Learning Decision Trees From Histogram Data

Ram B. Gurung
Dept. of Computer and Systems Sciences
Stockholm University, Sweden
Email: gurung@dsv.su.se

Tony Lindgren
Dept. of Computer and Systems Sciences
Stockholm University, Sweden
Email: tony@dsv.su.se

Henrik Boström
Dept. of Computer and Systems Sciences
Stockholm University, Sweden
Email: henrik.bostrom@dsv.su.se

***Abstract*—When applying learning algorithms to histogram data, bins of such variables are normally treated as separate independent variables. However, this may lead to a loss of information as the underlying dependencies may not be fully exploited. In this paper, we adapt the standard decision tree learning algorithm to handle histogram data by proposing a novel method for partitioning examples using binned variables. Results from employing the algorithm to both synthetic and real-world data sets demonstrate that exploiting dependencies in histogram data may have positive effects on both predictive performance and model size, as measured by number of nodes in the decision tree. These gains are however associated with an increased computational cost and more complex split conditions. To address the former issue, an approximate method is proposed, which speeds up the learning process substantially while retaining the predictive performance.**

***Index Terms*—histogram learning, histogram tree**

I. Introduction

Standard machine learning algorithms are designed for handling data represented by numeric and categorical variables. Even in cases when it is known that the data does have some structure, e.g., some groups of variables are related, such information is lost when the data is encoded as ordinary numeric and categorical variables and provided as input to the standard learning algorithms. One particular type of structure that we focus in this paper is histogram data, i.e., sets of variables representing the frequency distributions of some (implicit) variables. For example, we may use three variables (bins) to represent the relative frequency distribution of days during a month with average temperature lower than zero degrees, between zero and twenty degrees, and above twenty degrees. Histogram data is frequently encountered in domains where multiple observations are aggregated. One reason for aggregating data can simply be to save storage space, e.g., when dealing with big data, while in other cases the aggregation is necessary for being able to represent all data points (observations) on the same format, i.e., with the same number of variables. For example, if each customer in a database corresponds to one data point, where information on the purchase amounts should somehow be represented, then since the number of purchases may vary from customer to customer, each single purchase cannot be represented by a unique variable without introducing problems with missing variables and undesired ordering effects. Instead, the information can readily be represented by a histogram, e.g., where the different bins correspond to intervals for the purchase amounts. Histograms are also widely used to aggregate data streams where data are collected over time, e.g., readings in sensor networks.

Research on complex data structures, such as histograms, has been undertaken within the field of symbolic data analysis (SDA) [1]. Symbolic data represents complex data types which do not fall under the traditional categories of numeric and categorical variables. One specific type of histograms that have been studied are categorical in nature with a relative frequency assigned to each bin. Such histograms are classified as modal multi-valued variables in the terminology of the SDA framework, while Diday [2] refers to such histograms as categorical histogram data. More formally, for observations with n categorical histogram variables $X_i, i = 1...n$, with m_i bins $x_{ij}, j = 1...m_i$ each bin is assigned a relative frequency r_{ij} such that $\sum_{j=1}^{m_i} r_{ij} = 1$ and each observation is associated with a class label Y. For all observations, bin descriptions of a histogram variable are identical. This is the type of histogram we will be considering in this study.

Research on learning from histogram data is still at an early stage. To the best of our knowledge, no studies have been published on learning classifiers from histogram data. However, there have been some studies on applying linear regression [3], [4], PCA [5] and clustering [6] to histogram data. While most of the considered approaches take into account the actual bin boundaries, the work on adapting PCA for categorical histogram data [5] deals with data of the same type as considered here. It should be noted, however, that the approach in [5] is aimed for dimensionality reduction and not for performing classification. The type of histogram data considered in this study and in [5] is closely related to ”compositional” variables within compositional data analysis [7], where weights associated with each variable represent distributions over possible values. However, the research in compositional data analysis has not been on learning classifiers.

