GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 47-58

Discretization Techniques: A recent survey

Sotiris Kotsiantis, Dimitris Kanellopoulos

Educational Software Development Laboratory
Department of Mathematics, University of Patras, Greece
sotos@math.upatras.gr, dkanellop@teipat.gr

Abstract. A discretization algorithm is needed in order to handle problems with real-valued attributes with Decision Trees (DTs), Bayesian Networks (BNs) and Rule-Learners (RLs), treating the resulting intervals as nominal values. The performance of these systems is tied to the right election of these intervals. A good discretization algorithm has to balance the loss of information intrinsic to this kind of process and generating a reasonable number of cut points, that is, a reasonable search space. This paper presents the well known discretization techniques. Of course, a single article cannot be a complete review of all discretization algorithms. Despite this, we hope that the references cited cover the major theoretical issues and guide the researcher to interesting research directions and suggest possible combinations that have to be explored.

1 Introduction

Many Machine Learning (ML) algorithms are known to produce better models by discretizing continuous attributes [14]. Naive Bayes (NB) classifier requires the estimation of probabilities and the continuous explanatory attributes are not so easy to handle, as they often take too many different values for a direct estimation of frequencies. To circumvent this, a normal distribution of the continuous values can be assumed, but this hypothesis is not always realistic. The same phenomenon leads rules extraction techniques to build poorer sets of rules. DT algorithms carry out a selection process of nominal attributes and cannot handle continuous ones directly.
As result, a large number of ML and statistical techniques can only be applied to data sets composed entirely of nominal variables. However, a very large proportion of real data sets include continuous variables: that is variables measured at the interval or ratio level. One solution to this problem is to partition numeric variables into a number of sub-ranges and treat each such sub-range as a category. This process of partitioning continuous variables into categories is usually termed discretization. Unfortunately, the number of ways to discretize a continuous attribute is infinite. Discretization is a potential time-consuming bottleneck, since the number of possible discretizations is exponential in the number of interval threshold candidates within the domain [14].
The goal of discretization is to find a set of cut points to partition the range into a small number of intervals that have good class coherence, which is usually measured by an evaluation function. In addition to the maximization of interdependence be-

tween class labels and attribute values, an ideal discretization method should have a secondary goal to minimize the number of intervals without significant loss of class-attribute mutual dependence. Discretization is usually performed prior to the learning process and it can be broken into two tasks. The first task is to find the number of discrete intervals. Only a few discretization algorithms perform this; often, the user must specify the number of intervals or provide a heuristic rule. The second task is to find the width, or the boundaries, of the intervals given the range of values of a continuous attribute.

Usually, in discretization process, after sorting data in ascending or descending order with respect to the variable to be discretized, landmarks must be chosen among the whole dataset. In general, the algorithm for choosing landmarks can be either top-down, which starts with an empty list of landmarks and splits intervals, or bottom-up, which starts with the complete list of all the values as landmarks and merges intervals. In both cases there is a stopping criterion, which specifies when to stop the discretization process. Researchers in the ML community have introduced many discretization algorithms. Most of these algorithms perform an iterative greedy heuristic search in the space of candidate discretizations, using different types of scoring functions for evaluating a discretization. A recent overview of various types of discretization algorithms can be found in [28]. Thus, in this work apart from the brief description of the discretization process we refer to some more recent works than those in Liu's article as well as few articles that were not referred by Liu.

The next section covers wide ranged issues of discretization process. The categories of discretization methods are analysed in section 3, whereas the combination of discretization methods with DTs, RLs and BNs are presented in section 4. Finally, the last section concludes this work.

2 Discretization process

The term "cut-point" refers to a real value within the range of continuous values that divides the range into two intervals, one interval is less than or equal to the cutpoint and the other interval is greater than the cut-point. For example, a continuous interval [a, b] is partitioned into [a, c] and (c, b], where the value c is a cut-point. Cut-point is also known as split-point. The term "arity" in the discretization context means the number of intervals or partitions. Before discretization of a continuous feature, arity can be set to k—the number of partitions in the continuous features. The maximum number of cut-points is $k - 1$. Discretization process reduces the arity but there is a trade-off between arity and its effect on the accuracy.

