D-FS: A Novel Integration Method of Discretization and Feature Selection

Bin Fu, Hongzhi Liu*, Zhengshen Jiang, Zhonghai Wu
School of Software and Microelectronics,
Peking University, Beijing, 102600, P. R. China
Email: {binfu, liuhz, jiangzhengshen, wuzh}@pku.edu.cn

D. Frank Hsu
Department of Computer and Information Science,
Fordham University, New York, NY, 10023, USA
Email: hsu@cis.fordham.edu

***Abstract*—Discretization and feature selection are two basic preprocessing stages of data mining. However, it often results in information loss due to these two separate stages. This paper proposes a novel *supervised multivariate* discretizer integrated with feature selection, called *D-FS*. It takes into consideration of the interactions of both different cut-points and features, and achieves feature selection by discretization. *D-FS* can avoid the information loss caused by the independence of discretization and feature selection. Compared with several state-of-the-art discretizers, *D-FS* retains a smaller subset of both cut-points and features, while achieves competitive classification performance combined with different classifiers.**

***Keywords*—Discretization; Feature Selection; Integration Method;**

I. Introduction

Due to the development of Information Technology, the amount of accumulated data grows explosively. While, during this big data era, only a small proportion of the raw data contains the most useful information. To extract the useful information, data mining becomes more and more important.

There are several kinds of data mining methods, for example, classification aims to find a function which maps a set of predictive features *Xs* to a class feature *Y* [1]. However, the predictive features *Xs* from the raw data are often continuous. This is hard for some classifiers to learn the patterns from the data, especially for the ones which can only deal with categorical features. Other classifiers can also improve their performances based on the ordered discrete features [2]. So before the learning process, the discretization for numeric features is essential with consideration of the efficiency and effectiveness.

Discretization is a preprocessing step which transforms the continuous features into discrete ones, which means it finds a finite set of cut-points on these features. In general, the discretization process could be divided into two steps: fetching all the potential cut-points, and then picking up a subset from them. The former step finds the complete space of cut-points, which constrains the best subset and complexity in extreme case. The second step searches for the best subset according to the evaluation criterion, while this step is not always necessary. According to this subset, the continuous features can be discretized by associating each interval with a discrete value.

*Corresponding author

Discretization, as one of the common preprocessing methods, brings many benefits [22]. The discrete features lead to better performances for some classifiers. It becomes easier for human experts to analyse the data under those simpler and more intuitionistic features. Morever, the discrete data is compact, and requires less memory to store. With these benefits above, even though discretization causes some information loss, discretization is useful especially in this big data era.

There are different divisions for the present discretizers according to their properties: *supervised* / *unsupervised* / *semi-supervised*, *bottom-up(merge)* / *top-down(split)* / *hybrid*, *local* / *global*, *univariate* / *multivariate*, *dynamic* / *static*, *direct(parameterized)* / *incremental(non-parameterized)*, and evaluation measures [5] [10] [11] [21] [22] [24] [28].

Supervised discretization takes into consideration of the class information comparing with *unsupervised* discretization, and *semi-supervised* discretization uses both labeled and unlabeled instances. Two common unsupervised discretizers are the equal-width (*EW*) and the equal-frequency (*EF*). *EW* divides the continuous range into a user-specified number of equal-width intervals, while *EF* let each interval with same number of instances, and its variants are *PKID* [3] and *AEFD* [4]. These two discretizers perform well when the value meets uniform distribution [5] or normal distribution [4]. Clustering-based discretization clusters the data, and finds the maximum and minimum on each feature from each cluster, while its complexity depends on the cluster algorithm [8] [9]. *Supervised* discretization utilizes the class to refine the discretization scheme, which often leads to better performance comparing with *unsupervised* discretization [10]. [25] proposes a *semi-supervised* discretization for semi-supervised classification. Briefly, *supervised* discretization often yields higher accuracy which is fit for some specific applications, while *unsupervised* discretization is fit for multiple purposes [26].

To find the subset of cut-points from the complete set, *bottom-up*, *top-down* and *hybrid* [10] are the search directions. *Bottom-up*/*top-down* discretization starts with the complete/empty set, and iteratively removes/adds cut-points. *Hybrid* discretization combines both *bottom-up* and *top-down*. However, when the selected subset accounts for just a small proportion, computational complexity of the *bottom-up* is worse than the *top-down* [27].

The subset of cut-points can be evaluated on partial/all instances, this is called *local*/*global*. Since *local* discretization based on the local space, it seems more likely to select a subset with redundancy. Similarly, *univariate*/*multivariate* discretization selects the cut-points on one/all feature(s) each time. *Multivariate* discretization takes into consideration of the relevance between different features, which can reduce the redundancy of the subset [29].

Dynamic discretization accomplishes the discretization during the learning stage while *static* discretization discretizes the numeric features prior to the learning process. *Direct* discretization splits the range into k (defined by user) intervals, and *incremental* discretization starts from one subset and iteratively improves it until reaching the stop condition.

