A Novel Discretization for Parameter Learning in Bayesian Network using Dynamic Programming

Satyabrata Pradhan
Infosys Labs
Infosys Limited
Hyderabad, India
satyabrata_pradhan@infosys.com

P. Radha Krishna
Infosys Labs
Infosys Limited
Hyderabad, India
radhakrishna_p@infosys.com

Abstract—**In AI and machine learning techniques such as decision trees and Bayesian networks, there is a growing need for converting continuous data into discrete form. Several approaches are available for discretization, however finding an appropriate and efficient discretization method is a challenging task. In this paper, we present an impurity based dynamic multi-interval discretization approach for learning Bayesian networks. This approach is based on the dependencies among nodes existing in the network and capable of discretizing the nodes in the network using the interdependencies with multiple class variables. The simulation based experiments conducted in this paper, suggest that the proposed discretization method produces good inference with very minimum information loss. The dynamic programming approach used in the method, further reduces the space and time complexity of the complete discretization process.**

Keywords-Bayesian Network; Discretization; Information Gain; Dynamic Programming

I. INTRODUCTION

Bayesian Network (BN) is a model for reasoning about uncertainty, built on centuries-old Bayes theorem (invented by Thomas Bayes in 1763). Building Bayesian network has wide range of application areas such as Bioinformatics, Medicine, Image Processing, Manufacturing industries and Finance. Despite the wide applicability of Bayesian networks, requirement of discrete data is a limitation, as data for certain applications are routinely continuous, thus necessitating discretization. Moreover, while modeling a Bayesian network with continuous data or mixture of continuous and discrete data, it becomes extremely important that the discretization process should not be carried out in isolation as these attributes are connected to each other in the network. Discretization in isolation may lead to incorrect cause-effect analysis. Studies usually employ simple discretization techniques such as frequency-based discretization [5]. But by not addressing the ramifications of discretization, this process unknowingly loses information such as interactions and dependencies between variables and hence it impacts the learned parameters. Unfortunately, there is no consensus on a standard procedure for discretization. Consequently, it is still an unresolved research question as how best to handle continuous data.

Ideally, discretization method for Bayesian approach should have the following features:

1) Capability to split the numerical series of data to any number of buckets.
2) Splitting of any variable must be logical, i.e. it should be optimal in terms of information extracted from training data set, and the split of a variable should be based on its dependencies on other variables in the network.
3) Split must reduce the uncertainty as much as possible, though uncertainty can not be avoided completely.

B. *Motivation*

Many discretization works have been carried out [3] [4][11] for learning using continuous attributes. Specifically [3] [4] use the structure of a network for Bayesian learning. These works mainly deals with iterative binary splits or joins with some conditions, with a stopping criterion. For a predefined number of partitions k (i.e. k+1 discrete levels), recursive binary splits may not provide optimal discrete buckets. On the other hand, considering all combination of k-partitions out of possible (N - 1) partitions (N is the total number of records) gives us the optimal partition, but time consumption is very high for a large value of N. So, in this work, we attempt a dynamic programming based approach to find best possible k-partitions, which also reduces the time complexity of the process.

C. *Proposed method*

The proposed entropy based discretization methods have the following features:

1) Searching through all possible k-partitions instead of performing binary partitions to find best split.
2) Implementation of dynamic programming based Gain calculation to avoid redundant entropy computation.
3) Intelligent pre-storing of entropy measures in a matrix, which reduces both time and space complexity.
4) Capability of discretizing the nodes using multiple discrete nodes to capture all possible interdependencies.

In addition to above, existing theorems from the literature are also incorporated to optimize the overall performance of the complete discretization approach.

The rest of the paper is organized as follows. Related work on continuous feature discretization is presented in Section 2. Section 3 presents the details of the proposed solution. Simulation based experiments are conducted on a Bayesian network and results are discussed in Section 4. Finally the paper ends with conclusion in Section 5.

II. Related Work

Discretization process can be broadly classified in three different dimensions [1]: global vs. local, supervised vs. unsupervised, and static vs. dynamic.

