graph of the given toy dataset in Table 1.

random guessing has occurred. The values below 0.5 are not realistic as they can be negated by changing the decision criteria of the classifier.

The AUC value of a classifier is equal to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Hanley and McNeil[14] show that this is the same method employed by Wilcoxon test of ranks. This property of the AUC can be used to design a discretization algorithm. The details of such an algorithm will be given in the next section.

3. MAD Method

In this section, the details of the MAD method are discussed. First, the definition of concepts such as cut-points, ROC space and stopping criteria are presented. Next, the behavior of MAD in two-class datasets and multi-class datasets are examined in separate sections.

3.1. *Definition of cut-points*

Given an attribute A which has n instances with known values, let k be the number of distinct continuous values in the instance space of A. There are $k - 1$ candidate cut-points that can be used in discretization process. First, the instances are sorted (in this work in increasing order) according to their values for the attribute A. Then, each of the candidate cut-points is calculated by using Eq. (2).

$$C_i = (A_i + A_{i+1})/2, \tag{2}$$

where i is in $[1 \cdots k-1]$ and A_i and A_{i+1} are distinct and consecutive values in the sorted instance space.

3.2. *Definition of ROC space for discretization*

The numerical attribute values are taken as hypothetical classifier scores that are needed to obtain the ROC curve. In this matter, the cut-points calculated by Eq. (2) are used as the threshold values. According to Eq. (1), when the threshold value is $-\infty$, the TPR and FPR values are equal to 1 and corresponds to the coordinate (1,1) in the ROC space. The method will continue incrementally drawing the ROC curve by using each candidate cut-point as a threshold value. Finally, threshold $+\infty$ will be processed and the ROC point corresponding to the coordinate (0,0) will be obtained. The total number of ROC points for discretization is $k-1$ plus two for the trivial end points.

3.3. *Discretization measure*

As mentioned above different measures such as entropy, accuracy, dependency and the χ^2 statistic have been used in discretization methods. On the other hand, Provost *et al.* showed that AUC is a better performance metric for the comparison of the induction algorithms.[27] Therefore, in this work, the AUC of the ROC curve, obtained from these TPR and FPR values, is used as the measure to be optimized.

The motivation behind this approach is that an ROC curve results in a high AUC value when the **p**-labeled instances have higher score values then the **n**-labeled instances. Using this heuristic, the minimum number of ROC points that maximize the AUC value will be selected. That is, the minimum number of cut-points that rank positive-labeled instances higher than the negative-labeled instances will be selected. When the given attribute space has an ordering between negative and positive instances, a higher AUC value is obtained and according to the discretization measure of this method, a better discretization is achieved.

3.4. *Stopping criteria*

The MAD method is a merging discretization approach that continues to merge candidate cut-points to form larger intervals until the maximum AUC is obtained. The maximum AUC is defined by the convex hull formed by the ROC points in the given ROC space. The convex hull is a polygon with a minimum number of cut-points that encloses all other points in the ROC space. Theorem 1 shows that the maximum AUC can be obtained by finding the convex hull. The ROC convex hull is defined by Provost and Fawcett[26] to form classifiers that lead to maximum AUC values. In this work, a similar approach is used to select cut-points that maximize the AUC value.

Theorem 1. *If all the points forming the ROC curve are on a convex hull, the AUC is the maximum.*

Proof. (By contradiction) Assume that an ROC curve for a given space has a larger AUC. This curve should contain a point outside the convex hull to make area even larger. Since the convex hull encloses all points in the space, this is a contradiction. □

3.5. *Algorithm*

The MAD method finds the proper cut-points for the given instance space to maximize the AUC value. Since the maximum AUC value is obtained by the convex hull, these cut-points must lie on the convex hull. MAD method employs slightly different logic to find this convex hull for two-class and multi-class datasets. Two-class behavior is covered in Sec. 3.5.1 and Multi-class in Sec. 3.5.2. Before delving into details, we would like to point out the difference between two-class and multi-class discretization. As will be revisited in Sec. 3.5.1, there exists symmetry in every ROC curve for two-class datasets. If the labels of all instances are interchanged, that is, label **n**'s are replaced by **p**'s, and **p**'s are replaced by **n**'s, the ROC curve obtained in new setting is symmetric to the original on the $y = x$ line. That is, it is possible to find the discretization results by calculating one ROC curve that is independent of class label assignment. However, this symmetry does not exist for multi-class datasets. In that case, more than one ROC curves will be obtained. Therefore, two-class datasets and multi-class datasets require different treatments.

The overview of the algorithm is given in Fig. 3. In a nutshell, the MAD method starts by sorting the training instances. Then, ROC points are calculated by using every cut-point as a threshold. Finally, the cut-points forming the corner points of the convex hull are selected as final result.

3.5.1. *Discretization in two-class datasets*

For two-class datasets, calculating candidate cut-points represented by ROC points and the method that finds the convex hull are different than for the multi-class datasets in the MAD method. MAD method for two-class datasets is given in Fig. 4.

Some important points deserve further elaboration. For example, in order to calculate ROC points for a given attribute, the total number of instances predicted as **p** and **n** classes have to be counted. There are two possible ways to predict the labels of the instances: (a) label high scored instances as **p** and low scored instances as **n**,

```
MAD(trainInstances)
  begin
    sort(trainInstances);
    rocPoints= calculateROCPoints(trainInstance);
    cutPoints= findConvexHull(rocPoints);
    return cutPoints;
  end
```

Fig. 3. Outline of the MAD algorithm.

```
MAD2C (trainInstances)
  begin
     sort(trainInstances);
     rocPoints= calculateROCPoints(trainInstance);
     cutPoints= findConvexHull(rocPoints);
     return cutPoints;
  end

function calculateROCPoints(trainInstance)
  begin
     rocPoints<- (+ ,0,0),(- ,1,1);//initialize ROC points set
     for i=0 to N
        if(trainInstances[i]==positiveClass)
           totalPositive++;
        else totalNegative++;
     curPos=totalPositive;
     curNeg=totalNegative;
     for i=0 to N-1
        if(trainInstances[i]==positiveClass)
           curPos--;
        else curNeg--;
        if(trainInstances[i]==trainInstance[i+1])
           continue;
        cutValue=(trainInstances[i]
                  +trainInstances[i+1]/2);
        TPR= curPos/totalPositive;
        FPR= curNeg/totalNegative;
        If(upperTriangle(TPR,FPR)==true)
           rocPoints<- (cutValue,TPR,FPR);
        else rocPoints<- (cutValue,FPR,TPR);
     return rocPoints;
  end

function findConvexHull(rocPoints)
  begin
     pointsKept<-(+ ,0,0);
     currentSlope=slopeBetween(rocPoints[1],
                               rocPoints[0]);
     for i=2 to N
        nextSlope=slopeBetween(rocPoints[i],
                               rocPoints[i-1]);
        if(nextSlope<=currentSlope)
           concavityFound=true;
        else pointsKept<- rocPoints[i-1];
        currentSlope=nextSlope;
     pointsKept<-(- ,1,1);
     if(concavitiyFound)
        return findConvexHull(pointsKept);
     else return pointsKept;
  end
```

Fig. 4. MAD algorithm in two-class datasets.

