Technical Report

No: BU-CE-1001

A Discretization Method based on Maximizing the Area Under ROC Curve

Murat Kurtcephe, H. Altay Güvenir
January 2010

Bilkent University
Department of Computer Engineering
06800 Ankara, Turkey
http://www.cs.bilkent.edu.tr

A Discretization method based on Maximizing the Area under ROC Curve

Murat Kurtcephe[1], H. Altay Güvenir[1]

[1] Bilkent University, Computer Engineering Department
06800 Ankara, Turkey
{kurtcephe, guvenir}@cs.bilkent.edu.tr

Abstract. We present a new discretization method based on Area under ROC Curve (AUC) measure. Maximum Area under ROC Curve Based Discretization (MAD) is a global, static and supervised discretization method. It discretizes a continuous feature in a way that the AUC based only on that feature is to be maximized. The proposed method is compared with alternative discretization methods such as Entropy-MDLP (Minimum Description Length Principle) which is known as one of the best discretization methods, Fixed Frequency Discretization (FFD), and Proportional Discretization (PD). FFD and PD are proposed recently and designed for naïve Bayes learning. 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 our method is a candidate to be a good alternative to other discretization methods.

Keywords: Data Mining, Discretization, Area under ROC Curve

1 Introduction

As data comes in different forms, it is always possible to encounter continuous attributes in any datasets. Briefly, discretization methods aim to find the proper cut-points which form the intervals in the progress of discretization. A continuous attribute is then treated as a discrete attribute whose number of intervals is known on the continuous space.

Discretization methods have received great attention from researchers and different kind of discretization methods based on different metrics are proposed. Recently, Yang and Webb (2009) proposed two new discretization methods for the Naïve Bayes classifier**.** There are important reasons of this attention such as the inability of many machine learning algorithms which are not able to work with continuous values. Aggregating one-dependence estimators (AODE) is one of these algorithms which have been used in this research (Webb et al. 2005). It has been shown by Dougherty et al. (1995) that the discretization methods improve the predictive performance and make the algorithms work faster. These are other reasons for using discretization methods.

Liu et al. (2002) categorized discretization algorithms in four axes. These categories include *supervised vs. unsupervised, splitting vs. merging, global vs. local and dynamic vs. static*.

Simple methods such as equal width or equal frequency binning algorithms do not use the class labels of instances during the discretization process (Holte 1993). These methods are called *unsupervised discretization methods.* To improve the quality of the discretization, methods that use the class labels are proposed. Since these methods utilize class labels during discretization, they are referred as s*upervised discretization methods.*

Splitting methods take the given continuous space and try to divide it into small intervals by finding proper cut-points, whereas merging methods handle each distinct point on the continuous space as an individual candidate of cut-point and merges them to larger intervals.

Some of the discretization methods process localized parts of the instance space during discretization. As an example, C4.5 algorithm handles numerical values by using a discretization (binarization) method which is applied to localized parts of the instance space (Quinlan 1993, 1996). Therefore, these methods are called local methods. The methods which use whole instance space of the attribute that is going to be discretized are called global methods.

Dynamic discretization methods use whole attribute space during discretization. Dynamic discretization methods perform better on data with interrelations between attributes. On the other hand, static discretization methods discretize attributes one by one. These methods assume that there are no interrelations between attributes.

In this paper, we propose a discretization method called Maximum Area under ROC Curve Based Discretization (MAD). According to the categories defined above, MAD is categorized as a supervised, merging, global and static discretization method.

Splitting discretization methods usually aim to optimize measures such as entropy (Quinlan 1986; Catlett 1991; Fayyad and Irani 1992; Van de Merckt 1990; Cerquides and Mantaras 1997), dependency (Ho and Scott 1997) or accuracy (Chan et al. 1991) of values placed into the bins. On the other hand, merging algorithms proposed so far uses X^2 statistic (Kerber 1992; Liu and Setiono 1995; Wang and Liu 1998). As far as we know, Receiver Operating Characteristics (ROC) Curve has never been employed in discretization domain before.

1.1 Receiver Operating Characteristics (ROC)

The first application of ROC was the analysis of radar signals in World War II (Krzanowski & Hand, 2009). Later, it is used in different areas such as signal detection theory and medicine (Green and Swets 1966; Zweig and Campbell 1993; Pepe 2003). It was applied to machine learning by Spackman (1989) for the first time. According to Fawcett's definition, ROC graph is a tool that can be used to visualize, organize and select classifiers based on their performance (Fawcett 2006). They have become a popular performance measure in machine learning community after it is realized that accuracy is often a poor metric to evaluate classifier performance (Provost and Fawcett 1997; Provost et al. 1998; Ling et al. 2003).

ROC literature mostly depends on the classification problems with two classes (binary classification). In binary classification, each instance *I* has two different class labels, as **p** (positive) and **n** (negative). At the end of the classification phase, some classifiers simply map each instance to a class label (discrete output). Also there are classifiers which are able to estimate the probability of an instance belonging to a specific class (continuous valued output, also called as *score*). Classifiers produce discrete output represented by only one point in the ROC space since only one confusion matrix is produced from their classification output. Continuous output producing classifiers can have more than one confusion matrix by applying some thresholds to predict class membership. 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. The number of confusion matrices is equal to number of ROC points in an ROC graph.

1.1.1 ROC Space

ROC space is a two dimensional space whose range is [0.0, 1.1] on both axes. In ROC space y-axis represents the true positive rate (TPR) of a classification output and x-axis represents false positive rate (FPR) of output.

To calculate TPR and FPR values, the definitions of the elements in the confusion matrix should be given. The structure of a confusion matrix is shown in Fig. 1. True positives (TP) and false positives (FP) are most important elements of the confusion matrix for ROC graphs. TP is equal to the number of positive instances which are classified correctly. And false positive is equal to the number of negative instances which are not classified correctly. 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.

$$\begin{aligned} \mathrm{TPR} &= \mathrm{TP} / \mathrm{P}\,. \\ \mathrm{FPR} &= \mathrm{FP} / \mathrm{N}\,. \end{aligned} \qquad (\mathbf{1})$$

In this equation *N* is the number of total negative instances and *P* is the number of total positive instances.

