Draft. *Int. Journal of Approximate Reasoning* 15 (1996), 319–331

Global Discretization of Continuous Attributes as Preprocessing for Machine Learning

Michal R. Chmielewski and Jerzy W. Grzymala-Busse
Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66045
e-mail: jerzy@eecs.ukans.edu

Abstract. Real-life data usually are presented in databases by real numbers. On the other hand, most inductive learning methods require small number of attribute values. Thus it is necessary to convert input data sets with continuous attributes into input data sets with discrete attributes. Methods of discretization restricted to single continuous attributes will be called local, while methods that simultaneously convert all continuous attributes will be called global.

In this paper, a method of transforming any local discretization method into a global one is presented. A global discretization method, based on cluster analysis, is presented and compared experimentally with three known local methods, transformed into global. Experiments include ten-fold cross validation and leaving-one-out methods for ten real-life data sets.

Keywords: Discretization, quantization, continuous attributes, machine learning from examples, rough set theory.

1. Introduction

The process of converting data sets with continuous attributes into input data sets with discrete attributes, called *discretization*, was studied in many papers, see, e.g., [1–3, 5, 7, 10, 11, 13, 15]. We will assume that input data sets contain *examples*, characterized by *attribute* and *decision* values. A *concept* is defined as a set of all examples that have the same value *w* for decision *d*.

A data set may be either consistent or inconsistent. We need a measure of consistency for inconsistent data sets. Our measure, called a *level of consistency*, is based on *rough set theory*, a tool to deal with uncertainty, introduced by Z. Pawlak in [12]. Let *U* denote the set of all examples of the data set. Let P denote a nonempty subset of the set of all variables, i.e., attributes and a decision. Obviously, set P defines an equivalence relation $\wp$ on U, where two examples e and e' from U belong to the same equivalence class of $\wp$ if and only if both e and e' are characterized by the same values of each variable from P. The set of all equivalence classes of $\wp$, i.e., a partition on U, will be denoted P^*.

Equivalence classes of $\wp$ are called *elementary sets of* P. Any finite union of elementary sets of P is called a *definable set in* P. Let X be any subset of U. In general, X is not a definable set in P. However, set X may be approximated by two definable sets in P, the first one is called a *lower approximation of* X *in* P, denoted by $\underline{P}X$ and defined as follows

$$\cup\{Y \in P^* \mid Y \subseteq X\}.$$

The second set is called an *upper approximation of* X *in* P, denoted by $\overline{P}X$ and defined as follows

$$\cup\{Y \in P^* \mid Y \cap X \neq \emptyset\}.$$

The lower approximation of X in P is the greatest definable set in P, contained in X. The upper approximation of X in P is the least definable set in P containing X. A *rough set of* X is the family of all subsets of U having the same lower and the same upper approximations of X.

A data set is consistent with respect to decision d if and only if $\boldsymbol{A}^* \leq \{d\}^*$, where $\boldsymbol{A}$ is the set of all attributes. Furthermore, a *level of consistency*, denoted L_c, is defined as follows

$$\frac{\sum_{X \in \{d\}^*} |\underline{\boldsymbol{A}}X|}{|U|}.$$

In rough set theory, the level of consistency is known as the degree of dependency of $\{d\}$ from $\boldsymbol{A}$, see, e.g., [12]. For a data set that is consistent with respect to decision d, $L_c = 1$.

Let A be a continuous attribute, and let the domain of A be the interval $[a, b]$. A partition π_A on $[a, b]$ is defined as the following set of k subintervals

$$\pi_A = \{[a_0, a_1), [a_1, a_2), \ldots, [a_{k-1}, a_k]\},$$

where $a_0 = a$, $a_{i-1} < a_i$, for $i = 1, 2, ..., k$, and $a_k = b$. Thus, discretization is the process that produces a partition π_A on $[a, b]$.

