Discretization for Data Mining

Ying Yang
Monash University, Australia

Geoffrey I. Webb
Monash University, Australia

INTRODUCTION

Discretization is a process that transforms quantitative data into qualitative data. Quantitative data are commonly involved in data mining applications. However, many learning algorithms are designed primarily to handle qualitative data. Even for algorithms that can directly deal with quantitative data, learning is often less efficient and less effective. Hence research on discretization has long been active in data mining.

BACKGROUND

Many data mining systems work best with *qualitative* data, where the data values are discrete descriptive terms such as *young* and *old*. However, lots of data are *quantitative*, for example, with age being represented by a numeric value rather a small number of descriptors. One way to apply existing qualitative systems to such quantitative data is to transform the data.

Discretization is a process that transforms data containing a quantitative attribute so that the attribute in question is replaced by a qualitative attribute. A many to one *mapping function* is created so that each value of the original quantitative attribute is mapped onto a value of the new qualitative attribute. First, discretization divides the value range of the quantitative attribute into a finite number of intervals. The mapping function associates all of the quantitative values in a single interval to a single qualitative value. A *cut point* is a value of the quantitative attribute where a mapping function locates an interval boundary. For example, a quantitative attribute recording age might be mapped onto a new qualitative age attribute with three values, *pre-teen*, *teenage*, and *post-teen*. The cut points for such a discretization may be 13 and 18. Values of the original quantitative age attribute that are below 13 might get mapped onto the *pre-teen* value of the new attribute, values from 13 to 18 onto *teen*, and values above 18 onto *post-teen*.

Various discretization methods have been proposed. Diverse taxonomies exist in literature to categorize discretization methods. These taxonomies are complementary, each relating to a different dimension along which discretization methods may differ. Typically, discretization methods can be either *primary* or *composite*. Primary methods accomplish discretization without reference to any other discretization method. Composite methods are built on top of some primary method(s).

Primary methods can be classified as per the following taxonomies.

- **Supervised vs. Unsupervised (Dougherty, Kohavi, & Sahami, 1995):** Supervised methods are only applicable when mining data that are divided into classes. These methods refer to the class information when selecting discretization cut points. Unsupervised methods do not use the class information. For example, when trying to predict whether a customer will be profitable, the data might be divided into two classes *profitable* and *unprofitable*. A supervised discretization technique would take account of how useful was the selected cut point for identifying whether a customer was profitable. An unsupervised technique would not. Supervised methods can be further characterized as *error-based*, *entropy-based* or *statistics-based*. Error-based methods apply a learner to the transformed data and select the intervals that minimize error on the training data. In contrast, entropy-based and statistics-based methods assess respectively the class entropy or some other statistic regarding the relationship between the intervals and the class.
- **Parametric vs. Non-Parametric:** Parametric discretization requires the user to specify parameters for each discretization performed. An example of such a parameter is the maximum number of intervals to be formed. Non-parametric discretization does not utilize user-specified parameters.
- **Hierarchical vs. Non-Hierarchical:** Hierarchical discretization utilizes an incremental process to select cut points. This creates an implicit hierarchy over the value range. Hierarchical discretization can be further characterized as either *split* or *merge* (Kerber, 1992). Split discretization starts with a single interval that

encompasses the entire value range, then repeatedly splits it into sub-intervals until some stopping criterion is satisfied. Merge discretization starts with each value in a separate interval, then repeatedly merges adjacent intervals until a stopping criterion is met. It is possible to combine both split and merge techniques. For example, initial intervals may be formed by splitting. A merge process is then applied to post-process these initial intervals. Non-hierarchical discretization creates intervals without forming a hierarchy. For example, many methods forming the intervals sequentially in a single scan through the data.

- **Univariate vs. Multivariate (Bay, 2000):** Univariate methods discretize an attribute without reference to attributes other than the class. In contrast, multivariate methods consider relationships among attributes during discretization.
- **Disjoint vs. Non-Disjoint (Yang & Webb, 2002):** Disjoint methods discretize the value range of an attribute into intervals that do not overlap. Non-disjoint methods allow overlap between intervals.
- **Global vs. Local (Dougherty, Kohavi, & Sahami, 1995):** Global methods create a single mapping function that is applied throughout a given classification task. Local methods allow different mapping functions for a single attribute in different classification contexts. For example, decision tree learning may discretize a single attribute into different intervals at different nodes of a tree (Quinlan, 1993). Global techniques are more efficient, because one discretization is used throughout the entire data mining process, but local techniques may result in the discovery of more useful cut points.
- **Eager vs. Lazy (Hsu, Huang, & Wong, 2000, 2003):** Eager methods generate the mapping function *prior* to classification time. Lazy methods generate the mapping function as it is needed during classification time.
- **Ordinal vs. Nominal:** Ordinal discretization forms a mapping function from quantitative to ordinal qualitative data. It seeks to retain ordering information implicit in quantitative attributes. In contrast, nominal discretization forms a mapping function from quantitative to nominal qualitative data, thereby discarding any ordering information. For example, if the value range 0 – 29 were discretized into three intervals 0 – 9, 10 – 19 and 20 – 29, if the intervals are treated as nominal then a value in the interval 0 – 9 will be treated as dissimilar to one in 20 – 29 as it is to one in 10 – 19. In contrast, while ordinal discretization will treat the difference between 9 and either 10 or 19 as equivalent, it retains the information that this difference is less than the difference between 9 and 29.
- **Fuzzy vs. Non-fuzzy (Ishibuchi, Yamamoto, & Nakashima, 2001; Wu, 1999):** Fuzzy discretization creates a fuzzy mapping function. A value may belong to multiple intervals, each with varying degrees of strength. Non-fuzzy discretization forms exact cut points.

Composite methods first generate a mapping function using an initial primary method. They then use other primary methods to adjust the initial cut points.

MAIN THRUST

The main thrust of this chapter deals with how to select a discretization method. This issue is particularly important since there exist a large number of discretization methods and no one can be universally optimal. When selecting between discretization methods it is critical to take account of the learning context, in particular, of the learning algorithm, the nature of the data, and the learning objectives. Different learning contexts have different characteristics and hence have different requirements for discretization. It is unrealistic to pursue a universally optimal discretization approach that can be blind to its learning context.

Many discretization techniques have been developed primarily in the context of a specific type of learning algorithm, such as decision tree learning, decision rule learning, naive-Bayes learning, Bayes network learning, clustering, and association learning. Different types of learning have different characteristics and hence require different strategies of discretization.

For example, decision tree learners can suffer from the fragmentation problem. If an attribute has many values, a split on this attribute will result in many branches, each of which receives relatively few training instances, making it difficult to select appropriate subsequent tests. Hence they may benefit more than other learners from discretization that results in few intervals. Decision rule learners may require pure intervals (containing instances dominated by a single class), while probabilistic learners such as naive-Bayes do not. The relations between attributes are key themes for association learning, and hence multivariate discretization that can capture the inter-dependencies among attributes is desirable. If coupled with lazy discretization, lazy learners can further save training effort. Non-disjoint discretization is not applicable if the learning algorithm, such as decision tree learning, requires disjoint attribute values.

In order to facilitate understanding this issue, we contrast discretization strategies in two popular learning