A typical discretization process broadly consists of four steps: (1) sorting the continuous values of the feature to be discretized, (2) evaluating a cut-point for splitting or adjacent intervals for merging, (3) according to some criterion, splitting or merging intervals of continuous value, and (4) finally stopping at some point.

After sorting, the next step in the discretization process is to find the best "cut-point" to split a range of continuous values or the best pair of adjacent intervals to merge. One typical evaluation function is to determine the correlation of a split or a merge with the class label. There are numerous evaluation functions found in the literature

such as entropy measures and statistical measures (more details in the following sections). A stopping criterion specifies when to stop the discretization process. It is usually governed by a trade-off between lower arity with a better understanding but less accuracy and a higher arity with a poorer understanding but higher accuracy. The number of inconsistencies (inconsistency is defined later) caused by discretization—it should not be much higher than the number of inconsistencies of the original data before discretization. Two instances are considered inconsistent if they are the same in their attribute values except for their class labels.
Generally, the discretization methods can be categorised as: (1) supervised or unsupervised, (2) direct or incremental, (3) global or local, (4) static or dynamic, (5) top-down or bottom-up. We distinct these categories in the following section.

3 Categories of discretization methods

A distinction can be made dependent on whether the method takes class information into account to find proper intervals or not. Several discretization methods, such as equal width interval binning or equal frequency binning, do not make use of class membership information during the discretization process. These methods are referred to as unsupervised methods. In contrast, discretization methods that use class labels for carrying out discretization are referred to as supervised methods. Previous research indicated that supervised are better than unsupervised methods [12].
The two representative algorithms of unsupervised (or class-blind) algorithms are equal-width and equal-frequency discretizations. The equal-width discretization algorithm determines the minimum and maximum values of the discretized attribute and then divides the range into the user-defined number of equal width discrete intervals. The equal-frequency algorithm determines the minimum and maximum values of the discretized attribute, sorts all values in ascending order, and divides the range into a user-defined number of intervals so that every interval contains the same number of sorted values. The obvious weakness of this equal-width method is that in cases where the outcome observations are not distributed evenly, a large amount of important information can be lost after the discretization process. For equal-frequency, many occurrences of a continuous value could cause the occurrences to be assigned into different bins. One improvement can be after continuous values are assigned into bins, boundaries of every pair of neighboring bins are adjusted so that duplicate values should belong to one bin only.
Another dimension of discretization methods is direct vs. incremental process [14]. Direct methods divide the range of k intervals simultaneously (i.e., equal-width), needing an additional input from the user to determine the number of intervals. Incremental methods begin with a simple discretization and pass through an improvement process, needing an additional criterion to know when to stop discretizing [7].
The distinction between global and local discretization methods is dependent on when discretization is performed [28]. Global discretization handles discretization of each numeric attribute as a pre-processing step, i.e. before induction of a classifier whereas local methods, like C4.5 carry out discretization on-the-fly (during induction). Empirical results have indicated that global discretization methods often produced supe-