Local Discretization methods (ex. C4.5) perform partitioning of data points in localized regions of instance space. In global methods (ex. binning [2]), each continuous feature is discretized independent of other features in the whole space.

Discretization methods which do not use the instance label of existing discrete features on a data set are referred as unsupervised discretization (ex., equal width interval and equal frequency methods). On the other hand, discretization methods that use the available class labels are referred to as supervised discretization methods.

Determination of maximum number of intervals required in discretization plays an important aspect in classifying the processes as a static or a dynamic method. Methods such as binning and entropy based partitioning [3] perform discretization process on each feature to determine number of intervals (*k*) independent of other features. In contrast, dynamic methods search through all possible values of *k* for all features simultaneously.

In the context of Bayesian learning, discretization can be classified into two categories, depending on the time phase at which it is done. First, data can be discretized prior to and independent from the application of the learning algorithm (ex., equal-frequency/equal-width discretization [5]). Second, the discretization can be integrated into the parameter learning phase in an effort to exploit the synergies [6], [4], [7]. In the case of pre-discretization method [5], during selection of different ranking and bucket span, it is assumed that each variable is independent of others most of the time, which is not true in case of Bayesian network, as nodes are dependent on each other according to the connectivity (edges) in the network. During selection of bucket span, it does not consider other variables. Even if bucket span of dependent set of variables is considered, it is practically impossible to carry out this task for large size and complex networks.

Entropy based discretization methods such as ID3 [13] and C4.5 [12] use minimal entropy heuristic for discretization of continuous attributes. These methods try to find out binary cut for each attribute for Decision tree learning. Introduced by Fayyad and Irani [3], a multilevel discretization process recursively performs binary partition using a stopping criterion called minimum description length (MDL). This method automatically decides the number of discrete levels suitable for a continuous attribute. These methods are very useful in decision tree learning, but do not provide optimal partition for discretization as they depend on iterative binary partitioning.

The Chi-Merge algorithm described in [11] consists of an initialization step and a bottom-up merging process. Chi-Merge is initialized by first sorting the training examples according to the values of an attribute being discretized and then constructed the initial discretization where each example is put into its own interval (i.e., placing interval boundary before and after each example). In the merging process, intervals are continuously merged until a termination condition is met. The interval merging process contains two iterative steps, firstly computing the $\chi2$ value for each pair of adjacent intervals, then merging (combine) the pair of adjacent intervals with the lowest $\chi2$ value. Merging continues until all pairs of intervals have $\chi2$ values exceeding the parameter $\chi2$ -threshold; that is, all adjacent intervals are considered significantly different by the $\chi2$ independence test. This method also determines number of discrete levels required, like Entropy based methods by Fayyad [3].

Learning Vector Quantization (LVQ) [12] is a supervised learning algorithm based on neural networks, where code vectors W_i labeled by each class is fed into the space. This approach is used for discretization, where a potential cut point can be the middle of learned codebook vectors of two different classes [14].

A detailed comparison of the above methods along with the Histogram based approach is presented in [14].

III. Proposed Approach

A. *Discretize a continuous variable V*

The main steps of our approach to discretize a continuous node *V* are

1) Perform a breadth-first-search in the network to discretize the continuous nodes in the network, starting from the root node (see section III.B.).
2) Select the best suitable set of class nodes for the continuous node.
3) Sort the values in node *V* and arrange the class nodes in a set, say *S*, accordingly.
4) Find all possible cut points for each class node in *S*.
5) Pre-store entropy measure of each possible valid partition in a matrix corresponding to each class node in set S.
6) Compute all possible k-partitions from k-1 cut points of each class node using dynamic programming approach. Find best partition out of all computed partition in order to optimize Gain value of the continuous node with respect to each class node.

The above steps are discussed below in detail.

