Generation of Sufficient Cut Points to Discretize Network Traffic Data Sets

Sahisnu Mazumder[1], Tuhin Sharma[1], Rahul Mitra[1],
Nandita Sengupta[2], and Jaya Sil[1]

[1] Dept. of Computer Science and Technology,
Bengal Engineering and Science University,
Shibpur, Howrah -711 103, West Bengal, India
[2] University College of Bahrain, Bahrain
{sahisnumazumder,tuhinsharma121,rahulm9999}@gmail.com,
ngupta@ucb.edu.bh, js@cs.becs.ac.in

Abstract. Classification accuracy and efficiency of an intrusion detection system (IDS) are largely affected by the discretization methods applied on continuous attributes. Cut generation is one of the methods of discretization and by applying variable number of cuts (in a partition) to the continuous attributes, different classification accuracy are obtained. In the paper to maximize accuracy of classifying network traffic data either 'normal' or 'anomaly', the proposed algorithm determines the set of cut points for each of the continuous attributes. After generation of appropriate and necessary cut points, they are mapped into corresponding intervals following centre-spread encoding technique. The learnt cut points are applied on the test data set for discretization to achieve maximum classification accuracy.

Keywords: discretization, cut generation, center-spread encoding, nsl-KDDCUP'99 network traffic data set classification accuracy.

1 Introduction

The process of partitioning continuous variables into categories is termed as discretization. Discretization is a potential time-consuming bottleneck, since the number of possible discretization is exponential in the number of interval threshold candidates within the domain. The goal of discretization is to bind the domain of all continuous conditional attributes into some finite set of values. The finite values help to observe the patterns of objects and associated decision class labels appearing in the discretized training data set. Many of the Machine Learning algorithms [1,2,3] produce better models by discretizing continuous attributes. For example, Naive Bayes classifier [4,5] requires probability estimations with the help of continuous attributes. But it is difficult to handle as they often take too many different values for a direct estimation of frequencies. To alleviate this problem, normal distribution of the continuous values can be assumed, but it is not always realistic. The same phenomenon leads rule extraction techniques to build poorer sets of rules. Decision

B.K. Panigrahi et al. (Eds.): SEMCCO 2012, LNCS 7677, pp. 528–539, 2012.

Tree based algorithms [6,7,8] cannot handle continuous attribute directly rather nominal attributes. As a result, machine learning and statistical techniques are applied on the data sets to compose nominal variables. However, a very large proportion of real data sets include continuous variables i.e. variables measured at the interval or ratio level. One solution is to partition numeric variables into a number of sub-ranges [9] and treat each such sub-range as a category. In the paper, based on the observations, contribution of a given interval corresponding to a particular decision (normal or anomaly) has been measured. In addition, by maximizing the interdependence between class labels and attribute values, the paper aims at to develop an ideal discretization method with a secondary goal to minimize the number of intervals without significant loss of class-attribute mutual dependence.

Cut generation [10] is one of the widely used methods of discretization. By varying number of cuts, applied on continuous attributes classification accuracy is also varied. So, in order to maximize classification accuracy of the network traffic data, the classifier system should learn to determine the set of cut points for each of the continuous attributes. In the proposed classifier system, a heuristic based cut generation algorithm has been developed which generates a set of cut points that are sufficient to distinguish between a discernible pair of data objects.

The paper has been divided into four sections. Section 2 presents the proposed work while section 3 comprises of experimental results and comparisons with other discretization methods considering nsl-KDDCUP'99 network traffic data set [11,12]. Finally, conclusions are summarized in section 4.

2 Proposed Work

Discretization is performed prior to the learning process [13] by dividing the total task into three sub-modules. The first task is to find the number of discrete intervals, unlike other discretization algorithms where user must specify the number of intervals [14] or provide a heuristic rule [15]. The second task is to find the width or the boundaries of the intervals depending on the range of values of each continuous attribute. Finally, the attribute values from the continuous domain are mapped to the discrete domain.

2.1 Cut Points Generation

Heuristic based solution to determine the set of all necessary and sufficient cut points has been described below:-

Algorithm Cut Generation

Step 1: Consider, k^{th} conditional attribute values in a decision system and arrange them in ascending order.

Step 2: For each particular value, group the objects based on the decision attribute value (d) either ***normal*** (say, d=1) or ***anomaly*** (say, d=2).