rior results compared to local methods since the former use the entire value domain of a numeric attribute for discretization, whereas local methods produce intervals that are applied to subpartitions of the instance space.
The distinction between static and dynamic methods depends on whether the method takes feature interactions into account [32]. Static methods, such as binning, entropy-based partitioning and the 1R algorithm [21], determine the number of partitions for each attribute independent of the other features. In contrast, dynamic methods conduct a search through the space of possible k partitions for all features simultaneously, thereby capturing interdependencies in feature discretization.
Finally, the distinction between top-down and bottom-up discretization methods can be made. Top-down methods consider one big interval containing all known values of a feature and then partition this interval into smaller and smaller subintervals until a certain stopping criterion, for example Minimum Description Length (MDL), or optimal number of intervals is achieved. In contrast, bottom-up methods initially consider a number of intervals, determined by the set of boundary points, to combine these intervals during execution until a certain stopping criterion, such as a x^2 threshold, or optimal number of intervals is achieved [24].
In the discretization problem, a compromise must be found between information quality (homogeneous intervals in regard to the attribute to predict) and statistical quality (sufficient sample size in every interval to ensure generalization). The chi-square-based criteria focus on the statistical point of view whereas the entropy-based criteria focus on the information point of view. Other criteria (such as Gini or Fusinter criterion) try to find a trade off between information and statistical properties. There are still other approaches, similar to the wrapper approach involved in the feature selection field as well as evolutionary based approaches. We analyze all these approaches in the following sub-sections.

3.1 Chi-square based methods

Chi-square (x^2) is a statistical measure that conducts a significance test on the relationship between the values of a feature and the class. Kerber [24] argues that in an accurate discretization, the relative class frequencies should be fairly consistent within an interval (otherwise the interval should be split to express this difference) but two adjacent intervals should not have similar relative class frequency (in that case the adjacent intervals should be merged into one). The x^2 statistic determines the similarity of adjacent intervals based on some significance level. It tests the hypothesis that two adjacent intervals of a feature are independent of the class. If they are independent, they should be merged; otherwise they should remain separate.
The top-down method based on chi-square is ChiSplit. It searches for the best split of an interval, by maximizing the chi-square criterion applied to the two sub-intervals adjacent to the splitting point: the interval is split if both sub-intervals substantially differ statistically. The ChiSplit stopping rule is based on a user-defined chi-square threshold to reject the split if the two sub-intervals are too similar.
The bottom-up method based on chi-square is ChiMerge [24]. It searches for the best merge of adjacent intervals by minimizing the chi-square criterion applied locally to

two adjacent intervals: they are merged if they are statistically similar. The stopping rule is based on a user-defined Chi-square threshold to reject the merge if the two adjacent intervals are insufficiently similar. No definite rule is given to choose this threshold. Chi2 [30] is an automated version of ChiMerge. In Chi2, the statistical significance level keeps changing to merge more and more adjacent intervals as long as the inconsistency criterion [29] is satisfied. A method very similar to Chi2 is ConMerge [33] which also uses the x^2 statistic and the inconsistency measure. Instead of considering one attribute at a time, ConMerge chooses the lowest x^2 value among the intervals of all continuous features.
Boulle [5] proposes the discretization method Khiops, based on the chi-square statistic. In contrast with related methods ChiMerge and ChiSplit, this method optimizes the Chi-square criterion in a global manner on the whole discretization domain and does not require any stopping criterion. The Khiops method starts the discretization from the elementary single value intervals. It evaluates all merges between adjacent intervals and selects the best one according to the chi-square criterion applied to the whole set of intervals. The stopping rule is based on the confidence level computed with the chi-square statistic. The method automatically stops merging intervals as soon as the confidence level, related to the Chi-square test of independence between the discretized attribute and the class attribute, does not decrease anymore.

3.2 Entropy based methods

Other discretization methods use entropy measures to evaluate candidate cutpoints. This means that an entropy-based method will use the class information entropy of candidate partitions to select boundaries for discretization. Class information entropy is a measure of purity and it measures the amount of information which would be needed to specify to which class an instance belongs. It considers one big interval containing all known values of a feature and then recursively partitions this interval into smaller subintervals until some stopping criterion, for example Minimum Description Length (MDL) Principle or an optimal number of intervals is achieved. Other evaluation measures include Gini, dissimilarity and the Hellinger measure.
The MDL principle states that the best hypothesis is the one with minimal description length. As partitioning always decreases the value of the entropy function, considering the description lengths of the hypotheses allows balancing the entropy gain and eventually accepting the null hypothesis. Performing recursive bipartitions with this criterion leads to a discretization of the continuous explanatory attribute at hand.
FID [15] evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For each evaluation of a candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidate cut points. This binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.
It has been shown that optimal cut points for entropy minimization must lie between examples of different classes [15]. A similar result can be proved for Zeta maximiza-

