Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Learning Decision Trees from Histogram Data Using Multiple Subsets of Bins

Ram B. Gurung, Tony Lindgren and Henrik Boström
Dept. of Computer and Systems Sciences,
Stockholm University, Sweden
Email: gurung@dsv.su.se, tony@dsv.su.se, henrik.bostrom@dsv.su.se

Abstract

The standard approach of learning decision trees from histogram data is to treat the bins as independent variables. However, as the underlying dependencies among the bins might not be completely exploited by this approach, an algorithm has been proposed for learning decision trees from histogram data by considering all bins simultaneously while partitioning examples at each node of the tree. Although the algorithm has been demonstrated to improve predictive performance, its computational complexity has turned out to be a major bottleneck, in particular for histograms with a large number of bins. In this paper, we propose instead a sliding window approach to select subsets of the bins to be considered simultaneously while partitioning examples. This significantly reduces the number of possible splits to consider, allowing for substantially larger histograms to be handled. We also propose to evaluate the original bins independently, in addition to evaluating the subsets of bins when performing splits. This ensures that the information obtained by treating bins simultaneously is an additional gain compared to what is considered by the standard approach. Results of experiments on applying the new algorithm to both synthetic and real world datasets demonstrate positive results in terms of predictive performance without excessive computational cost.

Introduction

The standard decision tree learning algorithm is trained on data represented by numeric or categorical variables (Quinlan 1986; Breiman et al. 1984). The histogram decision tree learning algorithm is an extension of this standard decision tree algorithm that considers also the data in the form of histogram variables (Gurung, Lindgren, and Boström 2015) on the following form: each histogram variable $X_i, i = 1...n$, has m_i bins $x_{ij}, j = 1...m_i$ where each bin is assigned a relative frequency r_{ij} such that $\sum_{j=1}^{m_i} r_{ij} = 1$. Each single bin can of course be handled as a single feature, which means that the standard decision tree learning approach can be applied to histogram data. However, the extended algorithm was introduced to exploit dependencies among the bins, which are ignored when bins are represented as ordinary variables.

Research on histogram variables are not in abundance and this is also the case for research on learning classifiers from histogram variables. One exception to this is (Gurung, Lindgren, and Boström 2015), in which an algorithm for constructing decision trees that utilize histogram variable(s) is introduced. In that study, results from employing the histogram tree algorithm to both synthetic and real-world datasets were presented, showing that the exploitation of dependencies in histogram data may have positive effects on both predictive performance and model size. These gains were however associated with an increased computational cost, mainly as a result of the employed brute force approach for finding the best splitting hyperplane. In order to reduce this complexity, an approximation was proposed, which considered only a small number of data points to generate splitting hyperplanes. This approximation method was useful for handling large numbers of observations (n) in a node. However, the computational complexity also depends on the size (number of bins) of histogram variable. A straightforward approach of compromising the resolution of histogram by merging some bins was performed. This however would risk in giving away potentially useful information.

In this paper, we propose an extension of the histogram tree learning algorithm (Gurung, Lindgren, and Boström 2015) to address the important issue of handling large numbers of bins. In addition to this, the heuristic approach to select a small number of data points has been simplified, while at the same time, a technique to refine the approximate splitting hyperplane has been introduced to produce a better split.

In the next section, the proposed method is described in detail. The simplified method for finding relevant split points is first presented, followed by the algorithm for refining the splitting hyperplane. In the *Experiment* section, the experimental setup and results are presented. The empirical findings of the experiment and limitations of the proposed approach are discussed in the *Discussion* section. Finally, in *Concluding Remarks* section, we summarize the main conclusions and point out directions for future research.