Step 3: The ordered k^{th} conditional attribute values are marked as per following rule:

(i) Mark a value by "tick" if it corresponds to an object with decision attribute value *normal* (d=1).

(ii) Mark a value by "Encircle" if it corresponds to an object with decision attribute value *anomaly* (*d*=2).

Step 4: Scan the marked k^{th} attribute values and set the cut points between any two successive values (say, attr-*k*_value-1 and attr-*k*_value-2) accordingly:

(i) If attr-*k*_value-1 is marked as "encircled" (or "ticked") and attr-*k*_value-2 is marked as "ticked" (or "encircled"), select a cut point as the mid-point of the interval between attr-*k*_value-1 and attr-*k*_value-2.

(ii) If one of them (say, attr-*k*_value-1) is marked as both "encircled" and "ticked" (i.e. in both ways) and other one (say, attr-*k*_value-2) is marked as either "encircled" or "ticked" (i.e., only one of the two ways), select a cut point as the mid-point of the interval between attr-*k*_value-1 and attr-*k*_value-2, which are marked differently.

(iii) If attr-*k*_value-1 is marked as both "encircled" and "ticked" and attr-*k*_value-2 is also marked as both "encircled" and "ticked", select a cut point as the mid-point of the interval between attr-*k*_value-1 and attr-*k*_value-2.

Step 5: Do step 1 to step 4 for all conditional attributes.

The proposed method of generating a cut point between two successive attribute values is based on their marking either different way or in both ways. ***Therefore, the data objects having those values of the k^{th} attribute forms a discernible pairs belonging to that training data set.***

***Example 1*:** Consider the following continuous data set in decision system, given in the table 1. In this decision system, there are 3 conditional attributes (Attr-1, Attr-2, Attr-3) and one decision attribute (Decision) with two possible decision attribute values (Decision =1, representing *normal* and Decision =2, representing *anomaly*). There are 21 data objects or instances in the given continuous data set in table 1.

Table 1. A Continuous Decision System

Attr-1	Attr-2	Attr-3	Decision
13	1	1	1
0	2	2	1
0	2	2	2
7570	1	1	1
0	1	4	1
0	1	5	2
10	1	8	1
0	3	12	1
5	1	8	1
0	2	10	2
0	1	14	1
0	1	3	1
282	1	9	1
0	1	15	1
0	1	6	1
0	1	16	2
0	1	11	1
0	3	13	1
7951	1	1	1
0	1	7	1
0	1	17	1

Now, the proposed cut generation algorithm is applied on the data set in table 1, producing ordered set of values for **"Attr-1"** as:-{0, 5, 10, 13, 282, 7570, 759 }. The result of the proposed heuristic based solution is given in fig. 1 for Attr-1.

***Attr-1*:**

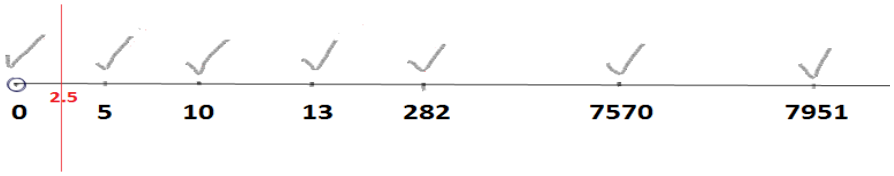

Fig. 1. Plotted Continuous Attribute values and Possible Cut Points for *Attr*-1

So, the only sufficient cut point we get is 2.5 and each one of the set of discernible pairs [16] (with respect to attr-1) consists of two data objects, one having the value from set {0} and the other having any value from the set {5,10,13, 282, 7570, 7951}.

Similarly, for two more conditional attributes, the proposed algorithm generates following cut point as shown in fig. 2 (for Attr-2) and fig. 3 (for Attr-3):-

***Attr-2*:**

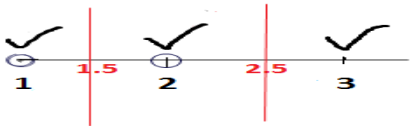

Fig. 2. Plotted Continuous Attribute values and Possible Cut Points for *Attr*-2

So, from fig. 2, the generated cut points for "*Attr-2*" are: **{1.5, 2.5}** and the discernible pairs [16] (with respect to attr-2) obtained from Fig. 2., are – {data object having *"Attr-2"* value as 1, data object having *"Attr-2"* value 2} ; { data object having *"Attr-2"* value as 2, data object having *"Attr-2"* value 3} and { data object having *"Attr-2"* value as 1, data object having *"Attr-2"* value as 3}.