(b) label low scored instances as **p** and high scored instances as **n**. However, there are domains where class label assignments do not follow such way. Therefore, it is proven in Theorem 2 that assigning either of the current class labels with **p** does not affect the discretization process. As a result, MAD method randomly picks one of current

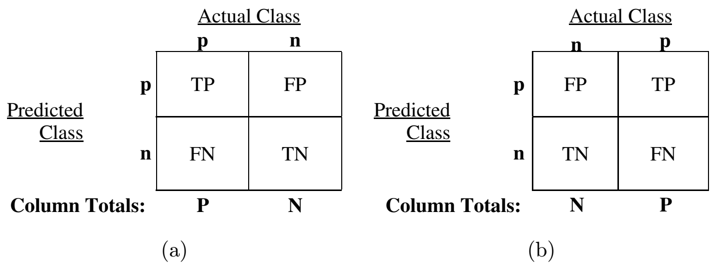
Actual Class
p
n
p
n
Predicted Class
TP
FP
FN
TN
Column Totals:
P
N
(a)
Actual Class
n
p
p
n
Predicted Class
FP
TP
TN
FN
Column Totals:
N
P
(b)

Fig. 5. (a) Confusion matrix for the case where one of the classes is labeled as **p** and other class as **n**. (b) Confusion matrix for the case when class labels interchanged.

class labels as **p**. Then, an ROC point for each candidate cut-points is calculated using Eq. (1).

Theorem 2. *In two-class problems, there exist two ROC points for the given candidate cut-point C and these points are symmetric about the $y = x$ line.*

Proof. In order to calculate ROC curve, one of the classes should be labeled as **p** and the other as **n**. Assume that an arbitrary class is labeled as **p** and the confusion matrix in Fig. 5(a) is obtained. The ROC point created from this confusion matrix is v and its coordinate is (x, y). The calculation of this coordinate is given in Eq. (3).

$$x = \mathrm{FPR} = \frac{\mathrm{FP}}{N}, \quad y = \mathrm{TPR} = \frac{\mathrm{TP}}{P}. \tag{3}$$

□

When the actual class labels are interchanged, the confusion matrix in Fig. 5(b) is formed. This new confusion matrix is equal to the original confusion matrix, where column values are interchanged. The new ROC point created from this matrix is v' represented by the (x', y') point. This ROC point is calculated using Eq. (4) and the coordinate of v' is equal to (y, x). Therefore, the points v and v' are symmetric about the $y = x$ line.

$$\begin{aligned} x' &= \mathrm{FPR}' = \mathrm{TP}/P, \\ y' &= \mathrm{TPR}' = \mathrm{FP}/N, \\ x' &= y \quad \text{and} \quad y' = x. \end{aligned} \tag{4}$$

Corollary 1. *Since there exists a symmetry between the ROC points, the one on or above the $y = x$ line is taken into consideration. The ROC points below the $y = x$ line are not candidate points for maximizing AUC since the default AUC value is* 0.5. *The upperTriangle function on 27th line of the algorithm given in Fig.* 4 *assures this property by checking on which side of the $y = x$ lies the given ROC point.*

The next step of the discretization is selecting the ROC points that form the convex hull. There are different methods for calculating the convex hull in the given n-dimensional space. One of these methods, proposed by Preperata and Shamos,[23] is called QuickHull. This method has $O(n \log n)$ expected time complexity and $O(n^2)$ in the worst case. In this work, a new method for calculating the convex hull for two-class problems is proposed.

The *findConvexHull* function is given in the 32nd line of the algorithm in Fig. 4. The main motivation for this function is the ordering of the ROC points. The first point created on the graph always corresponds to (1,1) and the last point to (0,0). The ROC points lying between these two trivial points have a monotonically non-decreasing property. For example, assume that v_1 is the point created just before v_2. Point v_1 always stands on the north, east or north-east side of v_2 assuming y-axis points north and x-axis points east. These points create a shape (possibly including concavities) when connected to each other with hypothetical lines during the ROC curve creation process. The *findConvexHull* method compares the slopes between every two consecutive ROC points in the order of the creation of the hypothetical lines and finds the junction points that cause concavities and eliminates them. As a result, due to the correspondence between ROC points and cut-points, *findConvexHull* eliminates the cut-points that are not on the convex hull, as well.

The *findConvexHull* method guarantees finding the convex hull in the best case $O(n)$ time and in the worst case $O(n^2)$. In the worst case, the method should leave at least one point out to call itself again, which leads to $O(n^2)$ complexity. In the best case, the method finds the convex hull in a linear time if the points already form a convex hull. The average case depends on the distribution of the points, which is random. However, we will investigate the average behavior in the empirical evaluation section.

With the MAD method, it is possible to visualize the discretization process. A toy dataset given in Table 2 will be used as an example to explain how the MAD method discretizes a feature in a visual way. The toy dataset contains 20 instances.

Each steps of the convex hull in the given ROC space process is visualized in Figs. 6 through 8. In Fig. 6, the ROC points generated for both classes assignment are shown. The $y = x$ line is drawn to show the symmetry between curves, which is proven in Theorem 2. According to Corollary 1, only the points on or above the $y = x$ line will be processed in the next step.

In the next step of MAD, points that cause concavity will be eliminated. Figure 7 shows the points left after the first pass of the method that finds the convex hull. Since the algorithm checks the concavity on a local base, it is possible to have a concave shape even after the first pass. The algorithm will continue recursively with the points left in each step until it converges to the convex hull.

Table 2. Toy dataset for visualization of MAD in two-class problems. A is the name of the attribute to be discretized.

Class Value	**n**	**p**	**n**	**p**	**n**	**n**	**n**	**p**	**n**	**p**	**n**	**p**	**n**	**n**	**p**	**p**	**p**	**n**	**p**	**p**
F1	1	2	3	4	5	6	7	8	9	9	10	11	12	13	14	15	16	17	18	19

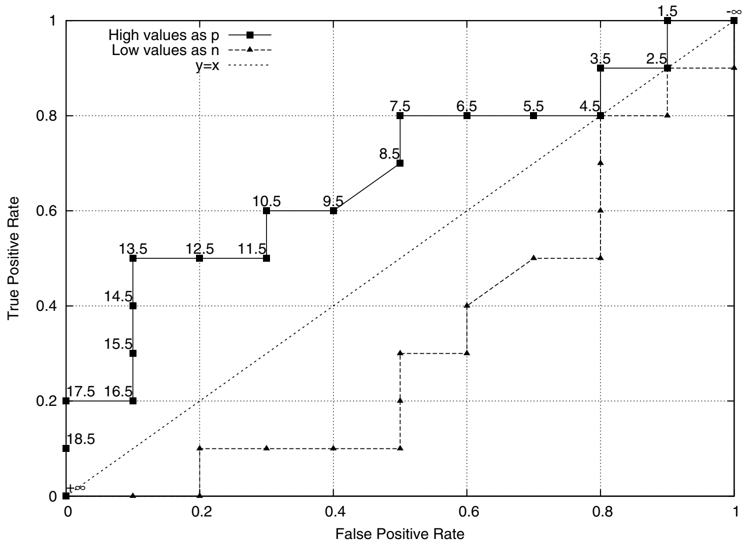

Fig. 6. Visualization of ROC point in two-class discretization.

In this example, the algorithm converges to the final convex hull after the second pass. The points left on the graph are the cut points to be used in the discretization. Figure 8 shows the final cut points left on the graph.

MAD method guarantees that any cut point left on the final graph does not divide a sequence of instances that belong to the same class. This can be proven for two class problems and can be extended to multi-class problems as well.

