DCR: Discretization using Class Information to Reduce Number of Intervals

Prachya Pongaksorn, Thanawin Rakthanmanon, and Kitsana Waiyamai

Data Analysis and Knowledge Discovery Laboratory (DAKDL),
Computer Engineering Dept, Faculty of Engineering, Kasetsart University, Bangkok, Thailand.
{g4865394, fengtwr, fengknw}@ku.ac.th

Abstract. Discretization techniques for data set features have received increasing research attention. Results using discretized features are usually more compact, shorter, and accurate than using continuous values. In this paper, an algorithm called Discretization using Class information to Reduce number of intervals (DCR) is proposed. DCR uses both class information and order between attributes to determine the discretization scheme with minimum number of intervals. Attribute discretization order is determined based on information gain of each attribute with respect to the class attribute. The number of intervals is reduced by deleting training data at each step of attribute discretization. Experiments are performed to compare the predictive accuracy and execution time of this algorithm with several well-known algorithms. Results show that discretized features generated by the DCR algorithm contain a smaller number of intervals than other supervised algorithms using less execution time, and the predictive accuracy is as high or higher.

Key words: discretization, continuous feature, data mining, classification

1 Introduction

Data mining is a powerful approach to extracting meaningful information from large and unwieldy databases. However, for efficiency, appropriate pre-processing of the input databases is needed. The majority of these databases usually come in a mixed format called "mixed-mode data" containing both discrete and continuous features, as shown in Table 1. In the table, feature2 is discrete while feature1, 3, and 4 are continuous. Some learning algorithms [4, 10, 12, 18] can handle only discrete-valued attributes, while some others can handle continuous attributes but still perform better with discrete-valued attributes [6, 11]. This drawback can be overcome by using a discretization algorithm as a pre-processing step for data mining.

Discrete values offer several advantages over continuous ones, such as data reduction and simplification. Quality discretization of continuous attributes is an important problem that has effects on speed, accuracy, and understandability of the classification models [20].

Although much research has been done in the area of discretization, many algorithms still do not take advantage of class information to increase their discretization performance. Thus, the resulting discretization schemes do not provide much efficiency when used in the classification process, e.g. they contain more intervals than necessary. Unsupervised discretization algorithms that do not use class information include the equal-width [1] and equal-frequency methods [17] that divide continuous ranges into sub-ranges. Supervised algorithms, such as statistics-based [11, 15], entropy-based [21], and class-attributes interdependency-based algorithms [13] use class information; however, these algorithms do not make use of relations between attributes in the database.

Table 1. Data set contanning both decrete and continuous attributes

ID	feature1	feature2	feature3	feature4	Class
1	17	Yes	49	33	Z
2	19	No	48	21	Y
3	21	No	50	50	Y
4	21	No	53	19	X
5	22	Yes	65	49	Y
6	35	Yes	70	55	Y
7	33	Yes	89	76	Z
8	42	No	48	80	Z
9	40	Yes	63	33	Y
10	22	Yes	72	21	X
11	23	Yes	80	10	X
12	20	Yes	73	9	X
13	19	No	65	43	Y
14	25	Yes	90	95	Z
15	29	Yes	73	21	Y

There are algorithms [2, 5] that use both class information and relations between attributes in their discretization process, however, they are not computationally efficient. Also, discretizations with the same accuracy but with fewer number of intervals are preferable to those with large number of intervals since they cause less fragmentation of the data in the sub-nodes of decision trees [16]. The Discretization Using Class Information to Reduce Number of Intervals (DCR) algorithm presented here uses both class information and order between attributes to determine an efficient discretization scheme. The order of attribute discretization is determined based on information gain of each attribute with respect to the class attribute. The number of intervals is successively reduced by deleting training data at each step of discretization. DCR is able to find the minimum number of discrete intervals while maintaining the accuracy of the classifier. Experimental results show that the discretized features generated using DCR nearly achieve the smallest number of intervals for a given level of accuracy. Further, DCR uses less execution time than well-known supervised discretization techniques such as CAIM and ChiMerge.