tion: a measure based on minimization of the error rate when each value of an independent variable must predict a different value of a dependent variable [20].
The approach of [9] seeks to maximize the mutual dependence as measured by the interdependence redundancy between the discrete intervals and the class labels, and can automatically determine the most preferred number of intervals for an inductive learning application. Because of its ability to automatically select the minimum number of intervals without significantly reducing useful mutual information, the method can speed up the learning time of most inductive learners with little classification accuracy loss.
USD [19] divides the continuous attributes in a finite number of intervals with maximum goodness, so that the average-goodness of the final set of intervals will be the highest. The main process is divided in two different parts: first, it calculates the initial intervals by means of projections, depending on the goodnesses obtained after carrying out two possible actions: to join or not adjacent intervals. The main features of the algorithm are: it is deterministic, does not need any user–parameter and its complexity is subquadratic.
The goal of the CAIM algorithm [26] is to maximize the class-attribute interdependence and to generate a (possibly) minimal number of discrete intervals. The CAIM algorithm maximizes mutual class-attribute interdependence and possibly generates the smallest number of intervals for a given continuous attribute. It was tested on several well-known data sets and compared with six other state-of-the-art discretization algorithms. According to the authors, the comparison shows that the CAIM algorithm generates discretization schemes with, on average, the lowest number of intervals and the highest dependence between class labels and discrete intervals, thus outperforming other discretization algorithms.

3.3 Wrapper based methods

While most discretization methods are employed as a preprocessing step to an induction algorithm, there are still other approaches, similar to the wrapper approach involved in the feature selection field. For example, the authors of [4] and [6] propose approaches that allow to refine the discretization of the continuous explanatory attributes by taking feedback from an induction algorithm. Error-based methods, such as [31], evaluate candidate cutpoints against an error function and explore a search space of boundary points to minimize the sum of false positive (FP) and false negative (FN) errors on the training set. In other words, given a fixed number of intervals, error-based discretization aims at finding the best discretization that minimizes the total number of errors (FP and FN) made by grouping together particular continuous values into an interval.
Another example of using accuracy for discretization is Adaptive Quantizer [8]. It considers how well one attribute predicts the class at a time. For each attribute, its continuous range is split into two partitions either by equal-frequency or by equal-width. The splitting is tested by running a classifier to see if the splitting helps improve accuracy. It continues binarizing subranges and the cut-point which gives the

minimum rate is selected. As it involves training a classifier, it is usually more time consuming than those without using a classifier.
Elomaa and Rousu [13] showed that in order to find Training Set Error (the optimal discretization in which the interval labeling differs least often from that of the data points) one only needs to examine a small number of all cut points, called alternation points. On the other hand, the authors prove that failing to check an alternation point may lead to a suboptimal discretization. Alternation points can be identified efficiently once the data has been ordered.
Elomaa and Rousu [14] presented techniques for speeding up the discovery of optimal—with respect to an evaluation function—multisplits along numerical value ranges. The techniques are based on proving some cut points or prefixes of partitions as suboptimal and thus discarding them from the search space of the algorithms. Overall, the quadratic-time dynamic programming algorithm runs twice as fast on the average and in some cases up to 90 percent faster than the baseline algorithm.
The objective of cost-based discretization approach [23] is to take into account the cost of making errors instead of just minimizing the total sum of errors, such as in error-based discretization. The specification of this cost function is dependent on the costs assigned to the different error types. In the special case where the cost of making errors is equal, the introduced method of cost-based discretization is in fact equal to error-based discretization.