B. *Parsing the BN for discretization*

The proposed method assumes that at least one attribute is in discrete form to start. We label the attribute as root node. Then, each node of the Bayesian network is processed using breadth-first-search and in each step, if the node is continuous then it is discretized using suitable class variable(s). At every iteration step of Algorithm 1, a continuous valued attribute gets discretized based on a chosen class attribute. The best cut points are decided based on entropy maximization theory. Here, an undirected version of the graph for the Bayesian network is considered for traversal purpose. The overall algorithm is as follows:

```
Algorithm 1: Main Discretization routine to parse all nodes in BN
Data: Bayesian network (BN) structure and records of all the nodes
Result: Discretized version of all nodes
Initialization:
 G = Bayesian network converted into undirected graph;
 C = Set of all continuous valued nodes in the Bayesian network;
 S = The root node which is initially discretized;
Insert S into a queue Q;
while Q !empty do
    V = extract node from Q;
    if $V \in C$ then
        C = C - V;
        Choose candidate class variable for node V;
        Perform Discretization on node V;
    end
    ADJ = Set of all adjacent nodes of V;
    Push all the nodes in ADJ to Q;
end
```

C. Selection of class variable(s)

In order to discretize a continuous variable *V* in each iteration of Algorithm 1, we need a class variable (in discrete form). The adjacent nodes of V can be considered as the possible set of class variables. To find the best splits according to the selected class variable(s), we use information Gain measure as given below.

$$\text{Gain}\ \text{V},\text{T}\ =\ \text{Info}\ \text{T}\ -\ \text{Info}(\text{V},\text{T}) \qquad (1)$$

where *T* is the class variable and *V* is the continuous variable. Info(T) is defined as

$$\text{Info}\ \text{T}\ =\ {}_{\text{i}=1}^{\text{n}}\,\text{p}_\text{i} * \log(\text{p}_\text{i}) \qquad (2)$$

where $P = (p_1, p_2, \ldots\ldots p_n)$ is the probability distribution of class variable *T*.

Info(V, T) for a given *k*-partition of *V* that divides the original class variables record set into $T = T_1, T_2, \ldots\ldots T_k$ is defined as,

$$\text{Info}\ \text{V},\text{T}\ =\ {}_{\text{i}=1}^{\text{n}}\,\frac{|\text{T}_\text{j}|}{|\text{T}|} * \text{Info}(\text{T}_\text{j}) \qquad (3)$$

Higher the Gain better is the split, and hence we choose the split that has highest Gain. To find a discretization process for a Bayesian network, a continuous variable should not just be converted into a discrete form with respect to a single node. Instead, we capture as much dependencies as possible from all the adjacent nodes. Here we choose a combination of class variables so that the discretization process maximizes the Gain value for each class variable. For this, we explore to find a cluster of class variables whose information gain values are distinct and better than the rest. Algorithm 2 presents the selection procedure to find the best set of the class variables.

```
Algorithm 2: Find best set of class variable(s)
Data:   Continuous node V and its adjacent set adj D(V ) which
        contains only discrete nodes
Result: Set S containing best set of class variable(s) for node V
Initialization:
 D  $\leftarrow$   data set containing all the highest Gain(V, $X_i$) values
        for each $X_i$ where $X_i \in$ adj D(V) ;
Sort D;
Q $\leftarrow$ queue having initial element as D;
while Q  !empty do
        C $\leftarrow$ extract the first set from Q;
        Call PPDP to divide the data set C into $C_L$ and $C_R$;
        $W_L$ = mean of $C_L$;
        $W_R$ = mean of $C_R$;
        if $| W_L - W_R | \geq \tau$ then
            insert $C_L$ into Q;
        else
            S $\leftarrow C_L$;
        end
end
```

Algorithm 2 comprises of two techniques: K-means [10] and Principal Direction Divisive Partitioning (PDDP) [8]. K-means is the most widely used clustering technique; and it is the best representative of the class of iterative centroid-based divisive algorithms. On the other hand, PDDP is a representative of the non-iterative techniques based on the Singular Value Decomposition (SVD) of a matrix built from the data-set. Detailed description of Algorithm 2 can be found in [9].