In this paper, we will propose an adaptation of the standard decision tree algorithm [8] to allow for learning from categorical histogram variables. We will compare the performance of the adapted learning algorithm to using the standard learning algorithm with histogram data represented by ordinary variables, i.e., with no structural information. The main contributions of the paper are:

- A novel approach for learning decision trees from histogram data, including an approximation to allow for substantial speedup
- An empirical evaluation comparing the new approach to the standard decision tree learning algorithm on both synthetic and real-world data sets
- Findings concerning the utility of exploiting the structure in histogram data when learning decision trees

In the next section, the novel approach for learning decision trees from histogram data is presented. In Section III, the experimental setup and results are presented. The empirical findings and limitations of the proposed approach are discussed in Section IV. Finally, in Section V, we summarize the main conclusions and point out directions for future research.

II. Method

The standard decision tree algorithm [8] was adapted to learn from histogram data. Therefore, the approach can be viewed as a generalization of the standard algorithm where the bins of each histogram is handled as a vector and partitioning takes place by finding a separating hyperplane in the corresponding space. Fig. 1 provides an illustration of the approach. The best split plane for each histogram variable is obtained and then compared to the best splits of the other histogram variables, as well as to splits obtained from the regular numeric and categorical variables. The split with the highest information gain is finally selected for partitioning the examples in the node. Similar approaches to employing multivariate splits have been proposed in the past, e.g., using linear combination of multiple features to perform splits at each intermediate nodes [9], [10]. In these approaches, all the features are considered simultaneously for splitting, while in our case, multiple variables considered for a split are bins of same histogram variables with unit sum constraint. Hence, there can be more than one histogram variable in a dataset that would require evaluation of multiple multivariate splits.

In this section, we first provide a formalization of the node-splitting part of the adapted decision tree learning algorithm and then illustrate its workings with two very simple examples. We proceed by providing an analysis of the computational complexity of the algorithm and end the section by proposing an approximation of the original method for speeding up the node-splitting process.

A. The node-splitting algorithm

The aim of the algorithm is to find the optimal node splitting hyperplane. Because of the unit sum constraint, a histogram with m bins is a vector point that lies in a hyperplane, which is represented by

$$x_1 + x_2 + ... + x_m = 1 \tag{1}$$

Let the equation for the linear splitting hyperplane be

$$c_1x_1 + c_2x_2 + c_3x_3 + ... + c_mx_m = 1 \tag{2}$$

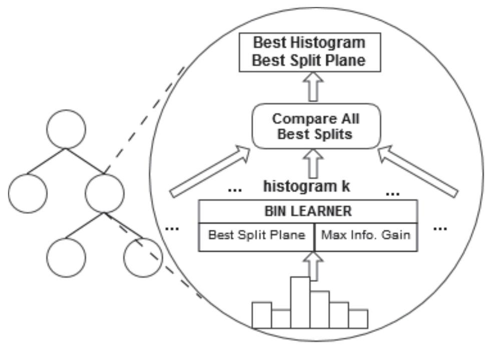

Fig. 1. Overview of the node splitting approach

where $C = (c_1, c_2, c_3, ..., c_m)$ are the unknown coefficients to be solved. Hyperplanes represented by equation 1 and 2 are assumed to be orthogonal, which results in

$$\begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ | \\ c_m \end{bmatrix} = 0$$

$$c_1 + c_2 + c_3 + ... + c_m = 0 \tag{3}$$

Solving for m unknowns in C requires m linear equations. In addition to equation 3, substituting m-1 points for $X = (x_1, x_2, ..., x_m)$ in equation 2 would give sufficient number of equations to solve for C. Selection of m-1 points out of n data points can be done in $\binom{n}{m-1}$ ways. The resulting system of linear equations can be solved as follows.

$$\begin{bmatrix} 1 & 1 & ... & 1 \\ x_{1,1} & x_{1,2} & ... & x_{1,m} \\ | & | & | & | \\ x_{m-1,1} & x_{m-1,2} & ... & x_{m-1,m} \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ | \\ c_m \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ | \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} c_1 \\ c_2 \\ | \\ c_m \end{bmatrix} = \begin{bmatrix} 1 & 1 & ... & 1 \\ x_{1,1} & x_{1,2} & ... & x_{1,m} \\ | & | & | & | \\ x_{m-1,1} & x_{m-1,2} & ... & x_{m-1,m} \end{bmatrix}^{-1} \times \begin{bmatrix} 0 \\ 1 \\ | \\ 1 \end{bmatrix}$$

Algorithm 1 specifies the node splitting process. For an m binned histogram, m-1 vector points are chosen in $\binom{n}{m-1}$ ways. Each combination is examined for a split plane (lines 5 to 24). The left hand sides of m linear equations are captured in mXm square matrix A (line 6). The right hand sides of these linear equations are represented as column vector B of size m (line 8). If the inverse of A exists, the product of the inverse of A and B results in coefficients of the split plane (line 9). For all the points in the node, the scalar product of the point and coefficient vector gives a value that determines whether to assign the point to the left or the right node (lines 10 to 17). The information gain obtained from the split can be calculated and compared with the previous best gain (lines 18 to 22).