Method

In the earlier algorithm for learning decision trees from histogram data (Gurung, Lindgren, and Boström 2015), the number of evaluated splits for a histogram variable X with

m bins, would be $\binom{n}{m-1}$, where n is number of observations. Clearly, the computation cost increases with both n and m. The problem of handling large n was addressed by an approximation that considered only a few candidate datapoints sp ($sp \ll$n) to be used for generating splitting hyperplanes. In order to handle the problem of large m, we here propose that instead of using all m bins simultaneously to find a splitting hyperplane, l bins can be considered, where $1 \leq l \leq m$. However, the algorithm still would have to evaluate $\binom{m}{l}$ combinations of bins which is expensive. So, out of all the possible combinations, we suggest to consider only those combinations that have adjacent bins in sequential order. Assuming that bins that are adjacent to each other are relevant to be considered simultaneously, we can limit the possible combinations to have only bins that are in consecutive order of their bin boundaries. In real world situations, bin boundaries of histograms are often naturally ordered, so considering adjacent bins together might be useful. This leaves us with only $m - l + 1$ sets of bins to evaluate. So, the complexity of finding the best split in a node for a histogram variable X is linear to $\binom{sp}{l}$. Using a smaller subsets of histograms could be useful if some of the bins are non informative or even noisy. The explanation of the algorithm follows in the various subsections below.

Using a sliding window to select subsets of bins

The number of bins of histogram variable to be evaluated simultaneously, l ranges from l_{min} to l_{max} where $1 \leq l_{min} \leq l_{max}$ and $l_{min} \leq l_{max} \leq m$. For a given value of l, sliding window of size l is assumed across an ordered set of bin indices. Bins are ordered according to their bin boundaries. Bins covered by the window are selected as set of bins to be evaluated together for determining best splitting hyperplane. For example, consider a histogram variable of 5 bins. So, the bin indices $b_{idx} = (1, 2, 3, 4, 5)$. Let $l_{min} = 3$ and $l_{max} = 4$ be range of window size l. Then, the set of indices of bins to be evaluated together would be $\{(1, 2, 3), (2, 3, 4), (3, 4, 5), (1, 2, 3, 4), (2, 3, 4, 5)\}$. The first part of the algorithm therefore is to find the set of all bins to be considered together given a range of window size. The procedure is presented in algorithm (1). Bins of histogram are indexed sequentially in variable b_idx. For increasing window size from l_{min} to l_{max}, consecutive bins are selected from b_idx and a list $bin\_combinations$ is formed (lines 2 to 9).

Split points selection method

Observations for each set of bins would be considered as a point in l dimensional space, if l bins are in the set. Only sp points out of n points are used for generating the splitting hyperplane such that $l \leq sp \leq n$. sp is updated as $sp = sp + number\_bins$ (line 1). Each point in l dimension space has a class label. The objective is to find the points around the class boundary region assuming that there exists such a boundary. The algorithm for selecting candidate split points is presented in algorithm 2. The centroid C_+ for all points of the positive class is calculated by taking mean and similarly centroid C_- is calculated for points of the negative class (lines 3 and 4). Euclidean distance of all points of negative class from C_+ and distances of all points of positive class from C_- point are calculated (lines 5 to 12). All the points are ordered in ascending order of its distance from opposite classed centroid (line 13). In the most ideal case, when there is a clear separation between two classes, the points closest to the opposite classed centroid will be the ones near the optimal boundary as shown in figure 1. The sp points from top of the ordered list are selected (line 14).

Algorithm 1 Find sets of bins to evaluate simultaneously

```
Input: $b\_idx$: ordered list of bin indices in a histogram
    $l_{min}$: minimum number of bins to be selected
    $l_{max}$: maximum number of bins to be selected
Output: $bin\_combinations$: list of set of bins to be evaluated
1: $m \leftarrow$ total number of bins, size of $b\_idx$
2: for all $l$ in between $l_{min}$ and $l_{max}$ do
3:     if $l \geq 1$ and $l \leq m$ then
4:         for all $i$ from 1 to $(m - l + 1)$ do
5:             $bin\_set \leftarrow$ indices in $b\_idx$ from $i$ to $(i + l - 1)$
6:             $bin\_combinations \leftarrow$ add $bin\_set$ to the list
7:         end for
8:     end if
9: end for
```