D. Cut-Points selection

Cut points in class variable T always occur on boundary between two different class values [3]. Thus the number of possible cut points can be greatly reduced by the approach presented in [3]. However this approach is not directly applicable when multiple class variables are considered for discretization. When we choose cut-points according to the criteria mentioned in [3], we may not find a partition that is common in all classes. So in order to come out of this situation, we keep the partitions based on the cut points for continuous variable with respect to each class variable. Once all possible partitions computed, we consider top n (n=100 for conducted experiments) partitions with highest information gain iteratively. In each step, we check the corresponding class variable of the Gain value and then fetch the Gain values of other class variable at the same partition either from the pre-computed Gain table in the database or by computing them freshly. Details of how to obtain the best partition which satisfies all the class variable distribution is discussed in section III.F.

E. Pre-storing Entropy measure

Assume T = {a, b, c, d, e, f, g, h} is a class variable. Consider the following two cases of a continuous variable *V*, where each case is partitioned into three buckets:

Case 1 : (a, b, c)(d, e)(f, g, h),
Case 2 : (a, b)(c, d, e)(f, g, h)

Here the third partition (T3) generated in both the cases are same. If we compute the Gain(V, T) for both the above cases, then we redundantly compute the term $\frac{|T_3|}{|T|}\mathrm{Info}(T_3)$ which is exactly same for both cases.

In order to avoid such redundant computation, we store all possible $\frac{|T_m|}{|T|}\mathrm{Info}(T_m)$ in a triangular matrix (*M*) of dimension n x n. Consider an element M_{ij} which denotes the function $Info(t_i, \ldots\ldots, t_j)$ of a partitioned interval containing sorted elements $\{t_i, \ldots\ldots, t_j\} \subset T$. If *k* number of partitions are required, an interval [ti, ……, t_j] in a data set of length *n* is defined as

$$j \le n - k + i \text{ and } i \ge (k - n - j) \qquad (4)$$

So, the total number of computations required to store all entropy measures is $\frac{n\ n+1\ - k\,(k-1)}{2}$, which is in the order of $O(n^2)$ compared to $O(k * n^k)$ if all computations of entropies carried out at run time.

An alternate pictorial presentation of the above derivation is given in section IV-D1.

F. Best partition by dynamic approach

In this paper we follow a dynamic approach to compute all possible combination of partitions out of *b* possible cut-points. Let B is the set of all cut-points. |B| = b, where b be the possible number of cut-points. Here, number of accesses to the entropy matrix *M* is also further reduced in an attempt to save more computing time in general and increases the overall performance of the proposed method. A typical example to illustrate the performance of the presented approach is given in section IV-D2. The procedure of this approach is given in Algorithm 3.

```
Algorithm 3: ComputeGain(K, p, Value)
Data:     An array B of all b possible cut-points, class variable T
          ∈ S, number of partitions k and Matrix M containing all
          intermediate entropy measures
Result: Hash Table H
Initialization: Hash table H; K [1 : b + 2]; $K_1$ = 1; $K_{b+2}$ = N, where N
is the length of total data set;
p = 1; value = 0;
Function ComputeGain(K, p, Value);
for i ← $K_p$ + 1 to b - k + p do
          if p = 1 then
                value = 0;
          end
          value = value + $M t_{B_{i-1,}}, t_{B_i}$ ;
          $K_{p+1}$ = i;
          if p < b then
                ComputeGain (K, p+1,value);
          else
                value = value + $M t_{B_{i,}+1}, t_{K_{b+2}}$ ;
                Store value in a hash table H with key as elements
                           in K;
                return;
          end
end
```

In Algorithm 3, hash table H is generated for each class variable in the set of best class variables *S*. To select the best partition, we need to leverage the corresponding values of a partition in each hash table.

Assuming a scenario, there are two class variables considered for partitioning a variable. The partition creating highest Gain value in each table may not unique. So, we need to find a unique partition that maximizes the Gain value for both class variables. We consider top *n* partitions, from a descending sorted pool of partitions and compute the Gain value for other class variables with respect to the same partition. Now assume for a partition that the gain value stored in both hash tables are [0.62, 0.46] in point form. Euclidian distances are computed between the virtual maximum valued point, taken from highest Gain point for each class and each partition. The point having shortest distance from the virtual max point is considered as the final partition. If there is a tie in Euclidian distance of multiple partitions, then lowest absolute difference between the Gain values in the partition is considered for final selection criteria of best partition.