Algorithm 1 Finding best split plane in a node

```
Input: $obs$: observations in a node
    $histogram\_variables$: names of histogram variables
Output: $best\_split\_plane$: coefficients of best split plane
 1: for all $histogram$ in $histogram\_variables$ do
 2:     $m \leftarrow$ number of bins in $histogram$
 3:     $h\_points \leftarrow histogram$ values in $obs$
 4:     $combinations \leftarrow$ ways of choosing $m$-1 points from $h\_points$
 5:     for all $combn$ in $combinations$ do
 6:         $A \leftarrow$ matrix of $m$-1 points in $combn$ with all elements of first row 1.
 7:         if $A^{-1}$ exists then
 8:             $B \leftarrow$ column vector of $m$-1 ones, first element 0.
 9:             $split\_plane\_coefs \leftarrow$ multiply $A^{-1}$ and $B$
10:             for all $point$ in $h\_points$ do
11:                 $value \leftarrow$ multiply $point$ and $split\_plane\_coefs$
12:                 if $value < 1$ then
13:                     $l\_obs \leftarrow$ assign $point$ to left node
14:                 else
15:                     $r\_obs \leftarrow$ assign $point$ to right node
16:                 end if
17:             end for
18:             $info\_gain \leftarrow$ get information gain of the split
19:             if $info\_gain$ is greater than previous best then
20:                 $best\_info\_gain \leftarrow info\_gain$
21:                 $best\_split\_plane \leftarrow split\_plane\_coefs$
22:             end if
23:         end if
24:     end for
25: end for
```

B. Examples

We illustrate the workings of the algorithm using histogram variables with two and three bins respectively. The left graph in Fig. 2 shows the splitting process when the histogram variable has two bins x_1 and x_2. All the points (x_1, x_2) lie on the line AB. The splitting line CD is orthogonal to AB and passes through a point in AB. The coefficients of CD, a and b, can be determined by solving two linear equations. The process is repeated allowing CD to pass through all the points and choosing the one that gives the highest information gain. The process is similar for a histogram variable with three bins, as illustrated by the right graph in Fig. 2, in which all vector points are spread in the 3-D plane ABC. A three-dimensional splitting plane DEFG can be defined by the equation $ax_1 + bx_2 + cx_3 = 1$. DEFG is orthogonal to ABC and passes through two vector points. Three linear equations on a, b and c can be formed to solve for these unknowns.

Figure 3 shows a 3-D scatter plot for a small sample set of 100 observations that has a histogram variable with three bins x_1, x_2 and x_3. A simple pattern was injected in the data, if $x_1 + x_3 < 0.3$ then class label $y = 1$ else $y = 0$. Green

