Fast Class-Attribute Interdependence Maximization (CAIM) Discretization Algorithm

Lukasz Kurgan [1], and Krzysztof Cios [2,3,4,5]
[1] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada
[2] Department of Computer Science and Engineering, University of Colorado at Denver, U.S.A.
[3] Department of Computer Science, University of Colorado at Boulder, U.S.A.
[4] University of Colorado Health Sciences Center, U.S.A.
[5] 4cData, LLC, Golden, Colorado, U.S.A.

Abstract – *Discretization is a process of converting a continuous attribute into an attribute that contains small number of distinct values. One of the major reasons for discretizing an attribute is that some of the machine learning algorithms perform poorly with continuous attribute and thus require front-end discretization of the input data. The paper describes a Fast Class-Attribute Interdependence Maximization (F-CAIM) algorithm that is an extension of the original CAIM algorithm. The algorithm works with supervised data by maximization of the class-attribute interdependence. The F-CAIM's improvement of the CAIM algorithm is significant shortening of the computational time required to discretize the data. It has all CAIM's advantages like fully automated generation of possibly minimal number of discrete intervals, achieving the highest class-attribute interdependency when compared with other discretization algorithms, and improving performance of machine learning algorithms that are subsequently used on the discretized data. We present the results based on extensive benchmarking tests of F-CAIM, CAIM and six other state-of-the-art discretization algorithms. The tests use eight well-known machine learning datasets consisting of continuous and mixed-mode attributes. They show that the F-CAIM's speed is comparable to the speed of the simplest unsupervised algorithms and better than these of other supervised discretization algorithms.*

Keywords: discretization, class-attribute inter-dependency maximization, CAIM, F-CAIM, machine learning, class boundary points, scalability.

1. Introduction

In the information-based society one of the challenges is to automate analysis of large data sources. Machine learning (ML) is one of the most successful techniques that helps in solving the problem. One of the main goals of ML algorithms is generation of knowledge from class-labeled (supervised) data examples that are described by a set of numerical, nominal or continuous attributes. Some of the ML algorithms, like AQ algorithm [20, 15], CLIP algorithms [5, 6, 7], DataSqueezer algorithm [18], and CN2 algorithm [8, 9], can handle only numerical or nominal data. Some other ML algorithms can handle continuous attributes but still perform better with discrete-valued attributes [2, 16]. The difficulty of dealing with continuous attributes can be solved by performing discretization prior to the learning process [2, 11, 13, 22].

Discretization is a process of dividing a continuous attribute into a finite set of intervals to generate an attribute with small number of distinct values, by associating discrete numerical value with each of the generated intervals. More information about the discretization process and algorithms can be found in [4, 16, 13, 14, 3, 5, 6, 17, 19].

A supervised discretization algorithm should automatically seek for a minimal number of discrete intervals since their large number slows the machine learning process [2]. It also should generate discrete intervals that are characterized by high interdependency with the class label. The proposed F-CAIM algorithm is based on our previous CAIM discretization algorithm [17, 19] and it inherits all its properties. Both CAIM and F-CAIM algorithms have these features:

- discretize attributes into possibly the smallest number of intervals

- maximize the class-attribute interdependency to improve results of the subsequently used machine learning
- do not require user interaction since they automatically pick proper number of discrete intervals.

The main design goal of the F-CAIM algorithm was to speed-up the original CAIM algorithm, while keeping all of its advantages, like the lowest number of discrete intervals, the highest interdependency between class labels and the discrete intervals, and improvement of classification accuracy and complexity of the models generated from the discretized data.

To show the above properties, a set of benchmarking tests were performed using F-CAIM and it was compared with seven well-known discretization algorithms:

- unsupervised algorithms: Equal Width and Equal Frequency [4]
- supervised algorithms: Patterson-Niblett [21], Maximum Entropy [25], Information Entropy Maximization (IEM) [14], CADD [3], and CAIM [17, 19].