3.4 Adaptive Discretization and Evolutionary based methods

Another discretization-based knowledge representation has been developed, called Adaptive Discretization Intervals (ADI) [3]. This representation is used inside a Pittsburgh LCS and uses rules that contain intervals built joining together the low level intervals provided by the discretization algorithm, thus collapsing the search space when it is possible. Also, this representation can use several discretization algorithms at the same time allowing the system to choose the correct discretization for each problem and attribute. The authors of [2] generalize the ADI representation approach (proposing ADI2) by also using heuristic non-uniform discretization methods. This representation evolves rules that can use multiple discretizations, letting the evolution choose the correct discretization for each rule and attribute. Moreover, the intervals defined in each discretization can split or merge among them through the evolution process, reducing the search space where it is possible. There are other systems that, like ADI, perform evolutionary induction of rules based on discretization [17], [10]. A comparison of ADI with these two methods is found in [1].
An elegant and robust approach for overcoming univariate supervised discretization methods' drawback is provided by evolutionary algorithms, which can be used for performing local multivariate discretization during the evolutionary learning process. In these methods numerical values are handled by means of inequalities that can be modified, by means of ad hoc operators, during the evolutionary process. These methods differ among each other mainly in the way inequalities, describing continuous attribute subranges, are encoded, and in the definition of suitable genetic operators for modifying inequalities. The authors of [11] analyze experimentally discretiza-

tion algorithms for handling continuous attributes in evolutionary learning. They consider a learning system that induces a set of rules in a fragment of first-order logic, and introduce a method where a given discretization algorithm is used to generate initial inequalities, which describe sub-ranges of attributes' values. Mutation operators exploiting information on the class label of the examples are used during the learning process for refining inequalities.

3.5 Other methods

In theory, given a k-valued classification variable C, a continuous variable X could be partitioned into k sub-ranges by calculating e.g. Zeta [20] for each of the possible assignments of the k-1 cut points and selecting that combination of cut points that gives the largest value of Zeta. In general such a method will not be practicable because the number of combinations of cut points is extremely large. Cerquides and Lopez [7] proposed a discretization method based on a distance metric between partitions that can be easily implemented in parallel. This method is very effective and efficient in very large datasets.

For discretization, two methods of cluster analysis: agglomerative (bottom-up) and divisive (top-down) have been also used [19]. In agglomerative techniques, initially each case is a single cluster, and then they are fused together, forming larger and larger clusters. In divisive techniques, initially all cases are grouped in one cluster, then this cluster is gradually divided into smaller and smaller clusters. In both methods, during the first step of discretization, cluster formation, cases that exhibit the most similarity are fused into clusters. Once this process is completed, clusters are projected on all attributes to determine initial intervals on the domains of the numerical attributes. During the merging step adjacent intervals are merged together.

The univariate case does not take into account any correlation between the explanatory attributes and fails to discover conjointly defined patterns. Many authors proposed multivariate methods that search for cut points simultaneously [27]. Ferrandiz and Boull [16] proposed an extension to the multivariate case, relying on the multivariate definition of discrete neighborhood by means of a non-oriented graph structure. A framework for supervised bipartitioning has been proposed, which applied recursively leads to a new multivariate discretization algorithm.

Non-Disjoint Discretization (NDD) forms overlapping intervals for a numeric attribute [35], always locating a value toward the middle of an interval to obtain more reliable probability estimation. It also adjusts the number and size of discretized intervals to the number of training instances, seeking an appropriate trade-off between bias and variance of probability estimation.

4 Combining discretization methods with DTs, RLs and BNs

The quadratic time complexity of the popular C4.5 DT learning algorithm for non-numeric data increases by a logarithmic factor when numerical attributes are proc-

essed. Many practical DTs—including C4.5—use in partitioning techniques that only look for the best halving of the domain. Recently, however, considering multisplits of the domain with more than two intervals has gained popularity. In general, obtained results (DTs, RLs) using discrete features are usually more compact, shorter and more accurate than using continuous ones, hence the results can be more closely examined, compared, used and reused. Supervised discretization techniques might reasonably be expected to lead to more accurate classification trees since the partitions they produce are directly related to the class to be predicted. On the other hand one might expect most of the unsupervised techniques to be considerably faster since they involve little more than sorting the data, an operation which is common to all discretization methods.