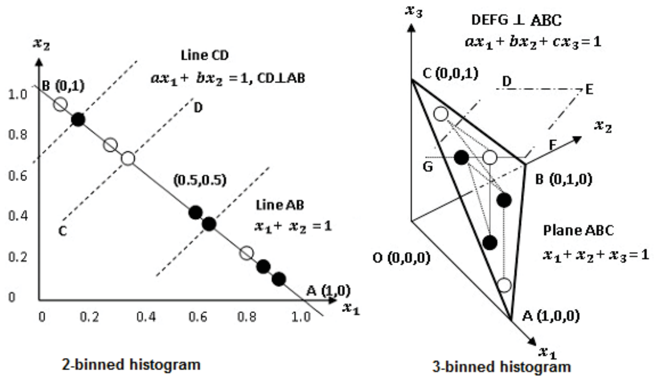

Fig. 2. Splitting in two and three binned histogram

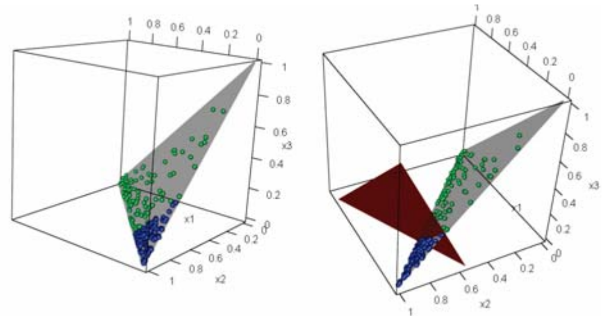

Fig. 3. Split plane in 3-binned histogram variable

and blue points correspond to the negative and positive cases respectively. The red plane is the splitting plane discovered by the algorithm. The equation of this splitting plane turns out to be: $0.885x_1 - 1.769x_2 + 0.885x_3 + 1 = 0$. Projecting the plane intothe 2-D plane of x_1 and x_3 would result in $x_1 + x_3 = 0.29$ which is approximately the same pattern injected in the data set. More experimental results will be presented in Section III.

C. Computational complexity

The computational cost of the proposed approach increases as the number of observations n and number of bins m increase. The cost is due to the combinatorial explosion of having to evaluate $\binom{n}{m-1}$ combinations while searching for the best separating hyperplane. Therefore, the computational complexity of the algorithm is proportional to $O(n^m)$.

D. Speeding up the node-splitting process

The computational cost can be reduced by either limiting the number of bins m or the number of observations n or both. The former was in this study handled in a straightforward manner, by merging bins for details see *Real-world Data* in the Experiments section. The latter, i.e., limiting the number of observations, was addressed using a more elaborate approximation method.

1) Approximation Approach: Each observations in a node for a histogram variable can be considered as a point in a m-dimension space, m being the number of bins. So, for convenience, observations in a node shall be referred as points henceforth. In this approach, instead of using all the points in a node to build and evaluate splitting planes as described in algorithm 1, we generate small number of candidate split points and then build and evaluate split planes out of those newly generated points. It should be noted that new points are synthetically generated from original ones as will be described later in algorithm 2. The parameter num_points is the number of such candidate split points, as a consequence the algorithm only needs to consider $\binom{num\_points}{m-1}$ combinations. By choosing $num\_points < n$, the computational cost can be significantly reduced. In order for such approximate plane to make a good split, the new points that the plane passes through should be carefully generated.

As shown in the right half of figure 4, if there exist an optimal decision boundary, we want $best\_split\_plane$ to pass along this decision boundary as shown by red line. This is only possible when newly created synthetic split points lie close to the optimal decision boundary as shown by asterisks (*). Therefore, the first step in the approximation approach is to generate new candidate split points that are likely to fall around optimal decision boundary. Algorithm 2 describes the process of generating new synthetic candidate split points.

The algorithm first tries to locate the boundary regions since new synthetic points should come from such regions. This is done by examining the neighborhood around each point. For each point, a certain number e.g. 10 nearest neighbors are taken to form a group, which we shall simply refer as a cluster. The size of a cluster i.e. the number of nearest neighbors around the point is treated as parameter (N_c). Basically, the algorithm builds a cluster of N_c neighboring points around each point in a node. If the cluster lies in boundary region, its member points will be of different classes. As shown in the left half of figure 4, cluster A and C have all its members of same class whereas cluster B has a mix of both classes. So, cluster B is an ideal type of clusters that the algorithm prioritizes. The entropy value of the cluster determines how ideal the cluster is. Higher entropy values are preferred. Once the entropy of the clusters is calculated, they are prioritized according to the entropy value. A certain number of best clusters (num_points) e.g. 15 are selected and their centers are obtained. The cluster centers are used as new synthetic candidate split points to build and evaluate an approximate split plane.

If we were to use a standard clustering algorithm e.g. K-means it probably would result in new split points from regions that are not useful in finding optimal splits, as K-means do not focus on finding clusters with high class entropy (actually it would penalize different classes). As shown in the left half of figure 4, split points around the region of cluster A and C will not contribute much in finding optimal split. However, evaluating these points to search for an optimal split plane would consume valuable search time. Therefore, the tailored approach of obtaining relevant split points as explained in the algorithm 2 was preferred.