In the next section, we present the class-based discretization process. In Section 3 and 4 we present the DCR discretization concepts and algorithm, resp. we discuss the results of comparative experiments in Section 5. Finally, Section 6 gives the conclusion and further work.

2 Class-based Discretization

In this section, we present class-based discretization process. First we describe the discretization process for each attribute; then we describe the information gain used for sorting the attributes to be discretized by the process that we purpose; the order of attribute in discretization makes the result different. Lastly, we describe the quanta-matrix [9] that shows the relation between class and discretization scheme.

2.1 Univariate discretization process

Discretization can be univariate or multivariate. Univariate discretization quantifies one continuous feature at a time while multivariate discretization simultaneously considers multiple features. We mainly consider univariate (typical) discretization in this paper. A typical discretization process broadly consists of four steps:

1. Sort the values of the attribute to be discretized.
2. Determine a cut-point for splitting or adjacent intervals for merging.
3. Split or merge intervals of continuous values, according to some criterion.
4. Stop at some point.

2.2 Information Gain

Information gain is used in C4.5 [7] to choose the best attribute (maximum information gain) for splitting the data, but this method can handle only discrete values. For continuous valued attributes, there is a need for a discretization algorithm that transforms continuous attributes into discrete ones. Using the data in Table 1 as an example, calculate information gain of features 1, 3, and 4 using the equal width discretization method, where the range of values of a feature is evenly divided into equi-width intervals, to discretize these attributes before calculating the information gain. If we discretize the features in Table 1 into five intervals, then the information gains of feature1, 3, and 4 are 0.78, 1.023, and 1.53, resp.

2.3 Class-attribute interdependency discretization

The main objective of this paper is to find the discretization scheme for each continuous attribute that contains the minimum number of intervals while minimizing the loss of class-attribute interdependency. The quanta-matrix plays a major role in achieving this purpose. Table 2 shows a quanta-matrix of feature f of a given discretization schema D. In this quanta-matrix, training dataset consists of M

examples, where each example belonging to only one of the S classes. If we discretize attribute f_i -- for which d_0 is the minimum value and d_n is the maximum value -- into n intervals, then the discretization scheme of attribute f_i is

$$D_i = \{[d_0,d_1], (d_1,d_2], \ldots, (d_{n-1},d_n]\}$$

Using discretization scheme D_i, each value of attribute f_i can be classified into one of the n intervals.

Table 2. Quanta-matrix of feature f of a given discretization scheme D

Class	Interval					Class Total
	$[d_0,d_1]$	..	$(d_{r-1},d_r]$	..	$(d_{n-1},d_{dn}]$	
C_1	q_{11}	..	q_{1r}	..	q_{1n}	q_{1+}
:	:	..	:	..	:	:
C_i	q_{i1}	..	q_{ir}	..	q_{in}	q_{i+}
:	:	..	:	..	:	:
C_S	q_{S1}	..	q_{Sr}	..	q_{Sn}	q_{S+}
Interval Total	q_{+1}	..	q_{+r}	..	q_{+n}	M

In Table 2, q_{ir} is the total number of continuous values belonging to the i^{th} class that are in interval $(d_{r-1},d_r]$; q_{i+} is the total number of objects belonging to the i^{th} class, and q_{+r} is the total number of continuous values of attribute f_i that are within the interval $(d_{r-1},d_r]$, for $i = 1, 2, \ldots, S$ and $r = 1, 2, \ldots, n$. Table 3 shows a quanta-matrix of feature4 (taken from Table 1) with discretization schema $D4 = \{[9,20], (20,65.5], (65.5,95]\}$.

Table 3. Quanta-matrix of feature4 from Table 1 with the discretization schema $D4 = \{[9,20], (20,65.5], (65.5,95]\}$

Class	Interval			Class Total
	$[9,20]$	$(20,65.5]$	$(65.5,95]$	
X	3	1	0	4
Y	0	7	0	7
Z	0	1	3	4
Interval Total	3	9	3	15

3 Discretization using Class Information to Reduce Number of Intervals

In this section, we describe the DCR supervised discretization algorithm whose objective is to maximize predictive accuracy while generating a (possibly) minimal number of discrete intervals. To maximize the predictive accuracy, DCR uses class intervals, reduces training transactions at each step of attribute discretization.