Dougherty et al. [12] carried out a comparative study of five discretization procedures using 16 data sets from the UC Irvine ML Database Repository. The methods compared were two unsupervised global methods (equal width interval procedures), two supervised global methods (1RD [21] and entropy minimisation [15]), and C4.5 which is a supervised local method. In all cases the tree was actually constructed by C4.5 but in the four global methods the data was pre-processed using the corresponding procedure to discretize all continuous variables. Somewhat surprisingly Dougherty et al [12] found only small differences, most of which were not statistically significant, between the classification accuracies achieved by resulting DTs. None produced the highest accuracy for all data sets. In particular the local supervised method, C4.5, showed no advantage over other supervised methods: Fayyad & Irani's method [15] achieved the best overall results.

Although there exist a number of efficient inference algorithms and implementations for probabilistic reasoning in BNs with discrete variables, few algorithms support efficient inference in hybrid BNs, BNs where continuous and discrete variables are intermixed. Exact probabilistic inference in hybrid networks can be reduced to taking multidimensional integrals in the same way that exact inference in discrete networks can be reduced to computing sums. However, computing integrals exactly is possible only for a restricted class of continuous functions. Kozlov and Koller [25] constructed an iterative anytime algorithm for BNs that gradually improves the quality of the discretization and the accuracy of the answer on a query.

For NB, one common discretization approach is fixed k-interval discretization (FKID). Another popular one is Fayyad and Irani's entropy minimization heuristic discretization (FID) [15]. Two recent alternatives are lazy discretization (LD) [22] and proportional k-interval discretization (PKID) [34]. LD [22] delays probability estimation until classification time. It waits until a test instance is presented to determine the cut points for each numeric attribute. For a numeric attribute value from the test instance, it selects a pair of cut points such that the value is in the middle of its corresponding interval whose size is the same as created by FKID with $k = 10$. LD tends to have high memory and computational requirements because of its lazy methodology. PKID [34] adjusts discretization bias and variance by tuning the interval size and number, and further adjusts the probability estimation bias and variance of NB classifiers to achieve lower classification error. The larger the interval $(a_i, b_i]$, the more instances in it, the lower the discretization variance, hence the lower the variance of NB' probability estimation. However, the larger the interval, the less distin-

guishing information is obtained about the particular value x_i of attribute X_i, the higher the discretization bias, hence the higher the bias of the probability estimation. So, increasing interval size (decreasing interval number) will decrease variance but increase bias. Conversely, decreasing interval size (increasing interval number) will decrease bias but increase variance. PKID aims to resolve this conflict by adjusting the interval size and number proportional to the number of training instances.

5 Conclusion

A number of ML algorithms can be applied only to data described by discrete numerical or nominal attributes (features). In the case of continuous attributes, there is a need for a discretization algorithm that transforms continuous attributes into discrete ones. Continuous-valued attributes are transformed into nominal ones by splitting the range of the attribute values in a finite number of intervals.
Since a large number of possible attribute values slows and makes inductive learning ineffective, one of the main goals of a discretization algorithm is to significantly reduce the number of discrete intervals of a continuous attribute. At the same time, the algorithm should maximize the interdependency between discrete attribute values and class labels, as this minimizes the information loss due to discretization. A satisfactory trade off between these two goals needs to be achieved. Many studies show induction tasks can benefit from discretization: rules with discrete values are normally shorter and more understandable and discretization can lead to improved predictive accuracy. In other words, the improvement can be measured in three aspects through before/after discretization comparison: (1) accuracy, (2) time for discretization and for learning, and (3) understandability of the learning result. There are numerous discretization methods available in the literature. Most of discretization algorithms perform an iterative greedy heuristic search in the space of candidate discretizations, using different types of scoring functions for evaluating a discretization. In this article, we presented a survey of discretization methods and discussed various dimensions in which these methods can be categorized. The key question is not whether a discretization method is superior to others, but under which conditions a particular method can significantly outperform others on a given problem.