The results show that the F-CAIM algorithm, in a manner similar to CAIM, generates the smallest number of discrete intervals, and retains the highest class-attribute interdependency. The F-CAIM algorithm is also shown to be the fastest among all five supervised discretization algorithms.

The data discretized using the F-CAIM algorithm and the other seven algorithms were used with two ML algorithms: CLIP4 [6, 7], and C5.0 [10] to generate the rules. The accuracy of the generated rules shows that the F-CAIM algorithm significantly improves the classification performance, and performs best among the seven discretization algorithms.

1.1. Some definitions of the class-attribute interdependent discretization

Let us assume that we have a mixed-mode data set consisting of M examples, and that each example belongs to only one of the S classes. F denotes continuous attributes. Then, there exists a discretization scheme D on F, which discretizes the continuous domain of attribute F into n discrete intervals bounded by the pairs of numbers (boundary points):

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

where d_0 is the minimal value and d_n is the maximal value of attribute F, and the values are arranged in the ascending order. These values constitute the boundary set $\{d_0, d_1, d_2, \ldots, d_{n-1}, d_n\}$ for discretization D.

In D each value belonging to attribute F can be classified into only one of the n intervals. The membership of each value in a certain interval for attribute F may change when the discretization intervals change. The class variable and the discretization variable of attribute F can be treated as two random variables defining a 2-D frequency matrix (called quanta matrix) that is shown in Table 1.

Table 1. 2-D quanta matrix for attribute F and discretization scheme D

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

In Table 1, q_{ir} is the total number of continuous values belonging to the ith class that are within interval $(d_{r-1}, d_r]$. M_{i+} is the total number of objects belonging to the ith class, and M_{+r} is the total number of continuous values of attribute F that are within the interval $(d_{r-1}, d_r]$, for $i=1,2\ldots,S$ and, $r= 1,2, \ldots, n$.

The F-CAIM algorithm discretizes the data using the class-attribute dependency information and the CAIM discretization criterion. The criterion measures the dependency between the class variable C and the discretization variable D for attribute F, for a given quanta matrix, and is defined as:

$$CAIM(C,D\,|\,F)=\frac{\sum_{r=1}^{n}\frac{\max_r^{\,2}}{M_{+r}}}{n}$$

where: n is the number of discrete intervals, r iterates through all intervals, i.e. r=1,2,...,n, max_r is the maximum value among all q_{ir} values (maximum value within the rth column of the quanta matrix), i=1,2,...,S, M_{+r} is the total number of continuous values of attribute F that are within the interval (d_{r-1}, d_r]. For more background information the reader is referred to [17, 19].

2. The F-CAIM Algorithm

The main goal of the F-CAIM algorithm is to do the necessary computations very fast so that it can be applied to continuous attributes that have large number of unique values. The other goals are to minimize the number of discrete intervals and to maximize the dependency relationship between the class labels and the discrete intervals.

The design of the F-CAIM algorithm is based on the CAIM algorithm. A weaker feature of the CAIM algorithm was selection of candidate boundary points. In the CAIM algorithm they were initialized with the min, max and all the midpoints of all the adjacent data points, so the number of boundary points was equal to M+1. The F-CAIM algorithm performs different initialization of the initial boundary points. It initializes them with the max, min, and midpoints of the adjacent data points, but only for the data points of different classes. This results in generation of maximum of M+1 boundary points, when in many real-life problems the number can be significantly smaller. The above idea is based on the work of Fayyad and Irani [14]. They proved that for the discretization that use the entropy-based criterion the generated boundary points are always between two data points that belong to two different classes. Such selection of boundary points significantly speeds up the discretization process since fewer number of candidate boundary points needs to be examined. This idea is used in the IEM algorithm [14], and the ID3 algorithm [23]. It is also used to speed up an algorithm that selects optimal partitions from supervised data [12].