3.1 Using class information to find the best cut points

To evaluate the relation between the discretization scheme and class for each attribute, a criterion called *DCR* is defined as (using the notation as in Table 2):

$$DCR = \frac{\sum_{r=1}^{n}\left(\sum_{i=1}^{S} q_{ir}^{2} \Big/ q_{+r}\right)}{n} \tag{1}$$

The DCR value is the average of the distribution of class ($\sum_{r=1}^{n}$) in each interval ($\sum_{i=1}^{S} q_{ir}^2 \Big/ q_{+r}$). It has the following properties:

- The algorithm is able to find the discretization scheme where each interval has one major class, consider each interval *r* in the quanta-matrix, q_{ir} value is in range [*0*, q_{+r}]. Thus, an interval has the maximal one major class when q_{ir} equals q_{+r}. Since *DCR* depends on $\sum_{i=1}^{S} q_{ir}^2$, *it* achieves its maximum when each interval has all of its values grouped within a single class label.

Table 4. Distribution of class for each interval

Class		Interval		Class Total
	..	$(d_{r-1},d_r]$	..	
C_1	..	*0*	..	
C_2	..	*5*	..	
C_3	..	*10*	..	
Total		*15*		

(a)

Class		Interval		Class Total
	..	$(d_{r-1},d_r]$	..	
C_1	..	*2*	..	
C_2	..	*3*	..	
C_3	..	*10*	..	
Total		*15*		

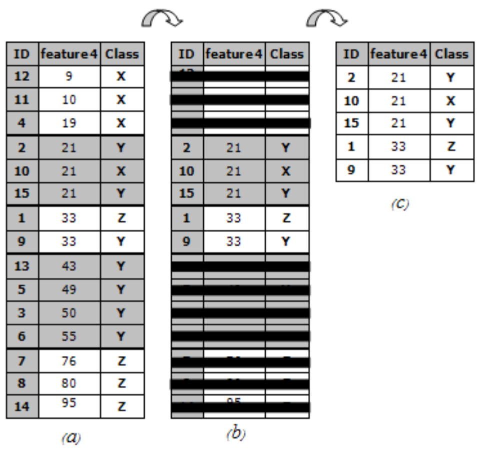
(b)

- The q_{ir}^2 values are used to compute the distribution of classes in each interval. To find the discretization scheme when each interval has one major class, the discretization scheme in Table 4*(a)* might be better than the scheme in Table 4*(b).* DCR is able to distinguish which of the two scenarios is better. Because the interval in 4*(a)* has a smaller distribution of classes and it may possibly use only one attribute to classify non-majority class. But the interval in Table 4*(b)* must use at least two attributes for classification.

- The q_{ir}^2 values are divided by q_{+r} for normalization. Numerical overflow errors can be avoided by calculating $q_{ir}^2 \big/ q_{+r}$ as $(q_{ir} / q_{+r})\, q_{ir}$, so the maximum value is q_{ir}.

3.2 Deleting training transactions to generate the minimum number of discrete intervals

We use the relation between attributes to reduce the size of training data by removing transactions for which the (continuous) values are in the interval having all of its values grouped within a single class label. Thus, the next continuous attribute has only unclassified transactions to be discretized. This process can also reduce the execution time for classifying other attributes. Further, the size of the training data is reduced at each attribute discretization. Fig. 1(a) shows the discretization scheme {[*9,20*], (*20,27*], (*27,38*], (*38,65.5*], (*65.5,95*]} of feature4.



