Discretizing Continuous Features for Naive Bayes and C4.5 Classifiers

Fatih Kaya
Department of Computer Science
University of Maryland, College Park
fatih@cs.umd.edu

ABSTRACT

In this work, popular discretization techniques for continuous features in data sets are surveyed, and a new one based on equal width binning and error minimization is introduced. This discretization technique is implemented for the UCI Machine Learning Repository [7] dataset, Adult database and tested on two classifiers from WEKA tool [6], NaiveBayes and J48. Relative performance changes for these classifiers show that this particular discretization method results in greater improvements in the classification performance of NaiveBayes as compared to the J48 classifier.

1. INTRODUCTION

Discretization of continuous attributes is both a requirement and a way of performance improvement for many machine learning algorithms. It is a requirement because those algorithms are only designed to work for nominal feature spaces, therefore even if the features are not discretized as a part of preprocessing work, the algorithm itself does some sort of discretization during the learning process. This is also the reason for aforementioned performance gains of using discretization beforehand.

This work will investigate the effects of *"discretization of continuous variables in a data set"* to the performance of two machine learning methods, a naive Bayesian classifier and a decision tree classifier, C4.5 [9]. The aim is to show that discretization will improve the accuracy of naive Bayesian classifier more than the decision tree classifier. The data set to be used is the *"Adult Database"* from the UCI Machine Learning Repository [7]. It contains information about 48842 examples, each listing 14 attributes of a human. Goal is to predict whether the income of the person exceeds 50K/year based on this data.

2. TASK AND ENVIRONMENT DESCRIPTION

In this project, task is to discretize the continuous features in the *"Adult Database"* and measure the effects of discretization on the learning performances of two machine learning algorithms, NaiveBayes and C4.5. On the other hand, environment is WEKA (Waikato Environment for Knowledge Analysis) [6], which is a suit of machine learning algorithms, and it includes the algorithms and testing constructs we will use. The C4.5 implementation in WEKA is named as J48, therefore this is the name we will refer to in the following experimental sections.

There are a number of variables that we can control, both in the discretization process and the environment itself. Adult database defines each person by 14 attributes, 6 of which are continuous (age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week). There are a few possible approaches to discretize each of these continuous variables. One option is to choose a threshold value and divide the instances into two sets as the ones below that threshold and the ones above the threshold. As an example consider the "age" attribute: this attribute can be discretized by labeling people with ages smaller than 50 as YOUNG and others as OLD. Yet another option would be to have more discrete values for the continuous variable (i.e. [age<20 => YOUNG], [20<=age<50 => MIDDLEAGE], [age>=50] => OLD), or we could define some minimum and maximum interval width values and have a label for each of these intervals, i.e. a label for each age interval of 5 years or 7 years or so on. Thus for the age attribute, we may choose minimum meaningful interval to be 5 years and maximum such interval to be 10, then starting with width 5, we would label ages 0-5 as L1, 6-10 as L2, etc for the first experiment. Then we would repeat the same process for interval width 6 for 2nd experiment, interval width 7 for 3rd up to width 10. The minimum and maximum width values for intervals can also be controlled. At this point, we can also control the step amount for the interval values between min and max interval widths. For example for the example given above this step value was assumed to be 1, thus intervals tested were of length 5, 6, ..., 10. Instead we could choose the step to be 2, then the interval widths would be 5, 7, etc.

Controllable variables for the environment include the type of validation tests for the learning algorithms and other options that can be set individually for the algorithms through WEKA interface. WEKA uses tests to validate the learning performance of the algorithms. The type of these tests is not preset and can be one of these options: using training set, using a separate test file or using k-fold cross validation. Training set is the set of instances fed to the learning algorithm; if this set is used also as test data (the first option above) there is a high probability to get higher accuracy values, in other words, results may be biased. On the other hand we may split our instances into two and use one of them to train the algorithm (training set), and use the other (test set) to test the learned classification, which corresponds to the second option of using separate test set. Final option is a technique mostly used when there are not many instances to split. K-fold cross validation is the process of partitioning instances into small subsets and training and testing the learning algorithm for each subset such that in each run only one of the subsets is used as test set and others as training set. Besides the testing options, we can also force some individual settings for the learning algorithms, such as controlling the pruning capabilities of C4.5. However, this may not be something useful to do, since most of the time we will not be able to define similar options for both of the algorithms.