When a subset of cut-points are chosen, there are many proposed evaluation measures to evaluate the efficiency of this subset, and they are: entropy-based, statistics-based, rough-set-based, learner-wrapper-based. When the data is discretized into hypercubes, and the entropy-based adopts the information entropy to measure the purity of these hypercubes, like Minimum Description Length Principle (*MDLP*) [30], Gini Index, *EDIC* [31], and *CAIM* [6]. Statistics-based discretization uses statistics methods, such as χ^2 statistic in *Extended-Chi2* [12], *Chi2* [19] and *ChiMerge* [20], conducts the significance test on adjacent intervals to determine whether to combine them or not. Rough-set-based methods, such as *SMD* [24] and *SMDNS* [33], keep the indiscernibility relation or the classification ability of decision table relatively unchanged when merging the adjacent intervals. Learner-wrapper-based, as a complex discretization [11], ultizes the feedback (accuracy [13] or error [14]) from the subsequent learning model to refine the discretization scheme, while the complexity increases sharply. [32] proves the entropy-based and the statistics-based are asymptotically equivalent and then provides a unified generalized entropy function.

Feature selection is another preprocessing step to select a feature subset with minimum redundancy and maximum relevance [15]. The difference between discretization and feature selection is that: the former discretizes the feature value as a way to simplify the inner space of the features, while the latter selects partial features to simplify the outer space. Discretization is often prior to feature selection due to the computation complexity. If any feature is discretized into one value (no cut-points are selected on the feature), this feature can be removed, and which means that this discretizer has the power of feature selection. A few discretizers, *Chi2* [19] and our *D-FS*, have this capability.

In this paper, we propose a novel *supervised multivariate* discretizer, called *D-FS* which integrates discretization with feature selection. *D-FS* starts with a empty set of cut-points, and iteratively adds a cut-point with highest information gain from the candidate cut-points to this set. This simple discretizer is *top-down*, *incremental*, *global* and *static*. The main contributions of this paper are summarized below:

- *D-FS* gives the definition of the integration problem and solves it by finding a suitable discretization scheme. It takes into consideration of the interactions of both cut-points and features simultaneously. This can effectively removes the irrelevant and redundant information in the preprocessing stage of data mining.
- The integration of the discretization and feature selection as only-one process can bring benefits, such as lowering down the information loss and complexity, and this is critical in the big data era.
- Extensive experiments on seventeen benchmark data sets from different fields indicates that *D-FS* can find a smaller set of both cut-points and features, while the performance is similar or better than other state-of-the-art discretizers.

The rest of this paper is organized as follows: Section II introduces the preliminary knowledge of *D-FS* and provides a unified definition of the integration problem. Section III describes our *D-FS* method in details. In Section IV, we provide the experiments and discussion. Section V concludes the paper with future work.

II. Preliminary

In this section, we will first introduce some terminologies used in this paper, and then presents the definition of the integration problem.

A. Preliminary Conceptions

Cut-point: a point with its value v on a feature f which cuts the range into two intervals, we use c represent it.

$$c = (f, v) \tag{1}$$

Discretization Scheme: a subset of cut-points S selected from the complete set C on all features F.

$$S \subseteq C; C = \bigcup_{f \in F} c \tag{2}$$

Discretization Partition: the data can be discretized into hypercubes according to the discretization scheme S. Like clustering, it assigns one label to all the instances in the same hypercube.

$$\overrightarrow{g(S)} \tag{3}$$

Mutual Information: a measure of the mutual dependence between two variables. We know that Shannon entropy for variables $X = \{x_i\}$:

$$H(X) = -\sum_i p(x_i)\text{log}p(x_i)$$

Mutual information is based on entropy between two random variables $X = \{x_i\}$ and $Y = \{y_j\}$:

$$I(X|Y) = H(Y) - H(X|Y)$$

where $H(X|Y)$ is the condition entropy:

$$H(X|Y) = -\sum_{y_j} p(y_j) \sum_{x_i} p(x_i|y_j)\text{log}p(x_i|y_j)$$

So mutual information is also formulated:

$$I(X;Y) = \sum_{x_i}\sum_{y_j} p(x_i, y_j)\log\frac{p(x_i, y_j)}{p(x_i)p(y_j)} \tag{4}$$

In *D-FS*, mutual information is used to measure the relevance between discretization partition $\overrightarrow{g(S)}$ and the class $\overrightarrow{Y}$.

$$\sigma(S) = I(\overrightarrow{g(S)}; \overrightarrow{Y}) \tag{5}$$

Information Gain: when a new cut-point c is added into the discretization scheme S, the mutual information will increase by:

$$\sigma(S \cup c) - \sigma(S) \tag{6}$$

B. Problem Formulation

According to the definitions in *Section II.A*, the discretization scheme contains a set of features and cut-points, so the main task of the integration of the discretization and feature seletion is to search for a suitable discretization scheme.

To evaluate the discretization scheme, we consider two aspects: *relevance* and *redundancy*. Then, we think the *maximal relevance* is the requirement of the optimal ones.

$$\max R(S) \tag{7}$$

While the discretization scheme selected by *maximal relevance* has much redundancy. Due to the Occam's Razor, we also hope the discretization partition is as simple as possible under *maximal relevance*. We define the complexity of the discretization scheme $\Omega(S)$ which also satisfies *minimum redundancy*.

$$\min \Omega(S) \tag{8}$$

Combining *Expression 7* and *8*, we give the generalized formal definition for this integration problem.

$$\min \Omega(S) \; under \; \max R(S); S \subseteq C \tag{9}$$

III. D-FS: Discretization and Feature Selection

First, we analyze the integration problem and give our solution. Then, we present how to implement the *D-FS* algorithm in details.

