Int. J. Appl. Math. Comput. Sci., 2017, Vol. 27, No. 4, 737–748
DOI: 10.1515/amcs-2017-0051

amcs

INTERPRETABLE DECISION–TREE INDUCTION IN A BIG DATA PARALLEL FRAMEWORK

ABRAHAM ITZHAK WEINBERG [a], MARK LAST [a,*]

[a] Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev, P.O.B. 653, Beer-Sheva 8410501, Israel
e-mail: weinberA@post.bgu.ac.il, mlast@bgu.ac.il

When running data-mining algorithms on big data platforms, a parallel, distributed framework, such as MAPREDUCE, may be used. However, in a parallel framework, each individual model fits the data allocated to its own computing node without necessarily fitting the entire dataset. In order to induce a single consistent model, ensemble algorithms such as majority voting, aggregate the local models, rather than analyzing the entire dataset directly. Our goal is to develop an efficient algorithm for choosing one representative model from multiple, locally induced decision-tree models. The proposed SySM (syntactic similarity method) algorithm computes the similarity between the models produced by parallel nodes and chooses the model which is most similar to others as the best representative of the entire dataset. In 18.75% of 48 experiments on four big datasets, SySM accuracy is significantly higher than that of the ensemble; in about 43.75% of the experiments, SySM accuracy is significantly lower; in one case, the results are identical; and in the remaining 35.41% of cases the difference is not statistically significant. Compared with ensemble methods, the representative tree models selected by the proposed methodology are more compact and interpretable, their induction consumes less memory, and, as confirmed by the empirical results, they allow faster classification of new records.

Keywords: big data, parallel computing, MAPREDUCE, decision trees, editing distance, tree similarity.

1. Introduction

With the vast growth of the information volume and variety in the recent years, many organizations focus on big data platforms and technologies (Bekkerman *et al.*, 2011). In order to perform big data analysis with data mining algorithms, there is a need for a framework such as MAPREDUCE, which allows parallel execution over distributed computing resources. This includes architectures that are designed for real-time or near real-time analytics (Fan and Bifet, 2013). Here, we limit our discussion to architectures that need the combining phase, known as REDUCE. More specifically, we focus on decision tree algorithms. Since decision tree models are simple and effective in classification (Ben-Haim and Tom-Tov, 2010), they are among the most popular methods for data mining analysts and users. In addition, they provide human-readable classification rules, which is considered important in data analysis.

In order to induce one representative decision tree in the big data distributed framework, we suggest a novel approach called—the syntactic similarity method SySM. The algorithm runs as part of the combining (REDUCE) phase and chooses one representative model from multiple trees. This is a different approach from ensemble methods that combine predictions of multiple decision trees, which serve as base estimators. SySM is also more computationally efficient than algorithms like those of Mehta *et al.* (1996) or Shafer *et al.* (1996), which use time consuming activities in the REDUCE phase such as sorting attribute values. In addition, SySM classification accuracy is comparable to ensemble majority voting.

One of the advantages of SySM is fast classification of new records, which is an important issue for systems dealing with high-speed data streams. Using SySM, the new records traverse one representative decision tree. In this way, we save the computational effort needed for traversing multiple trees in the ensemble approach or going back to the raw training data for additional computation. Though ensemble methods may yield higher accuracy, their model is less interpretable than a single

*Corresponding author

decision tree selected by SySM.

To deal with very large datasets, we use a distributed framework (Zhang *et al.*, 2012), where we divide the data into slices (subsets) by using randomization, and process each data slice independently. Our assumption is that in a dataset big enough, any slice of data has a reasonable probability of representing the entire dataset. The question is which model will be best to represent the entire large dataset in an interpretable way. The most important issues in this case are to save the model combining time and to keep the representative model as compact as possible. These issues guide the SySM approach.

The proposed SySM technique is based on computing syntactic similarity between locally induced decision trees. We assume that the main reason for two decision trees to produce the same outcome is their inner structure: the nodes and their order in the tree. This implies that for the same dataset, decision trees that are similar to each other should yield similar classification decisions and predictions.

According to Andrzejak *et al.* (2013), distributed learning approaches should also consider the practical limitations imposed by the computing environment, including constraints on the memory of individual nodes, the size of exchanged messages, and the communication patterns influencing scalability. This can be done by reducing memory and data transfer requirements at the cost of accuracy. The SySM approach adds only minor computational effort to the parallel induction phase, which is transformation of the trees into the tree bracket format and comparing them which each other. As a result, the SySM combining phase (REDUCE) takes less time than the other merging methods such as that of Andrzejak *et al.* (2013).

Summarizing, the original contributions of our approach are the following:

- We present a computationally efficient solution for the challenge of constructing interpretable and compact decision tree models in the big data parallel framework.
- The combined tree is chosen out of a set of locally induced trees rather than built from the global dataset.
- The representative tree is found by computing pairwise syntactic similarity of locally induced trees represented in the tree bracket format.
- Our experimental results show that the proposed method can find a representative model that has classification accuracy comparable to that of ensemble majority voting.

The rest of this paper consists of four sections. Section 2 discusses possible ways to compare decision trees and choose one representative tree out of many. Section 3 presents the SySM algorithm methodology. Section 4 discusses experimental results on four benchmark datasets. Finally, Section 5 summarizes the current work and future research directions.