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

Fig. 1. Structure of a confusion matrix.

1.1.2 Calculation of ROC Curve

As mentioned above, the classifiers which are producing continuous output can form a curve in ROC graph as they are represented by more than one point in the graph. 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. However, this operation is computationally costly and it is possible to form ROC curve more efficiently with other approaches.

As proposed by Fawcett (2006), in order to calculate ROC curve efficiently, classification scores are sorted in an increasing order first. Starting from $-\infty$, each distinct score element is taken as a threshold, TPR and FPR values are calculated using Eq. 1.

As an example, assume that the score values for test instances and actual class labels for a toy dataset are 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. Starting from $-\infty$, nine different thresholds are used since total threshold value is equal to the S+1 where S is the number of distinct classifier scores in the dataset. With this simple method it is possible to calculate the ROC curve in linear time.

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

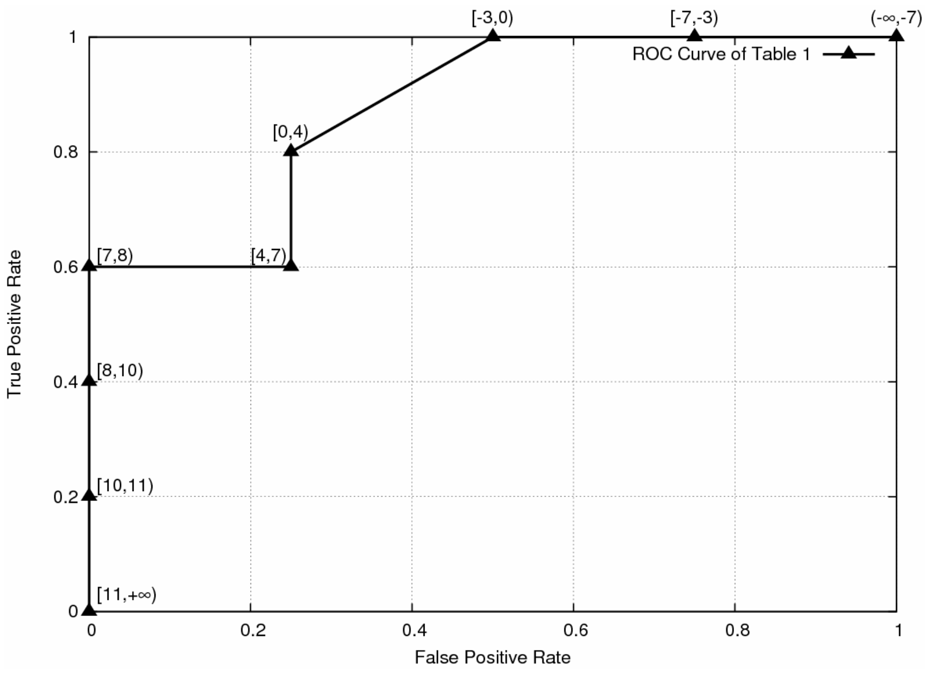

Fig. 2. ROC graph of the given toy dataset in table 1.

1.2 Area under ROC Curve (AUC)

ROC graphs are useful to visualize the performance of a classifier but a scalar value to compare classifiers is needed. In the literature, the area under the ROC curve (AUC) is proposed as a performance measure by Bradley (1997). According to the measure AUC, the classifier with a higher AUC value performs a 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 0.5 AUC value means random guessing and the values below 0.5 are not realistic as these values can be negated by changing the decision criteria of the classifier.

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 (1982) show that this is equal to the Wilcoxon test of ranks. This property of AUC of an ROC curve can be used to design a discretization algorithm. The details of such an algorithm will be given in the next section. Third section will present the empirical evaluations by using real world datasets. The last section will conclude with some future directions for improvement.

2 MAD Method

In this section, the details of MAD method will be given. Firstly, the definition of concepts such as cut-points, ROC space and stopping criteria will be given. After that, different behavior of MAD in two-class datasets and multi-class datasets will be examined on separate sections.

2.1 Definition of cut-points

Given an attribute *A* which has *N* instances with known values and let *C* be the number of distinct continuous values in A's instance space. There can be *C*-1 candidate cut-points to be used in discretization process.

First, the instances are sorted (in this work in increasing order) according their values for the attribute A. Then, each of these candidate cut-points are calculated by using Eq. 2.

$$C_n = (N_n + N_{n+1}) / 2, \tag{2}$$

where N_n and N_{n+1} are distinct and consecutive values in the sorted instance space

2.2 Definition of ROC space for discretization

The numerical attribute values are taken as hypothetical classifier scores that are needed to draw ROC curve; the cut-points are used as the threshold values.

According to Eq. 1 for the threshold value $-\infty$, the TPR and FPR values will be 1, corresponding to coordinate (1,1) in the ROC space. The method will continue incrementally drawing ROC curve by using each candidate cut-point as the threshold values. Finally, the ROC point corresponding coordinate (0,0) with the threshold $+\infty$ will be reached. Total number of ROC points for discretization is *C*-1 plus two for the trivial end points.

2.3 Discretization measure

As mentioned above different measures such as entropy, accuracy, dependency and X^2 statistic have been used in discretization methods. 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 the property that an ROC curve results in a high AUC value when the **p** labeled instances have higher score values then the **n** labeled instances. By using this heuristic, the minimum number of ROC points which maximize AUC value will be selected. This means minimum number of cut-points which 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 discretization measure of this method, a better discretization is achieved.

2.4 Stopping criteria

MAD is a merging discretization method that continues to merge candidate cut-points to 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.

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. ROC convex hull is defined by Provost and Fawcett (2001) to form classifiers which maximize the AUC value. A similar approach will be used to select cut-points which maximize 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 an ROC curve for a given space which has a larger AUC. This curve should contain a point outside the convex hull to make area even larger. Since convex hull enclose all points in the space, this is a contradiction.

2.5 Algorithm

MAD method finds the proper cut-points for the given instance space to maximize the AUC value. In turn, in order to maximize AUC, the convex hull is calculated in the given space.