References

[1] Aguilar, J., Bacardit, J., Divina, F. Experimental Evaluation of Discretization Schemes for Rule Induction. Genetic and Evolutionary Computation Conference (GECCO04). Lecture Notes in Computer Science 3102, 828-839

[2] Bacardit, J. Garrell J.M., Analysis and improvements of the Adaptive Discretization Intervals knowledge representation. Genetic and Evolutionary Computation Conference (GECCO04). Lecture Notes in Computer Science 3103, 726-738

[3] Bacardit, J. Garrell, J.M.. Evolving multiple discretizations with adaptive intervals for a Pittsburgh Rule-Based Learning Classifier System. Genetic and Evolutionary Computation Conference (GECCO03). Lecture Notes in Computer Science 2724, 1818-1831

[4] Bertelsen, R.,&Martinez, T. R. (1994). Extending ID3 through discretization of continuous inputs. In Proceedings of the 7th Florida Artificial Intelligence Research Symposium (pp. 122–125). Florida AI Research Society.
[5] Boulle, M., Khiops: A Statistical Discretization Method of Continuous Attributes. Machine Learning 55:1 (2004) 53-69
[6] Burdsall, B., & Giraud-Carrier, C. (1997). Evolving fuzzy prototypes for efficient data clustering. In Proceedings of the Second International ICSC Symposium on Fuzzy Logic and Applications (ISFL'97) (pp. 217–223).
[7] Cerquides, J. Lopez, R., Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method. III International Conference on Knowledge Discovery and Data Mining (KDDM97). Newport Beach (California USA, 1997) 139-142
[8] Chan, C.-C., Batur, C., and Srinivasan, A. 1991. Determination of quantization intervals in rule based model for dynamic. In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. Charlottesvile, Virginia, pp. 1719–1723.
[9] 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 (1995) 641-651
[10] Divina, F., Keijzer, M., Marchiori, E.: A method for handling numerical attributes in GA-based inductive concept learners. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Springer (2003) 898–908
[11] Divina, F., Marchiori, E., Handling continuous attributes in an evolutionary inductive learner. IEEE Transactions on Evolutionary Computation 9:1 (2005) 31-43
[12] Dougherty J, Kohavi R, Sahami M. Supervised and unsupervised discretization of continuous features. In: Proceedings of the 12th international conference on machine learning. San Francisco: Morgan Kaufmann; 1995. p. 194–202.
[13] Elomaa, T. Rousu, J., Fast Minimum Training Error Discretization. XIX International Conference on Machine Learning (ICML02). Sydney (Australia, 2002) 131-138
[14] Elomaa, T. Rousu, J., Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. Data Mining and Knowledge Discovery 8:2 (2004) 97-126
[15] Fayyad, U.M., Irani, K.B., Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. XIII International Joint Conference on Artificial Intelligence (IJCAI93). Chambéry (France, 1993) 1022-1029
[16] Ferrandiz, S., Boull, M., Multivariate discretization by recursive supervised bipartition of graph. IV International Conference in Machine Learning and Data Mining in Pattern Recognition (MLDM05). Lecture Notes in Computer Science 3587, 253-264
[17] Giráldez, R., Aguilar, J.S., Riquelme J.C., Natural Coding: A More Efficient Representation for Evolutionary Learning. Genetic and Evolutionary Computation Conference (GECCO03). Lecture Notes in Computer Science 2723, 979-990
[18] Giráldez, R., Aguilar, J.S., Riquelme J.C., Ferrer, J.F., Rodríguez, D.S., Discretization Oriented to Decision Rule Generation. VIII Intern. Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES02). Creme (Italy, 2002) 275-279
[19] Grzymala-Busse, J.W., Three strategies to rule induction from data with numerical attributes. Lecture Notes in Computer Science 3135 (2004) 54-62
[20] Ho, K.M., Scott, P.D., Zeta: A Global Method for Discretization of Continuous Variables. III Inter. Conference on Knowledge Discovery and Data Mining. USA (1997) 191-194
[21] Holte, R.C., Very simple classification rules perform well on most commonly used datasets. Machine Learning 11 (1993) 63-91
[22] Hsu, C.-N., Huang, H.-J., & Wong, T.-T. (2000). Why discretization works for naive Bayesian classifiers. 17th International Conference (ICML'2000) pp. 309–406.