IV. Experimental Results and Discussion

To demonstrate the proposed discretization method, a simulation based approach is followed to create continuous data set from a well defined Bayesian network structure [16]. This structure contains filled conditional probabilities and prior probabilities for its corresponding nodes.

A. Bayesian Network Source

Bayesian network structure considered in this work contains all discrete state nodes whose probability tables are already populated. For this purpose, a model (Power plant) of the correlations among the sensors in a coal-driven power plant is considered listed in [16]. As shown in Figure 1, it has 46 sparsely connected nodes, all of which are ternary. Nielsen and Jensen [15] have used this Bayesian network in their work for on-line alert system.

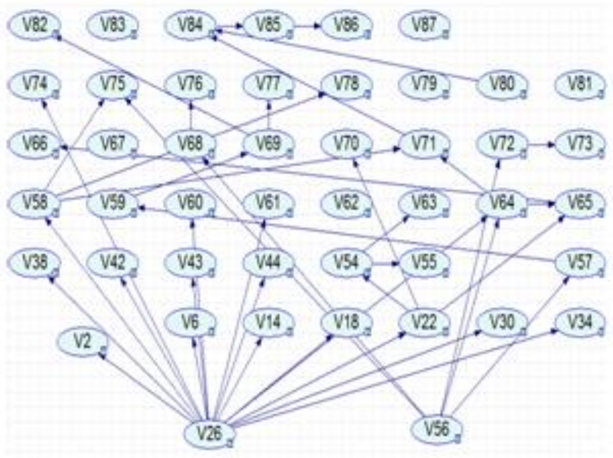

Figure 1. Power Plant Bayesian Network Structure

B. *Data Simulation*

The open source tool for Bayesian network, namely Genie, is used for analysis of inference. Using the tool, a data set for the corresponding network is simulated. The discrete states of the nodes are represented in terms of range. So, after the data set is simulated, a randomization algorithm is used to simulate continuous values for the discrete states of the nodes within the provided range for each record in the generated data set. The table I shows the format of the states used for a node in the supplied Bayesian network.

TABLE I. INTERPRETATION OF STATE NAMES FOR NODE "V26" IN THE NETWORK

State Name	Lower bound	Upper bound
x_-INF_2_42_(Low)	-∞	2.42
x_2_42_4_66_(Medium)	2.42	4.66
x_4_66_INF_(High)	4.66	∞

Multiple continuous valued records are simulated for each discrete record in the generated data set. As our discretization method requires at least one node to be in discrete state prior to the process, two nodes (”V26”, ”V56” that have only prior distribution) are kept in the original discrete form, with an assumption that an expert can provide discrete range of certain critical nodes in the Bayesian network. Once continuous attribute data set is generated, the proposed discretization process is executed on this data set and it is compared against the original discrete data set and the results are presented in the next section.

C. *Low information loss during inference*

To analyze the effectiveness of the proposed method, the Bayesian network is learned separately on both original simulated discrete data set and generated discrete data set from the proposed method. Now both the learned networks are compared based on the inference results (marginal distribution) for target nodes when similar sets of evidence are applied on the network. Table II shows inference results based on 6 different types of evidences on the Bayesian network.

Each set of inference results shows post marginal distribution of selected target nodes when the Bayesian network is updated based on a set of evidences. All the 6 sets of inference results suggest that the results obtained by the learned network on both original and predicted discrete data are very similar. Close inspection of these results shows that even though there are some variations in the post marginal probability, the order and skewedness of the distribution remains unchanged. This indicates that the information loss due to discretization is very minimal.

D. *Time Complexity Optimization*

The advantage of our approach is two-fold: (i) Optimization of Entropy computation and (ii) Generation of optimized candidate partitions for continuous data points using dynamic approach.