(a)

ID	feature4	Class
12	9	X
11	10	X
4	19	X
2	21	Y
10	21	X
15	21	Y
1	33	Z
9	33	Y
13	43	Y
5	49	Y
3	50	Y
6	55	Y
7	76	Z
8	80	Z
14	95	Z

(b)

ID	feature4	Class
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
2	21	Y
10	21	X
15	21	Y
1	33	Z
9	33	Y
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]

(c)

ID	feature4	Class
2	21	Y
10	21	X
15	21	Y
1	33	Z
9	33	Y

Fig. 1. At each attribute dicretization, trainning transactions are deleted to reduce the number of intervals

In Fig. 1, the class of continuous values that are in intervals [9,20], (38,65.5], and (65.5,95] are grouped within a single class label X, Y, and Z, resp, so these transactions are removed as they are classifiable transaction. Fig. 1(c) shows the remaining transactions that are used in next attribute to discretization.

In this method attributes are discretized one by one, and the order of attribute discretization may effect the final classifiers. Hence, the DCR algorithm uses the relation between attribute and class to compute the information gain value for each attribute with class and then discretize each attribute by its information gain value in descending order.

4 DCR Algorithm

The optimal discretization scheme can be found by searching over the space of all discretization schemes for the one with the highest *DCR* value; however, such a search is highly combinatorial and time consuming. Instead, the DCR algorithm uses a greedy approach, which searches for the approximate optimal value of the DCR criterion by finding locally maximum values of the criterion. Although this approach does not guarantee a global maximum, it is both computationally inexpensive and results in a near optimal discretization scheme, as shown in Section 5. The algorithm is composed of two principle steps:

1. Order the attribute to be discretized.
2. Discretize and reduce the size of current training data for each attribute

Pseudo code of the DCR algorithm is given in Fig. 2.

Input: Training data set *DB* consisting of continuous attributes F_i, and class attribute *C* from a total of *s* classes

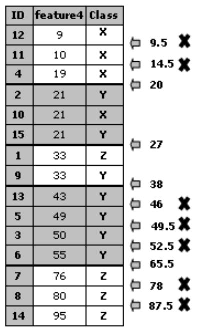
```
1.  for each $F_i$
2.    $E_i$ = information_gain($F_i$, C);
3.  arrange_order_desc(F, E);
4.  db = DB;
5.  for each $f_i$        // feature $F_i$ of current training data set db
6.    $d_{i0}$ = min($F_i$);
7.    $d_{in}$ = max($F_i$);
8.    $D_i$ = {[$d_{i0}$, $d_{in}$]}
9.    if (db $\neq \emptyset$) then
10.     k = 1;
11.     MaxDCR = 0;
12.     EB = essential_boundary_set($f_i$);
13.     Repeat
14.       DCR = compute_max_dcr_boundary($f_i$, $D_i$, EB);
15.       If (DCR > MaxDCR) or (k < s) then
16.         Update $D_i$ with a boundary that has the highest DCR
17.         MaxDCR = DCR;
18.         k = k + 1;
19.     Until (DCR <= MaxDCR) and (k >= s)
20.   db = reducing_transaction(db, $D_i$);
21.   D = D U {$D_i$}
```

Output: Set of all discretization scheme D

Fig. 2. Pseudocode of the DCR Algorithm

In steps 1-3 the algorithm orders attributes to be discretized based on the information gain value in descending order. Based on the information gain of attributes in section 2.2, the algorithm will discretize feature4, 3, and 1, resp., in step 5.

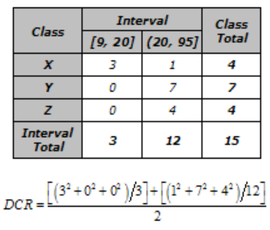
In the discretization process (steps 6-19), DCR starts with a single interval that covers all possible values of continuous attribute F_i and divides it interactively. In step 12, form a set of all distinct values of f_i in ascending order, and initialize all possible interval boundaries B with all the midpoints of all the adjacent pairs in the set, denoted by B = {$a_0, \ldots, a_m$}. If the instances that fall into the intervals (a_{i-1}, a_i] and (a_i, a_{i+1}] belong to the same class, remove a_i from set B until there are instances that fall into two adjacent intervals but do not belong to the same class. This results in an essential boundary set EB = {$b_0, \ldots, b_n$}, where $n < m$. For example, in Fig. 3 point 9.5 is the midpoint between transactions 11 and 12 both belonging to class X, so the point 9.5 is not included in set EB. The value 27 is the midpoint between feature value 21 (transactions 2, 10, 15) and feature value 33 (transactions 1 and 9) belonging to different class labels, hence 27 is added to set EB. Finally, the EB set for feature4 is {*20, 27, 38, 65.5*}.