A. D-FS Solution

To solve the problem *Expression 9*, first we need to find all most relevant discretization schemes and then choose the one with *minimum redundancy*, while this is a NP-Complete problem. Thus, heuristic rule, like greedy approach, is used in *D-FS*.

For *relevance*, the mutual information between discretization partition and the class is an alternative (see *Equation 5*). If this mutual information is higher, discretization partition is closer to the class, which means more relevance. However, it is hard to measure the *redundancy* directly, we use the size of the discretization scheme instead because *minimum redundancy* implies a smaller size.

Instead of finding the optimal discretization scheme, we use a linear search method to find the suboptimal one. There are some linear search methods with low complexity. Sequential forward search(*SFS*), sequential backward selection (*SBS*) and bi-directional search (*BDS*) [16] are options. *SFS* is a linear increamental search method: it iteratively selects one cut-point c_{max_IG} with the largest information gain into the selected subset S, and stops until no information gains. Then, for each step, the selected cut-point c_{max_IG} satisfies:

$$c_{max\_IG} = \underset{c \in C \backslash S}{\arg\max}\, \sigma(S \cup c) - \sigma(S) \; (> 0) \tag{10}$$

After the discretization scheme is found, if there are no selected cut-points on one feature, this feature will be discretized into one interval, while it is non-sense. These features should be removed from the feature set due to the *redundancy*.

B. D-FS Algorithm

The discretization of our *D-FS* can be summarized into two subprocesses [5]: initializing and search. Initializing subprocess produces a complete set of cut-points, which firstly sorts all the distinct values on each feature, then the midpoints of adjacent value pairs are calculated as the initial cut-points (see [6] [7]). Search subprocess finds the final discretization scheme from the complete set according to the evaluation measure.

The pseudo-code of *D-FS* algorithm is presented in *Algorithm 1*. Initializing subprocess (*lines 1-6*) finds the complete set of cut-points on all continuous features: *line 3* sorts the distinct values, and *line 4* calculates the midpoint of adjacent pairs on these arrays, then *line 5* simply combines the cut-points from all features into a complete set C. Search subprocess (*lines 7-21*) selects a subset S from C in a global view. Each time it chooses the cut-point with highest information gain. *D-FS* method is a top-down method and initializes all the instances into one hypercube (*line 7*). When a new cut-point c is tentatively added into S, the function $resplit()$ refreshes the discretization partition $\vec{g}'$ incrementally (see *Algorithm 2*). Then, the function $muInfor()$ measures mutual information between discretization partition $\vec{g}'$ and the class vector $\vec{Y}$ (see *Formula 4*). When mutual information σ does not increase, it stops the search subprocess.

Beyond above, a common filtering process (we do not present the pseudo-code) after *Algorithm 1* is added to check the data with the selected cut-points subset: if no cut-points on a feature are selected, this feature can be removed, else we discretize the feature.

In what follows, we estimate the time complexity. Firstly, we assume some variables: N (the number of instances), M (the number of continuous features), λ (the size of selected cut-points subset) and ρ (the size of the complete cut-points set). In initializing subprocess, it is determined by the calculation in *lines 3-4* which is $O(MNlogN)$. In search subprocess, we assume the times of outer loops is λ (*lines 8-21*), the inner loop is ρ (*lines 10-15*) and the complexity of both $resplit()$ and $muInfor()$ is $O(N)$, so the complexity of search subprocess is $O(\lambda\rho N)$. In summary, the complexity of *D-FS* is $O(MNlogN + \lambda\rho N)$. Due to $M < \rho < MN$, so the complexity belongs to $(O(MNlogN), O(\lambda MN^2))$.

Algorithm 1 *D-FS* algorithm

```
Input: Given train data $T$ consisting of $N$ instances, $M$ continuous features $F$ and the vector of the class $\vec{Y}$;
Output: Discretization scheme $S$
1:  Initialize the complete set of cut-points on all features $C \leftarrow \phi$
2:  for all feature $f \in F$ do
3:      Sort all distinct values on $f$ in ascending order: $\{v_1, v_2, ..., v_n\}$
4:      Calculate midpoints of all the adjacent pairs: $c_i \leftarrow (f, (v_i + v_{i+1})/2)$
5:      $C \leftarrow C \cup \{c_1, c_2, ..., c_{n-1}\}$
6:  end for
7:  Initialize the discretization scheme $S$, it corresponding discretization partition $\vec{g}$ and mutual information $\sigma$: $S \leftarrow \phi$, $\vec{g} \leftarrow (1^1, 1^2, ..., 1^N)$ , $\sigma \leftarrow muInfo(\vec{g}, \vec{Y})$
8:  while $C$ is not empty do
9:      Find the cut-point $c_{max\_IG}$ with highest information gain $\sigma_{max\_IG}$ according to Equation 10: $c_{max\_IG} \leftarrow \phi$, $\vec{g}_{max\_IG} \leftarrow \phi$, $\sigma_{max\_IG} \leftarrow 0$
10:     for all cut-point $c \in C$ do
11:         Check this cut-point $c$: $\vec{g}' \leftarrow resplit(\vec{g}, c, T)$, $\sigma' \leftarrow muInfo(\vec{g}', \vec{Y})$
12:         if $(\sigma' - \sigma) > \sigma_{max\_IG}$ then
13:             Update the cut-point $c_{max\_IG}$: $c_{max\_IG} \leftarrow c$, $\vec{g}_{max\_IG} \leftarrow \vec{g}'$, $\sigma_{max\_IG} \leftarrow (\sigma' - \sigma)$
14:         end if
15:     end for
16:     if $\sigma_{max\_IG} > 0$ then
17:         Select the cut-point $c_{max\_IG}$ and update: $C \leftarrow C \backslash \{c_{max\_IG}\}$, $S \leftarrow S \cup \{c_{max\_IG}\}$, $\vec{g} \leftarrow \vec{g}_{max\_IG}$, $\sigma \leftarrow \sigma + \sigma_{max\_IG}$
18:     else
19:         Break
20:     end if
21: end while
```