The simplest discretization scheme is one where the discretized attribute's domain is as small as possible, i.e., $|\pi_A| = 2$. Hence the simplest (not necessarily the best) discretization is binary. We exclude the case where the size of the new domain is one (unary discretization), as the information presented in such an attribute is lost. Although there are infinitely many binary discretizations for any interval $[a, b]$, any attribute in a data set of m examples can only take on m distinct values. Hence, up to $m - 1$ binary discretizations schemes are practically possible.

The simplest method to discretize a continuous attribute is partitioning its domain into equal width intervals is called *Equal Interval Width Method*.

A method of attribute discretization through *adaptive discretization* was proposed in [3]. The domain of an attribute is first partitioned into two equal width intervals and a learning system is run to induce rules. Then, the quality of the rules is tested by evaluating rule performance. If performance measure falls below a fixed threshold, one of the partitions is sub-divided further, and the process is repeated. The principal disadvantage of this method, however, is the repetition of the learning process until the final performance level is reached.

A discretization based on maximal marginal entropy was introduced in [15]. This process involves partitioning the domain of the continuous attribute such that the sample frequency in each interval is approximately equal and is called *Equal Frequency per Interval Method*. The only parameter supplied by the user is the number of intervals to be induced on the original domain.

Another discretization with class-entropy as a criterion to evaluate a list of "best" breakpoints which together with the domain boundary points induce the desired intervals (*Minimal Class Entropy Method*) was suggested in [7]. A similar method of discretization is used in C4.5 [13]. The class information entropy of the partition induced by a break-point q is defined as

$$E\,(A, q; U) = \frac{|S_1|}{|U|}\,\mathrm{Ent}(S_1) + \frac{|S_2|}{|U|}\,\mathrm{Ent}(S_2).$$

The break point q for which $E(A, q; U)$ is minimal among all the candidate break points is taken to be the best break point. This determines the binary discretization for attribute A. In this paper, in order to induce k intervals the procedure outlined above is applied recursively $k - 1$ times. Having computed the binary discretization on U that partitions U into two sets U_1 and U_2, we now compute binary discretization on U_1 and U_2. Let $E(A, q_1; U_1)$ be the class information entropy associated with the best break point in U_1. Let $E(A, q_2; U_2)$ be the class information entropy associated with the best break point in U_2. Then, if

$$E(A, q_1; U_1) > E(A, q_2; U_2)$$

we partition U_1. Otherwise we partition U_2. Thus, the worse of sets U_1 and U_2 is partitioned.

In this paper we propose a method to discretize attributes by using hierarchical cluster analysis. When clustering cannot be performed any further, suitable post-processing using class-entropy measure is performed to fuse neighboring intervals. This method of discretization is called *Cluster Analysis Method*.

The discretization methods presented here can be classified as either *local* or *global*. Local methods are characterized by operating on only one attribute. Methods cited above as Equal Interval Width Method, Equal Frequency per Interval, and Minimal Class Entropy are local, while Cluster Analysis Method is global. Global methods are characterized by considering *all* attributes (rather than one) before making a decision where to induce interval break points. The *Globalization of Local Discretization Methods* section discusses the steps we have taken to improve local methods (by minimizing user interaction).

2. Globalization of Local Discretization Methods

Local methods suffer from the inability to predict how many intervals should be induced for a given domain of a continuous attribute A. Often, only an expert can tell accurately into how many intervals the domain of A should be partitioned and how the partitioning ought to be done. Being unaware of how many intervals to induce, which method to use, and repeating this step n

times (for each continuous attribute) can seriously jeopardize the outcome of the discretization process.

We can, however, abide by the following guidelines that intuitively insure successful discretization:

- Complete discretization. We are seldom interested in discretization of just one continuous attribute (unless there is only one such attribute in a data set).

- Simplest result of discretization. In general, the smaller the size of an attribute's domain after discretization, the simpler the rules that are induced from discretized data. As a result, the knowledge encompassed by such an attribute is more general.