2. Background and related work

We can categorize big data approaches to decision tree induction as follows: building one big tree (Andrzejak *et al.*, 2013; Panda *et al.*, 2009; Ntoutsi *et al.*, 2008; Zhang and Jiang, 2012; Pawlik and Augsten, 2011; Narlikar, 1998; Sreenivas *et al.*, 2000; Goil and Choudhary, 2001; Amado *et al.*, 2001; Domingos and Hulten, 2000; Dai and Ji, 2014), transferring all decision trees into one rule base and back into a decision tree, ensemble approaches (Louppe and Geurts, 2012; Hansen and Salamon, 1990; Sollich and Krogh, 1996; Breiman, 1999), and others (e.g., Kargupta and Park, 2004) that do not build a new tree and use a combination of tree results. According to Ben-Haim and Tom-Tov (2010), another way to categorize the different types of algorithms for handling large datasets is to divide them into the following two groups: pre-sorting of data and using approximate representations of data. Under the first category we can mention SLIQ (Mehta *et al.*, 1996), its newer version SPRINT (Shafer *et al.*, 1996) and ScalParC (Joshi *et al.*, 1998). The second group includes algorithms that approximate representations of data by sampling and histograms construction. This group includes the following algorithms: BOAT (Gehrke *et al.*, 1999), CLOUDS (AlSabti *et al.*, 1998), and SPIES (Jin and Agrawal, 2003). Usually, pre-sorting techniques are more accurate but computationally intensive when running on big data sets.

As shown in Fig. 1, parallel decision tree construction algorithms can be grouped by different approaches to parallelism: task parallelism, data parallelism and hybrid parallelism (Srivastava *et al.*, 1995; Amado *et al.*, 2003). Task parallelism distributes the decision tree nodes among the processors in a dynamic way. Data parallelism distributes the training set among the processors so that each processor is responsible for a distinct part of the data. This category may be divided into two sub categories: horizontal partitioning vertical partitioning. The parallel strategy based on vertical data distribution (DeWitt *et al.*, 1991; Amado *et al.*, 2003; Kourtellis *et al.*, 2016) splits the data by letting each processor test different attributes, whereas horizontal parallelism partitions the data so that different processors see different records. Hybrid parallelism uses data parallelism as a combination between horizontal and vertical approaches. Its decision whether to use

horizontal or vertical parallelism is a function of the processing capability of each computing node and the constraints of the communication volume between them. Our method deals with decision trees constructed by the horizontal parallelism approach. We use the horizontal approach since our methodology is based on measuring the similarity of tree attributes.

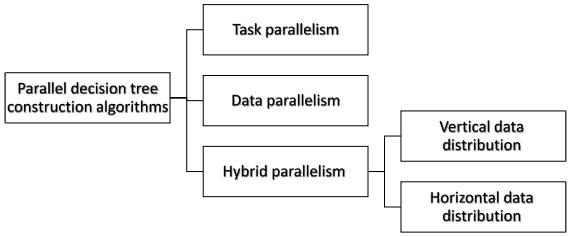

Fig. 1. Taxonomy of parallel decision tree construction algorithms.

Building a new decision tree from several induced decision trees is a well-known approach in big data. This approach usually excels in accuracy but needs significant computing resources (Ben-Haim and Tom-Tov, 2010). The computing resources are needed for controlling the parallel stage and for dividing the database in a specific way (Panda *et al.*, 2009) as well as for merging parts of trees in the post processing phase (Andrzejak *et al.*, 2013; Panda *et al.*, 2009; Ntoutsi *et al.*, 2008; Zhang and Jiang, 2012). The need for extensive computational resources and the long processing time are considered major disadvantages in cases where fast results are needed for decision making.

A framework for comparing decision trees in their induction stage is proposed by Pawlik and Augsten (2011). They use two types of similarity: semantic similarity and dataset similarity. The former is computed in terms of the agreement of the class predictions the decision trees return over the attribute space. The latter is based on the decision tree attribute space probability distribution, the attribute-class joint probability distribution and the attribute conditional class probability distribution. The presented framework can be used to decide when to update the global decision tree nodes and values as a result of a change in data, which is most relevant for the parallel stage. In addition, there is a need to go back to the raw data in order to compute probabilities.

Most of the methods mentioned above do not treat locally induced decision trees as final entities. They use induced decision trees as a basis for additional computations that go back and forth to the original dataset. From a computational point of view, this approach produces redundant work since it requires going back to the decision tree induction phase that was already completed and that is supposed to represent the original data.

Miglio and Soffritti (2004) present a similarity algorithm that combines the tree structure (syntactic) information with the agreement percentage (semantic similarity) on the testing set. The semantic similarity approach is based on the work of Shannon and Banks (1999). It is not suitable for big data environments where the running time is a critical constraint.

3. Methodology

SySM is aimed at selecting the best tree that can represent the entire dataset. We assume that the choice of the most representative tree should be based on its syntactic similarity to other locally induced decision trees. The syntactic similarity of two trees is calculated by a simple and fast editing distance algorithm called the RTED (robust tree edit distance) (Pawlik and Augsten, 2011), which counts the number of matching nodes having different labels. The RTED has lower computational complexity than the tree distance algorithm by Zhang and Shasha (1989) and it should be more robust than the Shannon and Banks (1999) approach as it is independent of the tree structure. The core idea in the RTED is dynamic decomposition. The RTED strategy recursively decomposes input trees into subforests by removing nodes from either the leftmost or the rightmost subtrees, whereas Zhang and Shasha's algorithm always removes subtrees from the right.

We assume that two decision trees induced from the same dataset using the same splitting metric will yield the same predictions if they both have the same internal nodes corresponding to the same attributes. We measure the similarity of internal nodes in the two compared decision trees using the following two parameters: the tested attribute name and the node position. These two parameters practically determine the tree structure. The node's position determines its relation to other nodes in the tree and hence its influence on the model outcome with no need to consider the derived split value. The RTED algorithm (Pawlik and Augsten, 2011) takes these two parameters into account.

When dealing with the original task of choosing one representative model out of many, our motivation is to minimize the need going back from one phase to its predecessor. For example, we are not interested in going back to the original training data or modifying induced decision trees. An additional assumption is that each of the decision trees produced in the parallel phase represents the corresponding dataset in its structure. When dealing with small datasets, each decision tree represents a local

Algorithm 1. Decision tree transformation into the tree bracket format.