A scatter plot of a small sample training set with three binned histogram is shown in figure 4. Blue and green points correspond to examples from each of the two classes, respectively. In the right part, the points marked with an asterisk (*) are the candidate split points generated by using algorithm 2. The approximate split plane obtained by using the algorithm is shown as red line which passes through two of these split points. In the left part, three clusters A, B and C are shown just for illustration with some cluster centers marked with asterisks (*).

Algorithm 2 Finding split points around decision boundary

```
Input: $h\_points$: observations for a histogram variable
    $class$: class label of $h\_points$
    $num\_points$: number of split points required
    $N_c$: number of nearest neighbors to consider
Output: $split\_points$: candidate split points
 1: if $|h\_points| > num\_points$ then
 2:     for all $point$ in $h\_points$ do
 3:         $cluster \leftarrow$ find $N_c$ nearest points around $point$
 4:         $center \leftarrow$ find center of $cluster$
 5:         $entropy \leftarrow$ get entropy of $cluster$ using $class$
 6:         $list \leftarrow$ save $center$ and $entropy$ in a list
 7:     end for
 8:     $d\_list \leftarrow$ sort $list$ in descending order of entropy
 9:     $new\_points \leftarrow$ get top $num\_points$ centers in $d\_list$
10:     $split\_points \leftarrow new\_points$
11: else
12:     $split\_points \leftarrow sample\_points$
13: end if
```

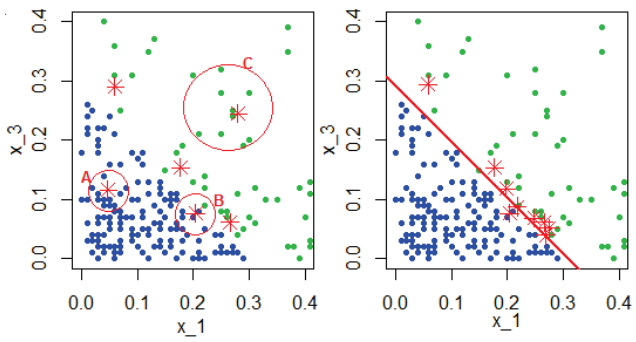

Fig. 4. Left: Generating split points, Right: Forming splitting plane

III. Experiments

The proposed approach has been implemented in R[1]. Experiments were performed on a synthetic, a semi-synthetic and a real-world data. The bin values in the synthetic data set were obtained using uniform random sampling. The bins of a histogram were later normalized to satisfy the unit sum constraint. Synthetic dependencies among the bins were then injected by labeling the examples according to a set of formulated rules. The purpose of synthetic data set is to show that the algorithm can exploit the dependencies in the bins better by treating them together compared to when bins are treated individually.

[1]http://www.r-project.org/

The semi-synthetic data set was derived from the publicly available 'iris' data set [11] where original numeric variables were converted into histogram variables. The purpose of the semi-synthetic experiment was to investigate the robustness of the algorithm when bins have no inherent dependencies, i.e., the class labels are not dependent on interactions among the bins. The real-world data was provided by the heavy truck manufacturing company Scania CV AB and consists of histogram variables that describe operational profiles of trucks. Certain pattern among the bins of a histogram might reveal information about the truck's usage that could be associated to the breakdown of various component in the truck. The goal of the algorithm therefore is to discover useful pattern among the histogram bins by treating them together. The predictive performance of both standard decision tree learning algorithm and the proposed approach are compared with respect to classification accuracy measured using cross validation. In addition to accuracy, the tree size, i.e., the number of nodes, is also presented for each method. We here report only the results of applying the histogram approach using the approximation method. Using the exact approach, i.e., using all samples for generating split planes, was practically infeasible due to the excessive computational cost. Brief descriptions of the data sets, the experimental settings and the results observed from the experiment with each data sets are provided in the following sub-sections.