- Consistency. Data sets with continuous attributes are almost always consistent ($L_c = 1$). When a discretization scheme applied to an attribute's values is "bad", an inconsistent data set may be obtained. When this happens, we lose valuable information. We should keep the level of consistency of the new discretized data set as close as possible to the original data's level of consistency.

With these points in mind the first step in transforming local discretization method into a global one is as follows. Using a chosen method (e.g., Equal Interval Width Discretization, Equal Frequency per Interval Discretization, Minimal Class Entropy Discretization), we partition each continuous attribute's domain into two intervals (binary discretization). Thus we achieve two out of three goals: complete and simplest discretization. At this time, exactly one of the following is true: (a) $L_c^D = L_c$ or (b) $L_c^D < L_c$, where L_c^D is the level of consistency of the new discretized data set and L_c is the level of consistency of the original data set.

If (a) is true we have produced the simplest discretization possible, discretized all candidate attributes, and maintained the original consistency level, thereby minimizing information loss due to discretization. If (b) is true we need to do more work. It is necessary to re-discretize an attribute's domain whose initial (binary) discretization was "poorest". Re-discretization will involve inducing $k + 1$ intervals on the domain of an attribute if k are currently present (using the same local discretization method). Determining which attribute's domain exhibits the "poorest" partitioning, however, is not straightforward. One way to do it is to play a series of what-if scenarios for each attribute. In turn, we compute the gain in the level of consistency of the data set as each attribute is re-discretized from k to $k + 1$ intervals. The attribute whose re-discretization produces the highest gain in level of consistency of the information system is selected and re-discretized. While this is an intuitive method it requires many time-consuming computations of the level of consistency to "play out" the various scenarios.

We propose a method to measure the performance of an attribute (after discretization) based on class entropy of attribute values. The rationale to use class entropy comes from the fact that if for some block $B \in \{A\}^*$ the class entropy of block B is zero then there exists a concept C in $\{d\}^*$ such that $B \subseteq C$, i.e., block is B is sufficient to describe (in part or whole) the concept C. This indicates that the current partitioning is good at least with respect to the block B.

The higher the entropy of the block B, the more randomness is encountered with respect to a concept and hence more chance of "poor" discretization. Since we are looking at whole attributes rather than just blocks of attribute-value pairs we will compute the average block entropy of an attribute A. We compute this measure according to the following formula

$$M_{\{A^D\}^*} = \frac{\sum_{B \in \{A^D\}^*} \frac{|B|}{|U|} \text{Ent}\,(B)}{|\{A^D\}^*|} \quad ,$$

where $\{A^D\}^*$ is the partition induced by the discretized attribute A^D. A candidate attribute for which $M_{\{A^D\}^*}$ is maximum is selected as the next attribute for re-discretization. The merit of computing average block entropy for an attribute is that we need only re-compute this measure for an attribute which was last picked for re-discretization. It is not a computationally intensive procedure.

3. Discretization Based on Cluster Analysis

The purpose of cluster analysis is to search for similar objects and group them into classes or clusters [6]. Objects are points in n-dimensional space which are defined by n characteristic values. During agglomerative cluster formation, objects that exhibit the most similarity are fused into a cluster. Once this process is completed, clusters can be analyzed in terms of all attributes.

The intent of the first part of this discretization method (cluster formation) is to determine initial intervals on the domains of the continuous attributes. During the second step (postprocessing) we hope to minimize the number of intervals that partition the domain of each continuous attribute.

3.1. Cluster Formation

There exist many different clustering techniques, none of which is a solution to all potential clustering needs. For our purposes, we have elected to use the Median Cluster Analysis method.