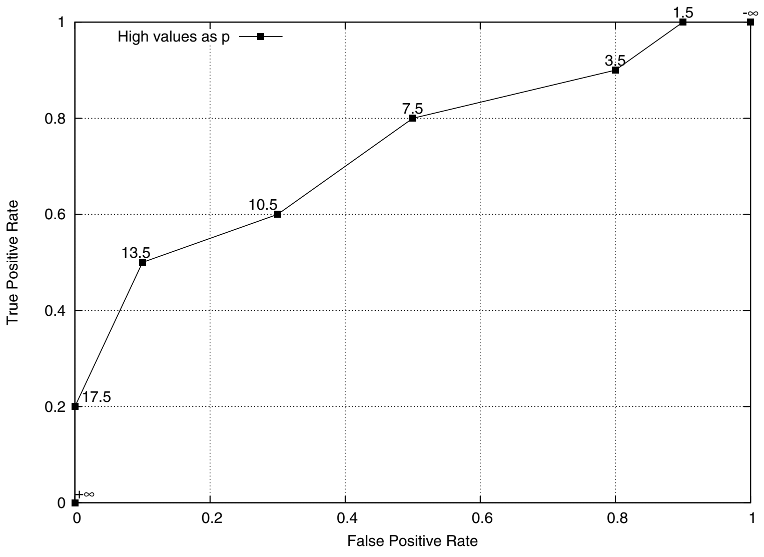

Fig. 7. First pass of the convex hull algorithm.

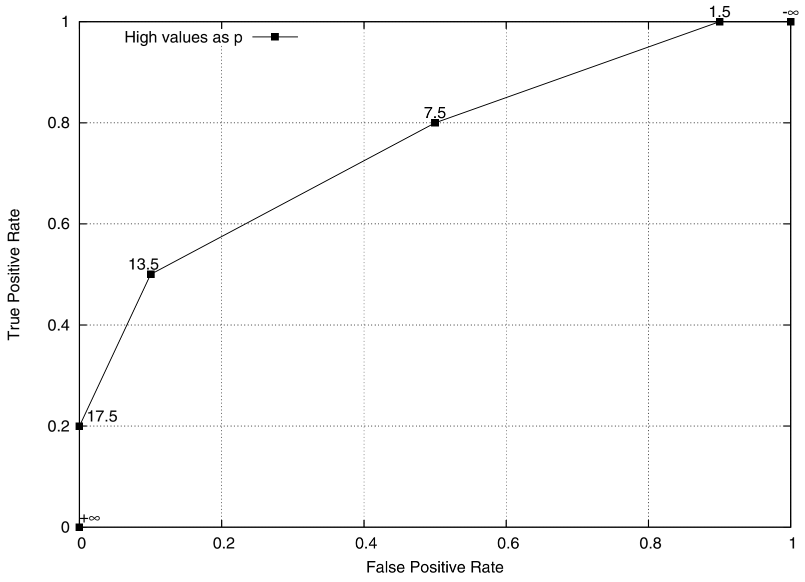

Fig. 8. Final cut points left after the second pass of convex hull algorithm.

Theorem 3. *Cut points found by MAD method in two class problem do not lie between two consecutive instances of the same class.*

Proof. (by contradiction) Assume that there exists such a cut-point C_i at the final ROC curve that divides a sequence of instances of the same class. Let C_{i-1} and C_{i+1} are the cut points before and after C_i, respectively. The total number of instances labeled as **p** is P and the total number labeled as **n** is N. The number of instances labeled as **p** and are higher than C_{i-1} is p'. The number of instances labeled as **n** and are higher than C_{i-1} is n'. The number of instances between C_{i-1} and C_i will be represented by m and the number of instances between C_i and C_{i+1} will be represented by l. If C_i divides an interval where all instances are labeled as **p**, the TPR and FPR values of C_{i-1}, C_i and C_{i+1} are given in Eq. (5). Since all these cut-points have the same TPR value, they lie on the same slope and the C_i point will be eliminated at the 40th line of the algorithm given in Fig. 4, which requires the slope between C_i and C_{i+1} to be strictly greater than the slope between C_{i-1} and C_i.

$$\begin{aligned} &\mathrm{TPR}_{i-1} = p'/P, \quad \mathrm{TPR}_i = p'/P, \quad\quad \mathrm{TPR}_{i+1} = p'/P, \\ &\mathrm{FPR}_{i-1} = n'/N, \quad \mathrm{FPR}_i = n' - m/N, \quad \mathrm{FPR}_{i+1} = n' - l/N. \end{aligned} \tag{5}$$

The other case is that C_i divides an interval where all instances are labeled as **n**. The TPR and FPR values of C_{i-1}, C_i and C_{i+1} are given in Eq. (6). In this case, all points have the same FPR value and these points lie on the same slope, as well. The algorithm shown in Fig. 4 will eliminate C_i. As a result in both cases the cut-point C_i is eliminated and it is a contradiction to have such a point in the final

ROC curve.

$$\begin{aligned}\mathrm{TPR}_{i-1} &= p'/P, & \mathrm{TPR}_{i} &= p'-k/P, & \mathrm{TPR}_{i+1} &= p'-l/P,\\ \mathrm{FPR}_{i-1} &= n'/N, & \mathrm{FPR}_{i} &= n'/N, & \mathrm{FPR}_{i+1} &= n'/N.\end{aligned} \tag{6}$$

$\square$

3.5.2. *Multi-class behavior*

In multi-class problems, the main problem is deciding how to choose the positive and the negative classes. Further, no symmetry exists between ROC curves of each class, as there is two-class problems. Therefore, in the multi-class MAD method for K number of classes, K different ROC curves are calculated.

The method used for two-class datasets can be extended to multi-class problems by relabeling one class as **p** and all others as **n** and obtaining the ROC curve. This technique is used by Provost and Domingos[24] in order to form ROC curves for multi-class datasets. For each class this relabeling operation is performed and the convex hull of the ROC curve is computed. All the points calculated in K different convex hulls are gathered in the same space and the final convex hull is found by using the QuickHull method. Outline of the multi-class MAD method is given in Fig. 9.

Multi-class MAD uses the same function to calculate ROC points (*calculate-ROCPoints* given in the algorithm in Fig. 4) and convex hull (*findConvexHull* given in the algorithm in Fig. 4) as is used in the two-class datasets. Therefore, Theorem 3 applies to the multi-class MAD method; that is, it is guaranteed that a cut-point does not lie between two consecutive instances of the same class. On the 6th line of the algorithm, QuickHull method is used. As mentioned in Sec. 3.5.1, *findConvexHull* assumes there exists a monotonically nondecreasing property among the ROC points. However, when K different ROC curves are gathered in the same space, this property is lost. Therefore, QuickHull, which is a more generic method, is used in multi-class problems.

An example visualization of the discretization process for multi-class datasets is given in Fig. 10. In this figure, an attribute belonging to a three-class dataset is being discretized. Each class label is represented by a convex hull and the points lying on the border of the shaded area are the final cut-points that will be used in the discretization process.

```
1 : MADMC (trainInstances)
2 :   Begin
3 :      sort(trainInstances);
4 :      for each class
5 :         mark current class as p others as n
4 :         rocPoints=calculateROCPoints(trainInstance);
5 :         totalrocPoints+=findConvexHull(rocPoints);
6 :      cutPoints= QuickHull(totalrocPoints);
7 :      return cutPoints;
8 :   end
```

Fig. 9. Multi-class MAD algorithm.