As MAD algorithm finds this convex hull by using different methods for two-class and multi-class datasets, both of these situations will be given in detail in Section 2.5.1 and Section 2.5.2. Outline of the algorithm is given in Fig. 3.

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

Fig.3. Outline of the MAD algorithm

As it will be revisited in section 2.5.1, there exist a symmetry in every ROC curves 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 is symmetric of the original about the *y*=*x* line. This means that it is possible to find the discretization result by calculating one ROC curve. Conversely, this symmetry does not exist for multi-class datasets. In this case, more than one ROC curves will be obtained. Therefore, two-class datasets and multi-class datasets require different treatment.

2.5.1 Discretization in two-class datasets

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

```
MAD2C (trainInstances)
   Begin
      sort(trainInstances);
      rocPoints= calculateROCPoints(trainInstance);
      cutPoints= findConvexHull(rocPoints);
      return cutPoints;
   end
function calculateROCPoints(trainInstance)
   begin
      rocPoints<- (+∞,0,0),(-∞,1,1);
      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)
      findConvexHull(pointsKept);
   else return pointsKept;
end
```

Fig. 4. MAD algorithm in two-class datasets

There are some important points that deserve elaboration. One of them is about the calculation of ROC points. In order to calculate ROC points for the given sorted attribute, total number of **p** and **n** classes should 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**, b) label low scored instances as **p** and low scored instances as **n**. The choice of the class labeled as **p** does not effect the discretization process according to the symmetry proved in Theorem 2. Then, for each candidate point TPR and FPR values are calculated by using Eq. 1.

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

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

$$\begin{gathered} x = FPR = FP / N \\ y = TPR = TP / P \end{gathered} \qquad (3)$$

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

$$\begin{gathered} x' = FPR' = TP / P \\ y' = TPR' = FP / N \\ x' = y \text{ and } y' = x \end{gathered} \qquad (4)$$

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

(a)

		Actual Class	
		n	**p**
Predicted Class	**p**	FP	TP
	n	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.

Corollary 1 Since there exist a symmetry between the ROC points, the one which is above $y=x$ line is taken into consideration. The points below $y=x$ line are not candidate points to maximize AUC since default AUC value is 0.5. The 17th line of the algorithm given in Fig. 4 assures this property.

Next step of the discretization is selecting the ROC points which form the convex hull. There are different methods to calculate convex hull in the given n dimensional space. One of these methods is called QuickHull by Preperata and Shamos (1985). This method has $O(n \lg n)$ time complexity in practical and $O(n^2)$. In this work, a new method to calculate convex hull for two-class problems is proposed.

FindConvexHull function is given on the 32th line of the algorithm given in Fig. 4. The main motivation for the function findConvexHull is the ordering of the ROC points. The first point created on the graph always corresponds to (1,1) and the last (0,0). The points lying between these two trivial points have a monotonically non-decreasing property. For example, assume that *v1* is the point which is created just before *v2*. Point *v1* always stands on the north, east or north-east side of the *v2*. These points create a shape (possibly include concavities) when the points are connected to each other with hypothetical lines during the ROC curve creation stage. FindConvexHull method compares the slopes in the order of creation of these hypothetical lines with each other and finds the junction points (cut-points), which cause concavities, and eliminates them.

FindConvexHull method guarantees to find the convex hull in best case $O(n)$ time and in worst case $O(n^2)$. In the worst case, the method should at least one point out to call itself again. This leads to $O(n^2)$ complexity. In best case, the method finds the convex hull in a linear time if the points are already forming a convex hull. Average case of the algorithm will be given by empirical results.

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

Table 2. Toy dataset for visualization of MAD in two-class problems. F1 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

Each step of calculation of the convex hull in the given ROC space is visualized in figures Fig. 6 thru Fig 8. In Fig. 6, generated ROC points for both classes 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 above *y*=*x* line will be processed in the next step.

