DISCRETIZATION

Rahul Yadav
Computer Science Department
Dronacharya College of Engineering , MDU, Gurgaon, Haryana

Abstract- **Discretization is an important preprocessing technique used in many knowledge discovery and data mining tasks. Its main goal is to transform a set of continuous aspects into discrete ones, by associating categorical values to intervals and thus transforming numerical data into qualitative data. In this manner, symbolic data mining algorithms can be applied over continuous data and the representation of information is simplified, making it more brief and specific. This paper objective are to study terms and symbolisations related to discretization, a typical discretization process and discretization current status. We have also discussed the future research for discretization.**

Index Terms- **Discretization, continuous aspects, symbolisation, research**

I. INTRODUCTION

Data usually comes in a mixed format: nominal, discrete, and/or continuous. Discrete and continuous data are ordinal data types with orders among the values, while nominal values do not possess any order amongst them. Discrete values are intervals in a continuous spectrum of values. While the number of continuous values for an aspect can be infinitely many, the number of discrete values is often few or finite. The two types of values make a difference in learning classification trees/rules. One example of decision tree induction can further illustrate the difference between the two data types. When a decision tree is induced, one feature is chosen to branch on its values. With the coexistence of continuous and discrete features, normally, a continuous feature will be chosen as it has more values than features of other types do. By choosing a continuous feature, the next level of a tree can quickly reach a "pure" state—with all instances in a child/leaf node belonging to one class. In many cases, this is equivalent to a table-lookup along one dimension which leads to poor performance of a classifier. Therefore it is certainly not wise to use continuous values to split a node. There is a need to discretize continuous features either before the decision tree induction or during the process of tree building. Widely used systems such as C4.5 (Quinlan, 1993) and CART(Breimanetal.,1984)organise various ways to avoid using continuous values directly. There are many other advantages of using discrete values over continuous ones. Discrete features are closer to a knowledge-level representation (Simon, 1981) than continuous ones. Data can also be reduced and simplified through discretization. For both users and experts, discrete features are easier to understand, use, and explain. As reported in a study(Dougherty et al., 1995), discretization makes learning more accurate and faster. In general, obtained results (decision trees, induction rules) 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. In addition to the many advantages of having discrete data over continuous one, a suite of classification learning algorithms can only deal with discrete data. Discretization is a process of quantizing continuous attributes. The successes of discretization can significantly extend the borders of many learning algorithms.

II. TERMS AND SYMBOLISATION

Feature: "Feature" or "Attribute" or "Variable" refers to an aspect of the data. Usually before collecting data, features are specified or chosen. Features can be discrete, continuous, or nominal. In this paper we are interested in the process of discretizing continuous features. Hereafter M stands for the number of features in the data.

Instance: "Instance" or "Tuple" or "Record" or "Data point" refers to a single collection of feature values for all features. A set of instances makes a data set. Usually a data set is in a matrix form where a row

corresponds to an instance and a column corresponds to a feature. Here afterwards N is the number of instances in the data.

Cut-point: The term "cut-point" refers to a real value within the range of continuous values that divides the range into two breaks, one interval is less than or equal to the cut- point 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.

Arity: 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 of classification and other task. A higher arity can make the understanding of a feature more difficult while a very low arity may affect predictive accuracy negatively.

III. CURRENT STATUS

In earlier days simple techniques were used such as equal-width and equal-frequency (or, a form of binning) to discretize. As the need for accurate and efficient classification grew, the technology for discretization develops rapidly. Over the years, many discretization procedures have been proposed and tested to show that discretization has the probable to reduce the amount of data while retaining or even improving predictive accuracy. Discretization methods have been developed along different lines due to different needs: supervised vs. unsupervised, dynamic vs. static, global vs. local, splitting (top-down) vs. merging (bottom- up), and direct vs. incremental. As we know, data can be controlled or unsupervised depending on whether it has class information. Likewise, supervised discretization considers class information while unsupervised discretization does not; unsupervised discretization is seen in earlier methods like equal-width and equal-frequency. In the unsupervised methods, continuous ranges are divided into subranges by the user specified width (range of values) or frequency (number of occurrences in each interval).This may not give good results in cases where the distribution of the continuous values is not uniform. Furthermore it is vulnerable to outliers as they affect the ranges significantly (Catlett, 1991).To overcome this shortcoming, controlled discretization methods were introduced and class information is used to find the proper intervals caused by cut-points. Different methods have been devised to use this class information for finding meaningful intervals in continuous attributes.