INPUT: x—decision tree
OUTPUT: TBF is the tree x in the tree bracket format

Function $UpdateNodes(x)$:
if $NumberOfChildrenOfNM$ (x) is not null **then**
$NM(x)$ is the list of node attribute names of x
Set TBF = TBF + '{ '+ $NM(x)$
for $j = 1$ to $NumberOfChildrenOfNM(x)$ **do**
Call $UpdateNodes(x)$
Set TBF = TBF+ '}'
end for
end if

subset of the training data. When dealing with big datasets, each data subset should be large enough to represent the entire dataset.

The SySM algorithm transforms each of the induced decision trees into the bracket tree format (BTF) as shown in Fig. 4. This is done by scanning the tree nodes from the top down. The BTF is a way of representing the tree as one long sequence of node labels (testing attributes) where each node is separated from another using the opening bracket symbol. Different levels of the nodes can be distinguished by the bracket symbol as well. Calculating the difference between the number of opened brackets to the closed ones determines the level of the specific node in the tree. For example, for the following BTF: $\{a\{b\}\{c\{d\}\{e\}\}\}$, we can deduce that b and c are at the same level since the difference between open brackets to closed brackets before b is 2 and before c is 2 as well. We can also infer from the bracket locations of the BTF that both d and e are siblings and children of c. As can be seen in Figs. 2 and 3, the BTF representation of a tree shown in Fig. 4 is more compact than other decision tree representations.

Using the SySM algorithm, we refer only to the structure of induced trees and hence there is no need to make any change to the tree structure or to perform additional computations on the raw data. The RTED algorithm (Pawlik and Augsten, 2011), which is the core of our similarity calculation procedure, finds the distance between each pair of induced trees by finding the mismatches of each node label at the equivalent positions of the compared trees. In the case of such a mismatch, there are three ways (editing operations) to 'fix' it: to delete a node, to insert it or to change it. The relevant changes are accumulated. Practically, one induced tree is transformed into another. The RTED (Pawlik and Augsten, 2011) is symmetric, so regardless of whether the first induced tree is compared with the second one or vice versa, the edit distance is the same. For multiple comparisons between one decision tree with the rest of the

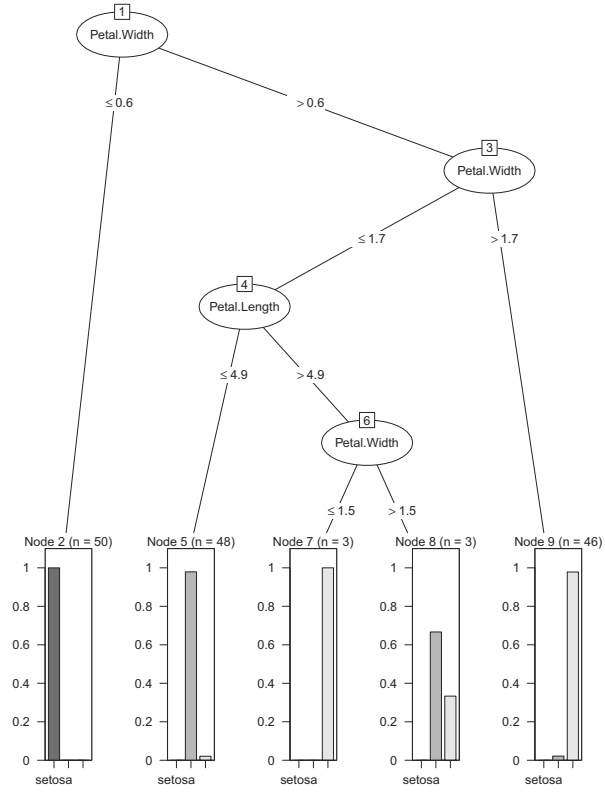

Fig. 2. J48 Visual representation of the IRIS dataset.

```
Model formula:
Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width

Fitted party:
[1] root
|   [2] Petal.Width <= 0.6: setosa (n = 50, err = 0.0%)
|   [3] Petal.Width > 0.6
|   |   [4] Petal.Width <= 1.7
|   |   |   [5] Petal.Length <= 4.9: versicolor (n = 48, err = 2.1%)
|   |   |   [6] Petal.Length > 4.9
|   |   |   |   [7] Petal.Width <= 1.5: virginica (n = 3, err = 0.0%)
|   |   |   |   [8] Petal.Width > 1.5: versicolor (n = 3, err = 33.3%)
|   |   [9] Petal.Width > 1.7: virginica (n = 46, err = 2.2%)

Number of inner nodes:    4
Number of terminal nodes: 5
```

Fig. 3. J48 Textual representation of the IRIS dataset.

```
{Petal.Width{Petal.Width{Petal.Length{Petal.Width}}}}
```

Fig. 4. J48 Bracket format representation (BTF) of the IRIS dataset.

Algorithm 2. SySM algorithm.