Algorithm 2 *resplit* function

```
Input: Given the discretization partition $\vec{g}$, the cut-point $c$ and train data $T$.
Output: The new discretization partition $\vec{g}'$.
1: Initialize $\vec{g}' \leftarrow \vec{g}$.
2: Get the properties of the cut-point $c$, and then the vector value of the feature $f$ from $T$: $(f, v) \leftarrow c$, $\vec{f} \leftarrow T[f]$.
3: for $i = 1$ to $N$ do
4:     $\vec{g}'[i] \leftarrow 2\vec{g}[i]$
5:     if $\vec{f}[i] > v$ then
6:         $\vec{g}'[i] \leftarrow \vec{g}'[i] + 1$
7:     end if
8: end for
```

IV. Experiments

To evaluate *D-FS*, we have done experiments comparing with several state-of-the-art discretization methods on benchmark data sets combined with different classifiers.

A. Experimental Setting

Seventeen public benchmark data sets from *UCI machine learning repository* [17], *KEEL data set repository* [18] and *NIPS Feature Selection Challenge*. Table I is a summary of these data sets.

TABLE I: Description of Experimental Datasets

Dataset	#instances	#numeric features	#classes
appendicitis	106	7	2
balance	625	4	3
banana	5300	2	2
bupa	345	6	2
haberman	306	3	2
monk2	432	6	2
titanic	2201	3	2
bank	1372	4	2
column2C	310	6	2
CTG	2126	21	3
ionosphere	351	33	2
pageblocks	5473	10	5
spambase	4601	57	2
transfusion	748	4	2
waveform	5000	40	3
coil2000	9822	85	2
MADELON	2600	500	2

In order to evaluate the performance, we use the classification quality on the data sets before and after discretization process. Two basic classifiers are chosen: Naive Bayes (*NB*), Decision Tree (*C4.5*). Naive Bayes classifier is a simple probabilistic classifier based on the Bayesian theorem with strong independence assumptions. *C4.5* is a state-of-the-art method for inducing decision trees using the concept of information entropy. Ten-fold cross-validation is adopted to ensure the reliability of the results. The discretization scheme is learnt from the train data, then is used to discretize both the train data and test data. The classifiers and classification quality measures are implemented in the publicly available *WEKA* tool [23].

B. Comparative Methods

Three comparative previous state-of-the-art methods are chosen for comparison in this experiment: *HCF* [9], *HDD* [7] and *Chi2* [19]. We also include the benchmark method *Orig*.

- *Orig* is a benchmark method which means that the raw data is not discretized, and directly fed into the classifiers.
- HyperCluster Finder (*HCF*) clusters the data into several hypercubes by using relative neighborhood graph. Then, it cuts the edges which links two instances from different classes. For each hypercube, maximum value and minimal value on each feature form the selected cut-points set.

- Hypercube division-based algorithm (*HDD*) divides the impure hypercubes into some smaller hypercubes to improve the purity of the hypercubes. It initializes all the instances in one hypercube, then chooses a feature and selects a subset of cut-points on it to split the residual hypercubes into smaller hypercubes. The purity measure is *CAIM*, and it finally stops until all the hypercubes are reaching a purity threshold. *CAIM* maximizes the class-feature interdependence:

$$Caim = \frac{\sum_{i=1}^{K} \frac{\max_j m_{ij}^2}{m_i}}{K} \tag{11}$$

where $\max_j m_{ij}$ is the frequence of the j-th class which dominates in the i-th hypercube. K is the number of hypercubes and m_i is the number of instances in i-th hypercube. The parameter *R* of *HDD* is specified as 0.8 in our experiment as in [7].
- *Chi2* is based on *ChiMerge* [20] which discretizes numeric features by the χ^2 statistic. *ChiMerge* is a bottom-up merging process until reaching a significance level α (manually given), and *Chi2* uses a hyperparameter σ (inconsistency threshold) to help find a proper α for each feature. Like our *D-FS*, *Chi2* has the capability of feature selection. Here we set σ 0 as a default value in [19].

HCF, *HDD*, *Chi2* are commonly used in practice and often achieve good performance. These methods and our *D-FS* are *supervised*, *multivariate* and *static*. Moreover, *Chi2* and *D-FS* can filter the irrelative and redundant features. These are the reasons why we only choose these methods for comparison. We implement them by using the *KEEL* library [18].