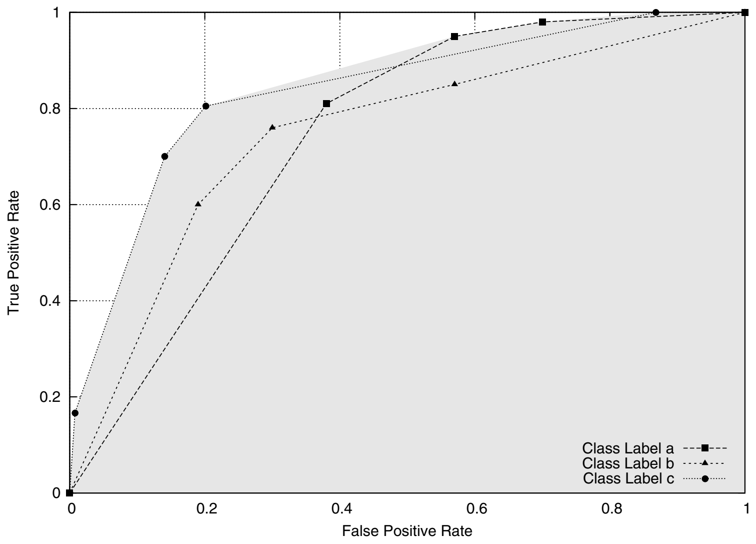

Fig. 10. Final cut points left after the second pass of convex hull algorithm.

By relabeling a class as **p** and marking others as **n**, the discretization method becomes sensitive to class distributions. If one of the classes in the dataset has a perfect ordering, only the points formed by that particular class will be selected by the QuickHull method and some valuable information can be lost. This drawback can be resolved by creating pairwise class ROC curves, which is similar to the method used in the M-Measure.[13] $K(K-1)$ different ROC curves must then be created, which will increase the computation time since the number of convex hulls to be calculated increases. Even with this disadvantage, the MAD method works well for multi-class datasets selected from the University of California Irvine (UCI) machine learning repository.[2]

4. Empirical Evaluation

In this section, the MAD discretization method is compared with the well-known Entropy-MDLP method proposed by Fayyad and Irani,[9] two other methods (FFD and PD) proposed recently by Yang and Webb[37] and ChiMerge proposed by Kerber.[17]

As a splitting method, Entropy-MDLP uses the entropy measure to select the proper cut-point to split an interval. An application of minimum description length principle called *information gain* is used as the stopping criteria. In a nutshell, this method selects the proper cut-points to minimize the entropy for the given interval and continues to discretize recursively until the information gain is below a threshold.

The unsupervised FFD and PD methods are designed to obtain high classification accuracy (lower classification error) by managing discretization bias and variance. The FFD method discretizes attributes into equal-sized intervals where each bin contains approximately 30 instances. The PD method also discretizes attributes into equal sized intervals but the number of instances in each interval is not fixed for each dataset. In PD, the desired interval frequency and number of intervals are calculated by using Eq. (7). Yang and Webb suggest that setting the interval frequency and number of intervals as the same value ($\sqrt{n}$) leads to lower bias and variance,[37] where n is the number of known instances. In this work, interval frequency and number of intervals are set to $\sqrt{n}$ for the experiments.

$$s * t = n, \tag{7}$$

where s is the interval frequency, t is the number of intervals and n is the number of known instances.

ChiMerge is one of the merging algorithms which uses χ^2 statistic in order to determine the similarities or differences between adjacent intervals. In this method, initially, all distinct values are considered as an interval similar to the MAD method. Then, the values of χ^2 statistic for all adjacent intervals are calculated by using the class frequencies. In each step, the adjacent intervals which have the lowest χ^2 value are merged until all adjacent intervals satisfy χ^2-threshold values which are selected from χ^2 table according to degree of freedom.[17]

The discretization results obtained by the MAD, ChiMerge, Entropy-MDLP, FFD, and PD methods in real-world datasets will be shown. All datasets used in the experiments are taken from the UCI machine learning repository and have at least one continuous variable. During the selection of datasets, only classification datasets are included in the experiments. Selected datasets cover both two-class and multi-class classification problems. Table 3 shows the properties of these datasets. In a recent study, Garcia *et al.* suggested accuracy, number of intervals and

Table 3. Dataset used in the experiments.

Name	# Instances	# Continuous Attributes	# Nominal Attributes	# Class Labels
Anneal	798	32	6	6
Bupa	345	6	0	2
Crx	653	9	6	2
Dermatology	366	33	1	7
German	1000	7	13	2
Glass	214	10	0	10
Heart (Statlog)	270	6	7	2
Ozone-onehr	2536	73	0	2
Ionoshpere	351	34	0	2
Mammography	961	1	4	2
Page-Blocks	5473	10	0	5
Pima-Indians	768	7	1	2
Sick-euthyroid	3163	7	18	2
SPECTF	267	44	0	2
Spambase	4601	58	0	2
Transfusion	748	5	0	2
Wisconsin	569	30	0	2
Yeast	1484	8	0	10

inconsistency as measures of performance in comparing discretization algorithms.[10] On this basis, the performance of the algorithms is evaluated in four different aspects: predictive performance, running time, inconsistency of intervals, and number of intervals found.

4.1. *Predictive performance*

As their ROC curve representation is more meaningful, the classifiers that associate the predicted class with a confidence value are preferred in this work. Two different classifiers, supporting this property, are selected. The first one is the Naïve Bayes classifier, which is one of the simplest and most effective classifiers. It is shown that using discretization with the Naïve Bayes algorithm increases predictive accuracy.[7] The other selected algorithm is AODE, which can process only categorical features.

The implementations of Naïve Bayes and AODE classification and Entropy-MDLP discretization methods are taken from the source code of the WEKA software package.[12] The Naïve Bayes algorithm is applied with default parameters which uses single normal distribution rather than kernel estimation. The FFD method is implemented by using WEKA's unsupervised discretization method by passing the number of intervals as a parameter. In implementing the PD, the number of known values for each attribute is calculated in order to find the number of intervals as shown in Eq. (7), and the number of intervals is passed as a parameter. ChiMerge implementation for WEKA is obtained from Zimek.[38] In empirical evaluations, the significance level is kept as the default value, 0.95.

In this section, six different cases will be considered for the Naïve Bayes algorithm: The Naïve Bayes algorithm with MAD, ChiMerge, Entropy-MDLP, FFD, PD discretization methods, and with continuous values (without discretization). Also five different cases will be considered for AODE algorithm such as AODE with MAD, ChiMerge, Entropy-MDLP, FFD, and PD discretization methods.

Two different measures have been used to evaluate predictive performance. The first measure is called M-Measure that is suitable for both calculating two-class AUC and multi-class AUC values. M-Measure is insensitive to class distributions and error costs. In addition to M-Measure metric, the predictive performance of MAD against other discretization methods is also measured by using accuracy metric. Stratified 10-fold cross validation is employed to calculate the M-Measure and accuracy values for each dataset. In each experiment, the standard deviation values are given since 10-fold cross validation is employed.

In order to measure the statistical significance of the differences between the MAD method and the other discretization methods, the Wilcoxon signed rank test is used. This statistic test is preferred since it is nonparametric and the distribution of the results could be non-normal. In each experiment, p-values related to the statistical tests is provided. P-values lower than 0.05 indicates that on average MAD method outperforms the corresponding method.

Table 4. Predictive performance of Naïve Bayes in terms of M-Measure under different discretization methods. Standard deviations (SD) are given in parentheses (Higher M-Measure values and lower SD values are better). *P*-values lower than 0.05 indicate significant difference wrt MAD.