Algorithm 2 Selecting split points for a given set of bins

```
Input: $h\_bins$: indices of bins in selected combination
    $h\_points$: points formed by bins in $h\_bins$
    $hp\_class$: class label of $h\_points$
    $sp$: number of split points required
Output: $split\_points$: candidate split points
1: $sp \leftarrow sp + |h\_bins|$
2: if $|h\_points| > sp$ then
3:     $C_+ \leftarrow$ get centroid of points of positive class
4:     $C_- \leftarrow$ get centroid of points of negative class
5:     for all $point$ in $h\_points$ do
6:         if $point$ is positive class then
7:             $dist \leftarrow$ distance from $point$ to $C_-$
8:         else
9:             $dist \leftarrow$ distance from $point$ to $C_+$
10:        end if
11:        $point\_dist\_list \leftarrow$ add $dist$ for $point$ and save
12:    end for
13:    $asc\_point\_dist \leftarrow$ sort $point\_dist\_list$ by ascending $dist$
14:    $split\_points \leftarrow$ take top $sp$ points in $asc\_point\_dist$
15: else
16:    $split\_points \leftarrow h\_points$
17: end if
```

Figure 1 shows split point selection process for synthetically generated experimental data that has single histogram variable with three bins. Only two bins (first and third) are plotted and the patterns injected is such that if $bin_1 + bin_3 < 0.5$ point belongs to positive class else it belongs to negative class. There is clear boundary between two classes as shown

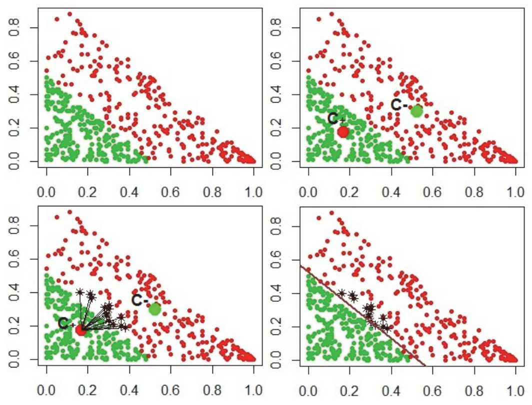

Figure 1: Selecting candidate split points

in top left figure. Centroids of two class are marked as two big circles as shown in top right part of the figure. Selected split points are marked as asterisk. The best splitting hyper plane is obtained by using the selected points as shown by straight line in the figure.

Generating splitting hyperplane

For each set of bins, corresponding best splitting plane is determined. The algorithm is presented in algorithm 3. Details of the algorithm can be found in(Gurung, Lindgren, and Boström 2015). If sp is the number of candidate split points and l is the number of bins used in a set, $\binom{sp}{l}$ split planes need to be evaluated (line 3). For each combination of l points, square matrix A is created (line 5). If inverse of A exists, it is multiplied by column vector B of length l that has value 1s (lines 7 and 8). Multiplication of inverse of matrix A and B gives the coefficients of the splitting plane $split\_plane\_coefs$(line 8). Information gain obtained by the split is obtained and the split plane is preserved if it is better than previous best split (lines 9 to 21).

Refining the best split plane

Once the best splitting hyperplane is obtained for one of the set of bins of a histogram variable, it can further be readjusted. The idea is to again find sp points that are closest to this splitting hyperplane and use those nearest sp points to find new splitting hyperplane. We select the new best splitting hyperplane if it is better than previous one. The algorithm for refining best split is presented in algorithm 4. sp nearest points to the current best split plane are selected (lines 2 to 7) which are then used to find new best split plane (line 12). The new best plane is preserved if it is better than the previous best plane (lines 14 to 18). Figure 2 shows the refining of the best split plane. The points closest to best split plane are marked by asterisks in the figure. The refined splitting hyperplane is shown as blue line in bottom right section of the figure.

Overall algorithm: putting the pieces together

The overall algorithm obtained by putting together all its subroutines, explained before, is presented in algorithm 5.