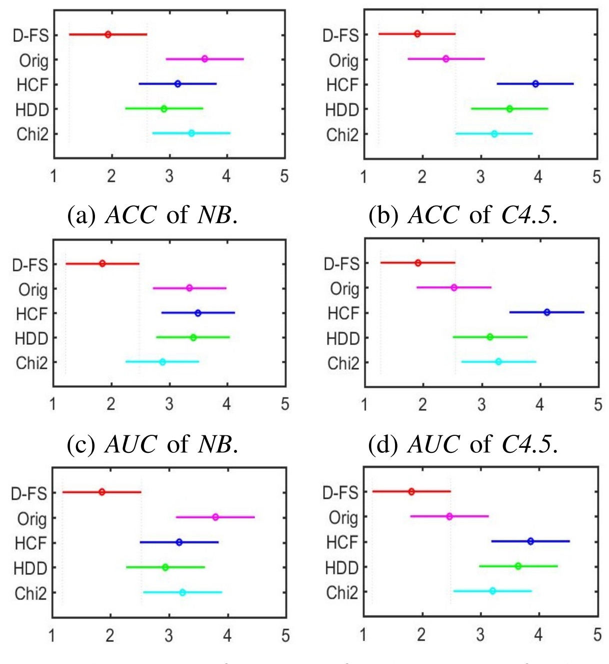
C. *Experimental Results*

To compare *D-FS* with other methods above, we try all the methods under the same environment conditions. The classification quality is measured using predictive accuracy (*ACC*), and we also give other measures: *AUC* (Area Under *ROC* Curve) and *F1-Measure*.

D-FS is an *incremental (non-parameterized)* method and the size of final selected subset is determined by the property of data. In some real applications, a trick can be used is that we can relax the stop condition: the mutual information value does not increase significantly.

Table II-IV show the average *ACC*, *AUC* and *F1-Measure* of different classifiers under different discretizers. Among at least half of the data sets, *D-FS* method can achieve highest *ACC*, *AUC* and *F1-Measure*. On average, *D-FS* method reaches highest *ACC*, *AUC* and *F1-Measure* and best ranks comparting with other methods.

To further compare these discretizers, the conjunction of Friedman test and Bonferroni test with significance level 0.1 is conducted on the ranks (see *Figure 1*). If two confidence intervals fail to overlap, there is a significant difference between them, but the reverse is not right. For *ACC*, *AUC* and *F1-Measure*, *D-FS* method is the best on Naive Bayes, and



(a) *ACC* of *NB*. (b) *ACC* of *C4.5*.
(c) *AUC* of *NB*. (d) *AUC* of *C4.5*.
(e) *F1-Measure* of *NB*. (f) *F1-Measure* of *C4.5*.

Fig. 1: The result of Friedman test for ranks of *ACC*, *AUC* and *F1-Measure* on Naive Bayes and Decision Tree.

better than *HCF*, *HDD* and *Chi2* on Decision Tree with a statistically significance.

When the cut-points subset is selected, the features with no selected cut-points are removed. *Table V* shows the average number of selected features and cut-points under *Orig*, *HCF*, *HDD*, *Chi2* and *D-FS*. For cut-point selection, *D-FS* is the smallest one (average 36.4 cut-points), which is far less than other discretizers. For feature selection, *D-FS* also selects least features (average 9.2 out of 46.5) comparing with the second least one *Chi2* (average 46.2). From *Table V*, we see that the capability of feature selection of *Chi2* is not as stable as *D-FS*. Especially on the large-scale data sets like *coil2000* and *MADELON*, *D-FS* chooses a reasonably smaller set of both cut-points and features than all other methods while its performance is similar or better.

D. *Discussions*

The complexities of above methods are: *HDD* ($O(MNlogN + MNC^3)$), *Chi2* ($O(KMNlogN)$), *D-FS* ($O(MNlogN + \lambda\rho N)$) and *HCF* ($O(N^3)$), where C (the average number of cut-points on each feature), K (the number of incremental steps in outermost loop in *Chi2*), λ (the number of selected cut-points in *D-FS*) and ρ (the size of all cut-points in *D-FS*), N (the number of the train data instances) and M (the number of features). Due to $\rho = MC$ and $1 < C < N$, when $\lambda < C^2$, *D-FS* method will be less time-cost than *HDD*.

On average, the best *ACC*, *AUC* and *F1-Measure* based on these two learning algorithms is achieved by *D-FS*. Combine *Table II-IV* and *Figure 1*, the comparison shows that *D-FS*

TABLE II: *ACC* under Discretizers(%)