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD	Without Discretization
Anneal	0.980 (0.024)	0.984 (0.031)	0.983 (0.018)	0.990 (0.012)	0.989 (0.014)	0.983 (0.007)
Bupa	0.756 (0.065)	0.654 (0.070)	0.530 (0.035)	0.686 (0.069)	0.683 (0.072)	0.633 (0.107)
Crx	0.925 (0.031)	0.920 (0.021)	0.929 (0.033)	0.930 (0.030)	0.931 (0.029)	0.900 (0.046)
Dermatology	1.000 (0.001)	1.000 (0.001)	1.000 (0.001)	0.999 (0.001)	0.999 (0.002)	0.998 (0.004)
German	0.794 (0.054)	0.737 (0.073)	0.773 (0.050)	0.790 (0.057)	0.790 (0.057)	0.784 (0.051)
Glass	0.929 (0.018)	0.839 (0.033)	0.929 (0.027)	0.922 (0.023)	0.908 (0.029)	0.880 (0.027)
Heart (Statlog)	0.905 (0.076)	0.874 (0.061)	0.900 (0.071)	0.906 (0.070)	0.906 (0.071)	0.900 (0.066)
Ionosphere	0.947 (0.046)	0.950 (0.030)	0.950 (0.035)	0.944 (0.039)	0.950 (0.034)	0.937 (0.056)
Mammography	0.903 (0.026)	0.898 (0.025)	0.902 (0.027)	0.902 (0.026)	0.901 (0.026)	0.897 (0.030)
Ozone-onehr	0.861 (0.069)	0.826 (0.088)	0.858 (0.070)	0.832 (0.082)	0.844 (0.074)	0.848 (0.067)
Page-Blocks	0.972 (0.010)	0.953 (0.017)	0.980 (0.010)	0.968 (0.012)	0.979 (0.010)	0.954 (0.016)
Pima-Indians	0.807 (0.064)	0.746 (0.065)	0.817 (0.049)	0.805 (0.054)	0.803 (0.039)	0.812 (0.066)
Sick-euthyroid	0.953 (0.019)	0.951 (0.012)	0.959 (0.013)	0.950 (0.011)	0.953 (0.015)	0.920 (0.037)
Spambase	0.965 (0.007)	0.968 (0.006)	0.965 (0.006)	0.950 (0.010)	0.957 (0.009)	0.940 (0.009)
SPECTF	0.867 (0.056)	0.834 (0.076)	0.823 (0.072)	0.846 (0.069)	0.844 (0.075)	0.850 (0.052)
Transfusion	0.723 (0.041)	0.665 (0.045)	0.701 (0.040)	0.693 (0.040)	0.688 (0.048)	0.713 (0.041)
Wisconsin	0.987 (0.013)	0.982 (0.016)	0.984 (0.014)	0.985 (0.014)	0.984 (0.014)	0.980 (0.020)
Yeast	0.842 (0.011)	0.781 (0.026)	0.835 (0.025)	0.818 (0.023)	0.819 (0.032)	0.879 (0.031)
Average	**0.895** (0.035)	0.865 (0.039)	0.879 (0.033)	0.884 (0.036)	0.885 (0.036)	0.878 (0.041)
P-value	1.000	0.001	0.306	0.007	0.044	0.005

The predictive performance evaluation results of Naïve Bayes obtained by using M-Measure is given in Table 4. Here it is clear that, on average, the MAD method outperforms all other discretization methods in terms of the M-Measure. Wilcoxon signed rank test method shows that in 95% confidence interval, only MAD method improves Naïve Bayes algorithm performance significantly compared to the performance obtained by using other discretization methods. The predictive performance of MAD method is significantly higher than all other discretization methods except Entropy-MDLP method according to M-Measure. However, MAD still outperforms Entropy-MDLP on average.

The predictive performance of Naïve Bayes in terms of accuracy metric is given in Table 5. The MAD method again outperforms all other methods on the average. This difference in the predictive performance is statistically significant for all discretization methods except ChiMerge method. According to the Wilcoxon signed rank test in 95% confidence, it is possible to say that all of the discretization methods used in this paper improve the performance of Naïve Bayes algorithm significantly.

The predictive performance of AODE algorithm in terms of M-Measure is given in Table 6. The AODE method is an extension to Naïve Bayes method in order to improve predictive performance, so it is natural to expect high performance from FFD and PD methods since they are Naïve Bayes optimal. But according to the

Table 5. Predictive performance of Naïve Bayes in terms of accuracy under different discretization methods SDs are given in parenthesis (Higher accuracy values and lower SD values are better). *P*-values lower than 0.05 indicate significant difference wrt MAD.

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD	Without Discretization
Anneal	95.35 (2.12)	96.99 (1.51)	95.11 (1.64)	94.98 (3.18)	95.35 (2.93)	63.61 (4.68)
Bupa	70.13 (5.60)	62.03 (8.00)	57.70 (2.63)	63.18 (7.59)	63.22 (6.38)	53.96 (7.94)
Crx	85.91 (3.91)	84.38 (4.17)	86.53 (3.59)	85.61 (3.95)	86.22 (3.81)	77.97 (4.80)
Dermatology	97.55 (2.25)	97.82 (1.63)	98.09 (2.14)	98.09 (1.25)	97.82 (1.63)	97.81 (1.63)
German	75.90 (4.30)	71.00 (4.20)	73.50 (3.61)	75.20 (4.24)	75.20 (4.24)	75.00 (3.55)
Glass	72.36 (6.00)	62.12 (7.36)	71.02 (7.59)	67.27 (9.83)	68.68 (9.41)	48.55 (8.13)
Heart (Statlog)	82.59 (6.21)	80.00 (8.80)	82.59 (8.93)	82.59 (8.29)	82.96 (6.67)	83.70 (7.07)
Ionosphere	90.02 (5.76)	89.17 (5.09)	89.17 (5.40)	89.73 (3.68)	88.31 (4.34)	83.18 (5.04)
Mammography	83.35 (3.52)	82.83 (3.84)	82.10 (3.55)	83.25 (3.68)	83.24 (3.42)	82.10 (3.44)
Ozone-onehr	78.34 (1.18)	93.02 (1.32)	79.88 (1.72)	87.81 (1.65)	83.31 (1.55)	70.77 (2.96)
Page-Blocks	94.54 (0.78)	94.04 (0.82)	93.42 (0.85)	93.40 (0.86)	92.40 (1.16)	90.15 (3.10)
Pima-Indians	74.87 (6.49)	70.04 (5.51)	76.04 (4.42)	73.95 (6.30)	73.16 (4.94)	76.16 (5.30)
Sick-euthyroid	95.89 (0.93)	95.67 (1.12)	96.02 (0.97)	95.07 (1.20)	95.32 (1.11)	84.22 (1.97)
Spambase	89.96 (1.40)	90.15 (1.28)	89.81 (1.51)	87.83 (1.47)	89.13 (1.36)	79.72 (1.84)
SPECTF	76.81 (6.36)	78.66 (7.42)	72.66 (3.75)	74.16 (7.85)	76.79 (6.54)	68.58 (6.99)
Transfusion	77.94 (2.88)	73.65 (3.75)	75.27 (2.62)	76.47 (2.46)	75.27 (3.87)	75.67 (1.90)
Wisconsin	94.37 (2.34)	94.20 (2.61)	94.19 (2.38)	94.19 (2.63)	93.67 (3.09)	93.49 (4.26)
Yeast	58.32 (3.24)	51.44 (3.61)	56.71 (4.27)	52.79 (3.63)	53.74 (3.68)	57.99 (4.12)
Average	**83.01** (3.63)	81.51 (4.00)	81.66 (3.42)	81.98 (4.10)	81.88 (3.90)	75.70 (4.37)
P-value	1.000	0.064	0.039	0.011	0.022	0.002

Table 6. Predictive performance of AODE in terms of M-Measure under different discretization methods SDs are given in parentheses (Higher M-Measure values and lower SD values are better). *P*-values lower than 0.05 indicate significant difference.