***Attr-3*:**

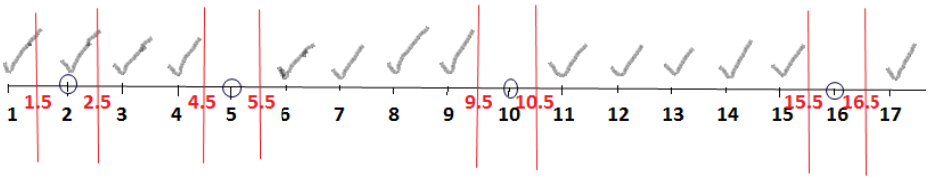

Fig. 3. Plotted continuous attribute values and possible cut points for attr-3

So, from fig. 3, the generated cut points for *Attr*-3 are: **{1.5, 2.5, 4.5, 5.5, 9.5, 10.5, 15.5, 16.5}.**

2.2 Centre-Spread Encoding Method

From the cut points of the decision system, intervals are generated and as a next step, the intervals are encoded with discrete values using Centre-Spread encoding technique. In the proposed approach, the mid-point of the interval represents the *centre* which is a discretized value of any continuous value belonging to that interval. The other variable *spread* provides limit of the continuous values on either side of the mid-point in discretized form. So, an interval is represented as (*centre*, *spread*) as described below:-

Step 1(Interval Generation):-The cut points are traced and two successive cut points are represented as an interval [*lw*,*up*) where lower one (*lw*) is included and upper one (*up*) is excluded in the interval.

Step 2(Centre-Spread encoding):- An interval denoted as [*lw*,*up*) is encoded as the centre (mid-point) of the interval and spread (span) of the interval from its mid-point to either side of the end-points.

Example 2: *The set of cut points of conditional attribute* ***"Attr-3"*** *is arranged in ascending order as:-* {1.5, 2.5, 4.5, 5.5, 9.5, 10.5, 15.5,16.5}. *Inclusion of minimum and maximum value of the attribute results ----* {**1,** *1.5, 2.5, 4.5, 5.5, 9.5, 10.5, 15.5,* 16.5**, 17**}.

From this set, following intervals are generated–

[1, 1.5):-Interval # "1"; [1.5, 2.5):-Interval # "2"; [2.5, 4.5):-Interval # "3"; [4.5, 5.5):-Interval # "4"; [5.5, 9.5):-Interval # "5"; [9.5, 10.5):-Interval # "6"; [10.5, 15.5):-Interval # "7"; [15.5, 16.5):-Interval # "8" and [16.5, 17):-Interval # "9"

Now, the intervals of ***"Attr-3"*** *are encoded using* ***Centre-Spread encoding*** *technique as follows given in table 2:-*

Table 2. Centre-Spread Encoding of Intervals

Intervals	Centre	Spread
[1, 1.5)	1.25	0.25
[1.5, 2.5)	2	0.5
[2.5, 4.5)	3.5	1
[4.5, 5.5)	5	0.5
[5.5, 9.5)	7.5	2
[9.5, 10.5)	10	0.5
[10.5, 15.5)	13	2.5
[15.5, 16.5)	16	0.5
[16.5, 17)	16.75	0.25

2.3 Discretization of Data Set

Data set has been discretized to bind the domain of continuous conditional attributes into some finite set of values so that patterns of all conditional attributes corresponding to each data object and its associated decision class has been observed. Using the Centre-Spread encoding technique, each attribute value of a data set is guaranteed to be in at most and at least one interval. The corresponding centre value is assigned as the discrete value for the respective attribute of the decision system.

Example 4: After encoding of intervals, all the center values of the set of intervals are obtained for each of the 3 continuous attributes and used to discretize corresponding continuous attributes. Finally, the discretized form of the given decision system is obtained and given in the table 3.