Let $m = |U|$ and let $\{A_1, A_2,..., A_i, A_{i+1},..., A_n\}$ be the set of all attributes, where attributes $A_1, A_2,..., A_i$ are continuous and attributes $A_{i+1}, A_{i+2},..., A_n$ are discrete ($1 < i \le n$). Let $e \in U$. We define the continuous component of e as

$$e_{continuous} = (x_1^e, x_2^e,..., x_i^e)$$

and the discrete component of e as

$$e_{discrete} = (x_{i+1}^e, x_{i+2}^e,..., x_n^e).$$

In cases where continuous attributes' values are not of the same scale (feet, pounds, light-years, etc.) we must standardize attribute values to zero mean and unit variance for clustering to be successful. This is done by dividing the attribute values by the corresponding attribute's standard deviation (derived from the complete set of values for the attribute) [6].

We begin cluster formation by computing an $m \times m$ distance matrix between every pair of continuous components of examples in U. The entries in the distance matrix correspond to squared Euclidean distances between data points in i-dimensional space. We initiate clusters by allowing each i-dimensional data point to be a cluster. Thus, we originally have m clusters, each of cardinality one.

New clusters are formed by merging two existing clusters that exhibit the most similarity between each other. In our case, this involves finding two clusters that are separated by the smallest Euclidean distance. When such a pair is found (clusters b and c), they are fused to form a new cluster bc. The formation of the cluster bc introduces a new cluster to the space, and hence its similarity (distance) to all the other remaining clusters must be re-computed. For this purpose, we use the Lance and Williams Flexible Method [6]. Given a cluster a and a new cluster bc to be formed from clusters b and c the distance from bc to a is computed as

$$d_{a(bc)} = d_{(bc)a} = \alpha_b d_{ab} + \alpha_c d_{ac} + \beta d_{bc} + \gamma |d_{ab} - d_{ac}|$$

where $\alpha_b = \alpha_c = \frac{1}{2}$, $\beta = -\frac{1}{4}$ and $\gamma = 0$ for the Median Cluster Analysis method.

At any point during the clustering process the clusters formed induce a partition on the set of examples U. Examples that belong to the same cluster are indiscernible by the subset of continuous attributes. Therefore, we should continue cluster formation until the level of consistency of the partition $\{K \mid K \text{ is a cluster}\}$ is equal to or greater than the original data's level of consistency L_c.

When the above condition fails, cluster formation stops, and we analyze the clusters to determine candidate intervals that will partition the domain of each of the i continuous attributes. Since the points in the clusters are defined by i attribute values, we can examine how the individual attributes define each cluster. Let r be the number of clusters produced. Let K be a cluster. The set of data points of an attribute A_j ($1 \le j \le i$) that define a cluster K is

$$DP_{A_j}^{K} = \{x_j^e \mid e \in K\}.$$

Hence, the defining interval I_{K,A_j} of cluster K_j with respect to attribute A_j is

$$I_{K,Aj} = [L_{K,A_j}, R_{K,A_j}] = [\min (DP_{A_j}^{K}), \max (DP_{A_j}^{K})]$$

It is possible that for a given attribute A_j, the domain that defines cluster K is a subdomain of a domain that defines another cluster K', i.e., ($L_{K,A_j} \geq L_{K',A_j}$ and $R_{K,A_j} \leq R_{K',A_j}$). From the perspective of the intervals

$$I_{K,Aj} = [L_{K,A_j}, R_{K,A_j}] \text{ and } I_{K',Aj} = [L_{K',A_j}, R_{K',A_j}]$$

we can safely eliminate the sub-interval $I_{K,Aj}$ from further consideration without compromising the discretization outcome.

At this time we are ready to induce intervals on each attribute's continuous domain. For each attribute A_j where $j \in \{1, 2,..., i\}$ we construct two sets, L_{A_j} and R_{A_j}, which contain the left and right boundary points respectively of the defining intervals $I_{K_l,Aj}$ for $l \in \{1, 2, ..., r\}$. Hence,

$$L_{A_j} = \{L_{K_l,Aj} \mid l \in \{1, 2,..., r\}\}$$

and

$$R_{A_j} = \{R_{K_l,Aj} \mid l \in \{1, 2,..., r\}\}.$$

The interval partition on the domain of the attribute A_j is equal to

$$\pi_{A_j} = \{[\min_1(L_{A_j}), \min_2(L_{A_j})), [\min_2(L_{A_j}), \min_3(L_{A_j})),..., [\min_r(L_{A_j}), \max(R_{A_j})]\},$$

where $\min_a(L_{A_j})$ corresponds to the a^{th} smallest element of L_{A_j}.