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD
Anneal	0.982 (0.025)	0.984 (0.031)	0.984 (0.022)	0.989 (0.014)	0.989 (0.016)
Bupa	0.761 (0.059)	0.651 (0.113)	0.530 (0.035)	0.661 (0.091)	0.658 (0.089)
Crx	0.929 (0.034)	0.932 (0.022)	0.932 (0.033)	0.932 (0.031)	0.929 (0.032)
Dermatology	1.000 (0.001)	1.000 (0.001)	1.000 (0.001)	1.000 (0.001)	0.999 (0.001)
German	0.795 (0.052)	0.761 (0.066)	0.783 (0.047)	0.789 (0.048)	0.789 (0.048)
Glass	0.926 (0.034)	0.872 (0.032)	0.934 (0.031)	0.934 (0.041)	0.931 (0.036)
Heart (Statlog)	0.915 (0.062)	0.881 (0.069)	0.905 (0.068)	0.908 (0.073)	0.896 (0.066)
Ionosphere	0.972 (0.026)	0.962 (0.028)	0.967 (0.026)	0.979 (0.011)	0.970 (0.016)
Mammography	0.902 (0.024)	0.892 (0.029)	0.905 (0.028)	0.900 (0.024)	0.901 (0.027)
Ozone-onehr	0.890 (0.049)	0.807 (0.059)	0.882 (0.063)	0.767 (0.064)	0.722 (0.079)
Page-Blocks	0.975 (0.014)	0.922 (0.018)	0.986 (0.014)	0.934 (0.018)	0.956 (0.016)
Pima-Indians	0.798 (0.057)	0.720 (0.064)	0.820 (0.050)	0.738 (0.060)	0.731 (0.045)
Sick-euthyroid	0.964 (0.012)	0.957 (0.015)	0.964 (0.010)	0.957 (0.011)	0.959 (0.014)
Spambase	0.977 (0.006)	0.971 (0.006)	0.980 (0.005)	0.945 (0.012)	0.959 (0.009)
SPECTF	0.871 (0.067)	0.823 (0.079)	0.821 (0.079)	0.827 (0.090)	0.787 (0.083)
Transfusion	0.703 (0.036)	0.641 (0.055)	0.727 (0.034)	0.663 (0.054)	0.656 (0.063)
Wisconsin	0.993 (0.008)	0.984 (0.015)	0.988 (0.014)	0.989 (0.011)	0.988 (0.015)
Yeast	0.834 (0.014)	0.773 (0.029)	0.833 (0.028)	0.812 (0.028)	0.817 (0.035)
Average	**0.899** (0.032)	0.863 (0.041)	0.886 (0.033)	0.874 (0.038)	0.869 (0.038)
P-value	1.000	0.001	0.698	0.013	0.003

Table 7. Predictive performance of AODE in terms of accuracy under different discretization methods SDs are given in parentheses (Higher values and lower SD values are better). P-values lower than 0.05 indicate significant difference.

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD
Anneal	96.74 (1.00)	96.74 (1.51)	96.61 (1.49)	96.61 (2.46)	96.99 (1.97)
Bupa	71.85 (5.06)	60.92 (9.84)	57.70 (2.63)	63.19 (7.27)	62.39 (11.68)
Crx	86.68 (2.82)	85.76 (4.05)	86.68 (4.27)	87.45 (4.27)	86.68 (3.29)
Dermatology	97.55 (2.25)	97.82 (1.63)	98.08 (1.77)	98.09 (1.25)	97.55 (2.25)
German	75.40 (2.54)	73.00 (4.29)	74.90 (3.59)	75.60 (3.47)	75.60 (3.47)
Glass	76.60 (6.69)	65.84 (7.10)	72.88 (6.35)	77.01 (6.04)	77.06 (7.25)
Heart (Statlog)	82.22 (7.55)	80.37 (8.77)	82.96 (8.15)	82.96 (8.95)	82.22 (7.73)
Ionosphere	91.73 (5.64)	89.74 (3.89)	90.87 (5.25)	91.15 (4.14)	91.16 (4.34)
Mammography	82.62 (3.83)	82.62 (3.10)	82.72 (3.60)	82.62 (3.74)	82.51 (3.59)
Ozone-onehr	96.21 (1.22)	97.12 (0.18)	88.76 (1.73)	96.65 (0.75)	96.88 (0.48)
Page-Blocks	96.62 (0.54)	94.86 (0.44)	96.97 (0.51)	95.74 (0.42)	96.18 (0.53)
Pima-Indians	73.94 (6.27)	67.70 (4.34)	76.57 (4.08)	70.83 (4.54)	69.40 (4.54)
Sick-euthyroid	96.65 (0.91)	95.73 (1.22)	96.52 (0.93)	95.26 (0.86)	95.92 (1.00)
Spambase	93.35 (0.87)	91.22 (1.18)	93.31 (0.97)	88.02 (1.65)	90.15 (1.54)
SPECTF	79.42 (4.71)	78.66 (5.81)	73.77 (3.82)	79.37 (4.61)	76.81 (8.29)
Transfusion	76.34 (2.31)	78.08 (2.18)	75.27 (2.62)	77.41 (2.13)	77.55 (2.59)
Wisconsin	95.78 (1.79)	93.67 (2.54)	96.12 (2.08)	95.60 (2.26)	94.91 (2.53)
Yeast	57.38 (4.14)	49.69 (2.92)	56.77 (4.03)	53.67 (3.48)	54.41 (3.41)
Average	**84.84** (3.34)	82.20 (3.61)	83.19 (3.22)	83.74 (3.46)	83.58 (3.92)
P-value	1.000	0.003	0.124	0.266	0.061

Wilcoxon signed rank test in 95% confidence interval MAD method outperforms both FFD and PD methods. Also the MAD method performs better than ChiMerge algorithm significantly and Entropy-MDLP method on the average.

The predictive performance of AODE algorithm used on discretized datasets in terms of accuracy metric is given in Table 7. According to this table, MAD method outperforms all other discretization methods on the average. However, only the difference between predictive performance of MAD and ChiMerge algorithm is statistically significant.

4.2. *Running time*

In machine learning research, performance on large datasets are essential and therefore, the running time of the proposed method is critical. The worst- and the best-case running time complexities of the MAD method are given in Sec. 3.5.1. In this section, the empirical evaluation of runtime of the MAD algorithm will be given. Since the actual running time of a method highly depends on its implementation, this evaluation will be based on the measured running times of the MAD algorithm on different datasets, rather than a comparison with other algorithms.

As mentioned in Sec. 2.5, the main time consuming step of MAD (after sorting) is finding the convex hull. In two class problems only one convex hull is calculated. On the other hand, in multi-class problems the number of convex hulls calculated is equal to the number of class labels. Each of these convex hulls is calculated by the method proposed in Sec. 2.5. Hence, the average running time of finding the convex hull method is the most prominent factor in the running time of the whole method. The proposed convex hull calculation method is invoked recursively until it converges to the convex hull. In order to give an insight into the running time of the algorithm in practice, Table 8 shows the average number of recursive call for an attribute to calculate convex hull. According to this table, it is possible to say that, regardless of the number of instances, the convex hull can be found by recursively calling the function at minimum one, maximum six and on average four times for the given datasets.