[23] Janssens, D., Brijs, T., Vanhoof, K., Wets, G., Evaluating the performance of cost-based discretization versus entropy- and error-based discretization. Computers and Operations Research 33:11 (2006) 3107-3123
[24] Kerber, R., ChiMerge: Discretization of Numeric Attributes. X National Conf. on Artificial Intelligence American Association (AAAI92). USA, 1992, 123-128
[25] Kozlov, A.V., Koller, D., Nonuniform Dynamic Discretization in Hybrid Networks. Thirteenth Annual Conference on Uncertainty in AI (UAI97). USA, 1997, 314-325
[26] Kurgan, L.A., Cios, K.J., CAIM Discretization Algorithm. IEEE Transactions on Knowledge and Data Engineering 16:2 (2004) 145-153
[27] Kwedlo, W., Kretowski, M.: An evolutionary algorithm using multivariate discretization for decision rule induction. In Proc. of the European Conference on Principles of Data Mining and Knowledge Discovery (1999) 392–397
[28] Liu, H., Hussain, F., Lim, C., Dash, M., Discretization: An Enabling Technique. Data Mining and Knowledge Discovery 6:4 (2002) 393-423
[29] Liu, H. and Setiono, Chi2: Feature selection and discretization of numeric attributes. VII IEEE Intern. Conf. on Tools with Artificial Intelligence (ICTAI95). USA, 1995, 388-391
[30] Liu, H. and Setiono, R. 1997. Feature selection and discretization. IEEE Transactios on Knowledge and Data Engineering, 9:1–4.
[31] Maass W. Efficient agnostic PAC-learning with simple hypotheses.Seventh annual ACM conference on computational learning theory. NewYork: ACM Press; 1994. p. 67–75.
[32] Steck, H., Jaakkola, T., Predictive discretization during model selection. XXVI Symposium In Pattern Recognition (DAGM04). Lecture Notes in Computer Science 3175, Springer-Verlag 2004, Tbingen (Germany, 2004) 1-8
[33] Wang, K., Liu, B., Concurrent Discretization of Multiple Attributes. 5th Pacific Rim International Conference on Topics in Artificial Intelligence (PRICAI98). Lecture Notes in Computer Science 1531, Springer-Verlag 1998, Singapore (Singapore, 1998) 250-259
[34] Yang, Y., & Webb, Non-Disjoint Discretization for Naive-Bayes Classifiers. XIX International Conference on Machine Learning (ICML02). Sydney (Australia, 2002) 666-673
[35] Yang, Y., & Webb, G. I. (2001). Proportional k interval discretization for naive-Bayes classifiers. 12th European Conf. on Machine Learning (ECML'01) pp. 564–575.

Biography

S. Kotsiantis received a diploma in mathematics, a Master and a Ph.D. degree in computer science from the University of Patras, Greece. His research interests are in the field of data mining and machine learning. He has more than 50 publications to his credit in international journals and conferences.	**D. Kanellopoulos** received a diploma in electrical engineering and a Ph.D. degree in electrical and computer engineering from the University of Patras, Greece. He has more than 40 publications to his credit in international journals and conferences.