3.2. Postprocessing

Postprocessing involves merging adjacent intervals in an attempt to reduce the domain size of each of the discretized attributes. This final processing task is divided into two stages: "safe" merging of intervals and merging of intervals.

Given an interval partition on the domain of an attribute A_j we can "safe" merge adjacent intervals providing that the outcome does not affect in any way the accuracy of the information system. Let $\pi_{A_j} = \{[a_0, a_1), [a_1, a_2),..., [a_{k-1}, a_k]\}$ be the interval partition of attribute A_j. Consider any pair of adjacent intervals $[a_{l-1}, a_l)$ and $[a_l, a_{l+1})$. We can fuse them safely into one interval $I_{l-1,l+1} = [a_{l-1}, a_{l+1})$ if and only if the class entropy of the block associated with $I_{l-1,l+1}$ is 0. A class entropy value of 0 indicates that the new interval $I_{l-1,l+1}$ describes one concept only (in part or in full). Therefore, the consistency of the data set is not violated. Safe merging of intervals is applied sequentially to all attributes for which intervals have been induced.

Merging involves exactly the same task as the previous step. However, since each of the merges performed may affect the consistency of the data set, two questions must be resolved first:

1. which attribute's intervals to merge first, and

2. which neighboring intervals to merge first.

In order to resolve these problems we utilize the class entropy function to determine the priority of interval merging. The rationale behind this choice is that we are interested in forming intervals whose blocks exhibit most class uniformity. Intervals whose blocks show high class entropy are not desirable because they may contribute adversely to the rule induction process.

To prioritize interval merging, we compute the class entropy of the block associated with every pair of adjacent intervals for all continuous attributes. We pick a pair of adjacent intervals for merging whose class entropy is the smallest amongst continuous attributes. By employing this procedure, we are not biasing merging to any particular attribute and hence point 1 above is resolved. There may be some natural bias, however, especially if data contained in an attribute

Table 1. Data Sets

Data set	Number of examples	Number of continuous attributes	Number of discrete attributes	Number of concepts
GM	200	20	0	2
rocks	1133	13	0	4
iris	150	4	0	3
bank	66	5	0	2
hsv-r	122	9	2	4
bupa	345	6	0	2
glass	214	9	0	7
wave	512	21	0	3
image	210	19	0	7
cars	193	14	9	6

show high class correlation.

Before a merge is performed, we must check if the accuracy of the data set will fall below a given threshold as a result of the merge. If the accuracy will still be adequate we perform the merge. Otherwise, we mark this pair of adjacent intervals as non-mergable and proceed to the next candidate. The process stops when each possible pair of adjacent intervals is marked as non-mergable.

4. Experiments

In order to compare the performance of discretization methods presented in this report, we have subjected a sample of ten data sets to discretization. The data contain real-life information from fields such as medicine, insurance, banking, and science and have been used previously in testing pattern recognition and machine learning methods. Table 1 gives a summary of data sets used in experiments.

The data sets *GM*, *rocks* and *bank* were donated by W. Koczkodaj from Laurentian University, Canada. The former contains financial data gathered by General Motors, the latter information about volcanic rocks. R. A. Fisher is credited for the well-known *iris* data set. The data set *bank* describing bankruptcy data was created by E. Altman and M. Heine at the New York University School of Business in 1968. The data set *hsv-r*, donated by R. Slowinski from Technical University of Poznan, Poland, represents raw data on treatment of duodenal ulcer by HSV. Remaining five data sets, as well as *iris* data set, were taken from the University of California at Irvine repository of machine learning databases. The data set *bupa*, describing a liver disorder, contain data gathered by BUPA Medical Research Ltd, England. The data set *glass*, representing glass types, has been created by B. German, Central Research Establishment, Home Office Forensic Science Service, Canada. The data set *waveform*, as described in [1], represents three types of waves. The data set *image* created in 1990 by the Vision Group, University of Massachusetts, represents image features: brickface, sky, foliage, cement, window, path, and grass. The data set *cars*, created in 1985 by J. C. Schlimmer, represents insurance risk ratings. All of the above ten data sets have the level of consistency equal to 100%.