The overall running times of all methods are measured using java virtual machine's CPU time and a hundred different runs are averaged in order to be objective. As mentioned earlier, for multi-class datasets, the MAD method calculates K different ROC curves, where K is the number of classes. It also combines these curves with the QuickHull algorithm whose complexity is no worse than $\mathrm{O}(n\log n)$ in practical, where n is number of instances. On the other hand, since the convex hull is found in a few iterations, MAD method works faster on two-class datasets. As it can be seen in Table 8.

The running times of the MAD algorithm on the datasets listed in Table 3 are measured using java virtual machine's CPU time and a hundred different runs are averaged in order to be objective. The results are shown in Table 8.

Table 8. Average number of recursive calls to calculate convex hull for an attribute and the average running time (in μs) for each datasets.

Name	#Recursive Calls	Running Time
Anneal	1	6630
Bupa	5	1528
Crx	5	2392
Dermatology	1	3165
German	3	2184
Glass	4	4524
Heart (Statlog)	2	732
Ionosphere	6	1092
Mammography	2	2589
Ozone-onehr	5	6358
Page-Blocks	6	67,033
Pima-Indians	5	1618
Sick-euthyroid	5	8134
Spambase	6	9075
SPECTF	4	648
Transfusion	4	2847
Wisconsin	6	1669
Yeast	3	28,002
Average	**4**	17,316

4.3. *Inconsistency of intervals*

First, the inconsistency of an interval has to be defined. Given an interval that consists of n instances, inconsistency of that interval is equal to value $n - c$ where c is the number of instances that belong to majority class on the given interval. For example, given an interval with 10 instances, if 7 of these instances belong to the class a and the others belong to the class b, the inconsistency of the given interval is $10 - 7 = 3$.

A smaller amount of inconsistency indicates a better discretization in general. As entropy is a measure that aims to obtain pure intervals, it is expected to achieve lower inconsistency values with Entropy-MDLP discretization method. However, Table 9 shows that on the average the MAD method forms intervals that are more consistent than Entropy-MDLP. Furthermore, ChiMerge, FFD and PD methods have lower inconsistencies but at the cost of producing a very high number of intervals.

A discretization method which maps all numerical values into different intervals can produce pure intervals. Therefore, in next section the number of intervals produced by each discretization method will be investigated.

4.4. *Number of intervals*

The MAD method is not designed to minimize the number of intervals; its main aim is to maximize the AUC. In contrast, the Entropy-MDLP method is known as

Table 9. Total inconsistencies for continuous attributes in given datasets (Lower results are better).

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD
Anneal	186	181	185	184	183
Bupa	138	111	143	135	129
Crx	216	161	221	218	215
Dermatology	214	213	216	214	213
German	296	263	299	296	296
Glass	107	65	114	113	106
Heart (Statlog)	89	81	93	89	88
Ionosphere	85	30	72	76	68
Mammography	271	267	271	270	270
Ozone-onehr	72	70	73	72	73
Page-Blocks	494	403	486	481	492
Pima-Indians	248	204	255	244	243
Sick-euthyroid	275	267	273	271	273
Spambase	1581	1508	1578	1544	1560
SPECTF	54	52	55	54	54
Transfusion	175	168	178	173	172
Wisconsin	122	30	127	125	123
Yeast	946	913	961	923	927
Average	309	**277**	311	305	305

producing less number of intervals. However, An and Cercone[1] showed that Entropy-MDLP method terminates too early on small datasets which results in less number of intervals but higher inconsistencies.

The MAD method outperforms the ChiMerge method. ChiMerge method has a known limitation to produce high number of intervals when the significance level is set to very low values.[17] However, in this work the significance level is used as 0.95 which is a proper value for discretization. As a result, ChiMerge method still produced the highest number of intervals when compared to other methods. It is possible to set the maximum number of intervals for this method. However, this approach contradicts the purpose of using stopping criteria (χ^2-threshold).

The average number of intervals per attribute is given in Table 10. According to these results, Entropy-MDLP outperforms MAD in this aspect. However, it is necessary to mention an important point about the results in this table. In six datasets, the Entropy-MDLP method discretizes all attributes into one interval, on average. When an attribute is discretized into one interval, the discretization method maps all elements to the same value such as $(-\infty \cdots + \infty)$. This situation occurs if the distribution of attribute values is not suitable according to the discretization methods measure. Thus, it is possible to say that Entropy-MDLP acts like a feature selection algorithm. In turn, as shown in Sec. 3.1, there exists predictive accuracy gain in some of these datasets. Therefore, in some cases the Entropy-MDLP method misses important information when it maps all instances to the same interval.

Table 10. Average number of intervals per attribute. (Lower results are better.)

Name	MAD	ChiMerge	Entropy MDLP	FFD	PD
Anneal	2	8	2	24	6
Bupa	7	47	1	11	17
Crx	10	114	1	20	24
Dermatology	3	6	1	11	5
German	5	65	1	31	15
Glass	8	72	2	7	14
Heart (Statlog)	5	22	1	9	8
Ionosphere	4	63	4	11	16
Mammography	6	16	2	29	10
Ozone-onehr	9	63	1	77	46
Page-Blocks	11	316	6	165	70
Pima-Indians	10	99	2	24	25
Sick-euthyroid	9	76	2	95	45
Spambase	8	132	2	139	63
SPECTF	7	23	1	9	15
Transfusion	9	36	1	23	25
Wisconsin	13	152	2	18	23
Yeast	11	47	2	45	27
Average	8	75	**2**	42	25

On average, MAD outperforms the FFD and PD methods in terms of the number of intervals. This was expected since FFD and PD methods are unsupervised and always have a large number of intervals due to their design.

As seen in Sec. 4.3 and in this section, number of intervals and inconsistency of intervals are related. When the number of intervals gets higher, the inconsistency of intervals gets lower naturally. However, a proper discretization algorithm should create consistent intervals without overfitting the data, creating higher number of intervals. On the other hand, creating very low number of intervals, as Entropy-MDLP does, is a sign of data over-generalizing. It is expected that better discretization algorithms have lower inconstancies without data over-generalizing. MAD is the only method which can produce consistent intervals without overgeneralization or overfitting.

5. Conclusion and Future Work

In this work, we proposed a novel approach called MAD, for discretization continuous attributes. A new discretization measure and stopping criteria are defined for this method. The theoretical evidence for using ROC curves and AUC values in discretization is given.

According to the empirical evaluations, the MAD method outperforms Entropy-MDLP which is one of the most well-known discretization methods in terms of predictive performance. The MAD method also outperforms ChiMerge, FFD and PD methods in terms of predictive performance. Since FFD and PD discretization methods are Naïve Bayes optimal, the significant gain in Naïve Bayes algorithm in terms of M-Measure is important. Through real-world datasets, we also show that the MAD method runs faster than other discretization methods for two-class datasets. In terms of inconsistencies of intervals, the MAD method outperforms Entropy-MDLP method on average but it is outperformed by ChiMerge, FFD and PD methods. This was expected, due to the inherent design of the FFD and PD methods, which are intended to produce large numbers of intervals that bring pure intervals naturally. ChiMerge algorithm still results in high number of intervals even with proper parameterization.

The main bottleneck of the MAD method is the time complexity of the convex hull computation for multi-class datasets. As a future work, a new method that will find the convex hull faster than the QuickHull algorithm will be sought. Such a faster convex-hull algorithm will improve the time complexity of the MAD method.