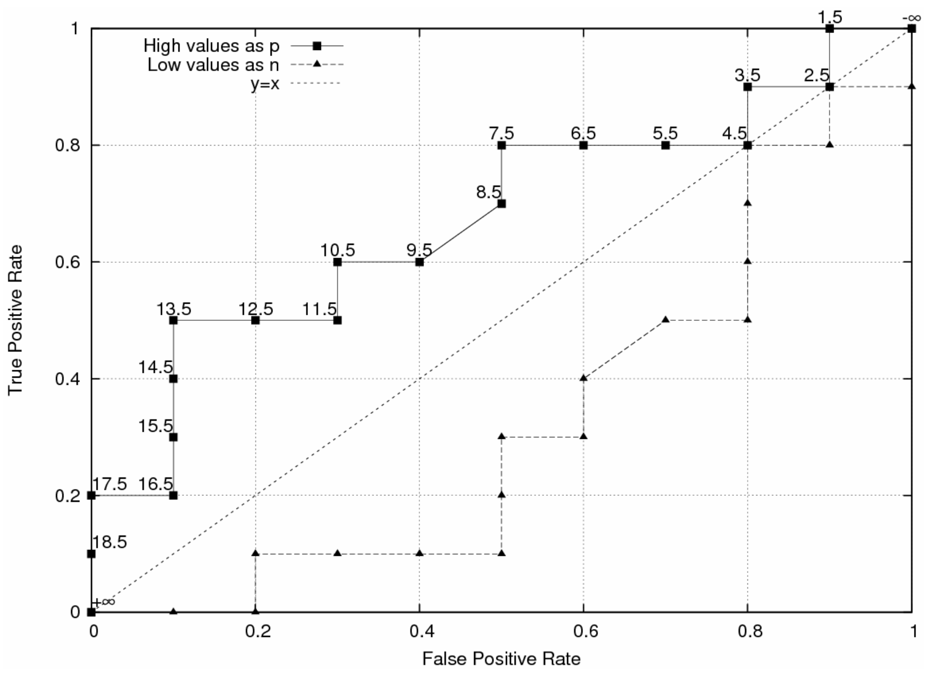

Fig. 6. Visualization of ROC point in two-class discretization

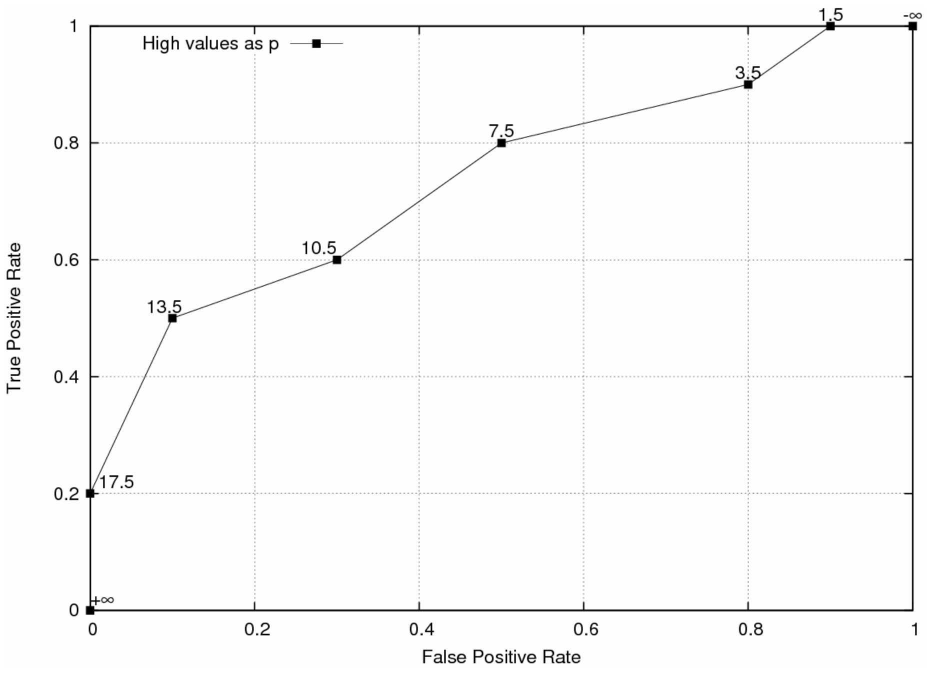

Fig. 7. First pass of the convex hull algorithm

In the next step of MAD, points which cause concavity will be eliminated. Fig. 7 shows the points left after the first pass of the method which 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.

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 which are going to be used in discretization. Fig. 8 shows the final cut points left on the graph.

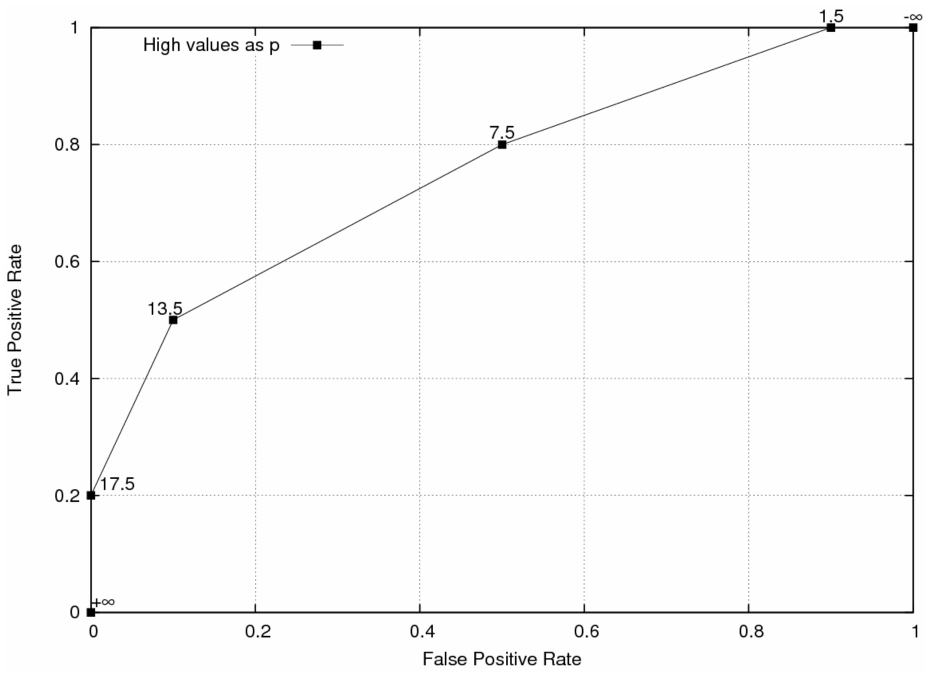

Fig. 8. Final cut points left after the second pass of convex hull algorithm.

MAD method guarantees that any cut point left at 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 it can be extended to multi class problems as well.

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

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

$$\begin{gathered}\mathrm{TPR}_{n-1} = p' / \mathrm{P},\ \mathrm{TPR}_{n} = p' / \mathrm{P},\ \mathrm{TPR}_{n+1} = p' / \mathrm{P} \\ \mathrm{FPR}_{n-1} = n' / \mathrm{N},\ \mathrm{FPR}_{n} = n'\text{-}k / \mathrm{N},\ \mathrm{FPR}_{n+1} = n'\text{-}l / \mathrm{N}\end{gathered} \tag{5}$$

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

$$\begin{gathered} \mathrm{TPR}_{\boldsymbol{n-1}} = p' / \mathrm{P},\ \mathrm{TPR}_{\boldsymbol{n}} = p'\text{-}k / \mathrm{P},\ \mathrm{TPR}_{\boldsymbol{n+1}} = p'\text{-}l / \mathrm{P} \\ \mathrm{FPR}_{\boldsymbol{n-1}} = n' / \mathrm{N},\ \mathrm{FPR}_{\boldsymbol{n}} = n' / \mathrm{N},\ \mathrm{FPR}_{\boldsymbol{n+1}} = n' / \mathrm{N} \end{gathered} \qquad \textbf{(6)}$$