Table 3. Discretized form of the Decision System

Attr-1	Attr-2	Attr-3	Decision
3976.75	1.25	1.25	1
1.25	2	2	1
1.25	2	2	2
3976.75	1.25	1.25	1
1.25	1.25	3.5	1
1.25	1.25	5	2
3976.75	1.25	7.5	1
1.25	2.75	13	1
3976.75	1.25	7.5	1
1.25	2	10	2
1.25	1.25	13	1
1.25	1.25	3.5	1
3976.75	1.25	7.5	1
1.25	1.25	13	1
1.25	1.25	7.5	1
1.25	1.25	16	2
1.25	1.25	13	1
1.25	2.75	13	1
3976.75	1.25	1.25	1
1.25	1.25	7.5	1
1.25	1.25	16.75	1

3 Time Complexity Analysis

Assume that the algorithm runs on a continuous valued decision system with *"m"* numbers of conditional attributes (say, attr-1,…, attr-*m*), one decision attribute (say, *d*) and "*n*" number of objects or instances (say, obj-1,…, obj-*n*). The decision system is the input to the proposed discretization algorithm.

3.1 Complexity of the cut generation algorithm

In step 1 of the cut generation algorithm, k^{th} conditional attribute ($1 \leq k \leq m$) is considered for discretization. Now, according to their decision attribute (d) values, k^{th} conditional attribute values along with marking (step 2 and step 3) are copied into a two dimensional array (with two fields value and class) of length n. Time complexity is O(n) to perform this task. Then, the array is sorted using heap sort algorithm because it is an in-place sort and has both average and worst case time complexity is O($n$lg$n$) for sorting "n" number of elements. In step 4, the sorted array is traversed to search the cut points and simultaneously storing these cut points into an array. To perform this operation, the algorithm takes O(n) running time.

So, for one conditional attribute, the cut generation algorithm takes {O(n)+O($n$lg$n$)+O(n)} running time which is O($n$lg$n$). Now, for "m" number of conditional attributes, the time complexity of the proposed algorithm is O(mnlg$n$). However, for a given decision system m is constant and does not change with time for a given data set, only no of instances may change. So, if we treat m as a constant, then the overall running time comes out to be O($n$lg$n$).

3.2 Complexity of the Centre-Spread Encoding Method

To encode the cut-points into intervals, the array which stores the cut-points has been traversed. To perform this task, the running time is O(n) and for all conditional attributes it becomes O(mn). If m is considered as constant, then time complexity of the center-spread encoding method is O(n).

3.3 Complexity of the Discretization Algorithm

To discretize the data set using encoded intervals, each continuous value of a conditional attribute is mapped into the center point of the corresponding interval. So to search that specific interval, binary search algorithm is employed with time complexity O(lg$n$). Therefore, to discertize all values of that attribute corresponding to n instances, it takes O($n$lg$n$) time and for the whole decision system, O(mnlg$n$) running time. If m is constant then time complexity of the discretization algorithm is O($n$lg$n$).

Therefore, time complexity of the proposed discretization algorithm is O(mnlg$n$) and if m is constant, it is O($n$lg$n$).

4 Experimental Results and Comparisons

Comparisons are performed between the discretized data set obtained using the proposed discretization process and the original continuous data set in terms of classification accuracy, mean absolute error, root mean square error, relative absolute error and root relative squared.

We have considered 10000 data objects from nsl-KDD Data set [11, 12], continuous in nature. Then the proposed discretization method is applied and 10 fold cross validation technique is used for measuring classification accuracy of different classifiers. The average classification accuracy in percentage of different well known classifiers considering both discretized and continuous data set has been shown in Fig. 4.

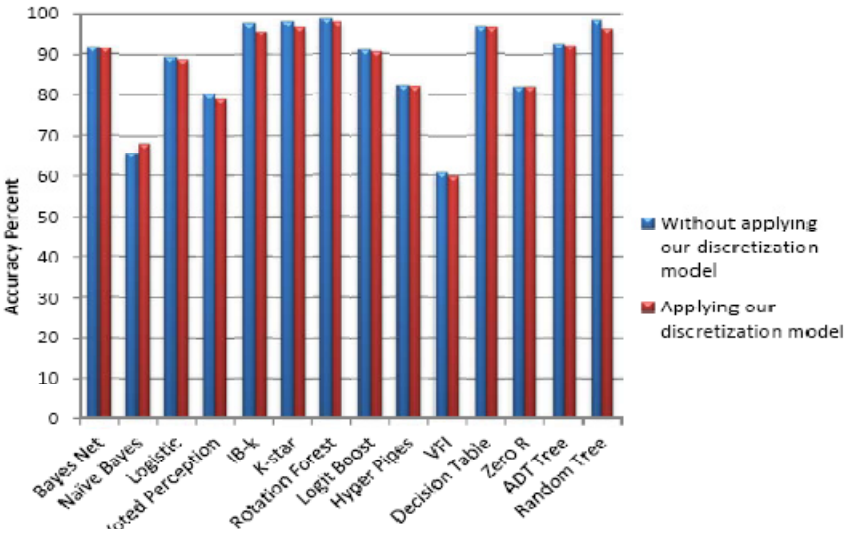

Fig. 4. Comparisons of classification accuracy considering both discretized and continuous data set