A. Synthetic Data

Two synthetic datasets, each consisting 1000 observations with equal proportions of observations labeled as positive and negative, were considered separately in two different experiments. First dataset consists of single histogram variable X with 4 bins. Simple pattern was injected in this dataset: if $X_1 + X_3 < 0.3$ then target class variable $Y = 1$, otherwise $Y = 0$. X_1, X_2, X_3 and X_4 are the bins of histogram X. Similarly, second dataset has two histogram variables; $X1$ with four bins and $X2$ with five bins. A more complex pattern was injected in the data which involve both histogram variable: if $(X1\_1 + X1\_3 < 0.3$ and $0.3 < X2\_1 + X2\_3 < 0.7)$ then $Y = 1$ else $Y = 0$, where Y is the target (output) variable. For both experiments parameter settings were identical. The termination condition for the tree building algorithm i.e., stop expanding the current node, is when the number of observations in a node drops below 5 or the split does not provide any information gain. Three values for the number of new split points (num_points in algorithm 2) to be used for forming splitting plane, were examined: 7, 11 and 15. This can be any value higher than number of bins but higher value result in longer model training time. In order to cover wider range, these 3 values were chosen. Three different cluster sizes i.e. number of points considered to form a cluster (N_c in algorithm 2), were examined: 10, 15 and 20. 10-fold cross-validation was performed. The outcome of the experiment with the first dataset is presented in table I whereas the results of the second dataset is presented in table II. The columns of tables respectively show the number of points used for approximating optimal split, the size of the cluster considered for generating new split point, the tree size and the accuracy, where the latter two correspond to averages over the ten folds.

TABLE I
SYNTHETIC DATASET: FOUR BINS

Split Points	Cluster Size	Tree Nodes	Accuracy
7	10	8.2	99.6
7	15	11.6	99.8
7	20	11	99.7
11	10	7	99.4
11	15	9.4	99.2
11	20	8.8	99.2
15	10	5.6	99.6
15	15	8.2	99.6
15	20	8.6	99.4
Bins Treated Individually (Standard Tree Algorithm)			
—	Decision Tree	38.8	98.2

TABLE II
SYNTHETIC DATASET: FOUR AND FIVE BINS

Split Points	Cluster Size	Tree Nodes	Accuracy
7	10	29.2	97.3
7	15	41.6	95.7
7	20	38.2	96.3
11	10	13.6	99.3
11	15	16.2	98.1
11	20	17	97.9
15	10	11.4	98.7
15	15	11.2	98
15	20	11	98.6
Bins Treated Individually (Standard Tree Algorithm)			
—	Decision Tree	50.8	95.9

B. Semi-synthetic Data

A semi-synthetic dataset was generated from the publicly available Iris dataset [11]. The dataset has four numeric variables: petal length, petal width, sepal length and sepal width. Each observation belongs to one of three classes: Iris-versicolor, Iris-setosa and Iris-virginica. The data set contains 150 observations, 50 from each class. Each of the four variables was used to generate a synthetic histogram variable by including two additional variables such that they satisfy unit sum constraint. For example, in order to transform numeric variable petal length into histogram variable, it was first normalized to lie between 0 and 1. Let it be X_1. For each X_1, two integers in the range of 1 to 100 were uniform randomly selected. These two integers, X_2 and X_3 were then scaled down as:
$X_2 => X_2 * (1 - X_1)/(X_2 + X_3)$
$X_3 => X_3 * (1 - X_1)/(X_2 + X_3)$
The new dataset hence has four histogram variables, each with three bins. 5-fold cross validation was performed. The same termination condition for the tree growing as applied in the previous experiment was used. Number of split points used for searching split planes were: 5, 7 and 9. The chosen number should be higher than number of bins in the histogram variable. Model training time would increase as we select higher numbers, so for simplicity only three values were

TABLE III
SEMI-SYNTHETIC: IRIS DATASET

Split Points	Cluster Size	Tree Nodes	Accuracy
5	10	13	91.67
5	15	12.6	92.33
5	20	13.8	89
7	10	12.2	91
7	15	10.6	89
7	20	11.4	85
9	5	11.4	91
9	10	10.6	92.33
9	15	10.6	88.33
Bins Treated Individually (Standard Tree Algorithm)			
—	Decision Tree	9.4	92.67

TABLE IV
OPERATIONAL DATA SET

Split Points	Cluster Size	Tree Nodes	Accuracy
13	10	82.4	59.17
13	15	68.8	59.33
13	20	77.8	57.83
15	10	73	57.17
15	15	65.8	56.17
15	20	68	58.67
17	10	67.8	61.67
17	15	64	59.83
17	20	58.2	57.33
Bins Treated Individually (Standard Tree Algorithm)			
—	Decision Tree	106.8	59

examined. Three different cluster sizes i.e. number of points used to form a cluster, were arbitrarily chosen: 5, 10 and 15. The results of the experiment are shown in table III.