Based on the above independent variables, descriptions, and modified training sets, we will keep track of a few dependent variables, which are either provided by WEKA or can be measured from the results WEKA supplies. WEKA returns a set of result tuples for each run on a training set with a particular algorithm. Among these tuples are correctly classified instances, incorrectly classified instances, mean absolute error and root mean squared error. These can be used to assess the accuracy of a single learning algorithm for

a particular set or to compare the performances of two algorithms on the same training set. To have an idea about the effect of discretization, multiple versions of the training set, ones which have continuous attributes and ones that have various levels of discretized attributes, should be fed to a single algorithm and its results should be compared. For this we will define new variables to compute the rate of changes in the accuracy of the classifiers. Our final goal will be to compare these rates of changes for NaiveBayes and C4.5 with respect to different level of discretizations on continuous attributes.

3. RELATED WORK

3.1 TAXONOMY FOR THE DISCRETIZATION OF CONTINUOUS FEATURES FOR MACHINE LEARNING ALGORITHMS

Generally three different dimensions have been used for classifying the discretization methods: “global vs. local”, “supervised vs. unsupervised”, and “static vs. dynamic” [1]. Local methods work on a single attribute, whereas global ones consider all of the attributes and partition each into independent regions. The second dimension refers to the use of class labels during discretization. Training data for classification learners include instances, each of which is a set of attributes and a class label. When the learning algorithm builds a model based on the training data, it is given a new instance and expected to guess its class label correctly. Unsupervised methods do not make use of these class labels for discretization. However, the supervised ones use them and try to create intervals where most of the instances of a single interval have same class label [5,11,13]. The last dimension regards the number of intervals to be created. This value is either inputted from the user or determined after one pass over the training data for each attribute by static methods. On the other hand dynamic methods search through all possible values for all features therefore provide means to observe dependencies among features. Below is a table that exemplifies some of the classification items given above.

	GLOBAL	LOCAL
SUPERVISED	Recursive Minimal Entropy Partitioning	Hierarchical Maximum Entropy
UNSUPERVISED	Equal Width Interval, Equal Frequency Interval	K-means Clustering

Table 1: A simple classification for the discretization methods

3.2 FOUR CLASSES OF DISCRETIZATION ALGORITHMS

There has been a lot of work done in continuous feature discretization field [1,2,4,5,8,12,13,14]. Below are four classes of those discretization algorithms and their performance comparisons.

3.2.1 Equal Width Binning and Equal Frequency Binning

Equal width binning [10,11] is one of the simplest approaches to unsupervised discretization process together with equal frequency binning. Basic step for the first method is to divide the range of values into k intervals of equal width. Similarly, equal frequency binning [10,11] divides the range into k bins of equal frequency so that at the end each bin has same number of instances (i.e. all/k). Since these methods do not make use of the class labels, the discretization process possibly loses the classification information, as instances of different classes can easily be grouped together. Additionally equal width binning is sensitive to outliers: consider the case where all instances have values between 1 and 20 except one that takes a value of 100 and k is given as 5. Then this method would produce approximately 15 empty bins resulting in a meaningless distribution of the attribute.

3.2.2 Recursive Minimal Entropy Partitioning

Recursive Minimal Entropy Partitioning (RMEP) is a supervised discretization method introduced by Fayyad and Irani [2]. RMEP aims to find intervals that minimize the class information entropy. Entropy here is the information entropy defined by Shannon [3].

Below is an example entropy computation. Let class label has three values {c1, c2, c3} and S' be a set with 10 instances, 2 of which are of class c1, 4 of c2 and 4 of c3. Then the entropy is calculated with the following formulation:

$$\text{Entropy}(S') = -(2/10)\log_2(2/10) - (4/10)\log_2(4/10) - (4/10)\log_2(4/10)$$

RMEP is a recursive algorithm, which given a set of instances, finds the interval boundary that provides the minimum entropy. Thus if S is the initial set of all instances and A is the attribute to be discretized, RPEM first finds the sets S1 and S2 (thus makes a binary discretization based on A) such that the following value is the minimum:

$$(|S1|/|S|)\text{Entropy}(S1) + (|S2|/|S|)\text{Entropy}(S2)$$

Then this is recursively applied to S1 and S2, until a stopping condition is reached, which is based on minimum description length principle. RPEM continuous dividing sets as long as the desired entropy value is not reached, therefore the created intervals will be smaller in length in regions where a high value of entropy is observed.