References

1. A. An and N. Cercone, Discretization of continuous attributes for learning classification rules, in *Proc. Third Pacific-Asia Conf. Knowledge Discovery and Data Mining*, Beijing, China (1999), pp. 509–514.
2. A. Asuncion and D. J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA (2007), http://www.ics.uci.edu/∼mlearn/MLRepository.html.

3. A. P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, *Pattern Recogn.* **30**(7) (1997) 1145–1159.
4. J. Catlett, On changing continuous attributes into ordered discrete attributes, in *Proc. Fifth European Working Session on Learning* (Springer-Verlag, Berlin, 1991), pp. 164–177.
5. J. Cerquides and R. L. Mantaras, Proposal and empirical comparison of a parallelizable distance-based discretization method, *KDD97: Third Int. Conf. Knowledge Discovery and Data Mining* (1997), pp. 139–142.
6. C. C. Chan, C. Batur and A. Srinivasan, Determination of quantization intervals in rule based model for dynamic systems, in *Proc. IEEE Conf. Systems, Man, and Cybernetics*, Charlottesvile, Virginia (1991), pp. 1719–1723.
7. J. Dougherty, R. Kohavi and M. Sahami, Supervised and unsupervised discretization of continuous features, in *Proceedings of the Twelfth International Conference on Machine Learning*, eds. A. Prieditis and S. Russell (Morgan Kaufmann, San Francisco, CA, 1995), pp. 194–202.
8. T. Fawcett, An introduction to ROC analysis, *Pattern Recogn. Lett.* **27** (2006) 861–874.
9. U. Fayyad and K. Irani, On the handling of continuous-valued attributes in decision tree generation, *Mach. Learn.* **8** (1992) 87–102.
10. S. Garcia, J. Luengo, J. Saez, V. Lopez and F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, *IEEE Trans. Knowl. Data Eng.*, in press, doi: 10.1109/TKDE.2012.35.
11. D. M. Green and J. A. Swets, *Signal Detection Theory and Psychophysics* (Wiley, New York, 1966).
12. M. Hall *et al.*, The WEKA data mining software: An update, *SIGKDD Explorations* **11**(1) (2009) 10–18.
13. D. J. Hand and R. J. Till, A simple generalization of the area under the ROC curve to multiple class classification problems, *Mach. Learn.* **45**(2) (2001) 171–186.
14. J. A. Hanley and B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, *Radiology* **143** (1982) 29–36.
15. K. M. Ho and P. D. Scott, Zeta: A global method for discretization of continuous variables, *KDD97: 3rd Int. Conf. Knowledge Discovery and Data Mining*, Newport Beach, CA (1997), pp. 191–194.
16. R. C. Holte, Very simple classification rules perform well on most commonly used datasets, *Mach. Learn.* **11** (1993) 63–90.
17. R. Kerber, Chimerge: Discretization of numeric attributes, in *Proc. AAAI-92, Ninth National Conf. Artificial Intelligence* (AAAI Press/The MIT Press, 1992), pp. 123–128.
18. W. J. Krzanowski and D. J. Hand, *ROC Curves for Continuous Data* (CRC Press, Taylor & Francis Group, 2009).
19. C. X. Ling, J. Huang and H. Zhang, AUC: A statistically consistent and more discriminating measure than accuracy, in *Proc. Eighteenth Int. Joint Conf. Artificial Intelligence (IJCAI)* (2003), pp. 329–341.
20. H. Liu and R. Setiono, Chi2: Feature selection and discretization of numeric attributes, in *Proc. Seventh IEEE Int. Conf. Tools with Artificial Intelligence*, Herndon, Virginia (1995), pp. 388–391.
21. H. Liu, F. Hussain, C. L. Tan and M. Dash, Discretization: An enabling technique, *Data Min. Knowl. Discov.* **6**(4) (2002) 393–423.
22. M. S. Pepe, *The Statistical Evaluation of Medical Tests for Classification and Prediction* (Oxford, New York, 2003).
23. D. F. Preperata and M. Shamos, *Computational Geometry. An Introduction* (Springer-Verlag, Berlin, 1985).

24. F. Provost and P. Domingos, Well-trained PETs: Improving probability estimation trees, CeDER Working Paper #IS-00-04, Stern School of Business, New York University, NY 10012 (2001).
25. F. Provost and T. Fawcett, Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions, in *Proc. Third Intl. Conf. on Knowledge Discovery and Data Mining (KDD-97)* (AAAI Press, Menlo Park, CA, 1997), pp. 43–48.
26. F. Provost and T. Fawcett, Robust classification for imprecise environments, *Mach. Learn.* **42**(3) (2001) 203–231.
27. F. Provost, T. Fawcett and R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in *Proc. Fifteenth Int. Conf. Machine Learning* (Morgan Kaufmann, 1998), pp. 445–453.
28. J. R. Quinlan, Induction of decision trees, *Mach. Learn.* **1** (1986) 81–106.
29. J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1993).
30. J. R. Quinlan, Improved use of continuous attributes in C4.5, *Journal of Artificial Intelligence and Research* **4** (1996) 77–90.
31. K. A. Spackman, Signal detection theory: Valuable tools for evaluating inductive learning, in *Proc. Sixth Int. Workshop on Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1989), pp. 160–163.
32. T. Van de Merckt, Decision trees in numerical attribute spaces, in *IJCAI Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Vol. 2 (1993), pp. 1016–1021.
33. D. Ventura and T. R. Martinez, An empirical comparison of discretization methods, in *Proc. 10th Int. Symp. Computer and Information Sciences (ISCIS)* (1995), pp. 443–450.
34. K. Wang and B. Liu, Concurrent discretization of multiple attributes, *Pacific Rim Int. Conf. AI* (1998), pp. 250–259.
35. G. Webb, J. Boughton and Z. Wang, Not so Naive Bayes: Aggregating one-dependence estimators, *J. Mach. Learn.* **58**(1) (2005) 5–24.
36. T. T. Wong, A hybrid discretization method for naïve Bayesian classifiers, *Pattern Recogn.* **45**(6) (2012) 2321–2325.
37. Y. Yang and G. Webb, Discretization for naive-Bayes learning: Managing discretization bias and variance, *Mach. Learn.* **74**(1) (2009) 39–74.
38. A. Zimek, Hierarchical classification using ensembles of nested dichotomies, Master's thesis, TU/LMU Munich (2005).
39. M. H. Zweig and G. Campbell, Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine, *Clin. Chem.* **39**(8) (1993) 561–577.

Murat Kurtcephe is a research assistant and M.Sc. student in the Computer Science Department of Case Western Reserve University, where he works with Professor Meral Ozsoyoglu. He received his first M.Sc. degree in Computer Science from Bilkent University (Ankara, Turkey) where he worked on machine learning and data mining, especially on discretization and risk estimation methods with Professor H. Altay Guvenir. His current research interest focuses on the areas of graph encoding and pedigree data querying.

H. Altay Guvenir received his B.S. and M.S. degrees in Electronics and Communications Engineering from Istanbul Technical University, in 1979 and 1981, respectively. He received his Ph.D. in Computer Engineering and Science from Case Western Reserve University in 1987. He joined the Department of Computer Engineering at Bilkent University in 1987. He has been a Professor and serving as the Chairman of the Department of Computer Engineering since 2001. His research interests include artificial intelligence, machine learning, data mining, and intelligent data analysis. He is a member of the IEEE and the ACM.