ID	feature4	Class
12	9	X
11	10	X
4	19	X
2	21	Y
10	21	X
15	21	Y
1	33	Z
9	33	Y
13	43	Y
5	49	Y
3	50	Y
6	55	Y
7	76	Z
8	80	Z
14	95	Z

Fig. 3. Finding essential interval boundaries (EB) of feature4 in Table 1

From all possible division points that are tried (with replacement) in step 14, the algorithm chooses the division boundary that gives the highest value of the DCR criterion. For example, in finding the division points of feature4 the initial discretization scheme D_4 is {[*9, 95*]} and the set of essential interval boundaries EB is {*20, 27, 38, 65.5*}*;* as shown in Fig. 3, the algorithm adds an inner boundary value that is not already in D_4, from EB, and calculates the corresponding *DCR* value. The algorithm accepts the boundary value with the highest value of *DCR*, e.g., for the first element of EB, point 20, the new discretization scheme D_4 is {[*9,20*], (*20,95*]} and the data in the quanta-matrix are as in Fig. 4. Thus, the *DCR* value of this discretization scheme is 4.25.



Class	Interval		Class Total
	[9, 20]	(20, 95]	
X	3	1	4
Y	0	7	7
Z	0	4	4
Interval Total	3	12	15

$$DCR = \frac{\left[\left(3^2+0^2+0^2\right)/3\right]+\left[\left(1^2+7^2+4^2\right)/12\right]}{2}$$

Fig. 4. The calculation of DCR value for feature4 in Table 1 where D_4 is {[*9,20*], (*20,95*]}

For boundary points 27, 38, and 65.5, the corresponding *DCR* values are 3.944, 3.41, and 4.25 resp. After all tentative additions have been tried, the point with the highest *DCR* value (20 in this example) is added to D_i in step 16. The algorithm assumes that every discretized attribute needs a number of intervals at least equal to the number of classes or that the *DCR* value shows improvement at each iteration, assuring that the discretized attribute can improve subsequent classification. Thus, the discretization scheme of feature4 is {[*9,20*], (*20,27*], (*27,38*], (*38,65.5*], (*65.5,95*]} as in Fig. 1(a). Step 20 creates a new training data set *db* by remove classifiable intervals as in Fig. 1(c).

5 Experimental Results

5.1 Experimental Set-up

The DCR algorithm is compared with five state of the art discretization algorithms including two unsupervised algorithms and three supervised algorithms. The unsupervised algorithms are equal width (EW) [1] and equal frequency (EF) [17]; supervised algorithms are the CAIM [13] splitting-based discretization, ChiMerge [11] merging-based discretization, and a discretization algorithm in the WEKA open-source data mining library.

Data for the experiments consist of six well-known continuous and mixed-mode data sets from the UCI repository of Machine Leaning Database [3]: Iris dataset (iris), Ionosphere dataset (ion), New-Thyroid dataset (thy), SatImage dataset (sat), Waveform dataset (wav), and Heart Disease dataset (hea). Properties of the data sets are listed in Table 5.

The unsupervised algorithms require the user to specify the number of discrete intervals. In the experiments, we set the number of intervals to be close to the number obtained with the DCR algorithm for purpose of comparison.

Table 5. Properties of data sets used in experiments.

Properties	Datasets					
	iris	ion	thy	sat	wav	hea
Number of classes	3	2	3	6	3	2
Number of examples	150	351	215	6435	3600	270
Number of attributes	4	34	5	36	21	13
Number of continuous attributes	4	32	5	36	21	6

5.2 Analysis of Results

In the experiments, we evaluated the results in terms of number of intervals, execution time, and accuracy of rules generated by the C5.0 algorithm.