2.5.2 Multi class behavior

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

The method used for the two-class datasets can be extended to the multi-class problems by relabeling one class as **p** and all others as **n** and obtaining the ROC curves. This technique is used by Provost and Domingos (2001) in order to calculate ROC curves for multi-class datasets. The convex hull of the ROC curve is computed. This process is repeated for all class labels. *C* many different convex hulls are summed together and final convex hull is found by using Quickhull method. Outline of the multi-class MAD method is given in Fig. 9.

```
MADMC (trainInstances)
   Begin
      sort(trainInstances);
      for each class
         mark current class as p others as n
         rocPoints=calculateROCPoints(trainInstance);
         totalrocPoints+=findConvexHull(rocPoints);
      cutPoints= QuickHull(totalrocPoints);
      return cutPoints;
   end
```

Fig. 9. Multi-class MAD algorithm

Multi-class MAD uses the same function to calculate ROC points (*calculateROCPoints* given in algorithm in Fig. 4) and convex hull (*findConvexHull* given in algorithm in Fig. 4) as it use in two-class datasets. Therefore, the Theorem 3 applies to the multi-class MAD algorithm; that is, it is guaranteed that a cut-point does not lie between two consecutive instances of the same class.

In Fig. 10 an example visualization of discretization process for multi-class datasets is given. 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 shaded area are the final cut-points that are going to be used in discretization process.

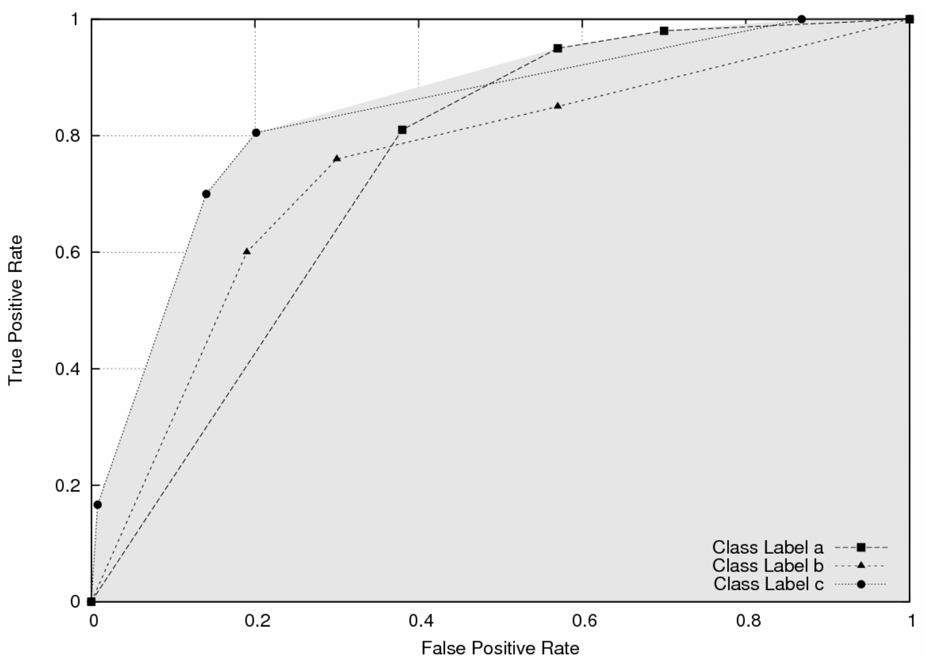

Fig. 10. Visualization of discretization process for multi-class datasets. An attribute of a three-class dataset is used. Gray shaded area represents ROC convex hull.

By relabeling a class as **p** and marking others as **n**, discretization method becomes to be sensitive to the 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 QuickHull method and some valuable information can be lost. This drawback can be solved by creating pairwise class ROC curves which is a similar method used in M-Measure (Hand and Till 2001). Then, $C(C\text{-}1)$ different ROC curves have to be created. On the other hand, this will increase the computation time since the number of convex hull has to be calculated increases. Even with this drawback MAD algorithm works well in multi class datasets in the UCI repository (Asuncion and Newman 2007).

3 Empirical Evaluations

In this section, MAD discretization method is compared with the well known Entropy-MDLP method proposed by Fayyad and Irani (1992) and two other methods (FFD and PD) proposed recently by Yang and Webb (2009).

As a splitting method, Entropy-MDLP method uses entropy measure to select proper cut-point to split an interval. An application of minimum description length principal called *information gain* is used as stopping criteria. In the nutshell, it selects the proper cut-points which minimize the entropy for the given interval and continues to discretize recursively until information gain is not sufficient.

The unsupervised FFD and PD methods are designed in order to obtain high classification accuracy (lower classification error) by managing discretization bias and variance. FFD discretizes attributes into equal sized intervals where each bin contains approximetely 30 instances. PD also discretizes attributes into equal sized intervals but the number of instances in each interval is not fixed for each dataset. In PD interval frequency is calculated by using Eq. 7.

$$s \text{ x } t = n, \quad s = t, \tag{7}$$

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

The discretization results obtained by MAD, Entropy-MDLP, FFD, and PD methods in real life datasets will be shown. All datasets used in the experiments are taken from the UCI Machine Learning Repository. Table 3 shows the properties of these datasets.

The performance of the algorithms is evaluated in four different aspects: predictive performance, running time, inconsistency of intervals, and number of intervals found.

Table 3. Dataset used in the experiments

Name	# Instances	# Continuous Attributes	# Nominal Attributes	# Class Labels
Abalone	4177	7	1	29
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
Page-Blocks	5473	10	0	5
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

3.1 Predictive performance

Classifiers that associate the predicted class with a confidence value are preferred in this work since their ROC curve representation is more meaningful. Two different classifiers, which are supporting this property, are selected. One of them is Naïve Bayes classifier. Naïve Bayes is one of the simplest and most effective classifiers and it is shown that using discretization with naïve Bayes algorithm increases predictive accuracy (Dougherty et al. 1995). The other algorithm selected, AODE, requires that all features are categorical.

The naïve Bayes, AODE, and Entropy-MDLP discretization implementations are taken from source codes of WEKA package (Hall et al. 2009). Naïve Bayes algorithm is used in default form which uses single normal distribution rather than kernel estimation. 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 number of intervals as shown in Eq. 7, and the number of intervals is passed as a parameter.