TABLE II. COMPARISON BETWEEN ORIGINAL AND PREDICTED DISCRETE DATA FROM INFERENCE RESULTS ON SAME EVIDENCE

	Original Data	Predicted Data
Evidence = [V26(L=1),V56(H=1)]		
Target V57	H=0.963,M=0.037,L=0.0	H=0.956,M=0.044,L=0.0
Target V66	H=0.009,M=0.902,L=0.089	H=0.013,M=0.884,L=0.103
Target V82	H=0.985,M=0.014,L=0.001	H=0.984,M=0.014,L=0.002
Evidence = [V26(H=1),V56(M=1)]		
Target V57	H=0.016,M=0.962,L=0.021	H=0.010,M=0.962,L=0.028
Target V66	H=0.073,M=0.922,L=0.005	H=0.050,M=0.939,L=0.011
Target V82	H=0.212,M=0.775,L=0.013	H=0.152,M=0.776,L=0.072
Evidence = [V56(H=1),V22(L=1)]		
Target V65	H=0.008,M=0.265,L=0.727	H=0.002,M=0.231,L=0.767
Target V69	H=0.988,M=0.012,L=0.000	H=0.986,M=0.012,L=0.002
Target V73	H=0.501,M=0.488,L=0.011	H=0.525,M=0.462,L=0.013
Evidence = [V56(M=1),V22(H=1)]		
Target V65	H=0.274,M=0.709,L=0.017	H=0.144,M=0.855,L=0.001
Target V69	H=0.585,M=0.408,L=0.007	H=0.617,M=0.369,L=0.014
Target V73	H=0.274,M=0.709,L=0.017	H=0.133,M=0.685,L=0.183
Evidence = [V26(L=1),V56(H=1)]		
Target V59	H=0.996,M=0.004,L=0.000	H=0.996,M=0.004,L=0.000
Target V65	H=0.008,M=0.265,L=0.726	H=0.002,M=0.221,L=0.777
Target V75	H=0.999,M=0.001,L=0.001	H=0.711,M=0.285,L=0.004
Evidence = [V56(H=1),V22(M=1)]		
Target V59	H=0.501,M=0.493,L=0.006	H=0.560,M=0.432,L=0.060
Target V65	H=0.274,M=0.708,L=0.018	H=0.001,M=0.956,L=0.043
Target V75	H=0.030,M=0.822,L=0.148	H=0.009,M=0.830,L=0.161

1) Optimization of Entropy computation: As described in section III-E, our approach optimizes the time complexity by reducing the computation of all intermediate Entropy measures, which is exactly $\frac{n \;\; n+1 \; - k\,(k-1)}{2}$. Here, n is the number of records and k is the number of partitions required. So time complexity of computing all intermediate entropies is reduced from $O(k * n^k)$ to $O(n^2)$. This reduction is very significant for large value of n and k. Moreover, the time complexity of our method is independent of k, i.e., number of partitions required.

Consider the matrix given in Figure 2. Assume a case where we have 8 data points and we need 4 partitions. The lower triangle shown in solid colored cells is redundant Entropy measures, as the computations are same as upper

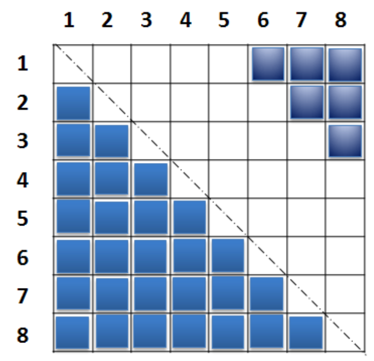

Figure 2. Pre-Storing of entropy measures in a matrix

triangle of the matrix ($\frac{n(n+1)}{2}$ cells). Further, consider cell (1, 6). The partition corresponding to this is not a valid Entropy measure, as there are only two data points left after this to be filled into 3 partitions, which is not possible. Similarly, the cells shown in solid colors on top-right corner ($\frac{k(k-1)}{2}$ cells) of the matrix are invalid Entropy measures, hence discarded. Thus the total number of valid computation of Entropy measures is $\frac{n \;\; n+1 \, - \, k \, (k-1)}{2}$.

