Discretization: An Enabling Technique

F. Hussain, H. Liu, C. L. Tan, and M. Dash
School of Computing, National University of Singapore, Singapore.
{farhad, liuh, tancl, manoranj}@comp.nus.edu.sg

Abstract

Discrete values have important roles in data mining and knowledge discovery. They are about intervals of numbers which are more concise to represent and specify, easier to use and comprehend as they are closer to a knowledge-level representation than continuous values. 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. Furthermore, many induction algorithms found in the literature require discrete features. All these prompt researchers and practitioners to discretize continuous features before or during a machine learning or data mining task. There are numerous discretization methods available in the literature. It is time for us to examine these seemingly different methods for discretization and find out how different they really are, what are the key components of a discretization process, how we can improve the current level of research for new development as well as the use of existing methods. This paper aims at a systematic study of discretization methods with their history of development, effect on classification, and trade-off between speed and accuracy. Contributions of this paper are an abstract description summarizing existing discretization methods, a hierarchical framework to categorize the existing methods and pave the way for further development, concise discussions of representative discretization methods, extensive experiments and their analysis, and some guidelines as to how to choose a discretization method under various circumstances. We also identify some issues yet to solve and future research for discretization.
Keywords: *Discretization, Continuous Feature, Data Mining, Classification.*

1 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 attribute 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 tantamount 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 [35] and CART [2] deploy 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 [40] 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 [11], 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 success of discretization can significantly extend the borders of many learning algorithms.

This paper is about reviewing existing discretization methods, standardizing the discretization process, summarizing them with an abstract framework, providing a convenient reference for future research and development. The remainder of the paper is organized as follows. In the next section, we summarize the current status of discretization methods. In Section 3, we provide a unified vocabulary for discussing various methods introduced by many authors, define a general process of discretization, and discuss different ways of evaluating discretization results. In Section 4, we propose a new hierarchical framework for discretization methods; describe representative methods concisely; while describing a representative method, we also provide its discretization results for a commonly used benchmark data set (Iris). Section 5 shows the results of comparative experiments among various methods and some analysis. The paper concludes in Section 6 with guidelines of choosing a discretization method and further work.

2 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 algorithms have been proposed and tested to show that discretization has the potential 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 **supervised** 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 instances 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 [4]. To overcome this shortcoming, supervised 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 literature 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 [35] while in the static approach discretization is done prior to the classification task. There is a comparison between dynamic and static methods in [11]. The authors reported mixed performance when C4.5 was tested with and without discretized features (static *vs.* dynamic).

Another dichotomy 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 [8]. 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' intervals 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 dimension 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 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 [5].

As shown above, there are numerous discretization methods and many different dimensions to group them. A user of discretization often finds it difficult to choose a suitable method for the data on hand. There have been a few attempts [11, 5] to help alleviate 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 systematic demonstration of different results by various discretization methods using a benchmark 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.

3 Discretization Process

We first clarify some terms used in different works followed by an abstract description of a typical discretization process.

3.1 Terms and Notations

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. Hereafter 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 intervals, 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 tasks. A higher arity can make the understanding of an attribute more difficult while a very low arity may affect predictive accuracy negatively.

3.2 A 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 an extension of univariate discretization.

A typical discretization process broadly consists of four steps (seen in Figure 1): (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. In the following we discuss these four steps in more detail.

3.2.1 Sorting

The continuous values for a feature is sorted in either descending or ascending order. Sorting can be computationally very expensive if care is not taken in implementing 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 repeatedly. 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 *local* treatment in which only a region of entire instance space is considered for discretization.

3.2.2 Choosing a cut-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 about these evaluation functions and their various applications is discussed in the following sections.

3.2.3 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

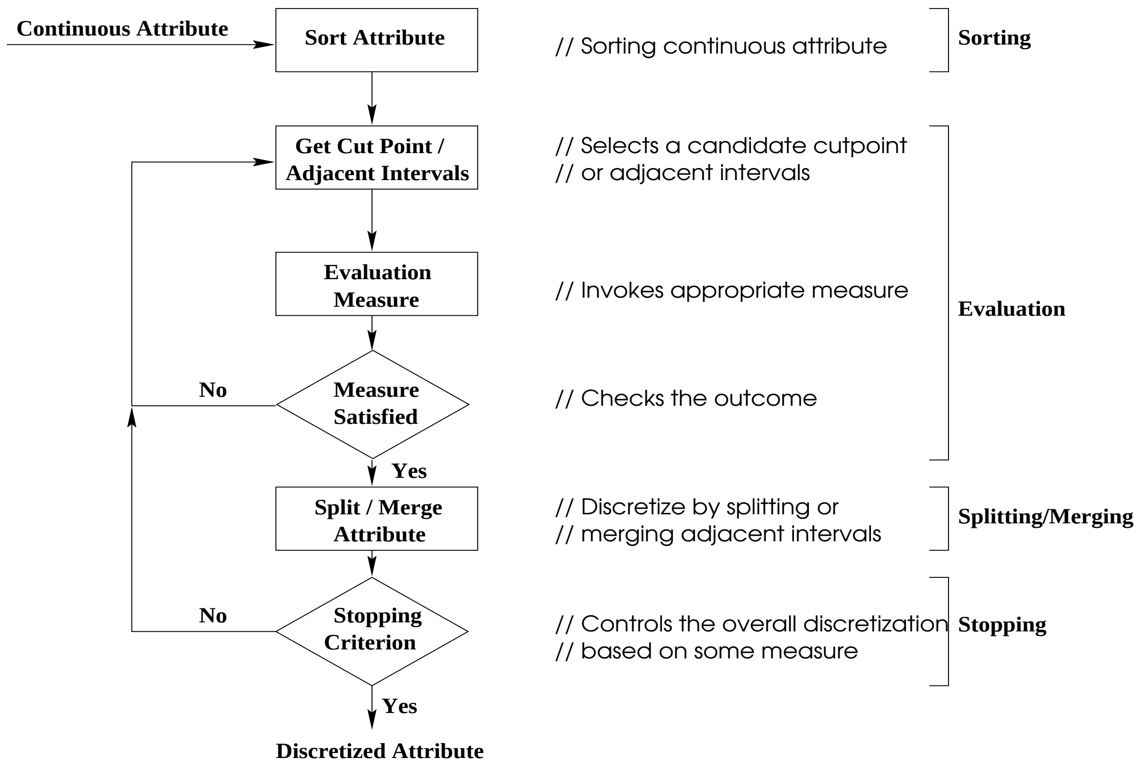


Figure 1: Discretization Process.

with each part (increased by one) until a stopping criterion 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 criterion is satisfied.

3.2.4 Stopping Criteria

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. We may consider k to be an upper bound for the arity of the resulting 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 evaluating a function. We describe different stopping criteria in the next section.

3.3 Result evaluation for discretization

Assuming we have many methods and each provides some kind of discretized data, which discretized data is the best? This seemingly simple question cannot be easily dealt with a simple answer. This is because the result evaluation is a complex issue and depends on a user's need in a particular application. It is complex because the evaluation can be done in many ways. We list three important dimensions: (1) The total number of intervals - intuitively, the fewer the cut-points, the better the discretization result; but there is a limit imposed by the data representation. This leads to the next dimension. (2) 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. If the ultimate goal is to generate a classifier from the data, we should consider yet another perspective. (3) Predictive accuracy - how discretization helps improve accuracy. In short, we need at least three dimensions: simplicity, consistency, and accuracy. Ideally, the best discretization result can score highest in all three departments. In reality, it may not be achievable, or necessary. To provide a balanced view of various discretization methods in terms of these measures is one of the objectives of this paper.