Five different cases are being considered for naïve Bayes algorithm in this section: The naïve Bayes algorithm with MAD discretization method, naïve Bayes with Entropy-MDLP method, naïve Bayes with FFD method, naïve Bayes with PD method, and naïve Bayes with continuous values (without discretization). Also four different cases will be considered for AODE algorithm with MAD discretization method, AODE algorithm with Entropy-MDLP discretization method AODE algorithm with FFD discretization method, and AODE with PD method.

Two different measures have been used to evaluate predictive performance. First measure is called M-Measure. This measure is suitable to calculate both two-class AUC and multi-class AUC values. M-Measure is insensitive to the class distribution and error costs. Since MAD is based on AUC, one might question the impartiality of using a performance metric that depends on AUC. To clear that question mark, predictive performance of MAD against other discretization methods is measured by using accuracy metric as well. Stratified 10-fold cross validation is employed in order to calculate M-Measure and accuracy values for each dataset.

The predictive performance evaluation result of naïve Bayes obtained by using M-Measure is given in Table 4. It is seen in this table that the MAD algorithm outperforms all other discretization methods in terms of the M-Measure on average. Paired-sample t-test method shows that in 95% confidence interval, MAD improves naïve Bayes algorithm performance significantly compared to the performance obtained with using FFD method, PD method or without using discretization. On the other hand, Entropy-MDLP method does not improve naïve Bayes algorithm performance statistically significantly according to M-Measure.

Table 4. Predictive performance of Naïve Bayes in terms of M-Measure under different discretization methods (Higher values are better).

Name	MAD	Entropy MDLP	FFD	PD	Without discretization
Abalone	0,650	0,598	0,643	0,637	0,642
Bupa	0,750	0,540	0,684	0,685	0,626
Crx	0,924	0,928	0,928	0,929	0,900
Dermatology	0,999	0,999	0,999	0,999	0,997
German	0,794	0,775	0,787	0,787	0,785
Glass	0,915	0,918	0,914	0,900	0,861
Heart (Statlog)	0,906	0,901	0,905	0,899	0,897
Ozone-onehr	0,859	0,853	0,831	0,841	0,843
Page-Blocks	0,969	0,978	0,966	0,977	0,951
Sick-euthyroid	0,952	0,959	0,950	0,953	0,920
SPECTF	0,965	0,964	0,950	0,957	0,940
Spambase	0,864	0,824	0,850	0,845	0,850
Transfusion	0,715	0,686	0,689	0,682	0,711
Wisconsin	0,987	0,986	0,987	0,986	0,980
Yeast	0,831	0,837	0,812	0,821	0,865
Average	**0,872**	0,850	0,860	0,860	0,851

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. According to the paired-sample t-test on 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.

Table 5. Predictive performance of Naïve Bayes in terms of accuracy under different discretization methods (Higher values are better).

Name	MAD	Entropy MDLP	FFD	PD	Without discretization
Abalone	25,79	25,26	25,91	26,89	23,95
Bupa	70,13	57,70	63,18	63,22	53,96
Crx	85,91	86,53	85,61	86,22	77,97
Dermatology	97,55	98,09	98,09	97,82	97,81
German	75,90	73,50	75,20	75,20	75,00
Glass	70,82	70,89	68,55	68,05	48,35
Heart (Statlog)	82,59	82,59	82,59	82,96	83,70
Ozone-onehr	78,34	79,88	87,81	83,31	70,77
Page-Blocks	94,54	93,42	93,40	92,40	90,15
Sick-euthyroid	95,89	96,02	95,07	95,32	84,22
SPECTF	89,96	89,81	87,83	89,13	79,72
Spambase	76,81	72,66	74,16	76,79	68,58
Transfusion	77,94	75,27	76,47	75,27	75,67
Wisconsin	94,37	94,19	94,19	93,67	93,49
Yeast	58,32	56,71	52,79	53,74	57,99
Average	**78,32**	76,83	77,39	77,33	72,09

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 paired-sample t-test on 95% confidence interval MAD method outperforms both FFD and PD methods. Also the MAD method performs better than Entropy-MDLP method on the average.

Table 6. Predictive performance of AODE in terms of M-Measure under different discretization methods (Higher values are better).

Name	MAD	Entropy MDLP	FFD	PD
Abalone	0,649	0,628	0,646	0,651
Bupa	0,752	0,540	0,656	0,665
Crx	0,929	0,930	0,932	0,928
Dermatology	0,999	0,999	0,999	0,999
German	0,793	0,783	0,788	0,788
Glass	0,920	0,925	0,940	0,915
Heart (Statlog)	0,907	0,904	0,904	0,895
Ozone-onehr	0,891	0,878	0,762	0,723
Page-Blocks	0,974	0,984	0,932	0,955
Sick-euthyroid	0,964	0,963	0,957	0,959
SPECTF	0,977	0,980	0,945	0,959
Spambase	0,865	0,820	0,829	0,794
Transfusion	0,697	0,707	0,661	0,654
Wisconsin	0,992	0,988	0,989	0,988
Yeast	0,822	0,833	0,802	0,813
Average	**0,875**	0,858	0,849	0,846

The predictive performance of AODE algorithms in terms of accuracy metric is given in Table 7. According to this table MAD method outperforms all other discretization methods on the average again.

Table 7. Predictive performance of AODE in terms of accuracy under different discretization methods (Higher values are better).

Name	MAD	Entropy MDLP	FFD	PD
Abalone	27,13	25,67	26,17	27,25
Bupa	71,85	57,70	63,19	62,39
Crx	86,68	86,68	87,45	86,68
Dermatology	97,55	98,08	98,09	97,55
German	75,40	74,90	75,60	75,60
Glass	75,50	73,20	77,88	76,47
Heart (Statlog)	82,22	82,96	82,96	82,22
Ozone-onehr	96,21	88,76	96,65	96,88
Page-Blocks	96,62	96,97	95,74	96,18
Sick-euthyroid	96,65	96,52	95,26	95,92
SPECTF	93,35	93,31	88,02	90,15
Spambase	79,42	73,77	79,37	76,81
Transfusion	76,34	75,27	77,41	77,55
Wisconsin	95,78	96,12	95,60	94,91
Yeast	57,38	56,77	53,67	54,41
Average	**77,00**	75,96	76,28	76,19

3.2 Running Time

In machine learning it is also essential to deal with large datasets. Therefore, the running time of the proposed method is critical. The worst and the best case running time complexity of the MAD algorithm are given in the Section 2.5.1. In this section, the running times on real life datasets are given empirically.