INPUT: Set $T = t_i, \ldots, t_n$ of induced decision trees, where $n =$ number of dataset slices
OUTPUT: Selected decision tree t_{RM}

if T is null **then**
  return *failure*
end if
for $i = 1$ to n **do**
  Set NM = list of node attribute names of $\{t_i\}$ (the nodes are in DFS order)
  Set $TBF = \{\}$, TBF is the tree bracket format
  Call UpdateNodes($NM(1)$), $NM(1)$ is the root node of the tree
  Set $TBF_i = TBF$
end for
Set $DM = 1$
for $i = 1$ to $(n-1)$ **do**
  for $j = (i+1)$ to n **do**
    Set DM_{ij} = $RTED$ $(t_i,\ t_j)$ {// DM_{ij} is the distance matrix, $RTED$ is the robust tree edit distance}
  end for
end for
Set $DM = DM + DM^{\top}$
for $i = 1$ to n **do**
  Set AED_i = $\sum_{j=1}^{n} DM_{ij}/n$ { // Compute the average edit distance per matrix row $tree$ }
end for
Set RM = $\arg\min_i$ $\{AED_i\}$ { // Choose the representative decision tree model (RM) by the minimal average edit distance tree}
Return t_{RM}

induced decision trees we build a distance matrix where each cell represents the mutual edit distance between the trees represented by the row number to each of the trees represented by the columns. For each row, the average of the edit distances represents the similarity of the tree to the rest of the decision trees. The representative model chosen by our algorithm is the model with the minimal average edit distance to other models.

We assume that the model chosen by the SySM algorithm is the best candidate for representing the entire dataset because of its highest syntactic similarity to other models. From the computational complexity perspective, the REDUCE phase of the SySM algorithm has a training complexity of $\mathcal{O}(n^2)$, where n is the number of induced trees/computation nodes, since the algorithm calculates the average edit distance of each induced tree to the rest of the trees. The testing (classification) complexity of our algorithm is $\mathcal{O}(m)$, where m is the number of new instances. In contrast, to it the testing complexity of ensemble methods is $\mathcal{O}(nm)$, since they require traversing n trees for each m new instances as opposed to SySM, which uses only one representative tree for classifying a new instance.

In the next section, we discuss the experimental results of our method.

Table 1. Datasets used for empirical evaluation.

ID	UCI Dataset name	Samples (number)	Attributes (number)	Classes (number)
DS1	Poker Hand	1025010	11	9
DS2	SUSY	5000000	18	2
DS3	Record Linkage Comparison Patterns	5749132	9	2
DS4	KDD Cup 1999	4898431	42	23

4. Results

In this section, we perform experiments to evaluate the performance of the SySM algorithm.

4.1. Design of experiments. We run the evaluated algorithms over four big multivariate datasets from the UCI repository, which are used in other papers on big data (e.g., Triguero *et al.*, 2015) and are shown in Table 1.

We applied the SySM algorithm to each dataset in six variations: 32 folds, 64 folds, 128 folds, 256 folds, 512 folds and 1024 folds. The folds are equal-size subsets (slices) of the training dataset assigned to different computation nodes. The training data are 90% of the randomly chosen tuples of the original datasets. J48 and CART decision trees were induced for each fold of every dataset. J48 trees were induced using the RWEKA package in R. J48 is induced from the POKER dataset by setting the M parameter (the minimum number of instances per leaf) to 30. For the SUSY dataset as well as KDDCUP, we used $M = 40$. For the RLCP dataset, J48 is induced using the default value of $M = 2$. The best value of M for each dataset was chosen by analyzing two random slices of 32 folds. For each value of M, we found model accuracy over training and testing datasets, the number of leaves and the size of the tree. We selected the value of M that leads to the highest testing accuracy of J48 under the memory constraints. The same method was used for choosing the best value of the CART *minbucket* parameter, which is the minimum number of observations in any terminal (leaf) node. In addition, *cp* (complexity parameter) was set to 0. For the POKER dataset, the best value of *minbucket* is chosen to be 250, for SUSY—500, for RLCP—500 and for KDDCUP—7 (the default value). As mentioned above, the values are the result of a search procedure aimed at maximizing classification accuracy with decision trees that fit in computer memory.

We use all computer cores for running the algorithms using R parallel packages. The computer has the following characteristics:

- processors: i7-4710 MQ,
- cores: 8 per processor (16 threads),
- clock speed: 2.50 GHz,
- cache: 256 MB,
- hard drive: 238.47 GB,
- RAM: 64 GB.

For each dataset, SySM obtains induced trees as input. The output of the SySM algorithm is one tree chosen out of all induced trees that is supposed to represent the entire training dataset. We compared the accuracy of the chosen tree over the testing data to the accuracy of ensemble majority voting and the majority rule that is induced from the entire training dataset.

Table 2. Dataset parameters for empirical evaluation.

ID	UCI Dataset name	J48 M	CART minbucket
DS1	Poker Hand	30	250
DS2	SUSY	40	500
DS3	Record Linkage Comparison Patterns	2	500
DS4	KDD Cup 1999	40	7

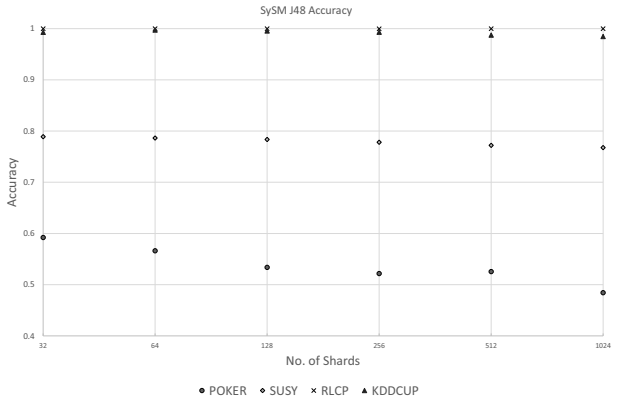

Fig. 5. SySM J48 accuracy.

In practice, the choice of the number of slices depends on the required processing speed and the available computational resources (number of CPUs). With an increase in the number of slices, the overall processing time will decrease. On the other hand, this will increase the number of CPUs used and their associated cost. As can be seen in Fig. 5, in smaller datasets (like Poker Hand), an increase in the number of slices may negatively affect the local models accuracy due to insufficient amount of training instances per each slice.

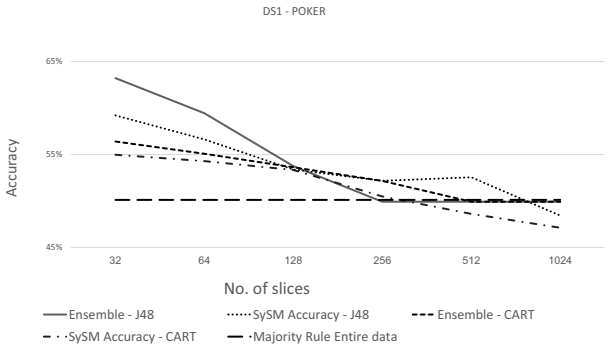

Fig. 6. DS1: Poker Hand algorithm accuracy per dataset slice.

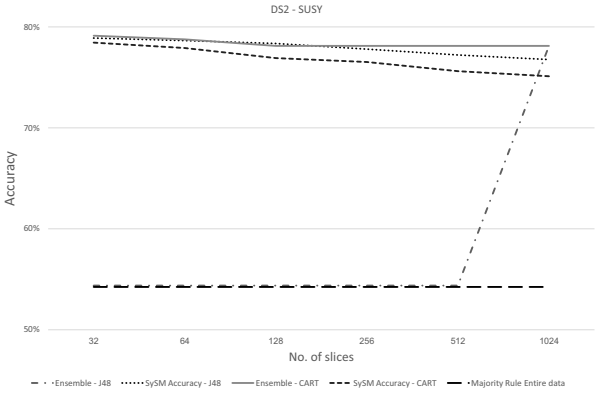

Fig. 7. DS2: SUSY algorithm accuracy per datasets slice.

4.2. Analysis of results. As we can be seen in Figs. 11 and 12, the more we partition the dataset, the lower the average tree distances are, which means that the induced trees become more similar to each other. When referring to Figs. 13 and 14, we can also see that the trees induced from smaller dataset slices have a smaller size (number of nodes).

The same decreasing trend applies to the models' accuracy, as can be seen in Figs. 6–9. The decrease in the tree size, the accuracy, and the other parameters mentioned above can be explained by the decrease in the number of training records per slice with an increase in the total number of slices. The difference between the maximum and the minimum accuracy shown in Figs. 8 and 9 is so low (0.0007 and 0.013, respectively) that it does not have any practical significance.

The high accuracy of ensemble majority voting is expected since each induced decision tree that sees another slice of the trained dataset takes part in voting for classifying each tuple in the testing dataset. In this way, although each induced decision tree sees only

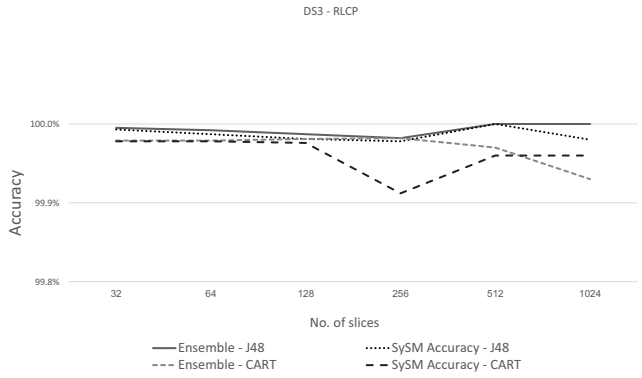

Fig. 8. DS3: Record linkage comparison pattern accuracy per dataset slice.

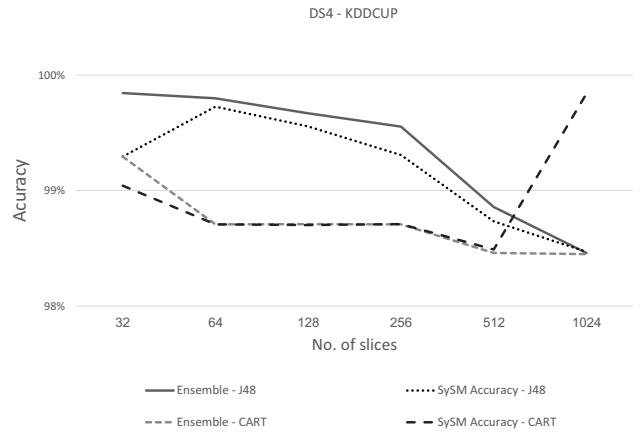

Fig. 9. DS4: KDD Cup 1999 algorithm accuracy per dataset slices.

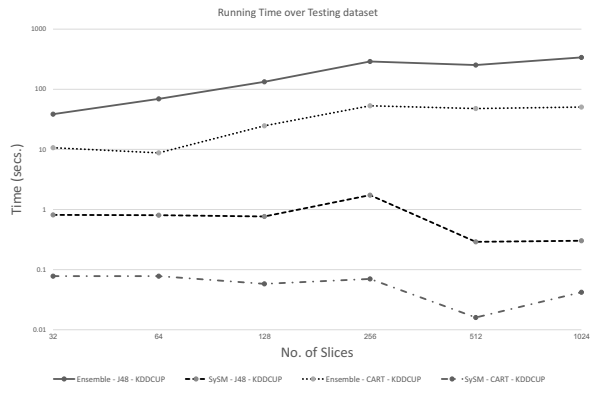

Fig. 10. Algorithm running time over the KDDCUP testing dataset.

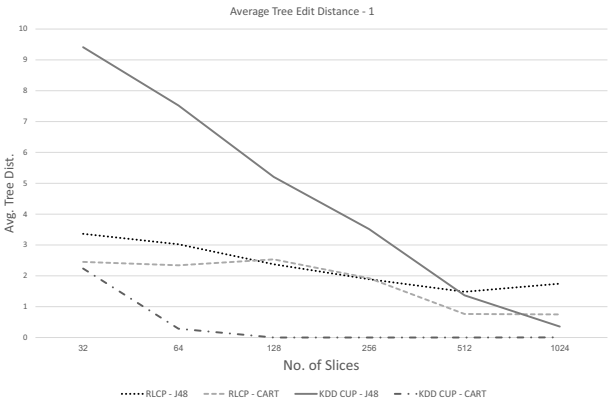

Fig. 11. Average tree edit distance: RLCP and KDD CUP.

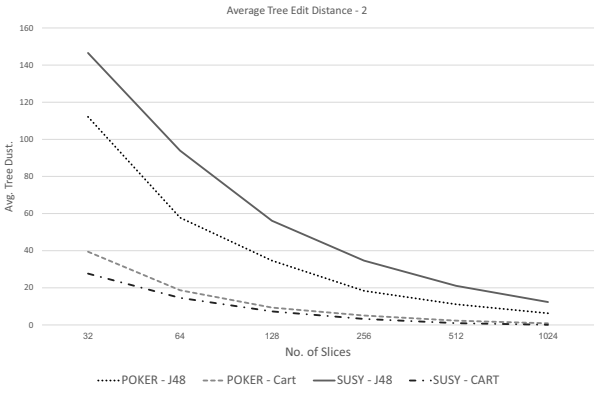

Fig. 12. Average tree edit distance: POKER and SUSY.

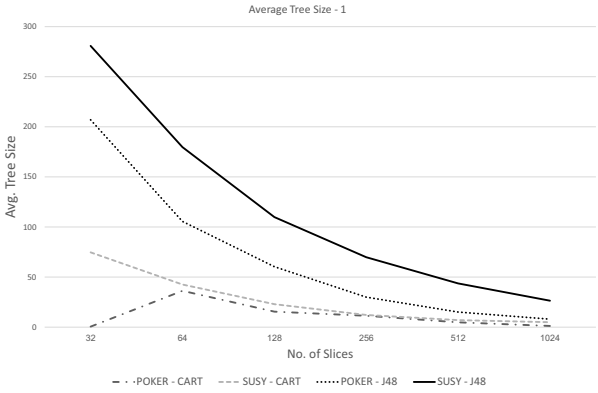

Fig. 13. Average tree size: POKER and SUSY.

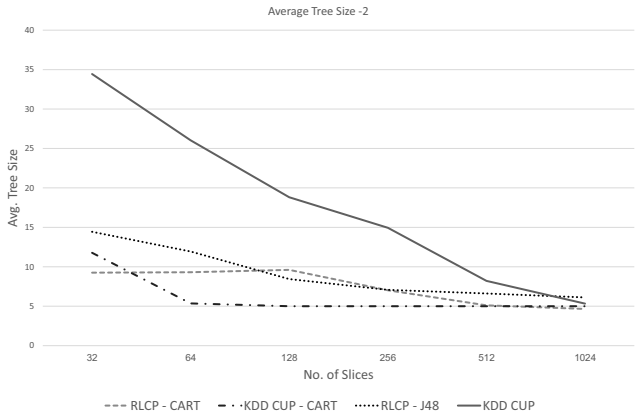

Fig. 14. Average tree size: RLCP and KDD CUP.

part of the trained dataset, the voting part combines their predictions over the testing dataset. However, the results of the majority voting algorithm are not easily interpretable, since they are not based on a single decision tree model. In addition, the classification phase of ensemble majority voting includes two computation steps: running the induced trees over the testing dataset and computing the majority voting result. The number of runs of ensemble induced trees over the testing dataset is equal to the number of slices per training dataset. In a distributed system, the induction of decision trees is done over each training dataset in parallel. However, for ensemble decision making there is a need to apply each decision-tree model to the same testing dataset. This adds computational and memory complexity to ensemble algorithms. As can be seen in Fig. 10, the total testing time is indeed higher for ensemble than for SySM at the order of the number of $folds \times tuples$.

In Appendix, we also compare the testing accuracy of the ensemble and SySM for every dataset and each decision tree algorithm (J48 and CART) using the t-test. The t-statistic is computed by the following formula:

$$\frac{\text{SySM}_{\text{Accuracy}} - \text{Ens}_{\text{Accuracy}}}{\sqrt{F(\text{SySM}_{\text{Accuracy}}) + F(\text{Ens}_{\text{Accuracy}})}},$$

where

$$F(x) = \frac{x(1-x)}{\text{number of testing records}}.$$

In the SUSY dataset, we see that most models in each shard, except the 1024 shard, have a low testing accuracy of 0.544. For the 32 shard, there are 30 decision trees (out of 32) with that accuracy, for 64 there are 62, for 128 there are 124, for 256 there are 125 out of 135, for 512 125 out of 136 and for 1024 only one. Since ensemble majority voting assigns the majority results to each record, the predicted class is supposed to be based on the trees having this low accuracy. On the contrary, SySM refers to the tree syntactic structure, rather than a statistical majority, and hence it selects a more accurate decision tree.

The overall results in Appendix show the following: SySM accuracy is significantly higher than that of the ensemble, at a 5% significance level in 18.75% of experiments with an average accuracy difference of 2.93%. In about 43.75% of the experiments, SySM accuracy is significantly lower than that of the ensemble but only by 0.57% on the average. In one case, the results are the same. In the rest of the cases (35.41%), the difference is not statistically significant. Specifically, for J48, in 33.3% of cases the difference is significantly negative, whereas in 29.2% of experiments the accuracy difference is significantly positive. For CART, in 50% of cases the difference is significantly negative, while in 4.2% of experiments the accuracy difference is significantly positive.

In general, we can conclude that SySM accuracy is comparable to ensemble. We also used the non-parametric Wilcoxon signed rank test for comparing between the accuracies of SySM and ensemble across all datasets. The results for J48 indicate that there is no statistically significant difference between the algorithms. For CART, we found a statistically significant difference between the algorithms at the p-value of 0.001. Hence, we can deduce that in most cases the ensemble approach provides slightly more accurate CART models than SySM.

5. Conclusions

In this paper, we proposed a novel approach, named SySM, for selecting one representative model out of several decision trees induced from different subsets of the same dataset. This approach is very useful for distributed systems as well as for cloud and big data environments. We suggest a syntactic approach that is based solely on the induced decision trees structure. This approach saves the computation time and the need to return back to the training dataset, which is critical in massive data environments. The SySM algorithm functions well under memory constraints. It might also fit real-time or near real-time environments such as data streams, since cache memory may be used for comparison of induced trees and for distance matrix manipulation, and there is no need to use disk memory. In addition, in the case of memory limitations, SySM can run when the ensemble cannot.

In order to evaluate the accuracy of our algorithm, we compared it with other well-known algorithms on four big data benchmark datasets. Each dataset was divided int. a number of slices from 32 to 1024. A slight decrease in SySM classification performance vs. the ensemble method that occurs in some cases is fully justified by

its improved interpretability and significantly higher classification speed. A common example of a data stream where a high classification speed is required is online advertising. The online advertising ecosystem involves streams of data generated simultaneously by millions of users including user profiles, clickstream data, cookies, recent browsing history, etc. One of the online advertising tasks is to assign ads to specific users in real time via the process of real-time bidding (RTB). According to the Google RTB requirements (`https://developers.google.com/ad-exchange/rtb/peer-guide`), the deadline for a response to a bid request is 100 ms including the network time. Our algorithm should be more suitable for such a task than the ensemble approach due to its shorter classification time.

In future work, one may utilize the fast computation characteristics of the SySM algorithm for distributed data streaming environment. In this environment, SySM may save expensive I/O disk operations, since all induced decision trees can fit in the memory of a single computer without the need to re-process the raw data. An additional future approach may use alternative data structures such as the DAG (directed acyclic graph) and additional tree similarity metrics. One may also experiment with other decision tree algorithms and node splitting metrics, and evaluate them on additional datasets. Choosing a representative decision tree in the case of vertical parallelism is another important challenge.

References

AlSabti, K., Ranka, S. and Singh, V. (1998). Clouds: Classification for large or out-of-core datasets, *Conference on Knowledge Discovery and Data Mining, New York, NY, USA*, pp. 2–8.

Amado, N., Gama, J. and Silva, F. (2001). Parallel implementation of decision tree learning algorithms, *in* P. Brazdil and A. Jorge (Eds.), *Progress in Artificial Intelligence*, Springer, Berlin/Heidelberg, pp. 6–13.

Amado, N., Gama, J. and Silva, F. (2003). Exploiting parallelism in decision tree induction, *ECML/PKDD Workshop on Parallel and Distributed Computing for Machine Learning, Cavtat/Dubrovnik, Croatia*, pp. 13–22.

Andrzejak, A., Langner, F. and Zabala, S. (2013). Interpretable models from distributed data via merging of decision trees, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Savannah, GA, USA*, pp. 1–9.

Bekkerman, R., Bilenko, M. and Langford, J. (2011). *Scaling up Machine Learning: Parallel and Distributed Approaches*, Cambridge University Press, Cambridge.

Ben-Haim, Y. and Tom-Tov, E. (2010). A streaming parallel decision tree algorithm, *The Journal of Machine Learning Research* **11**: 849–872.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line, *Machine Learning* **36**(1–2): 85–103.

Dai, W. and Ji, W. (2014). A MAPREDUCE implementation of c4.5 decision tree algorithm, *International Journal of Database Theory and Application* **7**(1): 49–60.

DeWitt, D.J., Naughton, J.F. and Schneider, D. (1991). Parallel sorting on a shared-nothing architecture using probabilistic splitting, *Proceedings of the 1st International Conference on Parallel and Distributed Information Systems, Miami Beach, FL, USA,* pp. 280–291.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA*, pp. 71–80.

Fan, W. and Bifet, A. (2013). Mining big data: Current status, and forecast to the future, *ACM sIGKDD Explorations Newsletter* **14**(2): 1–5.

Gehrke, J., Ganti, V., Ramakrishnan, R. and Loh, W.-Y. (1999). Boat—optimistic decision tree construction, *in* S. Davidson and C. Faloutsos (Eds.), *ACM SIGMOD Record*, Vol. 28, ACM, New York, NY, pp. 169–180.

Goil, S. and Choudhary, A. (2001). Parsimony: An infrastructure for parallel multidimensional analysis and data mining, *Journal of Parallel and Distributed Computing* **61**(3): 285–321.

Hansen, L.K. and Salamon, P. (1990). Neural network ensembles, *IEEE Transactions on Pattern Analysis & Machine Intelligence* **12**(10): 993–1001.

Jin, R. and Agrawal, G. (2003). Communication and memory efficient parallel decision tree construction, *Proceedings of the 3rd SIAM International Conference on Data Mining, San Francisco, CA, USA*, pp. 119–129.

Joshi, M.V., Karypis, G. and Kumar, V. (1998). SCALPARC: A new scalable and efficient parallel classification algorithm for mining large datasets, *Parallel Processing Symposium, Los Alamitos, CA, USA*, pp. 573–579.

Kargupta, H. and Park, B.-H. (2004). A Fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments, *IEEE Transactions on Knowledge and Data Engineering* **16**(2): 216–229.

Kourtellis, N., Morales, G.D.F., Bifet, A. and Murdopo, A. (2016). VHT: Vertical Hoeffding tree, *arXiv preprint*, 1607.08325.

Louppe, G. and Geurts, P. (2012). Ensembles on random patches, *in* P.A. Flach *et al.* (Eds.), *Machine Learning and Knowledge Discovery in Databases*, Springer, Berlin/Heidelberg, pp. 346–361.