Supervised and unsupervised discretization have their different uses. If no class information is available, unsupervised discretization is the sole choice. There are not many unsupervised methods available in the works which may be attributed to the fact that discretization is commonly associated with the classification task. One can also view the usage of discretization methods as dynamic or static. A dynamic method would discretize continuous values when a classifier is being built, such as in C4.5 (Quinlan, 1993) while in the static approach discretization is done prior to the classification task. There is a comparison between dynamic and static methods in Dougherty et al. (1995). The authors reported mixed presentation when C4.5 was tested with and without discretized features (static vs. dynamic).

Another contrast is local vs. global. A local method would discretize in a localized region of the instance space (i.e. a subset of instances) while a global discretization method uses the entire instance space to discretize (Chmielewski and Grzymala-Busse, 1994). So, a local method is usually associated with a dynamic discretization method in which only a region of instance space is used for discretization.

Discretization methods can also be grouped in terms of top-down or bottom-up. Top-down methods start with an empty list of cut-points (or split-points) and keep on adding new ones to the list by 'splitting' breaks as the discretization progresses. Bottom-up methods start with the complete list of all the continuous values of the feature as cut-points and remove some of them by 'merging' intervals as the discretization progresses.

Another measurement of discretization methods is direct vs. incremental. Direct methods divide the range of k intervals simultaneously (i.e., equal-width and equal-frequency), needing an additional input from the user to control 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 (Cerquides and Mantaras, 1997).

As shown above, there are many discretization methods and many different dimensions to group them. A user of discretization often finds it difficult to choose a suitable technique for the data on hand. There have been a few attempts (Dougherty et al., 1995; Cerquides and Mantaras, 1997) to help improve the difficulty. We carry on with this key objective to make a comprehensive study that includes the definition of a discretization process, performance measures, and extensive comparison. Contributions of this work are:

1. An abstract description of a typical discretization process,
2. A new hierarchical framework to categorize existing discretization methods in the literature,
3. A methodical demonstration of different results by various discretization methods using a standard data set,
4. A comparison of nine representative discretization methods chosen from the framework along two dimensions: times and error rates of a learning algorithm for classification over publically available benchmark data sets,
5. Detailed examination of comparative results, and
6. Some guidelines as to which method to use under different circumstances, and directions for future research and development.

IV. TYPICAL DISCRETIZATION PROCESS

By "typical" we mean univariate discretization. Discretization can be univariate or multivariate. Univariate discretization quantifies one continuous feature at a time while multivariate discretization considers simultaneously multiple features. We mainly consider univariate discretization throughout this paper and discuss more about multivariate discretization briefly at the end as a postponement of univariate discretization.