5.2.1 Number of intervals

Table 6. Number of intervals for each discretization method.

Discretization Method	Datasets						Rank mean
	iris	ion	thy	sat	wav	hea	
EW	12	**64**	15	180	63	**12**	2.0
EF	12	**64**	15	180	63	**12**	2.0
CAIM	12	**64**	15	216	63	**12**	2.3
ChiMerge	15	398	28	752	801	33	6.0
WEKA	10	117	**14**	475	81	13	3.8
DCR	**9**	**64**	**14**	**154**	**62**	**12**	**1.0**

Table 6 shows that the DCR algorithm generated discretization scheme with the smallest number of intervals for all data sets. A smaller number of discrete intervals reduces the size of the data and helps to better understand the meaning of discretized attributes. This is a major advantage of the DCR algorithm.

5.2.2 Discretization execution time

A comparison of the discretization times is given in Table 7. We implemented all discretization algorithms in the same programming language, except the WEKA algorithm and Built-in C5.0. Thus, they were not included in the comparison.

Table 7. Discretization execution time.

Discretization Method	Datasets						Rank mean
	iris	ion	thy	sat	wav	hea	
EW	0.110	**3.786**	0.231	1233.254	381.999	**0.300**	1.8
EF	**0.090**	3.806	**0.220**	1198.744	**337.575**	0.320	**1.5**
CAIM	2.004	77.862	4.740	2140.000	1260.000	13.489	4.3
ChiMerge	8.362	2399.089	45.375	**913.433**	517.164	39.817	4.0
DCR	0.631	14.962	1.752	1477.004	864.213	3.305	3.3

The comparison of execution times shows that the unsupervised discretization algorithms exhibit the shortest execution times; this is to be expected since they do not process any class-related information. Among the supervised algorithms, DCR exhibited the smallest execution time for four out of six data sets, but the second highest execution time (after ChiMerge) for *sat* and *wav*. Still, based on average rank, DCR ranked fastest among the supervised algorithms.

5.2.3 Accuracy comparison

The discretized data sets generated in Section 5.2.1, were used as input to C5.0 algorithms to generate classification rules. The accuracy of the resulting classification rules were compared. Since C5.0 can generate data models from continuous attributes, we compared its performance using generated rules from raw data against the results achieved using discretized data produced by the six algorithms. A 10-fold cross-validation test was performed using all data sets: each data set was divided into 10 parts of which nine parts were used as training data and the remaining one part as test data. The experiments were performed for all 10 choices of the test data. The final predictive accuracy was taken as the average of the 10 predictive accuracy values.

Table 8. Comparison of the accuracies achieved by the C5.0 algorithm for six data sets using the seven discretization schemes.

Discretization Method	Datasets						Rank mean
	iris	ion	thy	sat	wav	hea	
EW	97.3330	90.0285	86.0465	85.9518	74.6667	75.5556	5.0
EF	94.6667	81.7664	89.7674	85.2681	76.5000	80.0000	4.8
CAIM	94.0000	91.4530	94.8837	85.8430	77.0000	77.4070	4.2
ChiMerge	**97.3333**	92.0228	93.0233	83.3877	71.6111	76.2963	4.7
Built-in C5.0	95.3020	90.8571	91.5888	85.9341	75.8544	78.8104	4.2
WEKA	93.2886	94.0000	94.3925	**87.5971**	77.6605	**81.4126**	2.7
DCR	94.6667	**94.3020**	**96.2791**	85.9518	**78.7778**	78.5185	**2.2**

The DCR algorithm exhibited the highest accuracy for three of the six data sets; WEKA was most accurate for two datasets and nearly as accurate as DCR for three other data sets.

6 Conclusions and Future work

Experimental results comparing several discretization algorithms using standard data sets indicate that the DCR algorithm performs discretization with fewer intervals and overall lower run time while still providing classifiers with high predictive accuracy. On average it had the fastest run-time of all supervised algorithms. The resulting discretized data and classifiers were not only more compact, but resulted in high predictive accuracy for all six experimental data sets. The WEKA algorithm also showed high predictive accuracy, but required more discretization intervals.