2) Generation of optimized Candidate Partitions : In Algorithm 3 of section III-F, we also described the dynamic approach for finding all possible split points for partitioning the continuous data. Suppose, if we follow a simple approach by generating all candidate partitions without using dynamic approach, then total number of accesses to the entropy matrix M (discussed in section III-E) is $C_{k-1}^{n-1} \times k$, where as by following the dynamic approach provided in Algorithm 3, the total number of accesses to matrix M is

$$C_{k-1}^{n} + C_{k-1}^{n-1} - 1.$$

An analytical example is provided in Fig. 3. In this example, the size of records (n) is 6 and number of buckets is 4.

Similarly, if the size of n is 10 and k is 4, non-dynamic approach needs 336 accesses to matrix M, where as dynamic approach need only 203. This difference further increase as the size of n and k increases.

V. Conclusion

Most of the learning networks (such as Bayesian networks) need discretization of continuous data. In this paper, we presented a dynamic discretization method using entropy measure specific to Bayesian network. We modified the conventional method of partitioning using only single class variable to accommodate it in the context of Bayesian model. The time complexity of our approach is minimized by following pre-storage and dynamic approach. The results of inference using the output discrete data in the network show the viability of our approach. The results also show the strong interdependencies among the nodes in the network.

References

[1] James Dougherty, Ron Kohavi, Mehran Sahami Supervised and Unsupervised Discretization of Continuous Features. In Proceedings of the Twelfth International Conference on Machine Learning (1995), pp. 194-202.

[2] Michal R. Chmielewski and Jerzy W. Grzymala-Busse Global discretization of continuous attributes as preprocessing for machine learning. In International Journal of Approximate Reasoning, Volume 15, Issue 4, November 1996, Pages 319- 331.

[3] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In IJCAI93, pp. 1022-1027, 1993.

[4] Friedman N., Goldszmidt M. Discretizing Continuous Attributes While Learning Bayesian Networks. In: ICML, 1996.

[5] Hartemink A, Gifford D, et al. Combining Location and Expression Data for Principled Discovery of Genetic Regulatory Networks. In:PSB, 2002.

[6] Monti S, Cooper GF. A Multivariate Discretization Method for Learning Bayesian Networks from Mixed Data. In: UAI, 1998.

[7] Steck H, Jaakkola T. (Semi)-Predictive Discretization during Model Selection. AI Memo AIM-2003-002, 2003.

[8] D. L.Boley. Principal Direction Divisive Partitioning, Data Mining and Knowledge Discovery 2(4):325-344, 1998.

[9] S. Savaresi, D. Boley, S. Bittanti, and G. Gazzaniga. Choosing the cluster to split in bisecting divisive clustering algorithms. In Second SIAM International Conference on Data Mining (SDM'2002), pp. 299314, 2002.

[10] Jain, A.K., R.C. Dubes (1988). Algorithms for clustering data. Prentice-Hall advance reference series. Prentice-Hall, Upper Saddle River, NJ.

[11] Kerber, R. ChiMErge: Discretization of Numeric Attributes Learning: Inductive, AAAI 92, pp. 123-128. 1992.

[12] Quinlan, J.R. C4.5: Programs for Machine Learning, Morgan Kaufmann California, 1993.

[13] Quinlan, J.R. Introduction to decision trees. Machine Learning 1,pp. 81-106, 1986.

[14] P. Perner, S. Trautzsch, In: Advances in Pattern Recognition, A. Amin, D. Dori, P. Pudil, and H. Freeman (Eds.), Springer Verlag 1998, LNCS 1451, pp. 475-482.

[15] T.D. Nielsen, F.V. Jensen On-line alert systems for production plants: A conflict based approach. International Journal of Approximate Reasoning, 2007.

[16] Bayesian networks Repository. http://bndg:cs:aau:dkhtml/bayesian networks:html