Maximum **1.9%** drop in accuracy is observed in random tree classifier while there is **2.4%** increase in average classification accuracy considering Naïve Bayes Classifier. Due to information loss in the discretization process, reduction of accuracy is inevitable. The original data set has more information than the discretized one. When the training patterns are stored in multi-dimensional feature space (for lazy classifier) or represented using the rule set (for rule based classifier), or decision tree (for tree based classifier) is generated or the best component classifier for each data point (for meta-classifier [22]) is selected then it covers much more attribute domain than discretized data set. After discretization, several attribute values (on the basis of which decisions are made) which were in the rule set or decision tree or in the multidimensional feature space are lost due to the discretization and for those attribute values decisions made may be wrong. Similarly, for meta-classifier if some of the data points are lost then the selection of best component classifier may not be optimal. This explains the drop of classification accuracy.

Discretization provides an alternative to the probability density estimation when quantitive attributes are involved in naive-Bayes learning and under this probability density estimation, if the assumed density is not a proper estimate of the true density, the naïve-Bayes classification accuracy degrades [20]. Since for real world data, the true density is usually unknown, often unsafe assumptions are made resulting less

classification accuracy. A proper discretization method can circumvent this problem by tuning interval size and interval number to find a good trade-off between discretization bias [21] and discretization variance [21] and thus can achieve low learning error and higher classification accuracy. This explains the increase in classification accuracy of naïve-Bayes classifier (shown in figure. 4.) when it runs on the discretized data set generated from the continuous one by our proposed discretization method.

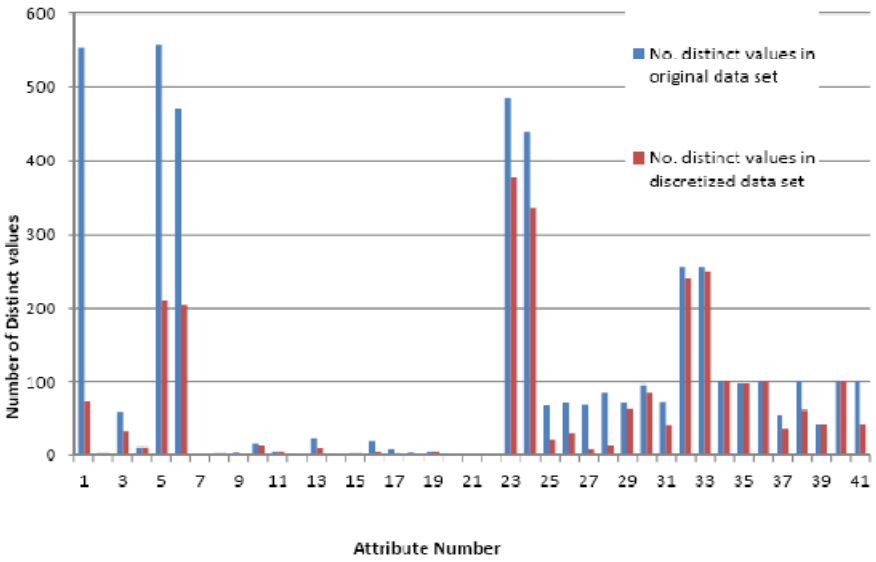

Fig. 5. Shows the difference between the distinct values in the original data set of different attributes with that of the discretized data set

Fig. 5 shows the comparisons of continuous data in the original data set into less number of distinct values in the discretized data set. Among the different attributes, the maximum reduction in data points occurred for Attribute 1 in the original data set having **550** distinct values and the discretized having only **70** distinct values. So the maximum ratio of reduction is (**550/70**) i.e. **7.85**.

In table 4, along with average classification accuracy [22] other statistics are provided showing error values which are nearly the same. Here, maximum difference in root mean square error in classification between discretized and continuous data set is 0.06. Furthermore, true positive [23], false positive [23] and F-measure [23] values are provided while classifying both the discretized and continuous data set. From the analysis of the results given in these figures, we can conclude that there is very little data loss in our proposed discretization process of a given continuous data set. Weka tool [24,25] is used for calculation of statistical parameters.