C. *Real-world Data*

Each observation in operational data is a snapshot of operational profile of a truck. Histogram variables in operational data holds information about how often the truck had operated under a particular feature range. The histogram variable, for example temperature, has 10 bins. Each bin measures the number of times the truck had operated within certain ambient temperature range. Temporal information about the truck's operation at various ambient temperature is transformed into relative frequency count as histogram over time. The histogram transformed data is extracted from the truck when it visits workshop for maintenance. Certain patterns within the bins of a histogram might reveal useful information about the truck's usage that are related to breakdown of certain components. The objective of the experiment is to distinguish trucks with battery failure from those without failure by using histogram feature variables. Bins of the histograms are normalized. The original data set had 33603 observations spread along 308 variables (counting bins as independent variables). There are 17 histogram variables of various length. The data set is very sparse and skewed in terms of battery failure as class label. Out of the 33603 observations, only 720 had battery problems.

For experimental purposes, a smaller data set was extracted. Given the computational cost that would incur, it was a necessary step. Out of these, only four histogram variables which were deemed as important were selected. Only histogram variables were selected for the experiment because the purpose here was to compare the performance when training tree as histogram against training by treating the bins individually. So the influence from any other variables either numeric or categorical was not desired. Issues related to missing values and skewed class distribution were set aside by including observations that had no missing values and selecting an equal number of positive and negative cases. Finally, a data set with 300 positive and 300 negative cases was extracted for the experiment. For confidentiality purpose, original variables are anonymised.

First histogram variable has 9 bins. Originally this variable was a 6X6 matrix. In order to handle computational complexity, adjoining 4 cells were merged resulting in 3X3 matrix. These 9 cells were then treated as bins. Second histogram variable also has 9 bins which was similarly transformed from 6X6 matrix. Third and fourth histogram variables have 10 bins. 10-fold cross validation was performed. The stop criteria was set to 15 or less observations in a node or if the split did not ensure any information gain. Three values for the number of split point used for forming split plane were examined:13, 15 and 17. Three different sizes of cluster (i.e. number of neighboring points used to form a cluster) were examined: 10, 15 and 20. The result of the experiment is shown in table IV and the description of the table is same as those in table II. Results are discussed further in *Discussion* section.

IV. DISCUSSION

Results in the tables I, II, III, IV show the performance of the algorithm at various parameter settings for four data sets. Performance of the standard decision tree approach has been shown at the bottom of the table as a base line. As observed from the synthetic data experiment results in table I and table II, the proposed method is better or at least as good as standard decision tree algorithm for all the parameter settings examined. Size of the tree in terms of number of nodes is significantly smaller than that of standard tree. When the number of approximate split points (cluster centers) increases, number of tree nodes decreases while accuracy increases as expected. Influence of the size of the cluster is however not clearly evident in the results.

Purpose of the experiment on semi-synthetic iris data set was to examine the robustness of the algorithm when the bins of histogram do not have any inherent dependencies. The result as shown in table III suggest that the performance of the proposed method does not suffer heavily because of non-informative bins. Accuracy performance was almost comparable with base line performance except in some cases where accuracy drops by around 7%. One reason for this high variation in performance could probably be attributed to the small size of data set which is only 150. On the best parameter settings, accuracy raised up to 92.33% which was very close to base line performance of 92.67%.

Experiment results on operational data presented in table IV could not decide clear winner. With some parameter settings, average accuracy of proposed method exceeded base line performance while at other instances performance dropped well below the base line. It should however be noted that since the purpose of the experiment is the comparison of proposed approach against standard tree method, poor performance in both approach should not be a concern. This poor performance could be attributed to the smaller size of dataset or insignificance of variables selected. Since, some of the variables were heavily reduced in size by merging the bins together, we might have lost the information about the patterns inherent in those bins. Presence of such inherent dependency among bins is where the proposed method thrives on. This probably could be one reason why proposed method could not perform better all the time.