Algorithm 3 Generate a split plane for a given set of bins

```
Input: $h\_points$: observations in a node for selected bins of a histogram variable
    $hp\_class$: class label of each $h\_points$
    $split\_points$: points to use for generating split plane
Output: $best\_split\_plane$: coefficients of best split plane
 1: $best\_info\_gain \leftarrow 0$, initialize variable
 2: $l \leftarrow$ number of bins considered in $h\_points$
 3: $split\_p\_combns \leftarrow$ ways to choose $l$ points in $split\_points$
 4: for all $p\_combn$ in $split\_p\_combns$ do
 5:     $A \leftarrow l$ by $l$ matrix of points in $p\_combn$
 6:     if $A^{-1}$ exists then
 7:         $B \leftarrow$ column vector of $l$ ones.
 8:         $split\_plane\_coefs \leftarrow$ multiply $A^{-1}$ and $B$
 9:         for all $point$ in $h\_points$ do
10:             $value \leftarrow$ multiply $point$ and $split\_plane\_coefs$
11:             if $value < 1$ then
12:                 $l\_obs \leftarrow$ assign $point$ to left node
13:             else
14:                 $r\_obs \leftarrow$ assign $point$ to right node
15:             end if
16:         end for
17:         $info\_gain \leftarrow$ info gain($split\_plane\_coefs$)
18:         if $info\_gain > best\_info\_gain$ then
19:             $best\_info\_gain \leftarrow info\_gain$
20:             $best\_split\_plane \leftarrow split\_plane\_coefs$
21:         end if
22:     end if
23: end for
```

Algorithm 4 Refining the best split plane in a node

```
Input: $h\_points$: observations of selected set of bins
    $current\_best\_plane$: coefficients of best split plane
    $sp$: number of split points required
Output: $refined\_split\_plane$: coefficients of refined plane
 1: if $|h\_points| > sp$ then
 2:     for all $point$ in $h\_points$ do
 3:         $dist \leftarrow$ find distance of $point$ from $current\_best\_plane$
 4:         $point\_dist \leftarrow$ append $dist$ with $point$ and save
 5:     end for
 6:     $asc\_dist \leftarrow$ sort $point\_dist$ by ascending value of $dist$
 7:     $refine\_points \leftarrow$ fetch first $sp$ points from $asc\_dist$
 8: else
 9:     $refine\_points \leftarrow h\_points$
10: end if
11: $best\_info\_gain \leftarrow$ info gain($current\_best\_plane$)
12: $new\_best\_plane \leftarrow$ find best plane($refine\_points$)
13: $new\_best\_info\_gain \leftarrow$ info gain($new\_best\_plane$)
14: if $new\_best\_info\_gain > best\_info\_gain$ then
15:     $refined\_split\_plane \leftarrow new\_best\_plane$
16: else
17:     $refined\_split\_plane \leftarrow current\_best\_plane$
18: end if
```

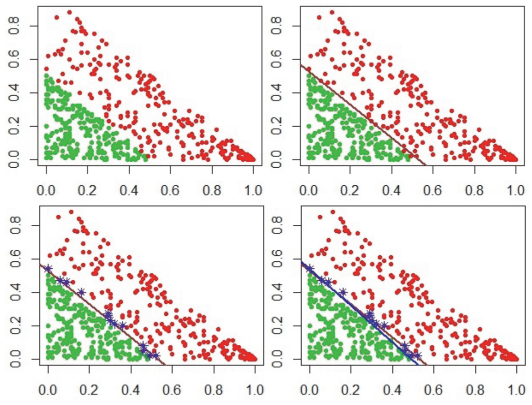

Figure 2: Refining best split plane

For each histogram variable, a list of the set of bins $list\_bin\_sets$ for window size from l_{min} to l_{max} is generated (line 8, algorithm 1). For each set of bins bin_set in $list\_bin\_sets$, split points are obtained (line 16, algorithm 2). The best splitting hyperplane is obtained by using $split\_points$ (line 17, algorithm 3). If the bin set has single bin, standard decision tree method to find best cutoff point is used (lines 12 to 15).

The best split planes of all the sets of bins for all histogram variables are sequentially compared which ultimately results in the best set (lines 19 to 25). Once the best set is selected, its best split plane is saved for further readjustment (line 29 to 31, algorithm 4). Readjustment / refinement is applied on the best split plane.