As mentioned in Section 2.5, the main time consuming step of MAD (after sorting) is finding the convex hull as fast as possible. In two class problems only one convex hull is calculated. On the other hand, in multi-class situation 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 Section 2.5. Hence, the average running time of finding the convex hull method is the most prominent part in the running time of whole method.

The proposed convex hull calculation method is invoked recursively until it converges to the convex hull. In order to give an insight of running time of algorithm in practice, Table 8 shows the average number of recursive call for an attribute to calculate convex hull. According to Table 8 it is possible to say that, regardless of the number of instances, the convex hull can be found by calling the function recursively minimum one, maximum six and on average four times for the given datasets.

Table 8. Average number of calling time to calculate convex hull for an attribute of each datasets.

Name	# Recursive Call
Abalone	4
Bupa	5
Crx	5
Dermatology	1
German	3
Glass	4
Heart (Statlog)	2
Ozone-onehr	5
Page-Blocks	6
Sick-euthyroid	5
SPECTF	4
Spambase	6
Transfusion	4
Wisconsin	6
Yeast	3
Average	**4**

The overall running time of all methods are measured. To be objective, the running times are measured by using java virtual machine's CPU time and hundred of different runs are averaged. Table 9 shows the average running times of all algorithms in all datasets, in microseconds (μs). According to this table, on the average, all other methods outperform MAD method. As mentioned above for multi-class datasets, the MAD method calculates *n* different ROC curves where *n* is the number of attributes. Also it combines these curves with QuickHull algorithm whose complexity (no worse than O(*n* log *n*) in practical) is higher than unsupervised discretization methods.

Table 9. Average running time results for all datasets (in μs) (lower results are better).

Name	MAD	Entropy-MDLP	FFD	PD	# Class label
Abalone	599955	87075	33437	62924	29
Bupa	1406	2250	1625	4593	2
Crx	3593	4661	4609	11484	2
Dermatology	7752	1732	1576	7274	7
German	5200	7165	7410	24352	2
Glass	6909	2465	1059	3541	10
Heart (Statlog)	1177	1430	1370	3894	2
Ozone-onehr	10928	14124	16511	167442	2
Page-Blocks	150625	69796	52562	120843	5
Sick-euthyroid	24753	26288	24429	250764	2
SPECTF	17834	25334	30000	148526	2
Spambase	1129	1207	1104	6946	2
Transfusion	2929	4570	5468	9179	2
Wisconsin	3588	5708	3072	14640	2
Yeast	53476	8886	7890	20332	10
Average	59417	17513	**12808**	57116	

On the other hand, we are expecting MAD algorithm to work faster on two-class datasets. As it can be seen in Table 10, MAD outperforms all other discretization methods in terms of running time on average for two-class datasets used in this paper.

Table 10. Average running time results for two-class datasets (in µs) (lower results are better).

Name	MAD	Entropy-MDLP	FFD	PD
Bupa	1406	2250	1625	4593
Crx	3593	4661	4609	11484
German	5200	7165	7410	24352
Heart (Statlog)	1177	1430	1370	3894
Ozone-onehr	10928	14124	16511	167442
Sick-euthyroid	24753	26288	24429	250764
SPECTF	17834	25334	30000	148526
Spambase	1129	1207	1104	6946
Transfusion	2929	4570	5468	9179
Wisconsin	3588	5708	3072	14640
Average	**7254**	9274	9560	64182

3.3 Inconsistency of intervals

Firstly, inconsistency of an interval should be defined. Given an interval which consists of *n* instances, inconsistency of that interval is equal to value *n-c* where *c* is the number of instances which 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 others to the class *b*, the inconsistency of the given interval is 10-7=3.

A smaller amount of inconsistency indicates a better discretization. As entropy is a measure that aims to obtain pure intervals, it is expected to achieve lower inconsistency values with Entropy-MDLP discretization method. On the other hand, Table 11 shows that on the average the MAD method forms intervals which are more consistent than the Entropy-MDLP method. On the other hand, FFD and PD methods have lower inconsistency on the cost of producing a very high number of intervals.

Table 11. Total inconsistencies for continuous attributes in given datasets. (Lower results are better).

Name	MAD	Entropy-MDLP	FFD	PD
Abalone	3067	3120	2957	3001
Bupa	138	143	135	129
Crx	216	221	218	215
Dermatology	214	216	214	213
German	296	299	296	296
Glass	107	113	113	106
Heart (Statlog)	89	93	89	88
Ozone-onehr	72	73	72	73
Page-Blocks	494	486	481	492
Sick-euthyroid	275	273	271	273
SPECTF	1581	1578	1544	1560
Spambase	54	55	54	54

Transfusion	175	178	173	172
Wisconsin	122	127	125	123
Yeast	946	961	923	927
Average	523	529	**511**	515

3.4 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, entropy-MDLP method is known as producing less number of intervals. It is even shown by An et al. (1999) that Entropy-MDLP method stops too early on small datasets.

The average number of intervals per attribute is given in Table 12. According to these results, Entropy-MDLP outperforms MAD on this aspect. On the other hand, an important point should be mentioned about the results in table. In eight datasets, the Entropy-MDLP method discretizes all attributes into one interval on the average. When an attribute is discretized into one interval this means that the discretization method maps all elements to the same value such as ($-\infty \ldots +\infty$). This situation occurs if the distribution of the attribute values is not suitable according to the discretization methods measure. It is possible to say that Entropy-MDLP is acting like a feature selection algorithm. In turn, as shown in Section 3.1, there exists predictive accuracy gain in some of these datasets. Therefore, in some cases the Entropy-MDLP method misses important information by mapping all instances to the same interval.