Each of the data sets in the sample was discretized completely, i.e., all continuous attributes were processed. We have used the globalized version of the local methods: Equal Interval Width, and Equal Frequency per Interval as well as the Cluster Analysis method to perform discretization. Furthermore, in order to minimize information loss due to discretization, we have maintained the original consistency level throughout the various discretization procedures. Hence, we have produced four consistent data sets from each data set in the sample.

As discretization is only a preprocessing step to concept acquisition through machine learning, it seemed worthwhile to investigate the quality of the knowledge base induced from the preprocessed data sets. To make this possible we have utilized the learning program LEM2 (Learning from Examples Module, version 2) to induce rules from the discretized data sets. LEM2 is a typical member of the family of learning algorithms in that it finds a minimal discriminant description, see, e.g., [4, 8].

The most important performance criterion of the discretization method is the accuracy rate. A complete discussion on how to evaluate the error rate (and hence the accuracy rate) of a rule set induced from a data set is contained in [14]. We have used the following cross validation guidelines in computing the accuracy rate:

- If the number of examples was less than 100, the leaving-one-out method was used to estimate the accuracy rate of the rule set. Leaving-one-out involves m learn-and-test experiments (where $m = |U|$). During the j^{th} experiment, the j^{th} example is removed from the data set, automatic concept acquisition is performed on the remaining $m - 1$ examples (using LEM2), and the classification of the omitted example by rules produced is recorded. The accuracy rate is computed as

$$1 - \frac{\text{total misclassifications}}{\text{number of examples}}$$

- If the number of instances in the data set was more than 100, the ten-fold technique was used. This technique is similar to leaving-one-out in that it follows the learn-and-test paradigm. In this case, however, the learning sample corresponds to 90% of the original data, the testing sample is the remaining 10%, and the experiments are repeated ten times. This method is used primarily to save time at the negligible expense of accuracy. The accuracy rate is computed in the same way as for the leaving-one-out method.

5. Conclusions

To analyze the results obtained, see Table 2, we have used the Wilcoxon matched pairs signed rank test, see, e.g., [9, pp. 546–561]. The purpose of this non-parametric test is to determine if significant differences exist between two populations. Paired observations from the two populations are the basis of the test, and magnitudes of differences are taken into account. This is a straightforward procedure to either accept or reject the null hypothesis which is commonly taken to be "identical population distributions".

The Cluster Analysis method had outperformed the Equal Interval Width and Equal Frequency per Interval methods. The Cluster Analysis method is better than Equal Interval Width Method with 1% significance level for one-tailed test. Also, the Cluster Analysis Method is better than the Equal Frequency per Interval Method with 0.5% significance level for one-tailed test. Minimal Class Entropy showed no significant performance difference to any other methods, i.e., the null hypothesis that the Minimal Class Entropy performs likewise any other method could not be rejected even at the 5% significance level for one-tailed test.

Following these results we would like to suggest that more study should be done to better understand how clustering based techniques can be used in discretization. Admittedly, we have selected one of the most straightforward clustering methods. It would be worthwhile to examine how different clustering methods perform in this global approach to discretization.