Dataset	*NB*					*C4.5*				
	Orig	HCF	HDD	Chi2	D-FS	Orig	HCF	HDD	Chi2	D-FS
appendicitis	85	84.2	86.2	**87**	84.2	83.3	86	83.4	82.5	**87**
balance	**90.2**	**90.2**	82.1	89.8	**90.2**	76.8	76.8	77	**77.6**	76.8
banana	61.3	57.1	65.4	**65.7**	63.8	**89.1**	88.7	**89.1**	88.9	**89.1**
bupa	54.1	65.2	62.8	64.2	**66.9**	67	62.6	**67.3**	66.1	66.3
haberman	74.8	**76.7**	75.1	74.5	76.1	73.2	72.8	73.2	71.5	**74.8**
monk2	91.9	**97.3**	**97.3**	**97.3**	**97.3**	**100**	95.9	97.3	**100**	**100**
titanic	75.3	**77.3**	75.3	74.7	75.3	**78.8**	**78.8**	**78.8**	77.9	**78.8**
bank	83.9	77.5	**89.6**	86.3	87.6	**98.6**	78.4	97.9	98.5	98.4
column2C	77.7	**79.7**	76.1	76.8	78.7	81.9	80.3	79.7	81.3	**83.2**
CTG	82	83	86.7	82.6	**87.2**	**93.2**	92.8	92.3	92.8	92.9
ionosphere	81.8	78.9	87.5	75.2	**91.1**	**90**	89.7	88	89.7	**90**
pageblocks	89.5	90.8	91.5	90.8	**92.5**	**97**	**97**	96.5	**97.2**	97
spambase	79.6	82.3	81	84.1	**92.1**	93	92.7	92.8	92.8	**93.2**
transfusion	75.1	74.2	74.6	73.9	**76.6**	77.3	77.4	77.3	77.6	**78.2**
waveform	80	54.3	80	**80.4**	78.7	75.1	57.3	75.1	74.8	**76.5**
coil2000	79.6	86.3	82.8	79.5	**89.1**	**94.2**	94.1	93.9	93.9	94.1
MADELON	59.5	51.1	59	59	**60.6**	69	52.5	69	69.8	**82.7**
Avg±Std	77.7±10.3	76.8±12.6	79.6±10	78.9±9.8	**81.6±10.5**	84.6±10.3	80.8±13.1	84±9.9	84.3±10.4	**85.8±9.3**
Rank(Avg±Std)	3.2±1.2	2.8±1.4	2.6±1	3±1.5	**1.6±.9**	1.9±1	3.1±1.1	2.7±1.1	2.5±1.1	**1.5±.8**

TABLE III: *AUC* under Discretizers

Dataset	*NB*					*C4.5*				
	Orig	HCF	HDD	Chi2	D-FS	Orig	HCF	HDD	Chi2	D-FS
appendicitis	.78	.83	.81	**.85**	.84	.74	.71	.71	.73	**.79**
balance	**.97**	**.97**	.93	**.97**	**.97**	.81	.81	**.83**	.82	.81
banana	.66	.62	.68	.68	**.69**	**.94**	.93	**.94**	.93	**.94**
bupa	.64	.69	.67	.68	**.71**	.65	.62	.66	**.68**	.65
haberman	.65	**.68**	.66	.66	**.68**	.60	**.62**	.60	.58	.61
monk2	.96	.96	**.99**	.97	.97	**1**	.99	.99	**1**	**1**
titanic	**.71**	.70	**.71**	**.71**	**.71**	**.73**	**.73**	**.73**	.72	**.73**
bank	**.94**	.86	.92	.93	**.94**	**.99**	.81	.98	**.99**	**.99**
column2C	**.89**	.87	.84	.85	.87	.82	.79	.81	.81	**.86**
CTG	.94	.94	.94	.94	**.95**	.91	.90	.91	.91	**.94**
ionosphere	.92	.90	.91	.88	**.93**	.87	.86	.87	.86	**.88**
pageblocks	.94	**.98**	.97	.97	.97	**.96**	.94	.94	.95	.94
spambase	.94	.95	.94	.95	**.97**	.94	.93	.94	.94	**.95**
transfusion	.70	.71	.70	.71	**.73**	.71	.71	.71	.70	**.72**
waveform	**.96**	.76	**.96**	**.96**	.93	.83	.78	.83	.82	**.88**
coil2000	.71	.69	.7	.7	**.72**	**.52**	.51	.5	.5	.51
MADELON	**.64**	.51	**.64**	**.64**	**.64**	.69	.53	.69	.7	**.84**
Avg±.Std	.82±.13	.8±.14	.82±.13	.83±.13	**.84±.12**	.81±.14	.77±.14	.8±.14	.8±.14	**.83±.14**
Rank(Avg±Std)	2.4±1.3	2.5±1	2.4±1	2.1±1	**1.3±.5**	1.9±.8	2.8±1	2.2±.8	2.3±.8	**1.5±.8**

can produce discrete data which improves *ACC*, *AUC* and *F1-Measure* of classification. In summary, the integration of discretiztion and feature selection can reduce information loss in the preprocess stage of data mining.

Moreover, *D-FS* finds less cut-points and features which can reduce the computation cost of the learning algorithms. *D-FS* takes into consideration of both feature-feature interactions and the class-feature interdependence, which is achieved by selecting cut-points in global view to maximize mutual information between discretization partition and the class. The size of final selected subset is determined by train data, which is far smaller than other discretizers as shown in *Table V*. In some extent, *D-FS* discretizes the raw data to improve the performance of learning algorithms.

V. Conclusion

In this paper, we propose a new integration method of discretization and feature selection: *D-FS*. It can select a smaller set of cut-points, and remove irrelative and redundant features via discretization. *D-FS* is a supervised top-down method, and finds out the suboptimal discretization scheme in greedy way: *D-FS* iteratively chooses one cut-point with highest mutual information gain and stops until no information gains. Compare with several well-known discretizers, *D-FS* method produces the discretization scheme with less features and cut-points which improves classification quality. The greatest advantage of *D-FS* is that it integrates the discretization and feature selection which lowers down the information loss during the preprocess of data mining.