Simplicity is defined by the total number of cut-points. Accuracy is usually obtained by running classifier in cross validation mode. Consistency is defined by having the least inconsistency count which is calculated in three steps: (in the following description a pattern is a set of values for a feature set while an instance is a pattern with a class label) (1) two instances are considered inconsistent if they are the same in their attribute values except for their class labels; for example, we have an inconsistency if there are two instances (0 1, a) and (0 1, $\bar{a}$) - class label is separated by "," - because of different classes a and $\bar{a}$. (2) the inconsistency count for a pattern is the number of times it appears in the data minus the largest number of class label: for example, let us assume there are n instances that match the pattern, among them, c_1 instances belong to label$_1$, c_2 to label$_2$, and c_3 to label$_3$ where $c_1 + c_2 + c_3 = n$. If c_3 is the largest among the three, the inconsistency count is $(n - c_3)$. (3) the total inconsistency count is the sum of all the inconsistency counts for all possible patterns of a feature subset.

4 Discretization Framework

There are numerous discretization methods available in the literature. These methods can be categorized in several dimensions as discussed earlier. We restate them here: dynamic *vs.* static, local *vs.* global, splitting *vs.* merging, direct *vs.* incremental, and supervised *vs.* unsupervised. One can construct different combinations of these dimensions to group the

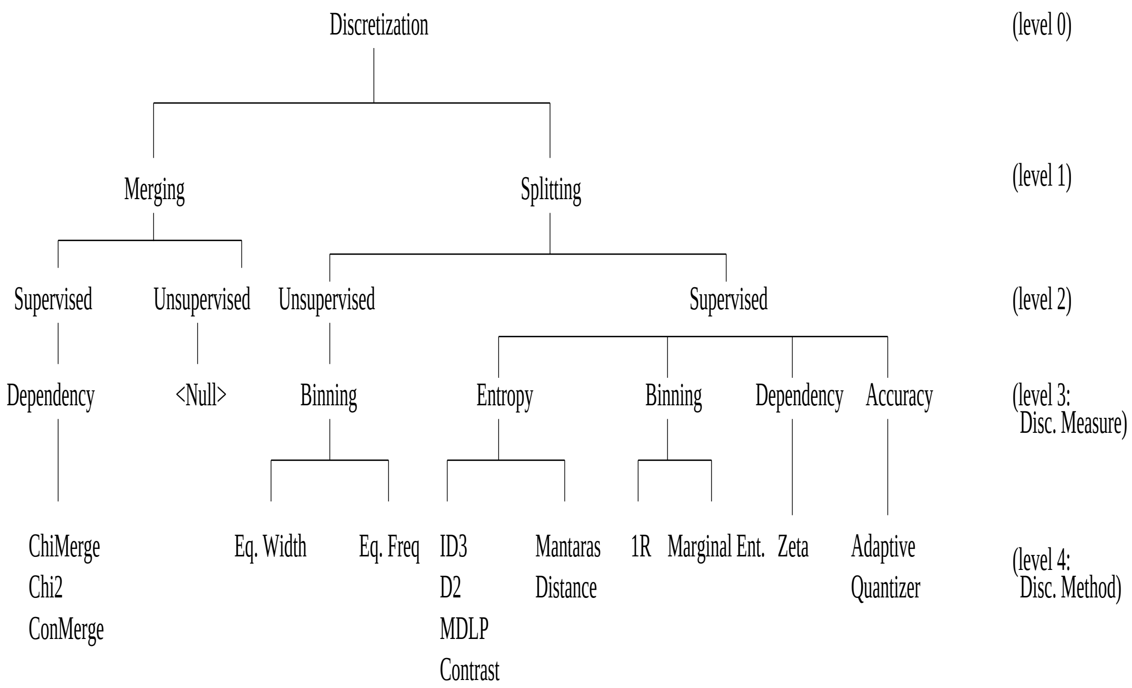


Figure 2: A Hierarchical Framework for Discretization Methods.

methods. But arbitrary combinations will not help in advancing the research of discretization. We wish to create a hierarchical framework that is systematic and expandable, and attempts to cover all existing methods. Each discretization method found in the literature discretizes a feature by either splitting the interval of continuous values or by merging the adjacent intervals. Both splitting and merging categories can further be grouped as supervised or unsupervised depending on whether class information is used. To repeat, supervised discretization methods use the available class information while the unsupervised methods do not.

With these considerations in mind, we propose a hierarchical framework in Figure 2. We describe different discretization measures according to two approaches: splitting and merging (level 1). We then consider whether a method is supervised or unsupervised (level 2). We further group together methods that use similar discretization measures (level 3) eg. binning and entropy. As is suggested in Figure 2, the supervised and unsupervised division determines different (non-overlapping) measures used. Hence this conceptually useful division will not be discussed in detail below. We will discuss the use of various measures under the categories of splitting and merging as shown in the table below (X indicates that a measure is not available in the category. We may find variations of these measures like Mantaras distance [5] which we categorize under entropy measure for their similarity.)

Splitting	**Merging**
binning	X
entropy	X
dependency	dependency
accuracy	X

In the following two subsections, some representative methods are chosen for in depth discussion. Their related or derived measures are also briefly mentioned. For each measure we give: (1) the definition of the measure; (2) its use in some discretization methods; (3) the stopping criteria used; and (4) its discretization results for the Iris data: cut-points for each attribute, how many of them, and how many inconsistencies resulted from discretization. The Iris data is used as an example in illustrating results of different discretization methods. The data is obtained from the UC Irvine data repository [29]. It contains 150 instances with four continuous features and three class labels.

4.1 Splitting methods

We start with a generalized algorithm for splitting discretization methods.

Splitting Algorithm
S = Sorted values of feature f

```
Splitting(S){
    if StoppingCriterion() == SATISFIED
        Return
    T = GetBestSplitPoint(S)
    S_1 = GetLeftPart(S, T)
    S_2 = GetRightPart(S, T)
    Splitting(S_1)
    Splitting(S_2)
}
```

The splitting algorithm above consists of all four steps in the discretization process, they are: (1) *sort* the feature values, (2) *search* for a suitable cut-point, (3) *split* the range of continuous values according to the cut-point, and (4) *stop* when a stopping criterion satisfies otherwise *go* to (2). Many splitting discretization measures are found in the literature: for example, binning [18], entropy [33, 4, 13, 43, 5], dependency [17], and accuracy [6].

4.1.1 Binning

It is the simplest method to discretize a continuous-valued attribute by creating a specified number of bins. The bins can be created by *equal-width* and *equal-frequency*.

Equal width or frequency: In both methods, arity k is used to determine the number of bins. Each bin is associated with a distinct discrete value. In *equal-width*, the continuous range of a feature is evenly divided into intervals that have an equal-width and each interval represents a bin. In *equal-frequency*, an equal number of continuous values are placed in each bin.

The two methods are very simple but are sensitive for a given k. For equal-frequency, for instance, many occurrences of a continuous value could cause the occurrences to be assigned into different bins. One improvement can be after continuous values are assinged into bins, boundaries of every pair of neighboring bins are adjusted so that duplicate values should belong to one bin only. Another problem is outliers that take extreme values. One solution can be to remove the outliers using a threshold.

<u>Stopping criterion:</u> as the number of bins is fixed there is no need for any other stopping criterion.

<u>Results:</u> We discretized the Iris data using equal-width and equal-frequency methods with arity $k = 4$. Thus we obtained three cut-points for each attribute using both methods as shown in the following tables.

Equal-Width

Feature	Cut Points	# Points
F_1	5.2, 6.1, 7.0	3
F_2	2.6, 3.2, 3.8	3
F_3	1.9, 3.9, 5.4	3
F_4	0.6, 1.3, 1.9	3

Equal-Frequency

Feature	Cut Points	# Points
F_1	5.1, 5.8, 6.4	3
F_2	2.8, 3.0, 3.3	3
F_3	1.6, 4.4, 5.1	3
F_4	0.3, 1.3, 1.8	3

As values are not evenly distributed, in most cases, cut-points obtained by the two methods are different.

1R: Binning methods mentioned above do not use class information even if it is available. *1R* [18] is a supervised discretization method using binning. After sorting the continuous values, *1R* divides the range of continuous values into a number of disjoint intervals and adjusts the boundaries based on the class labels associated with the continuous values. Each interval should contain a minimum of 6 instances with the exception for the final interval which would contain the remaining instances not yet grouped in any interval. The adjustment at the boundary does not allow to terminate an interval if the next instance has the same class label as the majority class label seen until then in that interval. This can be made clearer with the following simple example. The first row in the example is the values of a feature after sorting while the second row stands for the class label (R or C).

11	14	15	18	19	20	21	22	23	25	30	31	33	35	36
R	C	C	R	C	R	C	R	C	C	R	C	R	C	R
			C							C				R

1R would form an interval of class C stretching from 11 to 21, another interval of class C from 22 to 35, and the last of class R including just 36. The two leftmost intervals would then be merged, as they predict the same class based on row 3 above. There are 6 misclassifications after discretization.

Stopping criterion: The stopping criterion is indirectly specified by the minimum number of instances each interval should contain. The default value is 6.

Results: Discretization results of the Iris data set are shown below:

Feature	Cut Points	# Points
F_1	4.9, 5.5, 6.1, 6.6, 7.7, 7.9	6
F_2	2.7, 3.2, 4.2, 4.4	4
F_3	1.9, 6.9	2
F_4	0.6, 1.6, 2.5	3

Summary: Equal-width and equal-frequency are simple and easy to implement. This does not come without a price. First, arity k has to be specified beforehand. Because we usually do not know what a proper value k is, we need to resort to trial-and-error or specify a value randomly as we did for the Iris data. Second, even if class information is available, these two measures cannot make use of it. 1R is one way to improve in this regard. It relies on class information to overcome such problems that occur with equal-frequency which forces to put two instances with the same class label in two different intervals. *Maximum* *marginal entropy* [11] is another way to improve equal-frequency by using class information when adjusting boundaries of neighboring bins.

For the Iris data, we provide the discretization results for equal-width, equal-frequency, and 1R. The three methods give different cut-points. Their numbers of inconsistency are also different. The measures (number of cut-points and number of inconsistencies) are summarized in the following table. For the Iris data with $k = 4$, equal-frequency seems the best. For other k values more investigation will be needed.

Methods	Inconsistency	# Points
Equal-width	10	12
Equal-freq	6	12
1R	7	15

4.1.2 Entropy Measure

Entropy is one of the most commonly used discretization measures in the literature. Shannon defines entropy of a sample variable X as [39, 41]:

$$H(X) = -\sum_x p_x \log p_x$$

where x represents a value of X and p_x its estimated probability of occurring. It is the average amount of information per event where information of an event is defined as:

$$I(x) = -\log p_x.$$

Information is high for lower probable events and low otherwise. Hence, entropy H is the highest when each event is equi-probable, i.e., $p_{x_i} = p_{x_j}$ for all i, j; and it is the lowest when $p_x = 1$ for one event and 0 for all other events. Entropy measure is used in various applications. When used in discretization, entropy is usually used in a supervised manner.

ID3 type: ID3 [33] and C4.5 [35] are two popular algorithms for decision tree induction that use entropy measure. They construct an inductive decision tree by selecting a feature if its branching results in the overall minimum entropy at the next layer of the decision tree. A continuous feature has to be discretized to avoid creating too many branches for a node. ID3 employs a greedy search to find the potential cut-points within the existing range of continuous values using the following formula:

$$H = -p_{left} \sum_{j=1}^{m} p_{j,left} \log p_{j,left} - p_{right} \sum_{j=1}^{m} p_{j,right} \log p_{j,right}.$$

In this equation, m is the number of classes, p_{left} and p_{right} are probabilities that an instance is on the left or right side of a cut-point respectively. $p_{j,side}$ denotes the probability that an instance in the *side* (left or right) belongs to class j. The cut-point with the lowest entropy is chosen to split the range into two parts. Splitting continues with the each part until a stopping criterion is satisfied. In fact, it binarizes a range at every split.

Stopping criterion: When every leaf node is pure (all instances in the node belong to one class), it stops. The condition can be relaxed based on the needs.

Results: We do not provide discretization results here for two reasons. One is that classification algorithms like ID3 and C4.5 do discretization (binarization) for continuous features under any circumstances. They do that at each branching node. The other reason is that the cut-points thus obtained are usually only good for the classification algorithm. We discuss this discretization method (binarization) here because it is a base for its many successors such as D2, MDLP.

D2: This discretization method [4] applies entropy measure to find a potential cut-point to split a range of continuous values into two intervals. Unlike ID3 which binarizes a range of values while building a decision tree, D2 is a static method that discretizes the whole instance space. Instead of finding only one cut-point, it recursively binarizes ranges or subranges until a stopping criterion is met [1]. The discretized data is then used for building a classifier. So, D2 is a successor of ID3 discretization. A stopping criterion is essential to avoid over-splitting.

Stopping criterion: The stopping conditions used in D2 can be one of the following:

1. Stop if the number of instances to split is less than 14,
2. Stop if the number of intervals is more than 8,
3. Stop if the information gains on all cut-points are the same, or
4. Stop if all instances in the interval to be split belong to the same class.

One of the problems is that these stopping conditions are rather ad hoc. The next method (MDLP) provides a more principled way of determining when the recursive splitting should stop.

Results: For the Iris data, D2 gave the following results:

Feature	Cut Points	# Points
F_1	4.9, 5, 5.5, 5.6, 5.8, 6.3, 7, 7.9	8
F_2	2.3, 2.5, 2.8, 2.9, 3, 3.3, 3.4, 4.4	8
F_3	1.9, 4.5, 4.8, 5.1, 6.9	5
F_4	0.6, 1.4, 1.5, 1.8, 2.5	5

Entropy (MDLP): In [14], they propose that potential cut-points are those that form boundaries between classes after sorting the continuous feature values. Using the example for 1R, we can see that the number of boundaries is 12 which is less than $(k - 1 = 14)$. Therefore, its efficiency increases as fewer potential cut-points need to be checked.

11	14 15	18	19	20	21	22	23 25	30	31	33	35	36
R	C C	R	C	R	C	R	C C	R	C	R	C	R

However, not all boundaries are needed to serve as cut-points for discretization. A minimum description length principle (MDLP) is used to choose useful cut-points. When the selection is done, the discretization process completes. A generalization of this measure is used for discretizing continuous features while learning Bayesian networks in [15].

Stopping criterion: MDLP is used as stopping criterion. MDLP is usually formulated as a problem of finding the cost of communication between a sender and a receiver. It is assumed that the sender has the entire set of instances while the receiver has the class labels of the instances. The sender needs to convey the proper class labeling of the instances to the receiver. It says that the partition induced by a cut-point for a set of instances is accepted if and only if the cost or length of the message required to send before partition is more than the cost or length of the message required to send after partition. A detailed account how this is done can be found in [14].

Results: Entropy (MDLP) discretized the Iris data as shown below.

Feature	Cut Points	# Points
F_1	5.4, 6.1	2
F_2	4.4	1
F_3	1.9, 4.9, 6.9	3
F_4	0.6, 1.6, 2.5	3

Mantaras distance: Cerquides and Mantaras [5] introduced a distance measure (*Mantaras Distance* [28]) to evaluate the cut-points. Let us consider two partitions P_a and P_b on a range of continuous values, each containing n and m number of classes. The Mantaras distance between two partitions due to a single cut-point is given below.

[1] Without a stopping criterion, it could reach a status that each distinct value is a cut-point.

$Dist(P_a, P_b) = \frac{I(P_a|P_b)+I(P_b|P_a)}{I(P_a \cap P_b)}$

Since,

$$I(P_b|P_a) = I(P_b \cap P_a) - I(P_a)$$
$$Dist(P_a, P_b) = 2 - \frac{I(P_a)+I(P_b)}{I(P_a \cap P_b)}$$

where,

$$I(P_a) = -\sum_{i=1}^{n} P_i \log_2 P_i$$
$$I(P_b) = -\sum_{j=1}^{m} P_j \log_2 P_j$$
$$I(P_a \cap P_b) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} \log_2 P_{ij}$$
$$P_i = \frac{|C_i|}{|N|}$$
$$|C_i| = \text{total count of class } i$$
$$|N| = \text{total number of instances}$$
$$P_{ij} = P_i \times P_j$$

It chooses the cut-point that minimizes the distance.

Stopping criterion: It also uses the minimum description length principle discussed in Entropy (MDLP) as its stopping criterion to determine whether more cut-points should be introduced.

Results: Using this distance, we obtained results for the Iris data below.

Feature	Cut Points	# Points
F_1	5.7, 7.9	2
F_2	3, 4.4	2
F_3	1.9, 4.4, 5.1, 6.9	4
F_4	0.6, 1.3, 1.8, 2.5	4

Summary: ID3 type applies binarization to discretize continuous values while building a tree. D2 does better by separating discretization from tree building and by recursively binarizing ranges and/or subranges. Its problem is that there is no principled way to stop its recursive process of binarization. Entropy (MDLP) uses the minimum description length principle to determine when to stop discretization. It also suggests that potential cut-points are those that separate different class values. Based on the summary of results in the following table for the Iris data, MDLP is an obvious winner as it produces comparatively low inconsistencies and cut-points.

Methods	Inconsistency	# Points
D2	1	26
MDLP	3	9
Mantaras	8	12

Another work using entropy can be found in [43]. A *contrast* measure is introduced that uses the clustering concept to get the cut-point to induce a decision tree. The idea is to search for clusters that are contrasted as much as possible from the instance space proximity point of view. The cut-point for the maximum contrast is chosen. But the cut-point selected must be an informative one, i.e., it is uninformative to select a cut-point that separates instances belonging to the same class. So, an entropy measure is proposed to select cut-points further. As it is mainly suggested in the context of tree building, we stop short of discussing it here and refer the interesting reader to [43] for more details.

4.1.3 Dependency

Zeta: It is a measure of strength of association between the class and a feature. In [17], it is defined as the maximum accuracy achievable when each value of a feature *predicts* a different class value. We use a simple case to illustrate how Zeta is calculated. Assume that there is a continuous feature X with two classes (C_1 and C_2). The task is to find one cut-point (two intervals) at a time with the highest Z (zeta) value.

Feature	X_1	X_2
C_1	n_{11}	n_{12}
C_2	n_{21}	n_{22}

A modal class in interval i is determined as follows:
C_1 is the modal class, if $max(n_{1i}, n_{2i}) = n_{1i}$
C_2 is the modal class, if $max(n_{1i}, n_{2i}) = n_{2i}$
where
i takes a value (1 or 2) if there are two intervals, and
n_{1i} is the number of instances in interval i that belong to C_1.
A Zeta value for a cut-point is:

$$Z = \sum_{i=1}^{k} n_{f(i),i}$$

where
k = number of prespecified intervals (2 by default),
$f(i)$ = a class index that has the highest count of instances in interval i, and
$n_{f(i),i}$ = number of instances in interval i with class index $f(i)$ (modal class index).

For a feature with arity k, there could be $k-1$ potential cut-points. A cut-point with the highest Z value is selected if no neighboring pair of partitions predicts the same class (or two distinct class values should be predicted). This method continues binarizing subranges in the same spirit until the stopping criterion is met. A more complicated version of this method is when $k > 2$ which may incur an intractable discretization process [17]. Therefore, in practice k is set to 2.

Stopping criterion: The process stops when the specified number of intervals is reached for each continuous feature.

Results: The number of final intervals of a feature was specified as 4 (i.e., 3 cut-points). Zeta discretized the Iris data as follows:

Feature	Cut Points	# Points
F_1	5.4, 6.1, 7.9	3
F_2	2.9, 3, 4.4	3
F_3	1.9, 4.7, 6.9	3
F_4	0.6, 1.6, 2.5	3

Method	Inconsistency	# Points
Zeta	3	12

Zeta obtained reasonable discretization results for this data. Clearly, with a different number of final intervals, the results will certainly change. We could also allow different features to have different final intervals if we have some prior knowledge about them.

4.1.4 Accuracy

Accuracy measure usually means the accuracy of a classifier. An example of using accuracy for discretization is *Adaptive Quantizer* [6]. 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.

Stopping criterion: For each attribute, the discretization process stops when there is no improvement in accuracy.

Results: We used C4.5 as the classifier to see the improvement while splitting the attribute values using equal-width.

Feature	Cut Points	# Points
F_1	5.20, 6.10	2
F_2	2.60, 3.20	2
F_3	2.47, 3.95, 5.42	3
F_4	0.70, 1.30, 1.90	3

Method	Inconsistency	# Points
Accuracy	10	10

The choice of a classifier depends on a user's preferences. However, as the classifier need to be trained many times, it is necessary to choose one with small time complexity. For example, C4.5 runs reasonably fast with time complexity $O(N \log N)$ where N is the number of instances.

4.2 Merging methods

We start with a generalized algorithm for discretization methods adopting the *merging* or bottom-up approach.

Merging Algorithm
S = Sorted values of feature f
Merging(S){
  if StoppingCriterion() == SATISFIED
    Return
  T = GetBestAdjacentIntervals(S)
  S = MergeAdjacentIntervals(S, T)
  Merging(S)
}

The above algorithm consists of the four important steps in the discretization process. They are: (1) sorting the values, (2) finding the best two neighboring intervals, (3) merging the pair into one interval, and (4) stop when the chosen stopping criterion satisfies. Methods in this category [21, 25, 44] use the χ^2 statistic as one of the evaluation measures. Hence, we describe the measure first.

χ^2 measure: χ^2 is a statistical measure that conducts a significance test on the relationship between the values of a feature and the class. Kerber [21] 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 χ^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 formula for computing the χ^2 value is:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

where:
p = number of classes,
A_{ij} = number of distinct values in the ith interval, jth class,
R_i = number of examples in ith interval = $\sum_{j=1}^{p} A_{ij}$,
C_j = number of examples in jth class = $\sum_{i=1}^{m} A_{ij}$,
N = total number of examples = $\sum_{j=1}^{p} C_j$ and
E_{ij} = expected frequency of $A_{ij} = (R_i \times C_j)/N$.

ChiMerge: It is a supervised, bottom-up discretization procedure [21]. Initially each distinct value of the attribute is considered to be one interval. χ^2 tests are performed for every pair of adjacent intervals. Adjacent intervals with the least χ^2 value are merged together till the chosen stopping criterion satisfies. A higher value of significance level for χ^2 test causes over discretization while a lower value causes under discretization. The recommended procedure is to set the significance level between 0.90 to 0.99 and have a *max-interval* parameter set to 10 or 15. This *max-interval* parameter can be included to avoid the excessive number of intervals from being created.

Stopping criterion: The merging of adjacent intervals is repeated until χ^2 values of all pairs of adjacent intervals are smaller than a specified threshold value which is determined by a chosen significance level. The parameter *max-interval* is used to impose a constraint that the number of discretized intervals should be less than or equal to *max-interval*.

Results: Discretization results over all four features of the Iris data set are shown below:

Feature	Cut Points	# Points
F_1	4.3, 4.9, 5.0, 5.5, 5.8, 6.3, 7.1	7
F_2	2.0, 2.5, 2.9, 3.0, 3.4	5
F_3	1.0, 3.0, 4.5, 4.8, 5.0, 5.2	6
F_4	0.1, 1.0, 1.8	3

Chi2: It is an automated version of *ChiMerge*. In *Chi2* [26], the statistical significance level keep changing to merge more and more adjacent intervals as long as the inconsistency criterion [25] is satisfied. By inconsistency, it means that two patterns match but belong to different categories. The algorithm not only discretizes the continuous data set but also selects a subset of relevant features. Like ChiMerge, χ^2 statistic is used to discretize the continuous features until some inconsistencies are found in the data. Doing so some features are removed as irrelevant and hence a subset of relevant features is retained that is consistent.

Stopping criterion: The merging continues until the inconsistency is not more than the set limit. The inconsistency limit is 0 by default.

Results: Discretization results over all four features of the Iris data set with 0 inconsistency are shown below:

Feature	Cut Points	# Points
F_1	4.3, 4.9, 5.5, 5.8, 6.1, 7.1	6
F_2	2.0, 2.4, 2.9, 3.2, 3.4, 3.9	6
F_3	1.0, 1.9, 3.0, 3.6, 4.8, 5.2, 5.4	7
F_4	0.1, 1.0, 1.3, 1.8	4

One distinct feature of Chi2 is its capability to remove irrelevant attributes that do not help in classification. If we are allowed to tolerate some inconsistency, it is possible to merge the whole range of some attributes which are completely independent to the class. The following table shows that, by allowing 3% inconsistency, we could merge the first two attributes that are irrelevant.

Feature	Cut Points	# Points
F_1	Merged	0
F_2	Merged	0
F_3	1.0, 3.0, 4.8, 5.2	4
F_4	0.10, 1.00, 1.80	3

A method very similar to Chi2 is *ConMerge* [44] which also uses the χ^2 statistic and the inconsistency measure. Instead of considering one attribute at a time, ConMerge chooses the lowest χ^2 value among the intervals of all continuous features. It requires more dynamic space.

Summary: ChiMerge is one of the first methods that moves away from a splitting approach. It specifically considers the relations between a feature and the class using the χ^2 statistic. The basic idea is that if a merge of two continuous values or two intervals does not affect the differentiation of class values, the merge should be approved. χ^2 statistic allows some noise tolerance. The problem is how to set a proper threshold for each feature. Chi2 suggests to automate the threshold tuning via the inconsistency measure. Another advantage of using Chi2 is to allow noise tolerance which leads to removal of irrelevant features. As seen in the case of allowing 3% inconsistency in the Iris data, the values of its first two features are merged into one value only. This means these two features take a constant value and can thus be removed. The following table summarizes the discretization results. In this particular case, Chi2 with 3% inconsistency yields the best results: 4 inconsistencies and 7 cut-points.

Method	Inconsistency	# Points
ChiMerge	4	21
Chi2	0	23
Chi2(3%)	4	7

4.3 Remarks

We have reviewed representative discretization methods under two categories: splitting and merging. The majority of methods are found in the splitting category. We used the Iris data as an example to show the different results obtained by various discretization methods without resorting to any classification algorithms. An intuitive relation between the two evaluation measures (number of inconsistencies and number of cut-points) is that the more the cut-points, the fewer the inconsistencies. So a good set of results should be the one with low values in both evaluation measures. We indeed observe that some methods did better than others for the Iris data: for example, Entropy (MDLP) was the best in the category of splitting; Chi2 in the category of merging showed that it is possible for a trade-off between the two measures. That is, in addition to discretization, Chi2 can also remove features if some level of inconsistency is allowed.

The proposed framework has served one purpose so far: it helps us organize many discretization methods so that we can describe the methods in groups and observe their performance. Its other aspects are:

1. providing an overall picture of various methods and indicating their relations among them;
2. suggesting what is missing according to this framework and what methods are more similar than others; and
3. offering a starting point to think of new measures or methods; in other words, we hope that this framework makes the design of new methods easier.

If we reorganize the discretization methods according to the five different dimensions reviewed in Section 2, namely, global *vs.* local, supervised *vs.* unsupervised, direct *vs.*

Methods	Global/ Local	Supervised/ Unsupervised	Direct/ Incremental	Splitting/ Merging	Static/ Dynamic
Equal-width	Global	Unsupervised	Direct	Splitting	Static
Equal-frequency	Global	Unsupervised	Direct	Splitting	Static
1R	Global	Supervised	Direct	Splitting	Static
D2	Local	Supervised	Incremental	Splitting	Static
Entropy (MDLP)	Local	Supervised	Incremental	Splitting	Static
Mantaras	Local	Supervised	Incremental	Splitting	Static
ID3	Local	Supervised	Incremental	Splitting	Dynamic
Zeta	Global	Supervised	Direct	Splitting	Static
Accuracy	Global	Supervised	Direct	Splitting	Static
ChiMerge	Global	Supervised	Incremental	Merging	Static
Chi2	Global	Supervised	Incremental	Merging	Static
ConMerge	Global	Supervised	Incremental	Merging	Static

Table 1: Representative discretization methods in multiple dimensions. Double horizontal lines separate different combinations.

incremental, splitting *vs.* merging, and static *vs.* dynamic, we obtain another type of groupings in a multi-dimensional view as seen in Table 1.

5 Experiments and Analysis

Using the Iris data, we clearly see that discretization simplifies data (continuous values are quantized into intervals) without sacrificing the data consistency much (only a few inconsistencies occur after discretization). We are now ready to evaluate the ultimate objective of discretization - whether discretization helps improve the performance of learning and understanding of the learning result. 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. Thus, we need a classification learning algorithm. A critical constraint for choosing such a learning algorithm is that it should be able to run with both data types, i.e., continuous as well as discrete. Not every learning algorithm can do so. Naive Bayes Classifier [10, 22], as an example, can only run on discrete data. C4.5 [35] is chosen for experiments because it can handle both data types and it is conveniently available and widely used so that a reader can easily repeat the experiments here. Furthermore, C4.5 has become a de facto standard for comparison in machine learning. We have reimplemented the discretization methods used in the experiments based on the descriptions of the published papers. *The programs of these discretization algorithms are available through ftp or web access free of charge upon request.* We will examine the effect of discretization on C4.5 through comparisons before/after discretization. Before doing so, we outline the specific meaning of each aspect of evaluation.

- Accuracy - we wish to see if discretization would result in the decrease or increase of accuracy. The usual 10-fold cross validation procedure will be used.
- Time for discretization - we wish to see if a discretization method that takes more time would result in better accuracy.
- Time for learning with different data types of the same data - with which data types the learning requires more time to complete.
- Understandability - the learning result of C4.5 is a decision tree. This aspect is indirectly measured through the number of nodes in a tree.

	Data	Total#Instances	#Features
1	Australian	690	14
2	Breast	699	9
3	Glass	214	10
4	Heart	270	13
5	Vehicle	846	18
6	Iris	150	4
7	Wine	178	13
8	Pima	768	8
9	Bupa	345	6
10	Thyroid	215	5
11	Ionos	351	34

Table 2: Summary of data sets.

5.1 Experiment set-up

Eleven data sets are selected from the UC Irvine machine learning data repository [29] with all numeric features and varying data sizes. A summary of data sets can be found in Table 2. A total of 8 discretization methods excluding ID3 type are chosen to compare according to Table 1 excluding Equal-width, and Accuracy. Accuracy method takes too long time as every selection of a cut-point evokes a decision tree learning. ID3 type is used as the base for comparison, as we use C4.5 (an improved version of ID3) to provide the figures of performance before discretization. Equal-width is similar to Equal-Freq. We choose the latter based on our experience gained in Section 4.

Each experiment is conducted as follows:

- for each data set
  - test each discretization method by 10-fold cross validation of C4.5
    1. take every 9/10 of the data in each round (10 in total)
    2. run a discretization method to get cut-points, measure the time needed
    3. use these cut-points to discretize the rest 1/10 data
    4. use the above 9/10 data for *training* and the rest data for *testing*
  - report the average error rate and time of C4.5.

For the results without discretization, we skip steps 2 and 3 in the above procedure.

5.2 Results and analysis

C4.5 results are shown in Tables 3 and 4. Each result consists of mean and deviation of the reported error rates of 10-fold cross validation. The results of C4.5 without discretization are listed in the column "Continuous" in Tables 3 for comparison with the results after applying various discretization methods. The results are grouped in terms of the characteristics of discretization methods for easy comparison. Averages at the last rows of Tables 3 and 4 give an indication how the discretization methods affect predictive accuracy. Most discretization methods do not significantly increase error rates. In the meantime, we also need to look at other dimensions for performance evaluation.

Table 5 reports the time taken by each discretization method. It is clear that each method takes varying amount of time. This also serves only as an indication because of the flexibility shown in each method's stopping criterion. What is interesting to us is whether the more time spent on discretization for a method, the less error rate for C4.5. Figure 3 suggests that there is such a relation. A similar finding is reported in [17]: we measure the time only taken by the discretization step while they measure the time taken by discretization and by

Data	Continuous	Zeta	ChiMerge	Chi2
Australian	15.28±5.84	15.60±4.13	14.42±5.42	13.50±5.14
Breast	4.72±1.25	13.05±5.83	4.92±2.75	5.01±2.43
Glass	1.86±2.28	2.31±4.20	1.90±2.28	3.20±1.22
Heart	22.16±4.14	16.85±4.66	20.21±4.10	20.00±4.12
Vehicle	26.87±4.57	29.00±3.54	30.87±4.57	33.33±2.13
Iris	4.34±2.84	8.24±6.67	5.02±3.48	4.01±3.32
Wine	6.22±6.84	6.82±6.58	7.92±5.80	6.90±4.04
Pima	26.22±2.65	37.70±5.97	27.31±4.43	26.91±3.12
Bupa	33.13±5.70	34.73±6.45	33.99±8.39	32.09±5.55
Thyroid	8.00±4.31	23.77±9.16	8.91±4.43	9.21±2.22
Ionos	9.14±3.78	11.37±6.29	8.94±4.52	8.52±3.22
Average	14.36	18.13	14.95	14.79

Table 3: C4.5 error rates and standard deviations. For the column "Continuous", C4.5 obtained results without any static discretization. Average error rates of all data sets are shown at the last row (continued in Table 4).

Data	Eq-Freq	1R	D2	MDLP	Mantaras
Australian	14.51±6.08	13.00±5.47	14.13±5.96	14.00±5.90	13.82±5.64
Breast	7.65±3.59	13.27±3.32	5.30±3.09	6.37±4.07	7.79±3.03
Glass	22.43±11.09	18.79±8.04	2.79±4.18	2.31±3.08	2.77±4.27
Heart	22.86±6.42	20.00±7.17	22.13±4.71	20.35±5.75	16.07±4.29
Vehicle	31.39±5.00	28.57±3.97	27.90±4.26	29.47±5.03	29.81±6.44
Iris	8.14±5.75	6.07±6.73	5.13±5.79	4.25±4.58	10.23±5.10
Wine	7.96±5.17	6.84±6.67	6.78±5.49	7.95±7.27	7.53±8.35
Pima	27.68±4.67	25.17±4.20	24.42±4.32	25.21±4.23	22.91±8.65
Bupa	43.96±8.96	36.32±6.40	34.32±6.07	34.29±8.27	31.90±8.39
Thyroid	12.69±8.90	6.44±3.08	8.98±6.00	4.23±4.44	7.90±5.11
Ionos	9.67±5.74	11.98±6.23	8.58±5.10	9.15±5.38	10.69±5.32
Average	18.99	16.95	14.59	14.33	14.69

Table 4: C4.5 error rates and standard deviations (continued from Table 3).

Data	Eq-Freq	1R	D2	MDLP	Mantaras	Zeta	ChiMerge	Chi2
Australian	0.87	0.68	1.43	1.51	5.97	0.72	1.16	2.01
Breast	0.79	0.78	0.92	0.74	1.85	0.77	0.72	0.92
Glass	0.29	0.28	0.45	0.71	1.40	0.36	0.37	0.41
Heart	0.33	0.37	0.51	0.45	0.93	0.34	0.46	0.55
Vehicle	1.85	1.88	2.74	1.90	16.55	2.04	2.08	2.10
Iris	0.71	0.76	1.00	0.65	2.25	0.80	0.96	1.02
Wine	0.29	0.30	0.42	0.45	1.76	0.33	0.43	0.55
Pima	0.70	0.75	0.92	0.91	5.62	0.73	0.55	0.61
Bupa	0.24	0.26	0.33	0.33	0.46	0.28	0.29	0.31
Thyroid	0.13	0.15	0.21	0.22	0.80	0.18	0.19	0.21
Ionos	1.62	1.75	2.10	1.87	7.41	1.79	2.08	2.32
Average	0.71	0.72	1.00	0.89	4.09	0.76	0.84	1.00

Table 5: Time taken for discretization.

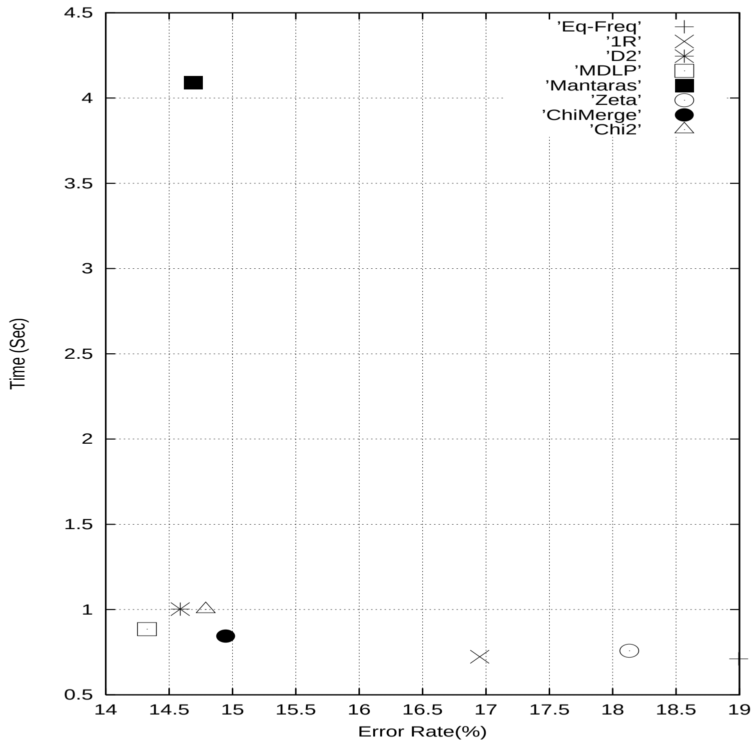


Figure 3: Error rate of C4.5 *vs.* Time of Discretization Methods. Error rates and times are shown as average values over data sets.

classification. The values in this figure are averaged over all data sets and taken from the last rows of Tables 3, 4, and 5. The idea is to see if there is any trade off between error rate of C4.5 on discretized data *vs.* speed of a discretization method. The three methods that spent least amount of time produced highest average error rates on the data sets. As seen in Figure 3, in general, the extra amount of time is worthy for the sake of keeping error rates low, except for the case of Mantaras. Other things being equal, a user may choose a method that gives less error rates.

How discretization affects the learning of a decision tree? As we suggested earlier, in addition to error rates, we can check learning time of C4.5 on the discretized data and number of nodes of a resulted decision tree with respect to one built from the original data. Tables 6 and 7 summarize the experimental results in these two aspects. The learning time is reduced to as much as less than half of the time on learning from continuous data. All discretization methods contribute to time saving in learning. This is consistent with some theoretical findings in [42, 30] that numeric data typically requires repetitive sorting, so it needs a $\log N$ factor at each node for C4.5; but not so for discrete data. The average number of nodes in a decision tree for all the data sets is also reduced.

In order to facilitate our understanding of different discretization methods in their groups, we show the eight discretization methods in Figures 4 (a), (b) and (c) using the result of C4.5 without discretization as the reference. A negative difference means an improvement in accuracy. Several points can be noted from Figure 4: (1) similar discretization methods show the same trend of increase or decrease of error rates for the 11 data sets. Eq-Freq and 1R, D2 and MDLP, ChiMerge and Ch2 behave similarly in pairs with the exception that Mantaras and Zeta have their distinct behaviors. (2) some methods change drastically (exceeding 7% difference) - Eq-Freq, 1R, Mantaras, and Zeta. Some change mildly (withing 7% difference) - Chi2. Some change a little (within 5% difference) - D2, MDLP, ChiMerge.

Data	Continuous	Eq-Freq	1R	D2	MDLP	Mantaras	Zeta	ChiMerge	Chi2
Australian	0.43	0.31	0.27	0.27	0.31	0.26	0.22	0.28	0.10
Breast	0.13	0.06	0.09	0.15	0.10	0.10	0.14	0.13	0.15
Glass	0.10	0.03	0.07	0.04	0.06	0.06	0.06	0.10	0.05
Heart	0.19	0.04	0.08	0.12	0.11	0.09	0.08	0.12	0.04
Vehicle	0.89	0.46	0.57	0.53	0.85	0.54	0.57	0.71	0.82
Iris	0.01	0.02	0.02	0.01	0.01	0.03	0.02	0.01	0.01
Wine	0.12	0.06	0.07	0.05	0.06	0.04	0.04	0.09	0.02
Pima	0.31	0.23	0.21	0.20	0.20	0.30	0.10	0.16	0.11
Bupa	0.11	0.12	0.12	0.07	0.15	0.11	0.04	0.11	0.09
Thyroid	0.05	0.04	0.02	0.06	0.04	0.02	0.02	0.09	0.02
Ionos	1.12	0.75	0.34	0.24	0.34	0.75	0.22	0.20	0.24
Average	0.31	0.19	0.16	0.15	0.20	0.20	0.13	0.18	0.15

Table 6: Time required to learn by C4.5 before and after discretization.

Data	Continuous	Eq-Freq	1R	D2	MDLP	Mantaras	Zeta	ChiMerge	Chi2
Australian	63	30	57	35	35	27	23	60	30
Breast	11	23	5	17	21	11	19	21	15
Glass	11	23	17	11	11	11	11	11	18
Heart	23	4	25	35	33	13	37	35	11
Vehicle	195	157	189	143	149	156	187	190	185
Iris	9	9	5	9	7	7	5	9	4
Wine	9	13	9	13	15	10	15	9	15
Pima	43	19	35	31	27	57	17	39	40
Bupa	51	61	51	29	51	51	19	51	51
Thyroid	17	11	17	17	15	15	13	17	19
Ionos	35	31	35	25	19	29	15	27	20
Average	42	34	40	33	34	35	32	42	37

Table 7: Number of nodes in C4.5 before and after discretization.

In this regard, D2 is the most stable with a difference less than 2% for all data sets. (3) No one discretization method can ensure a negative difference for all data sets. Combining the findings in Figures 3 and 4, we recommend D2, MDLP for a splitting approach and ChiMerge and Chi2 for a merging approach.

Data Set	#MethodBetter
1. Australian	7/8
2. Breast	0/8
3. Glass	0/8
4. Heart	7/8
5. Vehicle	0/8
6. Iris	2/8
7. Wine	0/8
8. Pima	4/8
9. Bupa	2/8
10. Thyroid	3/8
11. Ionos	4/8

(a)

Method	#DataBetter
Eq-Freq	1/11
1R	4/11
D2	4/11
MDLP	6/11
Mantaras	5/11
Zeta	1/11
ChiMerge	3/11
Chi2	5/11

(b)

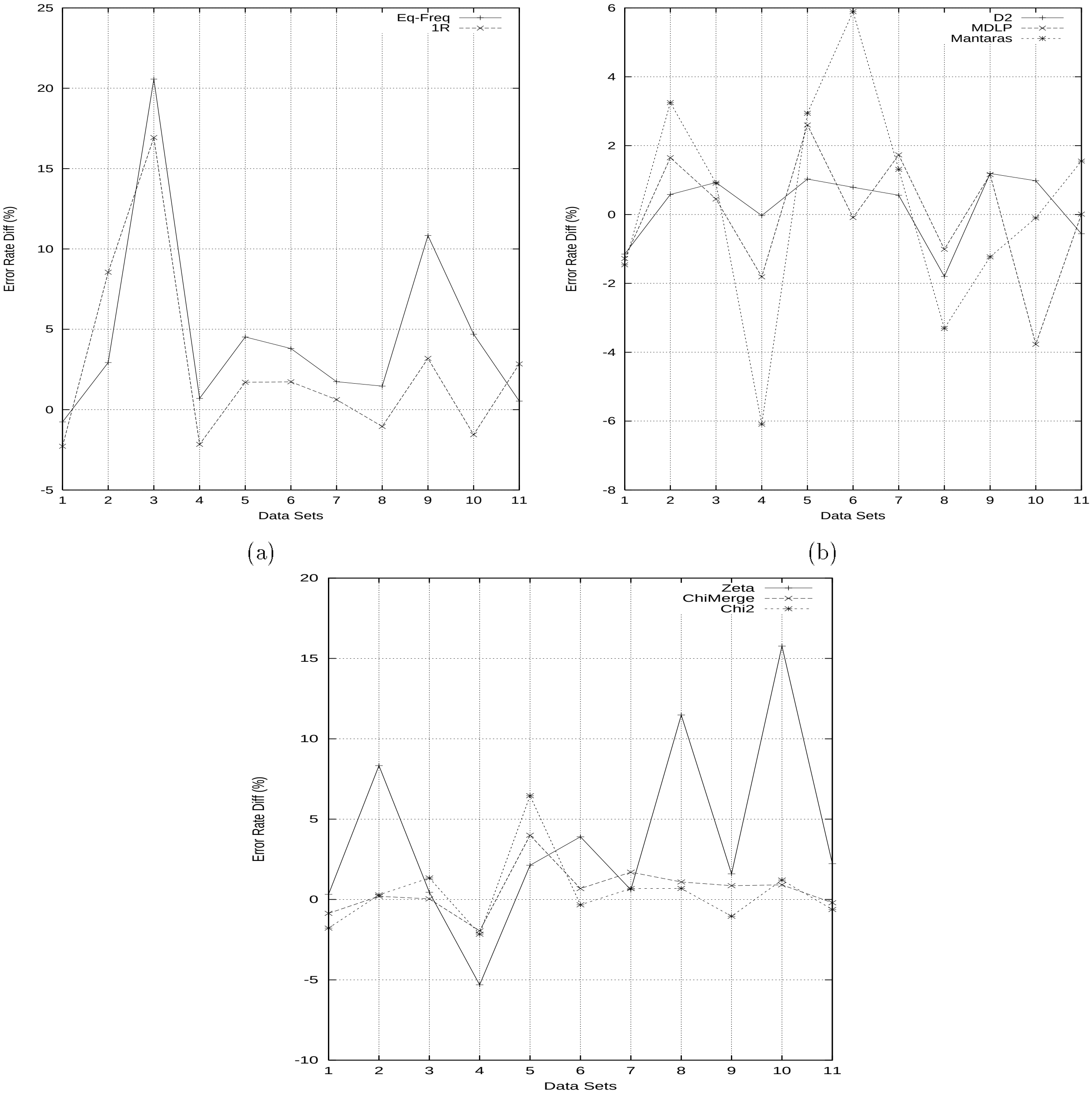
Table 8: Summary of results for C4.5: Table (a) shows the number of methods for each data set for which C4.5 performs better than without discretization. Table (b) shows the number of data sets for each method for which C4.5 performs better than without discretization.

Table 8 shows the summary of results across the data sets and across the discretization methods. As per Table 8, in 29 out of a total of 88 cases (11 'data sets' times 8 'methods') the error rate was less than or equal to that of C4.5 without discretization. The Entropy (MDLP) method gave the best results for error rate (6 out of 11 data sets) whereas Equal-freq and Zeta methods gave the worst results (1 out of 11 data sets). Similarly among the data sets, Australian and Heart showed most improvement after discretization (7 out of 8 methods) while Breast, Glass, Vehicle, and Wine showed least improvement (0 out of 8 methods).

6 Conclusion and Future Work

We present a survey of discretization methods and discuss various dimensions in which discretization methods can be categorized. A typical discretization process is described after introducing some common terms and notations. We then propose a hierarchical framework for discretization methods by considering some important dimensions. Representative methods are given from the perspective of splitting and merging and are further discussed according to the measures used. For each method, we discuss the method and the stopping criterion used, and present the discretization results for the Iris data in terms of number of inconsistencies and number of cut-points.

Experiments for chosen discretization methods have been conducted on 11 data sets with C4.5. The performance of discretization methods is evaluated with several dimensions: time for discretization and for learning, C4.5 error rates on discretized data with reference to C4.5 on original data, and number of nodes in a decision tree. It is observed that in general, more time on discretization leads to better accuracy for C4.5. Our findings in Sections 4 and 5 are pretty consistent as both point toward Entropy (MDLP) being identified as the first choice. However, choosing a suitable discretization method is generally a complex matter, and largely depends on a user's need and other considerations of discretization, as well as on what kind of data to be discretized. If the data does not have class information, only unsupervised methods can be applied. When class information is available, a supervised



(a) (b) (c)

Figure 4: Error rates of C4.5 Classifier before and after discretization. The error rates are shown relative to the classification error before discretization. The serial number of data sets are as given in Table 2. The results are summarized in Table 8.

method should be employed. Do we wish to remove redundant/irrelevant features? If so, Chi2 is a choice. If we need to incorporate discretization into a learning process, dynamic discretization methods such as ID3, Contrast should be considered. To reiterate, if we simply want to discretize data, other things being equal, Entropy (MDLP) should be the first choice to consider.

A reader may notice that every discretization method discussed takes it for granted 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 complexity for discretization. Using the inconsistency measure in Chi2 is one effort towards taking into account the joint contribution of features. With the availability of more powerful parallel computers or computer clusters, we may investigate 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 assumption, it seems practical. Sometimes, a data set consists of various types of features. In Chi2, concepts of over/under discretization are suggested to account for mixed types of features [26]. Again, mixed types of features would not cause a problem if the feature independence assumption is acceptable. It is obviously not the case in the context of multivariate discretization. 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 methods discussed here. Chi2 suggests to use the number of inconsistencies as a way to handle one type of noise. However, it seems an impasse when no prior knowledge about noise is available. All in all, this paper is not about a conclusion of discretization research. Instead, it is about a start of a new phase of discretization research. As we can see, a lot of work has been done, still many issues remain unsolved, and new methods are needed. We hope that this paper will provide a reference point to facilitate researchers and practitioners to embark on further research, development and application.

Acknowledgment

The authors wish to thank Jian Shu for his help in implementing some discretization methods and making the present form of experiments possible.

References

[1] T. L. Bailey and C. Elkan. Estimating the accuracy of learned concepts. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 95–112. Morgan Kaufmann Publishers, 1993.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[3] L. Breiman and P. Spector. Submodel selection and evaluation in regression the x-random case. *International Statistical Review 60(3)*, pages 291–319, 1992.

[4] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proc. Fifth European Working Session on Learning*, pages 164–177. Berlin: Springer-Verlag, 1991.

[5] J. Cerquides and R. L. Mantaras. Proposal and empirical comparison of a parallelizable distance-based discretization method. In *KDD97: Third International Conference on Knowledge Discovery and Data Mining*, pages 139–142, 1997.

[6] C.-C. Chan, C. Batur, and A. Srinivasan. Determination of quantization intervals in rule based model for dynamic. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pages 1719–1723. Charlottesvile, Virginia, 1991.

[7] D.K.Y Chiu, B. Cheung, and A.K.C Wong. Information synthesis based on hierarchical maximum entropy discretization. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:117–129, 1990.

[8] M. R. Chmielewski and J. W. Grzymala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. In *Third International Workshop on Rough Sets and Soft Computing*, pages 294–301, 1994.

[9] P. Chou. Optimal partitioning for classification and regression trees. *IEEE Trans. Pattern Anal. Mach. Intell*, 4:340–354, 1991.

[10] B. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In L. Saitta, editor, *Machine Learning: Proceedings of Thirteenth International Conference*, pages 105–112. Morgan Kaufmann Internationals, Inc., 1996.

[11] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proc Twelfth International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, Los Altos, CA, 1995.

[12] B. Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association 78(382)*, pages 316–330, 1983.

[13] U. Fayyad and K. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning. 8*, pages 87–102, 1992.

[14] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence pp*, pages 1022–1027. San Mateo, CA: Morgan Kaufmann, 1993.

[15] U. Fayyad and K. Irani. Discretizing continuous attributes while learning bayesian networks. In *Proc. Thirteenth International Conference on Machine Learning*, pages 157–165. Morgan Kaufmann, 1996.

[16] T. Fulton, S. Kasif, and S. Salzberg. Efficient algorithms for finding multi-way splits for decision trees. In *Proc Twelfth International Conference on Machine Learning*, pages 244–251. San Francisco, CA: Morgan Kaufmann, 1995.

[17] K.M. Ho and P.D. Scott. Zeta: A global method for discretization of continuous variables. In *KDD97: 3 rd International Conference of Knowledge Discovery and Data Mining*, pages 191–194. Newport Beach, CA, 1997.

[18] R.C Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993.

[19] R.C. Holte, L. Acker, and B.W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818. San Mateo, CA: Morgan Kaufmann, 1989.

[20] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Machine Learning Conference*, pages 121–129. New Brunswick, NJ: Morgan Kaufmann, 1994.

[21] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proc AAAI-92, Ninth National Confrerence Articial Intelligence*, pages 123–128. AAAI Press/The MIT Press, 1992.

[22] P. Kontkaren, P. Myllymaki, T. Silander, and H. Tirri. Bayda: Software for bayesian classification and feature selection. In *4th International Conference on Knowledge Discovery and Data Mining*, pages 254–258, 1998.

[23] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of the tenth national conference on artificial intelligence*, pages 223–228. AAAI Press and MIT Press, 1992.

[24] P. Langley and S. Sage. Induction of selective bayesian classifiers. In *Proceeding Conference on Uncertainty in AI*, pages 255–261. Morgan Kaufmann, 1994.

[25] H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In J.F. Vassilopoulos, editor, *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence, November 5-8, 1995*, pages 388–391, Herndon, Virginia, 1995. IEEE Computer Society.

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

[27] W. Maass. Efficient agnostic pac-learning with simple hypotheses. In *Proc Seventh Annual ACM Conference on Computational Learning Theory*, pages 67–75. New York, NY: ACM Press, 1994.

[28] R. L. Mantaras. A distance based attribute selection measure for decision tree induction. *Machine learning*, pages 103–115, 1991.

[29] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1996.

[30] T. Oates and D. Jensen. Large datsets lead to overly complex models: An explanation and a solution. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 294 – 298. AAAI Press / The MIT Press, 1999.

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

[32] B. Pfahringer. A new mdl measure for robust rule induction. In *ECML95:European Conference on Machine Learning (Extended abstract)*, pages 331–334, 1995.

[33] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[34] J. R. Quinlan. Decision trees and multi-valued attributes. *Machine Intelligence 11: Logic and the Acquisition of Knowledge*, pages 305–318, 1988.

[35] J. R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[36] J. R. Quinlan. Improved use of continuous attributes in c.45. *Artificial Intelligence Research*, 4:77–90, 1996.

[37] M. Richeldi and M. Rossotto. Class-driven statistical discretization of continuous attributes. In *Proc Of European Conference on Machine Learning*, pages 335–338. Springer Verlag, 1995.

[38] C. Schaffer. A conservation law for generalization performance. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 259–265. Morgan Kaufmann, 1994.

[39] C. Shannon and W. Weaver. *The Mathmatical Theory of Information*. Urbana: University of Illinois Press, 1949.

[40] H.A. Simon. *The Sciences of the Artificial*. Cambridge, MA: M.I.T. Press, 2 edition, 1981.

[41] C.J. Thornton. *Techniques of Computational Learning: an introduction*. Chapman and Hall, 1992.

[42] P. Utogoff. Incremental induction of decision trees. *Machine Learning*, 4:161 – 186, 1989.

[43] T. Van de Merckt. Decision trees in numerical attribute spaces. *Machine Learning*, pages 1016–1021, 1990.

[44] K. Wang and B. Liu. Concurrent discretization of multiple attributes. In *Pacific-Rim International Conference on AI*, pages 250–259, 1998.

[45] S. M. Weiss and N. Indurkhya. Decision tree pruning : Biased or optimal. In *Proceedings of the twelfth national conference on artificial intelligence*, pages 626–632. AAAI Press and MIT Press, 1994.