In the suggested globalization of local discretization methods, the search for a new attribute to be re-discretized is based on the maximum of the measure $M_{\{A^D\}^*}$. This way the search is

Table 2. Accuracy rate after discretization

Data set	Equal Interval Width	Equal Frequency per Interval	Minimal Class Entropy	Cluster Analysis Method
GM	68.0	59.0	73.0	69.0
rocks	57.5	54.2	55.6	53.0
iris	91.5	86.7	82.0	95.3
bank	77.3	95.5	84.9	97.0
hsv-r	42.5	35.8	46.7	48.3
bupa	41.9	39.7	41.3	42.5
glass	54.7	49.5	56.1	60.3
wave	99.4	99.4	99.4	99.8
image	69.0	70.0	73.8	77.6
cars	58.0	59.6	67.8	63.7

oriented for looking for the worst attribute that should be modified. Other criteria should be investigated as well, e.g., looking for the best attribute.

Finally, it is likely that certain local discretization methods work better with some type of data. It would be interesting to see whether a determination could be made to select a given discretization method based solely on the data characteristics of an attribute or a data set.

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. *Classification And Regression Trees*. Monterey, CA: Wadsworth & Brooks, 1984.

2. Catlett, J. "On changing continuous attributes into ordered discrete attributes". In Y. Kodratoff, (ed.), *Machine Learning—EWSL-91, Proc. of the European Working Session on Learning, Porto. Portugal, March 1991*, Lecture Notes in Artificial Intelligence. Berlin, Germany: Springer-Verlag, pp. 164–178, 1991.

3. Chan, C.- C., Batur, C, Srinivasasn, A. "Determination of quantization intervals in rule based model for dynamic systems". *Proc. of the IEEE Conference on Systems, Man, and Cybernetics, Charlottesville, VA*, pp. 1719–1723, Oct. 13-16, 1991.

4. Chan, C.- C., Grzymala-Busse, J. W. "On the attribute redundancy and the learning programs ID3, PRISM, and LEM2". Department of Computer Science, University of Kansas, TR-91-14, 20 pp., 1991.

5. Chiu, D. K. Y., Wong, A. K. C. Cheung, B. "Information discovery through hierarchical maximum entropy discretization and synthesis". In G. Piatetsky-Shapiro & W. J. Frawley (eds.), *Knowledge Discovery in Databases*. Cambridge, MA: MIT Press, pp. 125–140, 1991.

6. Everitt, B. *Cluster Analysis.* London, United Kingdom: Heinmann Educational Books, Second Edition, 1980.

7. Fayyad, U. M., Irani, K. B. "On the handling of continuous-valued attributes in decision tree generation". *Machine Learning* 8, pp. 87–102, 1992.

8. Grzymala-Busse, J. W. "LERS—A system for learning from examples based on rough sets". In R. Slowinski (ed.), *Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory.* Norwell, MA: Kluwer Academic Publishers, pp. 3–18, 1992.

9. Hamburg, M. *Statistical Analysis for Decision Making.* New York, NY: Harcourt Brace Jovanovich, Third Edition, 1983.

10. Lenarcik, A. Piasta, Z. "Probabilistic approach to design algorithm generation in the case of continuous condition attributes". *Proc. of the 1st Int. Workshop on Rough Sets: State of Art and Perspectives, Poznan-Kiekrz, Poland*, September 20–22, 1992, .

11. Pao, Y.-H. Bozma, Y.-H. "Quantization of numerical sensor data for inductive learning". In J. S. Kowalik (ed.), *Coupling Symbolic and Numerical Computing in Expert Systems*. Amsterdam, The Netherlands: Elsevier Science Publ., pp. 69–81, 1986.

12. Pawlak, Z. Rough sets. *International Journal Computer and Information Sciences* 11, pp. 341–356, 1982.

13. Quinlan, J. R. *C4.5*: *Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

14. Weiss, S. M., Kulikowski, C. A. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufmann Publishers, 1990.

15. Wong, A. K. C., Chiu, D. K. Y. "Synthesizing statistical knowledge from incomplete mixed-mode data". *IEEE Transaction on Pattern Analysis and Machine Intelligence* 9, pp. 796–805, 1987.