3.2.3 Error Based Discretization

Another supervised method is the error based discretization given by Maass [4] and analyzed by Kohavi and Sahami [5]. This algorithm takes into consideration the performance of the learning algorithm itself. It tries to find the optimal number of intervals by computing the error observed after running the learning algorithm only for the attribute that is being discretized. At the end optimal number of intervals found by

the algorithm will be the one that gives the minimum classification error on the training data.

3.2.4 SOM BASED DISCRETIZATION

Final algorithm is the Self Organized Map (SOM) based discretization given by Vannucci and Colla [8]. This one works similar to the k-means clustering algorithm, which generates arbitrary number k of partitions based on minimum square error partitioning method. The basic difference is that SOM does not use a fixed k value, rather defines a max value and then the algorithm finds out one that preserves the sample's distribution more than the others.

3.3 REPORTED RESULTS FOR NAIVE BAYES AND C4.5 CLASSIFIERS

Dougherty, Hohavi and Sahami evaluate numerous discretization methods for naive bayes classifier and C4.5 and show that almost any discretization method results in significant performance gains for naive bayes, while RMEP is the optimal one for C4.5 [1]. Kohavi and Sahami compare entropy-based methods with error-based methods and conclude that recursive minimal entropy partitioning outperforms error-based approaches most of the time [5]. Finally, Vannucci and Colla, compare their SOM-based method and k-means algorithms with other discretization algorithms and report that these two produce better results than classical methods as they reflect the original distribution better than the others [8].

4. RESEARCH QUESTION AND HYPOTHESES EXAMINED

Previous section gave information about various discretization techniques, which have been widely used for many classification methods. Although error-based approaches have been showed to be less effective than the entropy based ones [3], it is not clear if they will have any positive effects if they are used to assist the simpler methods such as equal width binning. Therefore, we are planning to investigate this issue by introducing some extensions to the equal width binning (EWB) approach that uses this error minimization metric while choosing the interval length.

As it was given in the previous section, EWB accepts a parameter k, i.e. the number of intervals to be created and discretizes the attribute's range into equal width partitions based on this value. Given k, EBW ends up with the following interval ranges:

```
[minValue+i*width,minValue+(i+1)*width]
for i={0,1,...,k-1}
where width = (maxValue– minValue)/k
```

The number of intervals is always fixed and independent of the specific properties of the training data. This restriction may have many unwanted side-effects. First consider the case, when the training data is a large set of instances and k is a small value. Then the produced bins will be overloaded, and will have the risk of grouping a large range of

instances together, therefore it will not have any auxiliary effect for the learning algorithm. On the other hand if k is too big then the bins will have very few elements and we will not be able to see any significant effect of discretization. Our intention for using error metric is to have some means of associating k value with the properties of training data and this builds our first hypothesis: *Dynamically searching through the possible values of k and recording corresponding classification performances should enable us to adjust k according to the properties of training data and find the optimal one.* Second weakness of EBW was due to its being unsupervised. Ignoring class labels could cause EBW to loss important classification information during discretization process. Our expectation for this part is also to benefit from the error minimization process. Since error-minimization indirectly evaluates class labels, accounting it, can minimize the negative effects of class-ignorant discretization.

Another part of our work will be to assess the combined effect of discretization of multiple attributes. We plan to search for optimal discretization intervals for each attribute separately and then discretize all the features with collected information. Next are the possible outcomes and our second hypothesis that we may observe: *Using optimal values for each attribute at the same time can result in the overall optimal set of interval values; or when we have simultaneous discretizations, one can suppress other's positive effect.* As an example, let's assume we are to discretize two continuous features A and B and our discretization method returns k_A and k_B as the optimal interval values. Assume also that discretizing the training data only for attribute A with k_A results in some improvement but then discretization of B does not produce any performance gains on the training data whose A attributes have been discretized. This constitutes a case where discretization of one variable suppresses the other one.

Overall reasoning behind the first hypothesis is to utilize error minimization for the simple approach of equal width binning. This frees us from statically determining the k value by guessing one that will efficiently divide instances to bins, and makes it possible to adjust it based on the characteristics of training data. The need for the second hypothesis is inspired from the fact that the training data we are using and in general most of the similar data sets have more than one continuous feature. Therefore, investigating the discretization of all continuous features instead of just one, can reveal more improved results. In order to provide this capability, we have to choose an optimal k value for each of the continuous features. In doing so, second hypothesis presents the simplest approach: to select each k separately based on the method supported by first hypothesis. We will introduce an alternative approach below. A third conclusive hypothesis is the one that compares the relative performance improvements seen for Naive Bayes classifier and the decision tree classifier C4.5, due to this discretization process. Here, we expect that Naive Bayes's improvement will outperform C4.5 and make it more advantageous.