Considering all four experiments as a whole, out of 36 occasions, histogram approach was better in 21 occasions although most of the wins were from synthetic data experiments. Although promising, there are some downsides to this approach. The histogram approach has some inherent limitations. The first limitation is due to number of bins, the higher the number of bins the higher the computational complexity. So, somehow, higher number of bins have to be merged to get fewer bins which will result in information loss. Another inherent limitation of the approach lies in the least number of observations required at each node for making a split decision. Since, solving the system of linear equations lies at the heart of the model, at least as many observations are needed as there are number of bins in order to be able to solve such a system. One of the inherent limitation of the proposed method is that it assumes linear separation in the data and tries to approximate linear separation when the decision boundary is nonlinear. This linear approximation of possibly nonlinear pattern in histogram variables in operational data could be another reason why the method was not always the winner.

Probably the most prominent limitation of the approach lies in interpretation of split condition. Unlike in standard decision tree where nodes store information about best split variable and split point, here each node stores the information about the best split plane. It is usually difficult to interpret the split plane in the context of the variable.

V. Concluding Remarks

Standard learning algorithms are designed to learn from numeric and categorical variables. However, practitioners in both academia and industry often have to deal with more complex variables. Histograms in the form of frequency distributions is one of such example. To the best of our knowledge, no previous learning algorithms have been designed to generate classifiers from histogram variables. Instead, bins of histograms are commonly handled as separate variables, ignoring the underlying structure. In this paper, we adapted the standard decision tree learning algorithm to learn from histogram variables. Experimental results from both synthetic and real-world datasets would suggest that gains in terms of predictive performance and a reduction of the number of nodes might be obtained by exploiting the underlying structure of histogram variables.

Although encouraging, the proposed approach could be refined in various ways. Some directions for improvement include investigating techniques for efficiently handling histogram variables with large numbers of bins which at the moment is addressed simply by merging some of them thereby losing information. Comprehensive study of comparing the performance of the proposed method against existing multivariate split methods can be done in future. This study has shown one of the ways to extend the basic decision tree learning algorithm to handle histogram data; directions for future research include similar extensions of the numerous other standard learning algorithms, e.g., SVMs, random forests etc.

Acknowledgment

FFI
Fordonsstrategisk
Forskning och Innovation

This work has been funded by Scania CV AB and the Vinnova program for Strategic Vehicle Research and Innovation (FFI)-Transport Efficiency.

References

[1] L. Billard and E. Diday, *Symbolic Data Analysis: Conceptual Statistics and Data Mining*, ser. Applied Optimization. Chichester, England ; Hoboken, NJ: John Wiley and Sons Inc., 2006.

[2] E. Diday, “Principal component analysis for categorical histogram data: Some open directions of research,” in *Classification and Multivariate Analysis for Complex Data Structures*, ser. Studies in Classification, Data Analysis, and Knowledge Organization, B. Fichet, D. Piccolo, R. Verde, and M. Vichi, Eds. Springer Berlin Heidelberg, 2011, pp. 3–15. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13312-1_1

[3] A. Irpino and R. Verde, “Linear regression for numeric symbolic variables: an ordinary least squares approach based on wasserstein distance,” 2012.

[4] S. Dias and P. Brito, “Distribution and Symmetric Distribution Regression Model for Histogram-Valued Variables,” *ArXiv e-prints*, Mar. 2013.

[5] P. Nagabhushan and R. Pradeep Kumar, “Histogram pca,” in *Advances in Neural Networks ISNN 2007*, ser. Lecture Notes in Computer Science, D. Liu, S. Fei, Z. Hou, H. Zhang, and C. Sun, Eds. Springer Berlin Heidelberg, 2007, vol. 4492, pp. 1012–1021. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72393-6_120

[6] A. Irpino and R. Verde, “A new wasserstein based distance for the hierarchical clustering of histogram symbolic data,” in *Data Science and Classification*, ser. Studies in Classification, Data Analysis, and Knowledge Organization, V. Batagelj, H.-H. Bock, A. Ferligoj, and A. iberna, Eds. Springer Berlin Heidelberg, 2006, pp. 185–192. [Online]. Available: http://dx.doi.org/10.1007/3-540-34416-0_20

[7] J. Aitchison, *The Statistical Analysis of Compositional Data*. London: Chapman and Hall, 1986.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[9] P. E. Utgoff and C. E. Brodley, “An incremental method for finding multivariate splits for decision trees,” in *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 1990, pp. 58–65.

[10] I. Sethi and J. Yoo, “Design of multicategory multifeature split decision trees using perceptron learning,” in *Pattern Recognition*, vol. 27, pp. 939–947.

[11] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml