Learning Random Forest from Histogram Data Using Split Specific Axis Rotation

Ram B. Gurung, Tony Lindgren, and Henrik Boström

***Abstract*—Machine learning algorithms for data containing histogram variables have not been explored to any major extent. In this paper, an adapted version of the random forest algorithm is proposed to handle variables of this type, assuming identical structure of the histograms across observations, i.e., the histograms for a variable all use the same number and width of the bins. The standard approach of representing bins as separate variables, may lead to that the learning algorithm overlooks the underlying dependencies. In contrast, the proposed algorithm handles each histogram as a unit. When performing split evaluation of a histogram variable during tree growth, a sliding window of fixed size is employed by the proposed algorithm to constrain the sets of bins that are considered together. A small number of all possible set of bins are randomly selected and principal component analysis (PCA) is applied locally on all examples in a node. Split evaluation is then performed on each principal component. Results from applying the algorithm to both synthetic and real world data are presented, showing that the proposed algorithm outperforms the standard approach of using random forests together with bins represented as separate variables, with respect to both AUC and accuracy. In addition to introducing the new algorithm, we elaborate on how real world data for predicting NOx sensor failure in heavy duty trucks was prepared, demonstrating that predictive performance can be further improved by adding variables that represent changes of the histograms over time.**

***Index Terms*—Histogram random forest, histogram data, random forest PCA. histogram features.**

I. Introduction

Learning algorithms for data where the features are expressed as histograms have not been widely explored yet. Histogram data is frequently encountered in domains where multiple observations are aggregated over time. The type of histogram that we focus in this study assumes that each histogram of a variable has the same number of bins and bin size (width) across all the observations. The bins of a histogram may have dependencies that are easily overlooked if the bins are treated as separate numeric variables. So, exploiting such dependencies in histogram variables may have a potential positive effect on predictive performance.

The field of Symbolic Data Analysis (SDA) attempts to address issues of dealing with complex data structures [1], with histograms being one example of such complex data structure. There are few studies on applying PCA [2] and clustering [3] on histogram data. These studies consider a slightly more general scenario than what is considered in this study, by allowing varying bin structures for different observations. However, none of these studies consider learning predictive models from multiple histogram variables, possibly mixed with standard numerical and categorical variables, with the exception of our own work on decision tree learning from histogram data [4], [5]. In these studies, decision trees for histogram data were shown to give better results compared to their standard counterparts. Such histogram decision trees may seem to be employed quite straightforwardly in random forests, to further improve predictive performance (at the cost of interpretability). However, the employed search heuristic in the algorithm for generating histogram trees has some limitations, which makes it less suited for generating forests (these limitations will be discussed in detail in the following section). In this paper, an alternative search heuristic is instead proposed to overcome those limitations by using the well-studied principal component analysis (PCA) method [6].

In the next section, the limitations of the existing algorithm for learning decision trees from histogram data are discussed together with some improvements that are proposed and discussed. In Section III, data preparation, experimental setup, and results from both synthetic and real world data are presented and discussed. In Section IV, we briefly discuss related approaches. Finally, in Section V, the main conclusions are summarized and directions for future research are pointed out.

II. Method

The binary decision tree algorithm [7] repeatedly tries to partition the examples in a node based on a variable (only numerical or categorical) that separates the examples in the best possible way until some stopping criteria are met. It may be adapted to handle features containing histograms. Bins of a histogram might have dependencies that can be captured if they are considered simultaneously, while making a node splitting decision, as illustrated by the algorithm proposed in [5]. When a histogram variable is selected for split evaluation by this algorithm, sets of bins are considered simultaneously. The number of bins to be considered simultaneously is a parameter of the algorithm, which employs a sliding window of this size over the ordered bins as illustrated by Fig. 1.

Histogram trees can be used as base models in a random forest [8]. Employing randomization when generating multiple trees has been shown to result in smaller error rates

Manuscript received September 5, 2017; revised January 30, 2018. This work was supported by Scania CV AB and the Vinnova program for Strategic Vehicle Research and Innovation (FFI) Transport Efficiency.

Ram B. Gurung and Tony Lindgren are with Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden (e-mail: gurung@dsv.su.se, tony@dsv.su.se).

Henrik Boström was with Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden. He is now at KTH Royal Institute of Technology, School of Information and Communication Technology, Kista, Sweden (e-mail: bostromh@kth.se).

doi: 10.18178/ijmlc.2018.8.1.666

compared to using single decision trees. Randomization is incorporated in the random forest algorithm by growing several trees from randomly generated bootstrap samples and by randomly selecting a few candidate variables to be evaluated for each split. For a histogram tree in a random forest, when a histogram variable is selected for split evaluation, not all the bin sets (obtained by the sliding window method) are evaluated. Instead, using the standard heuristics, a subset of the bin sets are randomly selected and hence considered as candidates for partitioning the examples. This includes randomization at yet another level, possibly resulting in even more diverse trees. In the following section, the previous approach for splitting a node is described in more detail, followed by a description of the proposed novel approach.

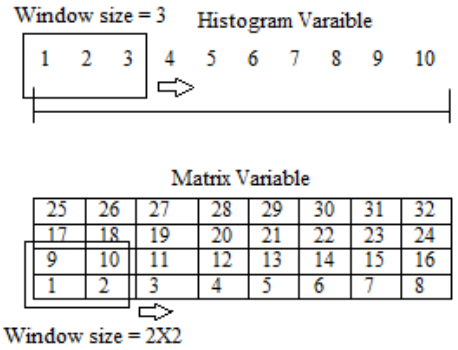

Fig. 1. Sliding window approach on histogram.

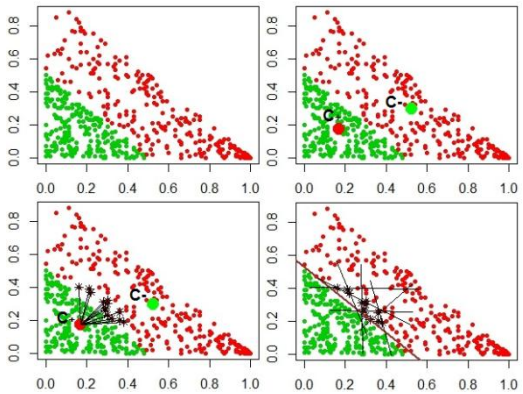

Fig. 2. Splitting hyperplanes

A. Current Node Splitting Method

In the algorithm proposed in [5], for a given histogram variable, each observation is considered to be a point in *d* dimensional space where the number of bins is equal to *d*. For example, consider a scenario when there are only two bins, as shown in the Fig. 2, where the bins are represented along *x* and *y* axes of each subplot. Assume there is a simple linear decision boundary between the points of each of the two classes. In order to find the best splitting hyperplane, the previous algorithm employed a crude approach where first a small set of special points (supposedly near the decision boundary) were selected to create possible splitting hyperplanes. A simple heuristic was used to select the special points, by choosing the centroids of each class and then calculating the distance of all points of one class against the centroid of the opposite class and vice versa, as illustrated by second and third subplots in Fig. 2. The number of such special points was determined by a parameter and the points with the shortest distances from opposite centroids were selected. If for example six such special points are to be chosen, two of them (determined by the number of bins in the set) are taken at a time to get a hyperplane that passes through them as

$$C_1.X + C_2.Y = 1 \quad (1)$$

If (x_1, y_1) and (x_2, y_2) are two randomly selected special points, the coefficients of a hyperplane through these two points can be calculated as

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} \times \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}^{-1} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (2)$$

Limitations: Apparently, the quality of the split depends on the selected special points, which in turn are dependent on the employed point selection heuristic. Moreover, the quality also depends on the number of such special points that are considered, as the number of possible hyperplanes is proportional to this number, and hence the chances of obtaining a good split increase with this number. However, the computational cost also increases rapidly with the number of selected special points. Moreover, the coefficients of the hyperplane can only be found when the inverse of the matrix created by using the selected points exists, otherwise the combination of selected points are simply discarded. Also, this heuristics works only for classification problems.

B. Proposed Node Splitting Method

The purpose of using multiple bins simultaneously for splitting is to make sure that the split utilizes relevant information from all the bins and adhere to possible dependencies among bins. However, searching for the splitting plane in d dimensions with a lot of freedom (any direction or inclination) results in a very large search space. Using heuristics to limit the search space has a possibility of missing good splits. Therefore, we propose to search for a split in a transformed space such that the new variable in the new space carries information from all the variables in original space. This new variable in the new space can be treated as a standard numerical variable and we can expect it to contain information about (linear) dependencies that are present in the original space. We use the Principal Component Analysis (PCA) method to transform the space such that each principal component is a variable in the new space. Principal components are linear combinations of the original variables and the new transformed space is obtained by rotating the original axis along the direction of maximum variance. We can expect to find the best split in the rotated space more easily by looking for splits orthogonal to the axis, as done in standard approach when handling numerical features. It should be noted that the rotation is performed only for evaluating splits of a histogram variable within a node, and the original values for the bins are kept intact (in the original space) in each resulting child node.

C. High Level Description of the Algorithm

1) Draw B bootstrap samples from the original data.

2) Grow a histogram tree from each bootstrap sample until the stopping criteria are met.
   - At each node of the histogram tree, randomly select a subset of p candidate variables.
   - For each numeric variable, find the most informative cutoff value.
   - For each histogram variable, randomly select $\sqrt{n}$ of the n possible sets of bins that can be generated (using a fixed window size for one-dimensional histograms and using a 2×2 window for two-dimensional histograms). Apply PCA and find the most informative principal component and cutoff value.
   - Partition according to the most informative split.
3) When a test example is to be classified, it is dropped down from the root node of all base trees. When a node with a histogram split variable is reached, the PCA rotation coefficient for the selected bin set is used to transform the bin set of a test case into the new space, using the saved cutoff value to determine which child node the example should be routed to.
4) The predictions of all base models are combined.

III. Empirical Evaluation

Our proposed random forest algorithm for handling histogram variables was implemented in *R* and evaluated on two synthetic datasets and a real world data set.

A. Synthetic Data

The first synthetic dataset has two histogram variables (with four and five bins respectively) where the decision boundary is defined by a linear pattern in each variable. The bins of each histogram sums to 1. The linear pattern for the first histogram variable is $h_1^1 + h_2^1 < 0.8$ while it is $h_1^2 + h_2^2 + h_3^2 < 0.8$ for the second histogram variable, where h_j^i represents j^{th} bin of i^{th} histogram. The observations are assigned the positive class if they fulfill the conditions of both histograms (with some noise injected around the boundary region by flipping the class labels randomly). The resulting dataset consists of 1912 observations of which 440 are positive.

The second synthetic dataset was generated using a nonlinear pattern for a single histogram variable with four bins. A nonlinear decision boundary is set as $(h_1^1 - 0.3)^2 + (h_2^1 - 0.3)^2 < 0.3^2$. Noise is injected along the boundary region using similar technique as in first data-set. The final data-set has 1912 examples out of which 624 are positive.

B. Real World Data: Heavy Duty Trucks

The objective here is to identify trucks with a faulty NOx sensor using information about the operation of individual trucks, which have been extracted during their workshop visits. Each extract is a snapshot covering various operational features, such as ambient temperature, vehicle speed, fuel temperature etc. stored as histogram variables. In order to limit our analysis to a homogeneous fleet, only trucks built for long haulage operation are considered. This data has been provided by Scania AB, a large truck manufacturing company in Sweden.

Typically, each truck has multiple snapshots extracted during each workshop visits and operational variables in these snapshots are cumulative in nature (value in histogram bins adds up) over corresponding snapshots. For positive cases, i.e., trucks with a faulty NOx sensor, the latest snapshot taken at least 7 days before the first time breakdown is considered, while for the negative cases, the last snapshot extracted at least 7 days before the final snapshot is selected, see Fig 3. Instead of discarding the remaining snapshots, they are used to track the change (increase) in value of bin over time (taking each snapshot as a time point). The rate of change of the bin value over time is then aggregated over number of snapshots. Two aggregation approaches are considered; simple average or weighted average. The use of weighted average can be justified by the assumption that recent activities (close to the breakdown point) are more relevant for the breakdown than the more distant ones, and hence the former should be given higher weight.

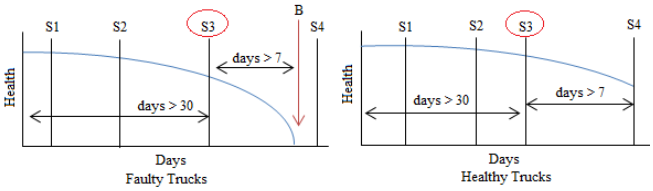

Fig. 3. Snapshot selection in a truck.

Let,

t_i = number of days since truck was delivered to when i^{th} snapshot was extracted such that $1 \le i \le n$ assuming that n^{th} snapshot is chosen for the truck.

b_{ij}^h = value in bin j of histogram h in i^{th} snapshot.

Rate of increase in bin value from $(i-1)^{th}$to i^{th}snapshot is calculated as:

$$r_{ij}^h = \frac{(b_{ij}^h - b_{(i-1)j}^h)}{(t_i - t_{i-1})}, \text{ where } b_{0j}^h = 0 \quad (3)$$

So, for bin b_j^h, Simple Average Rate (SAR) is

$$(SAR)_j^h = \frac{\sum_{i=1}^n r_{ij}^h}{n} \quad (4)$$

For weighted average, weight function is chosen to be a decaying function over time.

Let, $\gamma_i = \frac{t_i}{t_n}$, which ensures $0 \le \gamma_i \le 1$

The weight assigned to r_{ij}^h is set as

$$w_i = \gamma_i e^{(\gamma_i - 1)} \quad (5)$$

This function ensures that the weight *1* for the n^{th} (selected) snapshot and it gradually decreases as we go back in time.

Weighted Average Rate (WAR) for bin b_j^h is

$$(WAR)_j^h = \frac{\sum_{i=1}^n w_i r_{ij}^h}{n} \quad (6)$$

Finally, SAR and WAR for all bins of histogram variables are calculated for all trucks.

The original dataset has eight one-dimensional histogram variables, of which seven have 10 bins and one has 20 bins.

There is also one two-dimensional (matrix) histogram variable with 11× 12 (132) bins. So, in total there are 222 bins belonging to the original histogram variables. For each of these bins, the average rate of change is calculated. The resulting variables (here called shadow variables) are also treated as histograms. So, in total there are 444 variables associated to histogram variables. The considered dataset has around 12,500 trucks (of which only around 900 have a faulty NOx sensor) with overall 450 variables including technical specifications of trucks such as power, engine stroke volume, age, technical weight etc.

C. Experiment and Results

For the synthetic data sets, five-fold cross validation was performed. Five-fold cross validation was chosen instead of more common ten-fold to counter the computation time it incurs. For the random forests with histogram trees generated using the previous approach, the number of special points for generating a splitting hyperplane was varied between *x = 1, 2* and *3* plus the number of bins in the histogram (*number of special points = x + number of bins*). The number of trees in the random forest was set to *300*. Each tree was grown until each node was either pure or had only five observations. The results on the two synthetic datasets are shown in Table I and Table II.

TABLE I: SYNTHETIC DATA WITH LINEAR PATTERN

Random Forest Models	AUC	Accuracy	Leaf Nodes
Standard RF	0.9633	93.25	83.4
Histogram RF (*x*=1)	0.9852	94.19	62
Histogram RF (*x*=2)	0.9862	94.19	53
Histogram RF (*x*=3)	0.9868	94.24	47.8
Histogram RF (PCA)	0.9839	93.87	47.2

TABLE II: SYNTHETIC DATA WITH NON-LINEAR PATTERN

Random Forest Models	AUC	Accuracy	Leaf Nodes
Standard RF	0.9552	86.55	110.2
Histogram RF (*x*=1)	0.9580	87.02	115.2
Histogram RF (*x*=2)	0.9590	86.92	100.4
Histogram RF (*x*=3)	0.9597	87.12	92
Histogram RF (PCA)	0.9594	87.70	82.2

The results on the synthetic data shows that the new approach of using PCA in node splits outperforms the standard approach (Standard RF row in the table) but also performs at least as good as using the previous histogram approach. It can also be seen that when using PCA, the trees become less bushy. Improvements over the standard approach are more significant when the decision boundary is linear while still comparable to our earlier approach in terms of AUC and accuracy. However, computation time for PCA approach was not found to be any better than our previous approach.

For the NOx sensor data set, the experimental setup was identical to the one used for the synthetic datasets, with the exception that the number of split points parameter *x* was set to 1 (*number of split points equals x plus number of bins*). Setting the value for *x* larger than *1* was computationally very expensive and was therefore refrained. Here we also used the new shadow variables as we mentioned earlier, resulting in five different models and their results shown in Table III.

1) Standard RF: using original variables and weighted average rates.
2) Histogram RF (*x = 1*): using original variables and weighted average rates.
3) Histogram RF (PCA): using original variables and weighted average rates.
4) Histogram RF (PCA): using only original variables.
5) Histogram RF (PCA): using original variables and simple average rates.

TABLE III: NOX SENSOR FAILURE PREDICTION

RF Models	Description	AUC	Leaf Nodes
Standard RF	With weighted avg. rate	0.8519	397.8
Hist. RF (*x*=1)	With weighted avg. rate	0.8735	426.8
Hist. RF (PCA)	With weighted avg. rate	0.8809	312.8
Hist. RF (PCA)	With simple avg. rate	0.8622	315.8
Hist. RF (PCA)	Only original variables	0.8494	356.4

The results clearly show that the proposed approach outperforms the other in terms of AUC in the context when weighted average rate were used (comparing the methods with respect to accuracy is not very informative in this case due to the dataset being heavily imbalanced). In addition, the low average number of leaf nodes resulting from the PCA approach suggests that it was good at discovering informative node splits. The PCA approach was further compared with cases firstly when new variables were simple average rates of bins and secondly when the new variables were not considered at all. The comparison shows that using weighted average outperforms both cases. The AUC plot against the number of trees in Fig. 4 shows that the AUC gain of the model almost stalls after 300 trees.

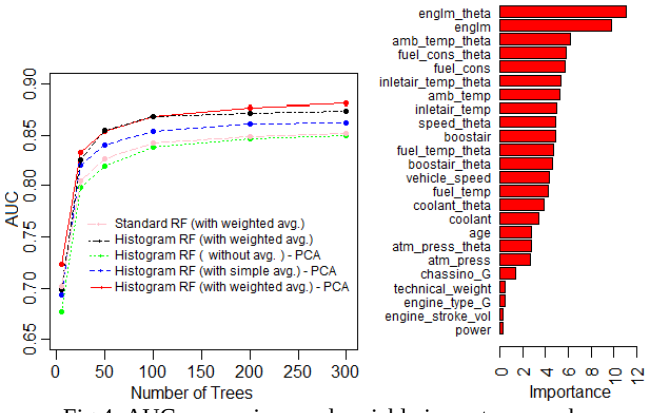

Fig 4. AUC comparison and variable importance rank.

Variables considered important are ranked as shown in Fig. 4 for the random forest model with PCA (with weighted average variables). Importance score was calculated based on their contribution on information gain while splitting the node. The names of the variable followed by *theta* suffix are the new shadow variables. It is interesting to see that in most of the cases, shadow variables are ranked higher than the corresponding original variables. Besides ranking based on information gain, the level or depth of the node where variable was used by the model to make the node split can also give additional information about how important the variable was. If the variable is used early in the tree, it can be considered important in general. Based on this assumption, minimum depth for each variable was

calculated and averaged over all trees. The minimal depth rank is compared with variable importance rank as shown in Fig. 5 (left subplot) and both approaches seem to agree in general.

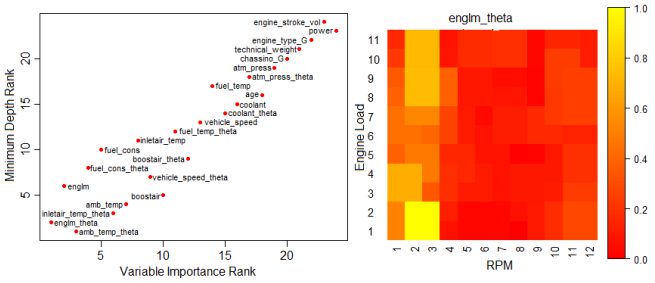

Fig 5. Important variable.

The shadow variable of Engine Loadmatrix *englm_theta* seems to be most significant one. This is a matrix of size 11×12, containing in total 132 bins. A square matrix of size 2×2 was used as sliding window to get 110 possible bin sets. 11 sets were randomly selected for node split evaluation, so each bin set gets a significance score according to information gain. Each cell in the selected bin set gets the same score. Based on this score, a heat map was plotted for each cell as shown in Fig. 5 (right). Low RPM regions seem to be more significant for classification as depicted by the yellow regions.

IV. Related Work

The idea of using PCA together with random forests is not entirely novel, and it has been considered in so-called rotation forests [9]. However, in that algorithm, the considered variables are not histograms and all the (numeric) variables are divided into a number of disjoint subsets and PCA is applied in turn to each subset to obtain the rotation matrix. The original dataset is multiplied by the rotation matrix which gives a new rotated dataset. This rotated dataset is then used to train base tree learner. So, in rotation forests, PCA is applied on (a bootstrap sample of) the whole training set before growing a tree, which is different from the approach proposed here, which uses PCA on a specific bin set, for split evaluation locally at each node. Since the transformation is applied only for split evaluation, it allows the random forest algorithm to estimate variable importance in the original space.

The use of multiple variables when evaluating node splits during decision tree construction has also been considered within multivariate decision trees [10], using methods such as LDA (Linear discriminant analysis) [11]. Clearly, these approaches can in principle be applied to include also the bins of histogram variables, hence potentially exploiting dependencies among the bins. However, it has not been investigated how well such trees would perform within forests, e.g., whether or not they are too stable to achieve optimal performance of the forest and how well they perform on data with histograms.

V. Conclusion

In this work, the random forest algorithm has been adapted to learn from histogram data by evaluating multiple bins together when splitting a node during tree growth, which allows for exploiting dependencies among the bins. However, searching for splitting hyperplanes in the same number of dimensions as there are bins, as done earlier in a previous approach, is computationally very costly.

In this work, a much more efficient approach is considered, which employs a sliding window to form groups of bins and then use PCA to generate principal components of the selected bins, for which the most informative cutoff point is selected.

Experimental evaluation based on synthetic data and real world data shows that this method seems to outperform the standard random forest approach and performs on par with the computationally costly approach. The high AUC measures and small tree sizes suggest that the proposed approach is highly competitive in finding informative splits. The use of shadow variables in the considered real-world dataset demonstrated that performance can be further significantly improved.

One direction for future work is to compare the proposed approach to other approaches for combining multiple features in decision trees and forests. Another direction for future research is to consider random survival forests [12], which can be used to predict the survival pattern of trucks. Rather than being able to say whether the truck is faulty or healthy, it could be more useful to be able to predict when a breakdown might happen.

Acknowledgment

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 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 Studies in Classification, Data Analysis, and Knowledge Organization*, B. Fichet, D. Piccolo, R. Verde, and V. Med, eds., Springer Berlin Heidelberg, 2011, pp. 3-15.

[3] A. Irpino and R. Verde, "A new wasserstein based distance for the hierarchical clustering of histogram symbolic data," in *Data Science and Classification 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.

[4] R. Gurung, T. Lindgren, and H. Bostrom, "Learning decision trees from histogram data," in *Proc. the 11th International Conference on Data Mining*, 2015, pp. 139-145.

[5] R. Gurung, T. Lindgren, and H. Bostrom, "Learning decision trees from histogram data using multiple subsets of bins," in *Proc. the 29th International Florida Arti_cial Intelligence Research Society Conference (FLAIRS)*, pp. 430-435, 2016.

[6] I. T. Jolliffe, *Principal Component Analysis*, Berlin, New York: Springer-Verlag Inc, 1986.

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

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5-32, 2001.

[9] J. J. Rodrguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1619-1630, 2006.

[10] C. Brodley and P. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, pp. 45-77, 1995.

[11] R. Todeschini, "Linear discriminant classification tree: A user-driven multicriteria classification method," *Chemometrics and Intelligent Laboratory Systems*, vol. 16, pp. 25-35, 1992.

[12] H. Ishwaran, U. Kogalur, E. Blackstone, and M. Lauer, "Random survival forests," *Ann. Appl. Statist.*, vol. 2, pp. 841-860, 2008.

Ram Bahadur Gurung is a Ph.D student in the Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden since 2013. He received his B. Eng. degree in computer engineering from Tribhuvan University, Nepal in 2006 and received his master's degree from Stockholm University, Sweden in 2013. His current research areas are data mining and machine learning.

Tony Lindgren Tony Lindgren was born in Hägersten, Stockholm in 1974. He received his master degree in computer and system sciences in 1999. In 2006 he received his Ph.D. degree in computer and system sciences. He has worked both in academia and industry since 2008, he is the inventor of numerous patents and has a permanent position as lecturer at the Department of Computer and System Sciences at Stockholm University since 2012. His main interest is in the field of machine learning, artificial intelligence and constraint programming.

Henrik Boström is a professor in computer science–data science systems at KTH Royal Institute of Technology. His expertise is within machine learning, primarily ensemble learning, rule and decision tree learning and conformal prediction. He is an action editor of the machine learning journal and on the editorial boards of Journal of Machine Learning Research, Knowledge Discovery and Data Mining, and Intelligent Data Analysis.