Experiments

The proposed approach has been implemented in R[1]. Experiments were performed on a synthetic and real-world datasets. The bin values in the synthetic dataset were obtained from 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 real-world dataset was provided by the truck manufacturing company Scania CV AB and consists of histogram variables that describe operational profiles of trucks. Operational profiles of trucks are used to classify trucks with broken NOx sensor from those that has fully functional NOx sensor. One publicly available dataset from the domain of image recognition was also used for the experiment. The predictive performance of both standard decision tree learning algorithm and the proposed approach are compared with respect to classification accuracy and area under ROC value as estimated using five fold cross validation. Results of previous histogram tree method are also presented where applicable. In addition to average accuracy and AUC, the tree size, i.e., the number of nodes, is also presented for each method. Average training time relative to average training time for standard decision tree is also reported. If the number of training examples in a node is equal or lower than 5 examples, no more splitting is done to that node. 20 percent of training data is set aside for post pruning. The number of split points to use, is set to 1, 5 and 7 for synthetic experiments and 1, 3 and 5 for all real world experiments. The window size range used is 1 to 4 in all the experiments. Brief descriptions of the datasets, the experimental settings and the results observed from the experiment are provided in respective sub-sections.

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

Algorithm 5 Overall algorithm

```
Input: $obs$: observations in a node
    $hist\_vars$: list of histogram variable names
    $sp$: Number of split points to use
    $l_{min}$: minimum size of sliding window
    $l_{max}$: maximum size of sliding window
Output: $best\_split\_plane$: coefficients of best split plane
 1: $best\_info\_gain \leftarrow 0$, initialize best information gain.
 2: $best\_var \leftarrow \emptyset$ stores best variable.
 3: $best\_split\_plane \leftarrow \emptyset$ initialize split plane.
 4: $best\_bin\_set \leftarrow \emptyset$ initialize best set of bins.
 5: for all $hist$ in $hist\_vars$ do
 6:     $m \leftarrow$ number of bins in $hist$
 7:     $b\_idx \leftarrow$ indices of bins in $hist$
 8:     $list\_bin\_sets \leftarrow find\_list\_bin\_sets(b\_idx, l_{min}, l_{max})$
 9:     for all $bin\_set$ in $list\_bin\_sets$ do
10:         $h\_points \leftarrow$ points in a node for $bin\_set$
11:         $hp\_class \leftarrow$ class label of each $h\_points$
12:         if $|bin\_set|$ is 1 then
13:             $all\_cutoffs \leftarrow$ candidate cutoff points for numeric variable
14:             $split\_plane \leftarrow$ find best cutoff point, standard tree method
15:         else
16:             $split\_points \leftarrow get\_split\_points(bin\_set, h\_points, hp\_class, sp)$
17:             $current\_split\_plane \leftarrow get\_best\_split(h\_points, hp\_class, split\_points)$
18:         end if
19:         $info\_gain \leftarrow get\_info\_gain(current\_split\_plane)$
20:         if $info\_gain > best\_info\_gain$ then
21:             $best\_info\_gain \leftarrow info\_gain$
22:             $best\_var \leftarrow hist$
23:             $best\_split\_plane \leftarrow current\_split\_plane$
24:             $best\_bin\_set \leftarrow bin\_set$
25:         end if
26:     end for
27: end for
28: $num\_bin\_bestvar \leftarrow |best\_var|$
29: if $num\_bin\_bestvar > 1$ then
30:     $best\_split\_plane \leftarrow refine\_split\_plane(h\_points, best\_split\_plane, sp)$
31: end if
```

Synthetic Data

Experiments on two synthetic datasets were conducted.