For future work, we try to adjust *D-FS* to the mixed-mode data and take into consideration of the interactions between

TABLE IV: *F1-Measure* under Discretizers

Dataset	*NB*					*C4.5*				
	Orig	HCF	HDD	Chi2	D-FS	Orig	HCF	HDD	Chi2	D-FS
appendicitis	.85	.85	.86	**.87**	.85	.82	**.85**	.83	.81	**.85**
balance	**.87**	**.87**	.79	.86	**.87**	.75	.75	.74	**.76**	.75
banana	.58	.55	.62	**.63**	.60	**.89**	**.89**	**.89**	**.89**	**.89**
bupa	.53	.64	.63	.64	**.65**	.67	.62	**.67**	.66	.66
haberman	.70	**.73**	.71	.69	**.73**	.69	.71	.69	.68	**.72**
monk2	.92	**.97**	**.97**	**.97**	**.97**	**1**	.96	.97	**1**	**1**
titanic	.75	**.76**	.75	.75	.75	**.76**	**.76**	**.76**	.75	**.76**
bank	.84	.77	**.90**	.86	.88	**.99**	.78	.98	**.99**	.98
column2C	.79	**.80**	.77	.77	.79	.81	.80	.78	.81	**.83**
CTG	.84	.84	.87	.84	**.88**	**.93**	**.93**	.92	**.93**	**.93**
ionosphere	.82	.79	.88	.76	**.91**	**.90**	**.90**	.88	**.90**	**.90**
pageblocks	.91	.92	.93	.93	**.94**	**.97**	**.97**	.96	**.97**	**.97**
spambase	.80	.82	.81	.84	**.92**	**.93**	**.93**	**.93**	**.93**	**.93**
transfusion	.71	.72	.71	.71	**.73**	.76	.76	.76	.76	**.77**
waveform	**.79**	.47	**.79**	**.79**	.78	.75	.56	.75	.75	**.76**
coil2000	.84	.88	.86	.84	**.9**	**.92**	.91	.91	.91	.91
MADELON	.59	.5	.59	.59	**.61**	.69	.41	.69	.7	**.83**
Avg±.Std	.77±.11	.76±.14	.79±.11	.79±.1	**.81±.11**	.84±.11	.79±.15	.83±.1	.83±.11	**.85±.1**
Rank(Avg±Std)	3.3±1.2	2.9±1.5	2.6±1.1	2.8±1.3	**1.6±.9**	2.1±1	3.2±1.1	3±1.2	2.6±1.1	**1.5±.7**

TABLE V: Number of Cut-points and Features under Discretizers

Dataset	#Cut-points					#Features		
	Orig	HCF	HDD	Chi2	D-FS	Orig/HCF/HDD	Chi2	D-FS
appendicitis	481	98.9	22.4	133.7	**10**	7	7	**5.3**
balance	16	20	**11.2**	15.1	16	4	4	4
banana	3847.2	**97.7**	3847.2	2143.9	209.8	2	2	2
bupa	311.6	230.8	311.6	173.9	**20.6**	6	6	6
haberman	87.1	72.6	87.1	49.7	**33.7**	3	3	3
monk2	11	13	8	**4.6**	6.1	6	**3.4**	4.1
titanic	5	9	5	**4.4**	5	3	3	3
bank	4548.6	16	3344.4	1659.7	**14.3**	4	4	4
column2C	1574.7	229	1301.7	629.9	**16.6**	6	6	**5.6**
CTG	1431.5	498.2	41	717.5	**29.1**	21	21	**15.7**
ionosphere	7313.2	738.3	6613.5	2068.5	**13.6**	33	33	**11.6**
pageblocks	8677.1	265.6	**40**	1696.9	42	10	10	**9.7**
spambase	14396.7	3164.5	13001.1	6016.7	**42.7**	57	57	**24.5**
transfusion	167.1	127.8	155.7	96.3	**53**	4	4	**3**
waveform	24565.3	239	24565.3	13506.5	**23.4**	40	40	**22.5**
coil2000	329.2	235.1	85	329.2	**64**	85	81.9	**14**
MADELON	68652.1	2126.8	68652.1	39251.3	**19.7**	500	500	**18.6**
Avg±Std	8024.4±16446.8	481.3±830.4	7181.9±16592.5	4029.3±9389.9	**36.4±46.2**	46.5±115.6	46.2±115.6	**9.2±7.1**

The *#Cut-points* column of *Orig* is the number of selected cut-points subset according to Algorithm 1. The *#Features* column of *Orig*, *HCF*, *HDD* are same and unified. The bold values indicate the best results.

categorical features and numeric features. We also hope to improve the search strategy to lower down the time cost especially on the large-scale data.

Acknowledgment

This work is partially sponsored by National Natural Science Fund of China (Grant No. 61232005) and National 863 Program of China (Grant No. 2015AA016009).

References

[1] T. Mitchell, *Machine learning*. McGraw Hill, 1997.

[2] J. Catlett, "On Changing Continuous Features into Ordered Discrete Features," *Proc. European Working Session on Learning*, pp. 164-178, 1991.

[3] Y. Yang, GI. Webb, "Proportional k-interval discretization for naive-Bayes classifiers," *Proc. of the Twelfth European Conf. on Machine Learning*, pp. 564-575, 2001.

[4] SY. Jiang, X. Li, Q. Zheng, LX. Wang, "Approximate Equal Frequency Discretization Method," *Global Congress on Intelligent Systems*, pp. 514-518, 2009.

[5] H. Liu, F. Hussain, CL. Tan, M. Dash, "Discretization: An Enabling Technique," *Data Mining and Knowledge Discovery*, vol.6, no.4, pp. 393-423, 2002.