In case of the CAIM algorithm, which used the class-attribute dependency information discretization criterion, we could not prove that boundary points would always be selected between two data points that belong to two different classes. Although we still cannot prove this property we decided to treat the above mechanism for selection of candidate boundary points as a heuristic that can be incorporated into the algorithm. The main reason was that it will speed up processing time of the algorithm. Also, we assume that applying the heuristic will not worsen the quality of discretization performed by the CAIM algorithm; this comes from our observations that almost all of the boundary points selected by the algorithm satisfy the above selection mechanism. All of the above lead to the development of the F-CAIM algorithm.

The main difference between the CAIM and F-CAIM algorithms is in step 1.2, where the initial boundary points are selected. The pseudocode of the F-CAIM algorithm follows:

```
Given:  Data set of M examples, S classes, and continuous attributes $F_i$
For every $F_i$ do:
Step1.
1.1  find maximum ($d_n$) and minimum ($d_o$) values of $F_i$
1.2  form a set of all distinct values of $F_i$ in ascending
     order and initialize all possible interval boundaries, B,
     with minimum, maximum and the midpoints of all the
     adjacent pairs in the set that belong to different classes
1.3  set the initial discretization scheme as
          D: {[$d_o$, $d_n$]}, set GlobalCAIM=0
Step2.
2.1  initialize k=1;
2.2  tentatively add an inner boundary, which is not
     already in D, from B, and calculate corresponding the
     CAIM criterion value
2.3  after all the tentative additions have been tried accept
     the one with the highest value of the CAIM criterion
2.4  if (CAIM > GlobalCAIM or k<S) then update D with
     the accepted in step 2.3 boundary and set
     GlobalCAIM=CAIM, else terminate
2.5  set k=k+1 and go to 2.2
Output:  Discretization scheme D
```

The expected running time of the F-CAIM algorithm is O(Mlog(M)). The time is calculated in the same way as for the CAIM algorithm [19]. Although the complexity did not change between CAIM and F-CAIM algorithms, experimental results show that significant improvement in the running time has been achieved, while keeping all other advantages of the CAIM algorithm.

3. Experiments

The eight datasets used to test the F-CAIM algorithm are: Iris Plants (*iris*), Johns Hopkins University Ionosphere (*ion*), Statlog Project Heart Disease (*hea*), Pima Indians Diabetes (*pid*), Statlog Project Satellite Image (*sat*), Thyroid Disease (*thy*), Waveform (*wav*), Attitudes Towards Workplace Smoking Restrictions (*smo*). The first seven datasets are from the UC Irvine ML repository [1], and the last one from the StatLog repository [24]. Detailed description of the datasets is shown in the Table 2. The experimental setup was identical to the setup described in [19].

3.1. Analysis of the results

The F-CAIM and the other seven discretization algorithms were used to discretize the eight datasets. The quality of the discretization was evaluated based on the CAIR criterion value, number of generated intervals, and the execution time. The CAIR criterion is defined as [26, 17, 19]:

$$R(C,D\,|\,F)=\frac{I(C,D\,|\,F)}{H(C,D\,|\,F)},$$

where $I(C,D\,|\,F)=\sum_{i=1}^{S}\sum_{r=1}^{n}p_{ir}\log_2\frac{p_{ir}}{p_{i+}p_{+r}}$ and

$H(C,D\,|\,F)=\sum_{i=1}^{S}\sum_{r=1}^{n}p_{ir}\log_2\frac{1}{p_{ir}}$; see Table 1.

The performance of the F-CAIM algorithm was compared with the six discretization algorithms. Also, direct comparison with the performance of the CAIM [19] algorithm was performed.

Table 3 shows the results of discretizing the datasets using the F-CAIM and CAIM algorithms. It shows mean and standard deviation values for the CAIR criterion, total number of intervals, and the execution time. It also shows if the discretization generated by the F-CAIM and CAIM are different, and how many attributes were discretized differently between the two. The results of other algorithms can be found in [19].

The comparison shows that the F-CAIM algorithm achieves a little worse results in terms of class-attribute interdependency, as measured by CAIR, the same results in terms of the number of discrete intervals, and significantly better results in terms of the execution time. For all eight datasets, the F-AIM algorithm was faster than the CAIM algorithm. The overall quality of discretization by the F-CAIM algorithm is similar to that of the CAIM algorithm but significant improvement in the execution time was achieved. We also note that two datasets were discretized identically, while for the remaining datasets the discretizations were very similar, except for the *ion* and *iris* datasets.

Table 4 compares results of the F-CAIM algorithm with the six other algorithms (all except the CAIM algorithm). It also shows evaluation for the CAIM algorithm and thus enables direct comparison of performance between the two. The table shows mean rank value for each of the algorithms, which is computed by ranking results for each of the datasets, and averaging the resulting scores.

Table 2. Major properties of datasets considered in the experimentation

Properties	**Datasets**							
	iris	**sat**	**thy**	**wav**	**ion**	**smo**	**hea**	**pid**
# of classes	3	6	3	3	2	3	2	2
# of examples	150	6435	7200	3600	351	2855	270	768
# of training / testing examples	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation
# of attributes	4	36	21	21	34	13	13	8
# of continuous attributes	4	36	6	21	32	2	6	8

Table 3. Comparison of results achieved by F-CAIM and CAIM algorithms (bold indicates better result)

Criterion	**Discretization Method**	**Dataset**															
		iris	std	**sat**	std	**thy**	std	**wav**	std	**ion**	std	**smo**	std	**hea**	std	**pid**	std
CAIR mean value	CAIM	**0.54**	0.01	0.26	0	**0.170**	0.01	0.130	0	**0.168**	0	0.010	0	0.138	0.01	0.084	0
	F-CAIM	0.52	0.01	0.26	0	0.168	0.01	0.130	0	0.164	0	**0.011**	0	0.138	0.01	0.084	0
total # of intervals	CAIM	12	0	216	0	18	0	63	0	64	0	6	0	12	0	16	0
	F-CAIM	12	0	216	0	18	0	63	0	64	0	6	0	12	0	16	0
time [s]	CAIM	0.05	0.01	53.36	1.90	11.50	0.47	46.13	3.68	2.51	0.25	0.64	0.01	0.13	0.01	0.70	0.01
	F-CAIM	**0.04**	0	**48.59**	1.14	**8.44**	0.23	**38.31**	0.68	**1.56**	0.02	**0.62**	0.02	**0.12**	0.01	**0.54**	0.01
The same discretization		NO		YES		NO		YES		NO		NO		NO		NO	
# different attributes		2		0		1		0		24		1		1		1	

Table 4. Comparison of results achieved by F-CAIM and CAIM algorithms, and the other discretization algorithms (bold indicates best results)

Criterion	CAIR mean value through all intervals		total # of intervals		time [s]	
Discretization Method	**mean rank** when comparing with CAIM	**mean rank** when comparing with F-CAIM	**mean rank** when comparing with CAIM	**mean rank** when comparing with F-CAIM	**mean rank** when comparing with CAIM	**mean rank** when comparing with F-CAIM
Equal Width	4.0	4.0	4.8	4.8	**1.3**	**1.3**
Equal Frequency	5.4	5.4	4.8	4.8	1.5	1.5
Paterson-Niblett	3.5	3.5	4.0	4.0	6.4	6.4
Maximum Entropy	5.9	5.9	4.4	4.4	**3.5**	3.8
CADD	3.4	3.4	3.6	3.6	6.6	6.6
IEM	3.1	3.0	2.1	2.1	4.1	4.5
CAIM / F-CAIM	**1.9**	**1.6**	**1.3**	**1.3**	4.1	**3.6**

The F-CAIM and CAIM algorithms achieve very similar results in terms of both the CAIR value and the number of discretization intervals when compared to other algorithms. Both were ranked as being the best among all other discretization algorithms.

The shortest execution time was obviously achieved by unsupervised discretization algorithms since they do not utilize class information. Among supervised algorithms the F-CAIM algorithm was the fastest. When analyzing performance of the CAIM algorithm, we note that it was the second fastest, with Maximum Entropy algorithm that was ranked best, and IEM algorithm that achieved the same result. Let us note that the F-CAIM algorithm is not only faster than the original CAIM algorithm but it also outperforms all other supervised discretization algorithms. This is a significant improvement that makes the F-CAIM algorithm applicable to large datasets with hundreds of thousands of data points and preferably small number of classes.

3.2. Analysis of classification results on the discretized data

The purpose of this experiment is to show the impact of selection of a discretization algorithm on performance of the subsequently used machine learning algorithm. The discretized datasets were used to generate classification rules by two ML algorithms: the rule algorithm called CLIP4 [6, 7], and the decision tree algorithm called C5.0 [10]. The results show accuracy and the number of the generated rules for the data discretized using the eight discretization algorithms, and for the original data in case of testing build-in discretization of the C5.0 algorithm.

Table 5 compares the results achieved by the F-CAIM and CAIM algorithms. It reports mean and standard deviation values for the accuracy and number of rules for rules generated by both CLIP4 and C5.0 algorithms. The results achieved by other dicretization algorithms can be found in [19].

The comparison shows that F-CAIM and CAIM algorithms achieve very comparable results for the rules generated by the CLIP4 algorithm. The results achieved for the C5.0 algorithm show that the data discretized using F-CAIM generates better results than the data discretized using CAIM. The accuracy of rules generated by C5.0 was better for three datasets for the data generated using F-CAIM. For five out of six datasets for which there was difference in discretization between CAIM and F-CAIM, the latter generates on average fewer number of rules. The F-CAIM generates data that results in generation of 75% fewer rules for the *pid* dataset, and 71% fewer rules for the *hea* dataset. This shows that the data discretized by F-CAIM is very well suited for decision tree algorithms. The main reason for this result is that the idea of using discretization boundaries, which lay on the class boundaries, which is applied in the F-CAIM algorithm, is also used in decision trees.

Table 5. Comparison of results achieved by F-CAIM and CAIM algorithms for the classification task performed on the already discretized data (bold indicates better results)

ML Algor.	Discretization Method	Datasets															
		iris		sat		thy		wav		ion		smo		pid		hea	
		acc	std	acc	std	acc	std	acc	std	acc	std	acc	std	acc	std	acc	std
CLIP4 accuracy	CAIM	92.7	8.0	76.4	2.0	97.9	0.4	76.0	1.9	**92.7**	3.9	**69.8**	4.0	**72.9**	3.7	79.3	5.0
	F-CAIM	92.7	8.0	76.4	2.0	**98.1**	0.7	76.0	1.9	91.8	4.9	69.0	2.9	72.5	3.9	**80.0**	6.3
C5.0 accuracy	CAIM	95.3	4.5	86.2	1.7	**98.9**	0.4	72.7	4.2	89.0	5.2	70.3	2.9	74.6	4.0	76.3	8.9
	F-CAIM	95.3	4.5	86.2	1.7	98.8	0.3	72.7	4.2	**90.0**	4.6	70.3	2.3	**74.7**	4.6	**76.9**	10.5
CLIP4 # rules	CAIM	**3.6**	0.5	45.6	0.7	7.0	0.0	14.0	0.0	1.9	0.3	**18.5**	0.5	**1.9**	0.3	**7.6**	0.5
	F-CAIM	4.5	0.9	45.6	0.7	7.0	0.0	14.0	0.0	1.9	0.3	18.8	0.4	3.3	0.7	7.8	0.4
C5.0 # rules	CAIM	3.2	0.6	332.2	16.1	10.9	1.4	58.2	5.6	7.7	1.3	**1.0**	0.0	20.0	2.4	31.8	2.9
	F-CAIM	**3.1**	0.3	332.2	16.1	**9.8**	0.8	58.2	5.6	**7.6**	0.5	2.2	1.4	**5.1**	0.9	**9.3**	0.8

Table 6. Comparison of results achieved by F-CAIM and CAIM algorithms, and the other discretization algorithms on the classification task performed on the already discretized data (bold indicates best results)

Algor.	Discretization Method	**mean rank** when comparing with CAIM	**mean rank** when comparing with F-CAIM
CLIP4 accuracy	Equal Width	4.6	4.6
	Equal Frequency	4.8	4.8
	Paterson-Niblett	4.3	4.1
	Maximum Entropy	5.3	5.3
	CADD	3.9	3.9
	IEM	2.9	2.8
	CAIM / F-CAIM	**1.8**	**2.0**
C5.0 accuracy	Equal Width	5.3	5.1
	Equal Frequency	6.0	5.0
	Paterson-Niblett	4.3	4.3
	Maximum Entropy	5.6	5.6
	CADD	5.4	5.5
	IEM	3.3	3.3
	CAIM / F-CAIM	**2.1**	**2.0**
	Built-in	3.3	3.4

Algor.	Discretization Method	**mean rank** when comparing with CAIM	**mean rank** when comparing with F-CAIM
CLIP4 # rules	Equal Width	3.8	3.5
	Equal Frequency	3.5	3.5
	Paterson-Niblett	2.6	**2.5**
	Maximum Entropy	3.6	3.6
	CADD	3.5	3.6
	IEM	3.0	2.9
	CAIM / F-CAIM	**2.1**	**2.5**
C5.0 # rules	Equal Width	4.9	4.9
	Equal Frequency	5.8	5.9
	Paterson-Niblett	3.3	3.3
	Maximum Entropy	5.8	5.9
	CADD	4.9	4.9
	IEM	3.5	3.6
	CAIM / F-CAIM	**1.9**	**2.5**
	Built-in	3.1	3.0

The accuracy and number of generated rules was compared between the six discretization algorithms and the F-CAIM algorithm. The same comparison was performed for the CAIM algorithm in [19]. The results are summarized using the rank values in Table 6. This enables direct comparison of performance between the F-CAIM and CAIM algorithms. The F-CAIM and CAIM achieve very similar results in terms of accuracy and number of rules when compared to other discretization algorithms. Both are ranked best among the considered discretization algorithms.

The results show that the F-CAIM algorithm generates the data that performs similarly as the data generated by the CAIM algorithm and better than the data generated by other discretization algorithms when subsequently used for supervised learning.

4. Summary and Conclusions

Discretization is a preprocessing step and thus should be characterized by very low complexity. To this end we proposed new discretization algorithm, called F-CAIM.

The F-CAIM algorithm is an extension of the CAIM algorithm. It preserves all advantages of the CAIM algorithm, and performs significantly faster than its predecessor especially on larger datasets. The F-CAIM algorithm was shown to be the fastest supervised discretization algorithm among all considered.

Like the CAIM algorithm, the F-CAIM algorithm discretizes the data in a way that results in the smallest number of intervals and the highest class-attribute interdependency when compared with other state-of-the-art discretization algorithms. The data discretized using F-CAIM significantly improves the accuracy of results achieved by the subsequently used ML algorithms. F-CAIM is better suited than CAIM to generate data for decision trees while both algorithms are similarly good for rule algorithms. Both F-CAIM and CAIM are better than the other discretization algorithms when analyzing results achieved by ML algorithms on the discretized data. Finally, F-CAIM, like CAIM, automatically selects the number of intervals, which is in striking contrast to many discretization algorithms.

In a nutshell, the results show high applicability of the F-CAIM algorithm for large datasets. It is scalable and accurate and can be used to perform supervised discretization tasks for a variety of real life problems.