Table 1: Synthetic Data: Linear pattern

Split pts.	Win.	Nodes	Time	Acc.	AUC
Histogram Approach with sliding window					
1	1-4	56.2	1.24	92.05	0.953
5	1-4	41	1.48	92.67	0.970
7	1-4	42.2	2.20	93.14	0.976
Histogram Approach without sliding window					
1	—	57	1.98	90.69	0.928
5	—	32.2	1.41	92.57	0.952
7	—	35.4	1.28	93.41	0.963
Bins Treated Individually (Standard Tree Algorithm)					
—	—	77.8	1	90.01	0.947

Table 2: Synthetic Data: Nonlinear pattern

Split pts.	Win.	Nodes	Time	Acc.	AUC
Histogram Approach with sliding window					
1	1-4	94.2	0.97	85.25	0.935
5	1-4	77.8	1.05	85.61	0.939
7	1-4	75.8	1.24	86.24	0.938
Histogram Approach without sliding window					
1	—	83.4	2.19	84.83	0.925
5	—	63.4	1.23	85.40	0.923
7	—	57.8	0.95	84.62	0.927
Bins Treated Individually (Standard Tree Algorithm)					
—	—	110.6	1	84.83	0.934

Histogram with Linear Pattern This dataset has two histogram variables. One histogram variable has four bins and other has five bins. A linear pattern was injected among some of the bins of these histograms. The first injected pattern in the first histogram is $V1_1 + V1_2 < 0.8$. Similarly the second injected pattern in the second histogram is $V2_1 + V2_2 + V2_3 < 0.8$ where Vi_j represent bin j of histogram variable i. An indicator variable was assigned to each histogram that is set TRUE if condition as specified in the pattern is satisfied. Observations are assigned as positive class if indicator variables for both histograms are TRUE. In order to blur the boundary region, 25 percent of the points, $(V1_1, V1_2)$ in first variable and $(V2_1, V2_2, V2_3)$ in second variable, that are closest to their respective injected boundary patterns, were selected, and for 10 percent of them, their indicator variable was flipped. Again the class label for each observation is re-evaluated. This dataset has 1912 observations out of which 440 are positive examples and 1472 are negative examples after noise injection.

Histogram with Nonlinear Pattern The second synthetic dataset has one histogram variable with 4 bins. A non linear pattern was injected in this histogram as $(V_1 - 0.3)^2 + (V_2 - 0.3)^2 < 0.3^2$ which is a circle with radius 0.3 centered at (0.3, 0.3) in 2D space of first and second bin. Any points (V_1, V_2) inside the circle was assigned the positive class while others were assigned the negative class. The boundary region was blurred by using similar technique of noise injection. This dataset has 1912 observations out of which 624 are positive examples while 1288 are negative examples after noise injection.

The new improved histogram tree was trained on these two synthetic datasets. The results of the experiments are shown in Table 1 and Table 2. The columns of each table, respectively show the parameters: number of split points, range of window size, average number of nodes in tree model, average training time relative to standard approach, average accuracy and average value of Area Under ROC over five folds.

Real-world dataset

Two datasets from real world that have histogram as attributes were used for the experiment.

Table 3: Real Data: Operational Dataset

Split pts.	Win.	Nodes	Time	Acc.	AUC
Histogram Approach with sliding window					
1	1-4	43.4	1.58	94.81	0.830
3	1-4	49.4	2.43	94.61	0.828
5	1-4	51	5.44	94.86	0.840
Histogram Approach without sliding window					
1	—	36.6	3.82	94.56	0.721
Bins Treated Individually (Standard Tree Algorithm)					
—	—	57.8	1	94.80	0.837

Operational data of heavy trucks Each observation in this dataset is a snapshot of operational profile of a truck. Experiment was conducted to classify trucks with faulty NOx sensor from those with functional ones (healthy). Each snapshot consists of six histogram variables, four of them have 10 bins, one has 20 bins and the sixth one has 132 bins. This dataset has 5884 trucks out of which 272 have faulty NOx sensor while 5612 are healthy. The dataset has no missing values. The results of the experiment are presented in Table 3. Previous implementation of histogram tree was not able to train on this dataset when parameter *number of split points* was set to 3 and 5.

Corel Image Dataset This dataset consist of 1000 images of ten different categories such as human, buildings, bus, animals etc. Each category has 100 images[2]. Each picture is represented as two histogram variables each with 512 bins. The experiment is set up as binary classification to classify first category from all the remaining categories. So, 100 observations of first category are considered to be positive cases while all the remaining 900 observations are considered to be negative cases. The results of the experiments are reported in table 4. Earlier implementation of histogram tree could not be trained on this dataset, as it cannot cope with the size of the dataset / histogram.