In the future work, we will focus on increasing the efficiency of discretization in the context of mixed-mode data. Another interesting research direction is to investigate other measures of interestingness [14, 19] as a way of optimizing the attribute discretization order.

Acknowledgment. Thanks to J. E. Brucker for reading and comments of this paper.

References

1. A.K.C. Wong, D.K.Y. Chiu: Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data. IEEE Trans. Pattern Analysis and Machine Intelligence, pp. 796--805 (1987)
2. Chan, C.-C., Batur, C., Srinivasan, A.: 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 (1991)
3. C.L. Blake, C.J. Merz: UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html
4. Cost, S., Salzberg, S.: A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning 10(1), pp. 57--78 (1993)
5. Ho, K.M., Scott, P.D.: 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 (1997)
6. J. Catlett: On Changing Continuous Attributes into Ordered Discrete Attributes. Proc. European Working Session on Learning, pp. 164--178 (1991)
7. J. R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
8. J. R. Quinlan: Induction of Decision Trees. Machine learning, vol.1**,** pp. 81--106 (1986)
9. J.Y. Ching, A.K.C. Wong, K.C.C. Chan: Class-Dependent Discretization for Inductive Learning from Continuous and Mixed Mode Data. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, no. 7, pp. 641--651 (1995)
10. K.A. Kaufman, R.S. Michalski: Learning from Inconsistent and Noisy Data: The AQ18 Approach. Proc. 11th Int'l Symp. Methodologies for Intelligent Systems (1999)
11. Kerber, R.: Chimerge: Discretization of Numeric Attributes. In Proc. AAAI-92, Ninth National Confrerence Articial Intelligence. AAAI Press/The MIT Press, pp. 123-128 (1992)
12. K.J. Cios, L. Kurgan: Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms. New Learning Paradigms in Soft Computing, L.C. Jain and J. Kacprzyk, ed., pp. 276--322 (2001)
13. L. A. Kurgan, K. J. Cios: CAIM Discretization Algorithm. IEEE Trans. Knowledge And Data Engineering, February vol. 16, no.2, pp. 145--153 (2004)
14. Lenca P., Lallich S., Do T.-N., Pham N.-K.: A Comparison of Different Off-Centered Entropies to Deal with Class Imbalance for Decision Trees. Pacific-Asia Conference on Knowledge Discovery and Data Mining (eds. Washio T., Suzuki E., Ting K. M., Inokuchi A.), Lecture Notes in Computer Science, 5012, Springer, Osaka, Japan, pp. 634--643 (2008)
15. Liu, H., Setiono, R.: Chi2: Feature Selection and Discretization of Numeric Attributes. IEEE Computer Society, November 5–8, pp. 388--391 (1995)
16. M. Boulle: MODL: A Bayes Optimal Discretization Method for Continuous Attributes. Machine Learning, vol. 65, No. 1. pp. 131--165, (2006)
17. Nguyen, H.S., Nguyen, S.H.: Discretization Methods in Data Mining. In: Polkowski, L., Skowron, A. (Eds.), Rough Sets in Knowledge Discovery. Physica, pp. 451--482 (1998)
18. P. Clark, T. Niblett: The CN2 Algorithm. Machine Learning, vol. 3, pp. 261--283 (1989)
19. Pham N.-K., Do T.-N., Lenca P., Lallich S.: Using Local Node Information in Decision Trees: Coupling a Local Decision Rule with an Off-Centered Entropy. The International Conference on Data Mining (eds. Stahlbock R., Crone S. F., Lessmann, S.), CSREA Press, Las Vegas, Nevada, USA, pp. 117--123, July 14-17 (2008)
20. T. Qureshi, D. A. Zighed: Discretization of Continuous Features by Resampling. Proc. 8emes Journees fancophones Extraction et Gestion des Connaissances (2008)
21. U.M. Fayyad, K.B. Irani: On the Handling of Continuous-Valued Attributes in Decision Tree Generation. Machine Learning, vol. 8, pp. 87--102 (1982)