References

[1] Blake, C.L. & Merz, C.J., UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science, 1998

[2] Catlett, J., On Changing Continuous Attributes into ordered discrete Attributes, *Proceedings of the. European Working Session on Learning,* pp.164-178, 1991

[3] Ching J.Y., Wong A.K.C. & Chan K.C.C.: Class-Dependent Discretization for Inductive Learning from Continuous and Mixed Mode Data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:7, pp. 641-651, 1995

[4] Chiu D., Wong A. & Cheung B., Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis, In: Piatesky-Shapiro G., Frowley W.J. (Eds.) *Knowledge Discovery in Databases*, MIT Press, 1991

[5] Cios, K. J., Pedrycz, W. & Swiniarski, R., Data *Mining Methods for Knowledge Discovery*, Kluwer, 1998

[6] Cios K. J. & Kurgan L., Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms. In: L. C. Jain, and J. Kacprzyk (Eds.) *New Learning Paradigms in Soft Computing*, Physica-Verlag (Springer), pp.276-322, 2001

[7] Cios, K.J., & Kurgan, L., Hybrid Inductive Machine Learning Algorithm that Generates Inequality Rules, *Information Sciences*, Special Issue on *Soft Computing Data Mining*, accepted, 2002

[8] Clark, P., and Niblett, Y., The CN2 Algorithm, *Machine Learning*, 3, pp.261-283, 1989

[9] Clark, P., and Boswell, R., Rule Induction with CN2: Some Recent Improvements, Lecture Notes in Artificial Intelligence, *Proceedings of the European Working Session on Learning*, Springer-Verlag, 1991

[10] Data Mining Tools, http://www.rulequest.com/ see5-info.html, 2002

[11] Dougherty J., Kohavi R. & Sahami M., Supervised and Unsupervised Discretization of Continuous Features, *Proceedings of the 12th International Conference on Machine Learning*, pp.194-202, 1995

[12] Elomaa, T., and Rousu, J., Speeding up the Search for Optimal Partitions, *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, Berlin, Heidelberg, Springer-Verlag. Lecture Notes in Artificial Intelligence, vol.1704, pp.89-97,1999

[13] Fayyad U.M. & Irani K.B., On the Handling of Continuous-Valued Attributes in Decision Tree Generation, *Machine Learning*, 8, pp.87-102, 1992

[14] Fayyad, U.M., and Irani, K.B. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, San Francisco, CA, Morgan Kaufmann, pp.1022-1027, 1993

[15] Kaufman, K.A., and Michalski, R.S., Learning from Inconsistent and Noisy Data: The AQ18 Approach, *Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems*, Warsaw, 1999

[16] Kerber R., ChiMerge: Discretization of Numeric Attributes, *Proceedings of the 9th International Conference on Artificial Intelligence* (AAAI-91), pp.123-128, 1992

[17] Kurgan L. & Cios K.J., Discretization Algorithm that Uses Class-Attribute Interdependence Maximization, *Proceedings of the 2001 International Conference on Artificial Intelligence* (IC-AI 2001), pp.980-987, Las Vegas, Nevada, 2001

[18] Kurgan, L. & Cios, K.J., DataSqueezer Algorithm that Generates Small Number of Short Rules, *IEE Proceedings: Vision, Image and Signal Processing*, submitted, 2002

[19] Kurgan, L., & Cios, K.J., CAIM Discretization Algorithm, *IEEE Transactions of Knowledge and Data Engineering*, accepted, 2003

[20] Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N., The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, *Proceedings of the Fifth National Conference on Artificial Intelligence*, Morgan-Kaufmann, pp.1041-1045, 1986

[21] Paterson, A. & Niblett, T.B., *ACLS Manua*l, Edinburgh: Intelligent Terminals, Ltd, 1987

[22] Pfahringer B., Compression-Based Discretization of Continuous Attributes, *Proceedings of the 12th International Conference on Machine Learning*, pp.456-463, 1995

[23] Quinlan, J.R., Induction of Decision Trees, *Machine Learning*, 1, pp.81-106, 1986

[24] Vlachos P., StatLib Project Repository, http://lib.stat.cmu.edu, 2000

[25] Wong A.K.C. & Chiu D.K.Y., Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9, pp.796-805, 1987

[26] Wong A.K.C. & Liu T.S., Typicality, Diversity and Feature Pattern of an Ensemble, *IEEE Transactions on Computers*, 24, pp.158-181, 1975