Here Precision of a class is the number of True Positives divided by the total number of elements belonging to the positive class. Recall is defined as the number of

True Positives divided by the number of elements actually belong to the positive class.

$$Prediction(p) = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad ;$$

$$Recall(r) = \frac{True\ Positive}{True\ Positive + False\ Negetive} \quad ;$$

$$F\ measure(F) = \frac{2(p * r)}{p + r}$$

Table 4. Comparisons of Correctly classified percentage, Root Mean Squared error in classification, true positive, false positive, and F-measure values

Classifier Type	Classifier Name	Without Applying our Discretization model					Applying our Discretization model				
		Correctly Classified Instance (%)	Root Mean Squared error	True Positive Rate (Class)	False Positive Rate (Class)	F – measure (Class)	Correctly Classified Instance (%)	Root Mean Squared error	True Positive Rate	False Positive Rate	F – measure
BAYES	Bayes Net	91.62	0.27	0.92(1)	0.10(1)	0.92(1)	91.45	0.27	0.92(1)	0.19(1)	0.94(1)
				0.89(2)	0.07(2)	0.79(2)			0.89(2)	0.08(2)	0.73(2)
	Naïve Bayes	65.47	0.57	0.97(1)	0.46(1)	0.74(1)	67.8	0.55	0.64(1)	0.19(1)	0.76(1)
				0.83(2)	0.38(2)	0.46(2)			0.80(2)	0.35(2)	0.47(2)
FUNCTION BASED	Logistic	89.38	0.28	0.97(1)	0.46(1)	0.93(1)	88.85	0.29	0.97(1)	0.47(1)	0.93(1)
				0.53(2)	0.02(2)	0.53(2)			0.52(2)	0.03(2)	0.63(2)
	Voted Perception	80.09	0.44	0.88(1)	0.58(1)	0.87(1)	79.05	0.45	0.91(1)	0.76(1)	0.87(1)
				0.41(2)	0.11(2)	0.43(2)			0.23(2)	0.08(2)	0.28(2)
LAZY	IB-K	97.59	0.15	0.98(1)	0.06(1)	0.98(1)	95.69	0.20	0.97(1)	0.13(1)	0.97(1)
				0.93(2)	0.01(2)	0.93(2)			0.86(2)	0.02(2)	0.88(2)
	K-Star	98.03	0.11	0.98(1)	0.05(1)	0.98(1)	96.91	0.15	0.98(1)	0.10(1)	0.98(1)
				0.94(2)	0.01(2)	0.94(2)			0.89(2)	0.01(2)	0.91(2)
META	Rotation Forest	98.46	0.10	0.99(1)	0.03(1)	0.99(1)	97.74	0.13	0.98(1)	0.06(1)	0.98(1)
				0.96(2)	0.01(2)	0.96(2)			0.93(2)	0.01(2)	0.93(2)
	Logic Boost	90.89	0.27	0.97(1)	0.38(1)	0.94(1)	90.49	0.27	0.97(1)	0.40(1)	0.94(1)
				0.61(2)	0.02(2)	0.71(2)			0.59(2)	0.02(2)	0.69(2)
MISC	Hyper Pipes	82.01	0.50	1.00(1)	0.98(1)	0.90(1)	81.81	0.5	1.00(1)	0.99(1)	0.90(1)
				0.01(2)	0.00(2)	0.03(2)			0.01(2)	0.00(2)	0.01(2)
	VFI	60.63	0.49	0.52(1)	0.04(1)	0.68(1)	59.86	0.49	0.52(1)	0.05(1)	0.67(1)
				0.95(2)	0.47(2)	0.47(2)			0.95(2)	0.48(2)	0.45(2)
RULE BASED	Decision Table	96.64	0.16	0.98(1)	0.12(1)	0.98(1)	96.5	0.16	0.98(1)	0.13(1)	0.97(1)
				0.87(2)	0.01(2)	0.90(2)			0.86(2)	0.01(2)	0.90(2)
	Zero R	81.68	0.38	1.00(1)	1.00(1)	0.89(1)	81.71	0.38	1.00(1)	1.00(1)	0.89(1)
				0.00(2)	0.00(2)	0.00(2)			0.00(2)	0.00(2)	0.00(2)
TREE BASED	ADT Tree	92.56	0.24	0.97(1)	0.30(1)	0.95(1)	92.22	0.24	0.96(1)	0.28(1)	0.95(1)
				0.69(2)	0.02(2)	0.77(2)			0.71(2)	0.03(2)	0.77(2)
	Random Tree	98.24	0.13	0.98(1)	0.04(1)	0.98(1)	96.34	0.18	0.97(1)	0.10(1)	0.97(1)
				0.95(2)	0.01(2)	0.95(2)			0.89(2)	0.02(2)	0.89(2)