Table 12. Average number of intervals per attribute. (Lower results are better).

Name	MAD	Entropy-MDLP	FFD	PD
Abalone	27	6	125	58
Bupa	7	1	10	16
Crx	10	2	19	23
Dermatology	2	1	10	4
German	5	1	30	14
Glass	7	1	6	13
Heart (Statlog)	4	1	8	7
Ozone-onehr	9	1	76	45
Page-Blocks	10	5	164	69
Sick-euthyroid	9	2	94	44
SPECTF	8	2	138	62
Spambase	7	1	8	15
Transfusion	9	1	22	24
Wisconsin	13	3	17	22
Yeast	11	2	44	26
Average	9	**2**	51	29

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

4 Conclusions

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

According to the empirical evaluations MAD method outperforms Entropy-MDLP method which is one of the most well known discretization method in terms of predictive performance. MAD also outperforms FFD and PD methods in terms of predictive performance. Since these two discretization methods are naïve Bayes optimal, the significant gain in naïve Bayes algorithm in terms of M-Measure is important. It is also shown by experiments that MAD method runs faster than other discretization methods for two-class datasets. In terms of inconsistencies of intervals MAD method outperforms Entropy-MDLP method on average but it is outperformed by FFD and PD methods. This was expected, due to the inherent design of FFD and PD methods, they will intend to produce large number of intervals which brings pure intervals naturally. Especially the gain in the predictive performance and faster running time results show that MAD method is a good alternative to other key discretization methods.

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

References

An A and Cercone N (1999) Discretization of Continuous Attributes for Learning Classification Rules, Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining. Beijing, China. pp. 509-514.

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

Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7):1145–1159.

Catlett J (1991) On changing continuous attributes into ordered discrete attributes. In Proc. Fifth EuropeanWorking Session on Learning. Berlin: Springer-Verlag, pp. 164–177.

Cerquides J and Mantaras RL (1997) Proposal and empirical comparison of a parallelizable distance-based discretization method. In KDD97: Third International Conference on Knowledge Discovery and Data Mining, pp. 139–142.

Chan CC, Batur C, Srinivasan A (1991) Determination of quantization intervals in rule based model for dynamic. In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. Charlottesvile, Virginia, pp. 1719–1723.

Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In A. Prieditis & S. Russell (Eds.), Proceedings of the Twelfth International Conference on Machine Learning (pp. 194–202). San Francisco, CA: Morgan Kaufmann.

Fawcett T (2006) An introduction to ROC analysis. Pattern Recognition Letters, 27:861-874.

Fayyad U and Irani K (1992) On the handling of continuous-valued attributes in decision tree generation. Machine Learning, 8:87–102.

Green DM and Swets JA (1966) Signal Detection Theory and Psychophysics. Wiley, New York.

Hall M et al. (2009) The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

Hand DJ and Till RJ (2001) A simple generalization of the area under the ROC curve to multiple class classification problems. Machine Learning 45 (2), 171–186.

Hanley JA and McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29–36.

Ho KM and Scott PD (1997) Zeta: A global method for discretization of continuous variables. In KDD97: 3rd International Conference of Knowledge Discovery and Data Mining. Newport Beach, CA, pp. 191–194.

Holte RC (1993) Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11:63–90.

Kerber R (1992) Chimerge: Discretization of numeric attributes. In Proc. AAAI-92, Ninth National Confrerence Articial Intelligence. AAAI Press/The MIT Press, pp. 123–128.

Krzanowski WJ and Hand DJ, (2009) ROC curves for continuous data, CRC Press, Taylor & Francis Group.

Ling CX, Huang J, Zhang H (2003) AUC: a Statistically Consistent and more Discriminating Measure than Accuracy. Proceedings of the Eighteenth International Joint Conference of Artificial Intelligence (IJCAI), pp. 329-341.

Liu H and Setiono R (1995) Chi2: Feature selection and discretization of numeric attributes. In Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence. Herndon, Virginia, pp. 388–391.

Liu H, Hussain F, Tan CL, Dash M (2002) Discretization: An enabling technique. Data Mining and Knowledge Discovery, 6(4):393–423.

Pepe MS (2003) The statistical evaluation of medical tests for classification and prediction. Oxford, New York.

Preperata DF and Shamos M (1985) Computational Geometry. An Introduction. Springer-Verlag, Berlin.

Provost F and Domingos P (2001). Well-trained PETs: Improving probability estimation trees, CeDER Working Paper #IS-00-04, Stern School of Business, New York University, NY 10012.

Provost F and Fawcett T (1997) Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: Proc. Third Internat. Conf. On Knowledge Discovery and Data Mining (KDD-97). AAAI Press, Menlo Park, CA, pp. 43–48.

Provost F, Fawcett T (2001) Robust classification for imprecise environments. Machine Learning 42(3), 203–231.

Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing induction algorithms. In Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann, pp. 445–453.

Quinlan JR (1986) Induction of decision trees. Machine Learning, 1:81–106.

Quinlan JR (1993) C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.

Quinlan JR (1996) Improved Use of Continuous Attributes in C4.5. Journal of Artificial Intelligence and Research 4:77-90.

Spackman KA (1989). Signal detection theory: Valuable tools for evaluating inductive learning. Proceedings of the Sixth International Workshop on Machine Learning. San Mateo, CA: Morgan Kaufmann. pp. 160–163.20.

Van de Merckt T (1990) Decision trees in numerical attribute spaces. Machine Learning, 1016–1021.

Wang K and Liu B (1998) Concurrent discretization of multiple attributes. In Pacific Rim International Conference on AI, pp. 250–259.

Webb G, Boughton J, Wang Z (2005) Not so naive bayes: Aggregating one-dependence estimators. Journal of Machine Learning 58(1):5-24.

Yang Y and Webb G (2009) Discretization for naive-Bayes learning: managing discretization bias and variance. Machine Learning 74(1):39–74.

Zweig MH and Campbell G (1993) Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. Clinical chemistry 39 (8):561–577.