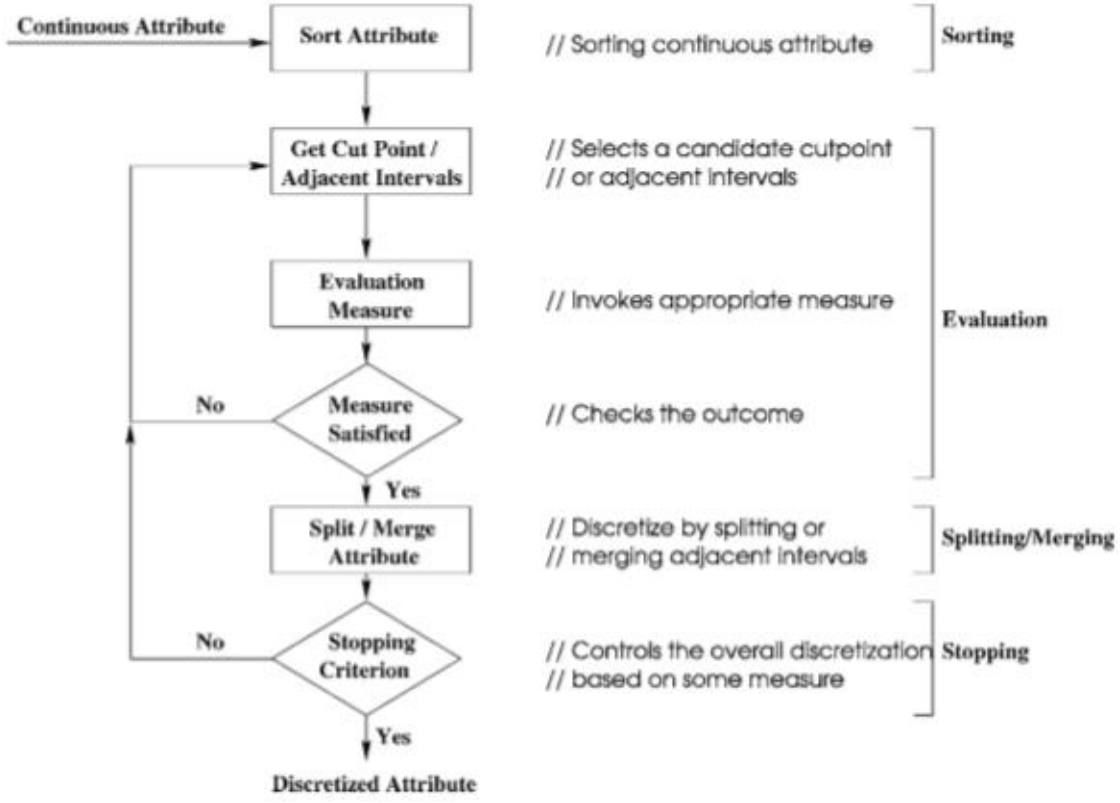

Figure : Discretization process.

A typical discretization process broadly consists of four steps (seen in figure above):
(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.

Sorting: The continuous value for a feature is sorted in either descending or ascending order. Sorting can be computationally very expensive if care is not taken in executing it with discretization. It is important to speed up the discretization process by selecting suitable sorting algorithms. Many sorting algorithms can be found in classic data structures and algorithms books. Among these "Quick-sort" is an efficient sorting algorithm with a time complexity of O(N log N).
Another way to improve efficiency is to avoid sorting a feature's values repetitively. If sorting is done once and for all at the beginning of discretization, it is a global treatment and can be applied when the entire instance space is used for discretization. If sorting is done at each iteration of a process, it is a native treatment in which only a region of entire instance space is considered for discretization.
Choosing a cut-point: After sorting, the next step in the discretization process is to find the best "cut-point" to riven a range of continuous values or the best pair of adjacent intervals to merge. One classic 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 arithmetical measures.
Splitting/merging: As we know, in a top-down approach, intervals are split while for a bottom-up approach intervals are merged. For splitting it is required to evaluate 'cut- points' and to choose the best one and split the range of continuous values into two partitions. Discretization continues with each part (increased by one) until a stopping standard is satisfied. Similarly for merging, adjacent intervals are evaluated to find the best pair of intervals to merge in each iteration. Discretization continues with the reduced number (decreased by one) of intervals until the stopping standard is satisfied.

Stopping criteria: A stopping criterion specifies when to stop the discretization process. It is usually ruled 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. We may consider k to be an upper bound for the arity of the subsequent discretization. In practice the upper bound k is set much less than N, assuming there is no repetition of continuous value for a feature. A stopping criterion can be very simple such as fixing the number of intervals at the beginning or a more complex one like assessing a function. We describe different stopping criteria in the next section.

V. FUTURE RESEARCH

Every discretization method takes it for arranged that each feature independently determines the class. Therefore, all these methods are univariate methods for the sake of efficiency. As we know, the assumption may not be valid. When we discretize, we may need to consider multiple features at a time, the so-called multivariate discretization. Doing so would inevitably increase time difficulty for discretization. With the availability of more powerful parallel computers or computer clusters, we may examine the possibility of using these computers for multivariate discretization. Parallel discretization algorithms are surely welcome when a large number of continuous features should be quantized. Can we extend the methods here to parallelized versions? With the feature independence supposition, it seems practical. Noise handling is another important issue of discretization in practice. To allow a certain degree of tolerance via thresholding is a common practice for noise handling in the discretization methods.

REFERENCES

[1] Huan Liu, Farhad Hussain, Chew Limtan, Manoranjan Das. "Discretization: An Enabling Technique"
[2] Salvador Garcıa, Julian Luengo, Jose Antonio Saez, Victoria Lopez, and Francisco Herrera. "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning"
[3] Fayyad, U. and Irani, K. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In Proc. Thirteenth International Joint Conference on Artificial

Intelligence. San Mateo, CA: Morgan Kaufmann. 1022–1027.

[4] http://en.wikipedia.org/wiki/Discretization

[5] Pfahringer, B. 1995a. Compression-based discretization of continuous attributes. In Proc. Twelfth International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann, pp. 456–463.

[6] Liu, H. and Setiono, R. 1997. Feature selection and discretization. IEEE Transactios on Knowledge and Data Engineering, 9:1–4.