[6] LA. Kurgan, KJ. Cios, "CAIM Discretization Algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol.16, no.2, pp. 145-153, 2004.

[7] P. Yang, JS. Li, YX. Huang, "HDD: a hypercube division-based algorithm for discretization," *International Journal of Systems Science*, vol.42, no.4, pp. 557-566, 2011.

[8] A. Gupta, KG. Mehrotra, C. Mohan, "A clustering-based discretization for supervised learning," *Statistics & Probability Letters*, vol.80, no.9C10, pp. 816-824, 2010.

[9] F. Muhlenbach, R. Rakotomalala, “Multivariate supervised discretization, a neighborhood graph approach,” *IEEE International Conference on Data Mining*, pp. 314-321, 2002.

[10] S. Garcia, J. Lueng, JA. Saez, V. Lopez, F. Herrera, “A Survey of Discretization Techniques Taxonomy and Empirical Analysis in Supervised Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol.25, no.4, pp. 734-750, 2013.

[11] S. Kotsiantis, D. Kanellopoulos, “Discretization Techniques: A recent survey,” *GESTS International Transactions on Computer Science and Engineering*, vol.32, no.1, pp. 47-58, 2006.

[12] CT. Su, JH. Hsu, “An extended Chi2 algorithm for discretization of real value features,” *IEEE Transactions on Knowledge and Data Engineering*, vol.17, no.3, pp. 437-441, 2005.

[13] CC. Chan, C. Batur, A. Srinivasan, “Determination of quantization intervals in rule based model for dynamic,” *In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pp. 1719-1723, 1991.

[14] D. Janssens, T. Brijs, K. Vanhoof, G. Wets, “Evaluating the performance of cost-based discretization versus entropy- and error-based discretization,” *Computers and Operations Research*, vol.33, no.11, pp. 3107-3123, 2006.

[15] H. Peng, F. Long, C. Ding. “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol.27, no.8, pp. 1226-38, 2005.

[16] M. Dash, H. Liu, “Feature selection for classification,” *Intelligent Data Analysis*, vol.1, no.3, pp. 131-156, 1997.

[17] A. Asuncion, DJ. Newman, UCI machine learning repository, 1997.

[18] J. Alcal-Fdez et al. “KEEL: A software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput.*, vol. 13, no. 3, pp. 307-318, 2009.

[19] H. Liu, R. Setiono, “Feature Selection via Discretization” *IEEE Transactions on Knowledge and Data Engineering*, vol.9, no.4, pp. 642-645, 1997.

[20] R. Kerber, “ChiMerge: Discretization of Numeric Features,” *National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, 1992.

[21] Y. Sang, Y. Jin, K. Li, H. Qi, “UniDis : a universal discretization technique,” *Journal of Intelligent Information Systems*, vol.40, no.2, pp. 327-348, 2013.

[22] R. Dash, RL. Paramguru, “Comparative Analysis of Supervised and Unsupervised Discretization Techniques,” *International Journal of Advances in Science and Technology*, vol.2, no.3. 2011.

[23] M. Hall et al. “The WEKA data mining software: an update,” *Acm Sigkdd Explorations Newsletter*, vol.11, no.1, pp. 10-18, 2008.

[24] F. Jiang, Z. Zhao, Y. Ge, “A supervised and multivariate discretization algorithm for rough sets,” *Rough Set & Knowledge Technology-international Conference*, LNAI 6401, pp. 596-603, 2010.

[25] A. Bondu, M. Boulle, V. Lemaire, “A Non-parametric Semi-supervised Discretization Method,” *Eighth IEEE International Conference on Data Mining*, vol.24, no.1, pp. 53-62, 2008.

[26] DM. Maslove, T. Podchiyska, HJ. Lowe, “Discretization of continuous features in clinical datasets,” *Journal of the American Medical Informatics Association Jamia*, vol.20, no.3, pp. 544, 2013.

[27] CJ. Tsai, CI. Lee, WP. Yang, “A discretization algorithm based on Class-Feature Contingency Coefficient,” *Information Sciences*, vol.178, no.3, pp. 714-731, 2008.

[28] S. Ramłrez-Gallego, S. Garcła, JM. Benłtez, F. Herrera, “Multivariate Discretization Based on Evolutionary Cut Points Selection for Classification,” *IEEE Transactions on Cybernetics*, vol.46, no.3, pp. 595-608, 2015.

[29] W. Kwedlo, M. Kretowski, “An Evolutionary Algorithm Using Multivariate Discretization for Decision Rule Induction,” *European Conference on Principles of Data Mining & Knowledge Discovery*, no.1704, pp. 392-397, 1999.

[30] UM. Fayyad, “Multi-Interval Discretization of Continuous-Valued Features for Classification Learning,” *International Joint Conference on Artificial Intelligence*, no.2, pp. 1022-1027, 1993.

[31] RP. Li, ZO. Wang, “An entropy-based discretization method for classification rules with inconsistency checking,” *International Conference on Machine Learning & Cybernetics*, vol.1, no.1, pp. 243-246, 2002.

[32] R. Jin, Y. Breitbart, C. Muoh, “Data Discretization Unification,” *IEEE International Conference on Data Mining*, vol.19, no.1, pp. 183-192, 2007.

[33] F. Jiang, Y. Sui, “A novel approach for discretization of continuous features in rough set theory,” *Knowledge-Based Systems*, vol.73, no.1, pp. 324-334, 2015.