5 Conclusions

In the paper, we have proposed a discretization method and applied on network traffic data set. Results are analyzed and classification accuracy is compared using different error values. It has been observed that in the proposed discretization method very little data loss is occurred. Here, sufficient number of cut points has been generated for each continuous attributes and maintains consistency in the decision system after

getting discretized using encoded intervals generated from those cut points. Therefore, each data objects in the discretized data set preserves its integrity [20] even after the proposed discretization process as it has been in continuous data set. Therefore, from this discussion, we conclude that using the proposed discretization method cut points are generated and using encoded intervals, an efficient technique of discretization has been achieved.

References

1. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification
2. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow Clustering Using Machine Learning Techniques. In: Passive & Active Measurement Workshop, France (April 2004)
3. Dunnigan, T., Ostrouchov, G.: Flow Characterization for Intrusion Detection, Technical Report, Oak Ridge National Laboratory (November 2000)
4. http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf
5. Chai, K., Hn, H.T., Chieu, H.L.: Bayesian Online Classifiers for Text Classification and Filtering. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 97–104 (August 2002)
6. Badulescu, L.A.: Data Mining Algorithms Based On Decision Trees, Annals of the Oradea University. Fascicle of Management and Technological Engineering, vol. V(XV), pp. 1621–1628. Publishing House of Oradea University (2006) ISSN:1583 - 0691
7. Chaudhuri, S., Fayyad, U., Bernhardt, J.: Scalable Classification over SQL Databases. In: Proc. ICDE 1999, Sydney, Australia, pp. 470–479. IEEE Computer Society (1999)
8. Du, W., Zhan, Z.: Building Decision Tree Classifier on Private Data. In: IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining, Conferences in Research and Practice in Information Technology, Maebashi City, Japan, vol. 14. Australian Computer Society, Inc. (2002)
9. Kotsiantis, S., Kanellopoulos, D.: Discretization Techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering 32(1), 47–58 (2006)
10. Xu, T., Yingwu, C.: Half-global discretization algorithm based on rough set theory. Journal of Systems Engineering and Electronics 20(2) (April 1, 2009)
11. Nsl-kdd data set for network-based intrusion detection systems (2009), http://nsl.cs.unb.ca/NSL-KDD/
12. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A Detailed Analysis of the KDD CUP 99 Data Set
13. http://werner.yellowcouch.org/phd03/PhdOnTheWeb/node8.html
14. Ching, J.Y., Wong, A.K.C., Chan, K.C.C.: Class-Dependent Discretization for Inductive Learning from Continuous and Mixed Mode Data. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(7), 641–651 (1995)
15. Ren, Z., Hao, Y., Wen, B.: A Heuristic Genetic Algorithm for Continuous Attribute Discretization in Rough Set Theory
16. Komorowski, J., Polkowski, L., Skowron, A.: Rough Set: A tutorial
17. Kruse, R.L., Ryba, A.J.: Data structures and program design in C++. Prentice Hall (1998) ISBN-13: 9780137689958
18. Boritz, J.E.: IS Practitioners' Views on Core Concepts of Information Integrity. International Journal of Accounting Information Systems (retrieved August 12, 2011)

19. Morariu, D.I., Vintan, L.N., Tresp, V.: Meta-Classification using SVM Classifiers for Text Documents World Academy of Science, Engineering and Technology 21 (2008)
20. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proc. of the Twelfth International Conf. on Machine Learning, pp. 194–202 (1995)
21. Yang, Y., Webb, G.I.: On Why Discretization Works for Naive-Bayes Classifiers
22. Han, Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2001)
23. Neyman, J., Pearson, E.S.: The testing of statistical hypotheses in relation to probabilities a priori. Joint Statistical Papers, pp. 186–202. Cambridge University Press (1933, 1967)
24. Weka 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/
25. Weka User Manual http://www.gtbit.org/downloads/dwdmsem6/dwdmsem61man.pdf, http://kent.dl.sourceforge.net/project/weka/documentation/3.6.x/WekaManual-3-6-2.pdf