Discussion

Results of experiments on synthetic data shows that treating bins as histogram was better when there was dependencies

[2]http://users.dsic.upv.es/~rparedes/english/research/rise/MiPRCV-WP6-RF/

Table 4: Real Data: Corel Image Dataset

Histogram Approach with sliding window

Split pts.	Win.	Nodes	Time	Acc.	AUC
1	1-4	9.4	3.08	94.0	0.885
3	1-4	9.8	8.14	94.5	0.888
5	1-4	9	20.04	94.3	0.888
Bins Treated Individually (Standard Tree Algorithm)					
—	—	10.6	1	93.9	0.889

among bins. In both synthetic experiments, histogram approach had better accuracy and AUC measures compared to standard decision tree approach and earlier implementation of histogram tree. However, growing the histogram trees were slightly slower, around 2.2 times slower than standard approach in worst case. Linear patterns were learned better compared to non linear patterns as shown in Table 1 and 2. The size of the trained tree for the histogram approaches were smaller in general compared to the standard approach. In general, the size of tree dropped as the number of split points increased whereas accuracy and AUC increased.

However, the results of experiments on real world data did not show a clear gain by using histogram approaches. In Table 4 for corel dataset, histogram approach was better than standard approach by narrow margin in terms of accuracy where as AUC was almost equivalent. In case of operational dataset as shown in Table 3, histogram approach was almost equivalent to standard approach.

In all the experiments, window size varied from 1 to 4. When window size was 1, each bin was individually evaluated for splitting the node just as in standard decision tree algorithm. This would ensure that performance of the proposed approach in general is at least as good as standard decision tree algorithm. Any gain obtained by using multiple bins simultaneously would then be additional information. One of the reasons why the results of experiments on real world data did not show any considerable gain could be because of algorithm's limitation to capture only linear pattern. This was hinted by comparing AUC values in the results of synthetic experiment for data set with linear and nonlinear pattern. So, in the future, focus could be on implementing methods to capture non linear patterns as well. Simpler splits in terms of number of bins, are always preferred in case of tie during the split. Ties, in cases of equal number of bins, are however, not addressed at the moment but is something that could be addressed in the future.

Concluding Remarks

The histogram tree classifier learns from histogram variables in addition to standard numeric and categorical variables. It exploits dependencies among bins of histogram by treating them together (simultaneously) during node splitting process. However, high computational complexity has been one major drawback of the method, specially when histogram variables have large number of bins. So, in this paper an approximation method was introduced such that only small chunk of bins are used simultaneously at a time during the node splitting phase. The size of the chunk can be varied as a parameter. Some of the real world datasets in the experiments conducted had histogram variables of length 132 and some even of length 512. It was practically impossible to the train the histogram tree on these big histograms with the earlier implementation. However, with the current implementation this is not a problem anymore. The results from both synthetic and real-world datasets suggest that gains in terms of predictive performance and AUC, and a reduction of the number of nodes might be obtained with slight increased learning time compared to using a standard decision tree.

In the future a comprehensive study of comparing the performance of the proposed method against existing multivariate split methods such as Linear Discriminant Trees (John 1995; Loh and Vanichsetakul 1988), and Perceptron Trees (Utgoff and Brodley 1990; Sethi and Yoo) are planned. Approaches for non-linear split conditions shall also be examined.

Acknowledgment

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

References

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

Gurung, R.; Lindgren, T.; and Boström, H. 2015. Learning decision trees from histogram data. In *In Proceedings of the 11th International Conference on Data Mining*, 139–145.

John, G. H. 1995. Robust linear discriminant trees. In *Fifth Intl Workshop on Artificial Intelligence and Statistics*. 285–291.

Loh, W.-Y., and Vanichsetakul, N. 1988. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association* 83:715–725.

Quinlan, J. R. 1986. Induction of decision trees. *MACH. LEARN* 1:81–106.

Sethi, I., and Yoo, J. Design of multicategory multifeature split decision trees using perceptron learning. In *Pattern Recognition*, volume 27. 939–947.

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