5. EXPERIMENTAL DESIGN

As mentioned in the previous section, experiments will be conducted in the WEKA environment [6], using the NaiveBayes and J48 classifiers. For the experiments regarding the first hypothesis, primary independent variable will be the number of intervals,

therefore various files, produced from the training set by discretizing at different levels and for different variables, will be used. On the other hand, primary dependent variable will be the number of correctly classified instances when a classifier is trained with these training files. Learned classifiers will be tested on both training data via cross-folding and also on separate test files, which contain instances that are not used to train the classifiers. For the experiments regarding the second hypothesis, number of continuous features that are discretized will be the independent variable; and dependent variable will again be number of correctly classified instances. Finally for the comparison part, we will have a dependent variable that codes the performance gain acquired by a level of discretization. This is basically the ratio of correctly classified instances between discretized and not-discretized training files.

In the first phase of experiments, we will find the optimal interval values for each of the continuous features that Adult Database has [7]. This training set includes six continuous attributes: age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week. Finding the optimal k value means, searching through all possible values of k i.e. k', discretizing training file using k' and feeding it to the classifiers. After which, the learned classifiers are tested using cross-validation or a test file. Value that results in the maximum number of correctly classified instances will be kept as the optimal one. For each attribute, we will have one optimal k for Naive Bayes and one for C4.5 (i.e. J48). In second phase, we will be using the optimal values from the first phase. Below is the experiment outline for this phase suggested by the second hypothesis. Note that these steps will be repeated once for each of Naive Bayes and C4.5 classifiers.

```
Outline 1
opt_age <- find_optimal(
      training_file, age)
opt_fnlwgt <- find_optimal(
      training_file, fnlwgt)
opt_education-num <- find_optimal   (
      training_file, education-num)
opt_capital-gain <- find_optimal (
      training_file, capital-gain)
opt_capital-loss <- find_optimal(
      training_file, capital-loss)
opt_hours-per-week <- find_optimal(
      training_file, hours-per-week)

discretized_file <- discretize_file(
      training_file with
      opt_age, opt_fnlwgt,
      opt_education-num,
      opt_capital-gain,
      opt_capital-loss,
      opt_hours-per-week)

compute_performance_gain(
      training_file, discretized_file)
```

Figure 1: First outline for discretizing multiple continuous features

One alternative to this approach that is slightly different would be to use intermediate discretized files. When an optimal k value for an attribute is found, the current training file is discretized with that value and used in the next step. Below is the corresponding experiment outline.

```
Outline 2
opt_age <- find_optimal(
      training_file, age)
intmed_file_age <- discretize_file(
      training_file, age with opt_age)
opt_fnlwgt <- find_optimal(
      intmed_file_age, fnlwgt)
intmed_file_fnlwgt <- discretize_file(
      intmed_file_age,
      fnlwgt with opt_fnlwgt)
...
discretized_file <- discretize_file(
      intmed_file_capital-loss,
      hours-per-week with
      opt_hours-per-week)

compute_performance_gain(
      training_file, discretized_file)
```

Figure 2: Second outline for discretizing multiple continuous features

6. RESULTS

We have used the Adult database to test the discretization method. The database has 32561 instances. Below is the range information for the six continuous features over these instances.

Continuous Variable	Minimum Value at the Adult Database	Maximum Value at the Adult Database
age	17	90
fnlwgt	12285	1484705
education-num	1	16
capital-gain	0	99999
capital-loss	0	4356
hours-per-week	1	99

Table 2: Range information for the continuous features in the training dataset

Initial experiment was to investigate the rough effects of discretization to the learning time and prediction accuracy of both methods. To figure out that, we have first run these algorithms without any discretization using a training file that has 2000 training instances chosen from the Adult database and got the following results.

Learning Method	Time Taken to Build Model	Number of Correctly Classified Instances
NaiveBayes	42 milliseconds	1705
J48(C4.5 imp in WEKA)	250 milliseconds	1721

Table 3: Performance measures for an unmodified training file of size 2000

Then we applied discretization to the "hours-per-week" attribute and run the tests again to have an informal idea about the behavior. The hours-per-week variable takes values in the range [1-99]. We have chosen to discretize this field using the length 38 for NaiveBayes and length 28 for J48, which turned out to be the best values for discretization of this field for the used training file. Below are the results.

Learning Method	Time Taken to Build Model	Number of Correctly Classified Instances
NaiveBayes	16 milliseconds	1708
J48	150 milliseconds	1742

Table 4: Performance measures for a training file of size 2000 discretized for "hours-per-week" attribute

Examining the results, we may informally argue that the optimal level of discretization improves both the model construction time and prediction accuracy for both methods. However; it is not true that we get improvements in the prediction accuracy for any discretization level. As an example for discretization with interval length 10, NaiveBayes predicts only 1700 of the instances correctly for the same file used above, which is worse than the unmodified version.

Following these initial experiments, the first step was to find the optimal discretization interval lengths for each of the continuous attributes for NaiveBayes and J48, separately, for the whole training data of Adult database. Interval length refers to the difference between the minimum and maximum values that can be included in a bin. As an example, if it is 5 then, any attribute that has value [1-5] will be labeled as *bin1* and any attribute with value [6-10] will be labeled as *bin2*, and so on.

Notice from table 2 that two of the features have relatively large ranges: "fnlwgt" and "capital-loss". Since it would be costly and take too much time to examine all possible interval length values for these ranges, we have omitted to discretize them. Therefore, only four of these features were actually discretized. Below are the optimal interval length values, found using all of the training data with 10-fold cross-validation.

CONTINUOUS VARIABLES	Naive Bayes	J48
age	4	8
education-num	4	2
capital-loss	11	30
hours-per-week	6	12

Table 5: Optimal discretization interval lengths for outline 1

Each optimal value has been found by using the fresh copy of the training file, in other words, we have followed the first outline given above. Thus for "education-num" attribute, we have discretized the training file 16 times, separately, for each of the interval length values {1,2,...,15,16}. For each discretized file, we have made both of the classifiers learn the file and via cross-validation we have noted their classification performances. The value, which resulted in the greatest number of correctly classified instances, has been noted as the optimal one.

Below is the table that shows time taken to build model and number of correctly classified instances for the original training file and the files, which have been created by discretizing only one of the features. First row is for the unmodified training file, for which NaiveBayes classifies 27171 of the instances correctly and J48 classifies 28607 correctly. Second row shows that when only age is discretized NaiveBayes classifies 27243 of the instances correctly. Similarly, fourth row shows that when only capital-loss is discretized J48 classifies 28632 of the instances correctly, and so on.

DISCRETIZED CONTINUOUS VARIABLE	NaiveBayes		J48	
	time to build model (ms)	# of correctly classified instances	time to build model (ms)	# of correctly classified instances
none	425	27171	10056	28607
age	371	27243	8012	28648
education-num	355	27209	8420	28598
capital-loss	360	27673	8653	28632
hours-per-week	322	27203	7908	28649

Table 6: Performance measurements for independent discretization of each feature

Table 6 shows that discretization process results in shorter times for building models for both classifiers. NaiveBayes exhibits an increase in classification accuracy for each of four features. This is also almost true for J48 except the education-num attribute for which classification accuracy slightly decreases.

Next comes the table that shows the effect of discretization when it is applied to a combination of continuous features. Second row in Table 7 is for the file, where only age is discretized, third one is for the file where both age and education-num is discretized, fourth is similarly for the file where age, education-num and capital-loss is discretized. And finally the last one is the file with all four features discretized.

DISCRETIZED CONTINUOUS VARIABLE	NaiveBayes		J48	
	time to build model (ms)	# of correctly classified instances	time to build model (ms)	# of correctly classified instances
none	425	27171	10056	28607
age	371	27243	8012	28648
+ education-num	312	27248	7640	28544
+ capital-loss	313	27240	5508	28371
+ hours-per-week (i.e. all 4 discretized)	183	27704	5635	28365

Table 7: Performance measurements for combined discretization of features

Results show that "using the optimal values from independent searches described in first outline" works for NaiveBayes but not for J48. Discretization of all features enables NaiveBayes to increase its correctly classified instances from 27171 to 27704; however, it results in important accuracy decrease for J48: form 28607 to 28365. On the other hand significant gains are observed in the model building times for both of the classifiers.

Next are the optimal values for the second outline. Optimal value for age is found using the original training file; however in the following steps the original file is not used. To find the optimum interval length value for education-num, training file is discretized for age feature with the optimal value found in the previous step (i.e. 4 or 8) and then the search is conducted using this file. Similarly, when an optimal value is found for education-num, training file is discretized for both age and education-num before the search for capital-loss's optimal values.

CONTINUOUS VARIABLES	NaiveBayes	J48
age	4	8
education-num	11	6
capital-loss	5	26
hours-per-week	1	46

Table 8: Optimal discretization intervals for outline 2

Here comes the classification performance table for the second approach.

DISCRETIZED CONTINUOUS VARIABLE	NaiveBayes		J48	
	time to build model (ms)	# of correctly classified instances	time to build model (ms)	# of correctly classified instances
none	425	27171	10056	28607
age	371	27243	8012	28648
+ education-num	316	27254	7106	28553
+ capital-loss	224	27703	6399	28500
+ hours-per-week (i.e. all 4 discretized)	202	27642	4957	28511

Table 9: Performance measurements for outline 2

Table 9 reveals that this alternative approach works better for J48, as it outputs 28511 correct instances as opposed to the 28365 instances of first outline; however, it is still not enough to get any accuracy improvements. NaiveBayes again records significant gains in both model build time and overall classification accuracy.

7. CONCLUSION

As we have defined in our hypotheses, we have always observed significant gains in classification accuracy for NaiveBayes; however, J48 showed fluctuating results and no positive change for overall discretization. On the other hand, runtimes for model construction are always positively affected for both classifiers. For the overall discretization of four features, NaiveBayes decreased its model construction time from 425 to 183 milliseconds, and increased its classification accuracy from 27171 to 27704 correct instances. If we are to compare this to the model construction time and accuracy of J48 on unmodified training file, which are 10056 milliseconds and 28607 correct instances, it is a significant improvement.

For the overall results that are obtained, there are many threats for internal and external validity. Although many different training files are produced and used to explore the behavior of discretization method, only the original training file is used to obtain the final results, which exhibits a threat to the internal validity. Besides, for testing phase, only cross-validation is used and in order to get healthier results different test files should be gathered and deployed. Using a single database will hurt the external validity of the project. Therefore, above experiments should be repeated on multiple datasets and

checked if they produce similar results or not. Another enhancement would be to find an efficient way of discretizing the two variables, fnlwgt and capital-gain, which have been omitted due to their large ranges; as the results show that discretizing features with large ranges result in greater performance gains (see capital-loss in Table 6).

8. REFERENCES

[1] Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proc. Twelfth International Conference*, pp. 194-202. San Francisco, CA: Morgan Kauffmann.

[2] Fayyad, U. M. & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022-1027. San Francisco, CA: Morgan Kauffmann.

[3] Information theory. (2007, December 7). In *Wikipedia, The Free Encyclopedia*. Retrieved 10:33, December 9, 2007, from http://en.wikipedia.org/wiki/Information_theory

[4] Maass, W. (1994). Efficient agnostic PAC-learning with simple hypotheses. In *Proc. Seventh Annual ACM Conference on Computational Learning Theory*, pp. 67-75.

[5] Kohavi, R., & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proc. Second International Conference on Knowledge Discovery in Databases*, pp. 114-199. San Mateo, CA: AAAI Press.

[6] Witten, I. H., & Frank, E. (2005). Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco.

[7] Asuncion, A & Newman, D. J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[8] Vannucci, M., & Colla, Valentina. (2004). Meaningful discretization of continuous features for association rules mining by means of a SOM. In *Proc. European Symposium on Artificial Neural Networks*, pp. 489-494. Bruges, Belgium.

[9] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

[10] Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *Proc. European Working Session on Learning*, pp. 164-178.

[11] Kerber, R. (1992). Chimerge: Discretization for numeric attributes. In *National Conf. on Artificial Intelligence*, pp. 123-128.

[12] An, A., & Cercone, N. (1999). Discretization of continuous attributes for learning classification rules. In *Proc. Third Pacific-Asia Conf. on Methodologies for Knowledge Discovery and Data Mining*, pp. 509-514.

[13] Ho, K. M., & Scott, P. D. (1997). Zeta: A global method for discretization of continuous variables. In *Proc. Third International Conf. on Knowledge Discovery and Data Mining*, pp. 191-194.

[14] Pazzani, M. J. (1995). An iterative improvement approach for the discretization of numeric attributes in Bayesian classifiers. In *Proc. First International Conf. on Knowledge Discovery and Data Mining*, pp. 228-233.