Mehta, M., Agrawal, R. and Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining, *in* P. Aspers *et al.* (Eds.), *Advances in Database Technology*, Springer, Berlin/Heidelberg, pp. 18–32.

Miglio, R. and Soffritti, G. (2004). The comparison between classification trees through proximity measures, *Computational Statistics & Data Analysis* **45**(3): 577–593.

Narlikar, G.J. (1998). A parallel, multithreaded decision tree builder, *Technical report*, DTIC Document, `http://www.dtic.mil/docs/citations/ADA363531`.

Ntoutsi, I., Kalousis, A. and Theodoridis, Y. (2008). A general framework for estimating similarity of datasets and decision trees: Exploring semantic similarity of decision trees, *in* C. Apte *et al.* (Eds.), *SIAM Conference on Data Mining, SIAM*, Philadelphia, PA, pp. 810–821.

Panda, B., Herbach, J.S., Basu, S. and Bayardo, R.J. (2009). Planet: Massively parallel learning of tree ensembles with MapReduce, *Proceedings of the VLDB Endowment* **2**(2): 1426–1437.

Pawlik, M. and Augsten, N. (2011). RTED: A robust algorithm for the tree edit distance, *Proceedings of the VLDB Endowment* **5**(4): 334–345.

Shafer, J., Agrawal, R. and Mehta, M. (1996). Sprint: A scalable parallel classifier for data mining, *International Conference on Very Large Data Bases, Mumbai (Bombay), India*, pp. 544–555.

Shannon, W.D. and Banks, D. (1999). Combining classification trees using MLE, *Statistics in Medicine* **18**(6): 727–740.

Sollich, P. and Krogh, A. (1996). Learning with ensembles: How overfitting can be useful, *in* D.S. Touretzky *et al.* (Eds.)*Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge, MA, pp. 190–196.

Sreenivas, M.K., AlSabti, K. and Ranka, S. (2000). Parallel out-of-core decision tree classifiers, *in* H. Kargupta and P. Chan (Eds.), *Advances in Distributed and Parallel Knowledge Discovery*, Cambridge, MA, pp. 317–336.

Srivastava, A., Han, E.-H., Kumar, V. and Singh, V. (1995). Parallel formulations of decision-tree classification algorithms, *Data Mining and Knowledge Discovery* **3**(3): 237–261.

Triguero, I., Peralta, D., Bacardit, J., García, S. and Herrera, F. (2015). MRPR: A MAPREDUCE solution for prototype reduction in big data classification, *Neurocomputing* **150**(A): 331–345.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems, *SIAM Journal on Computing* **18**(6): 1245–1262.

Zhang, X. and Jiang, S. (2012). A splitting criteria based on similarity in decision tree learning, *Journal of Software* **7**(8): 1775–1782.

Zhang, Y., Gao, Q., Gao, L. and Wang, C. (2012). IMAPREDUCE: A distributed computing framework for iterative computation, *Journal of Grid Computing* **10**(1): 47–68.

Abraham Itzhak Weinberg has spent over 25 years in the fields of software and information systems. He had served for six years in the IAF (Israeli Air Force) and retired as a captain. In recent years, he has managed a BI (business intelligence) unit and data warehousing projects, and has consulted data science projects as well as projects integrating big data and cybersecurity. His academic background consists of a BSc in industrial engineering and management as well as computer science, and an MSc in industrial engineering. Nowadays, in addition to working in cybersecurity industry, he is pursuing his PhD studies at the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel.

Mark Last is a full professor at the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel, and the head of the Data Mining and Software Quality Engineering Group. In the years 2009–2012, he served as the head of the Software Engineering Program at Ben-Gurion University. Prof. Last obtained his PhD degree from Tel Aviv University, Israel, in 2000. He has published over 190 peer-reviewed papers and 10 books on data mining, text mining, and software engineering. He currently serves as an associate editor of *IEEE Transactions on Cybernetics* and an editorial board member of *Data Mining and Knowledge Discovery*. From 2007 to 2016, he served as an associate editor of *Pattern Analysis and Applications*. His main research interests are focused on data mining, cross-lingual text mining, cyber intelligence, and medical informatics.

Appendix

SySM accuracy vs. ensemble majority voting (3 digits)

Table A1. J48 testing accuracy.

Dataset	Number of slices	Ensemble J48 accuracy	SySM J48 accuracy	Difference	P-value	Testing size
Poker	32	0.632	0.592	-0.04	<0.001	100000
Poker	64	0.594	0.566	-0.028	<0.001	100000
Poker	128	0.537	0.533	-0.003	0.051	100000
Poker	256	0.499	0.521	0.022	<0.001	100000
Poker	512	0.499	0.525	0.026	<0.001	100000
Poker	1024	0.499	0.484	-0.014	<0.001	100000
SUSY	32	0.543	0.788	0.245	<0.001	500000
SUSY	64	0.543	0.786	0.242	<0.001	500000
SUSY	128	0.543	0.783	0.239	<0.001	500000
SUSY	256	0.543	0.778	0.234	<0.001	500000
SUSY	512	0.543	0.772	0.228	<0.001	500000
SUSY	1024	0.781	0.767	-0.013	<0.001	500000
RLCP	32	0.781	0.767	-2E-05	0.281	100000
RLCP	64	0.999	0.999	-5E-05	0.137	100000
RLCP	128	0.999	0.999	-6E-05	0.144	100000
RLCP	256	0.999	0.999	-4E-05	0.327	50000
RLCP	512	1	1	0	–	10000
RLCP	1024	1	0.999	-0.0002	0.078	10000
KDD CUP	32	0.998	0.993	-0.005	<0.001	150000
KDD CUP	64	0.997	0.997	-0.0007	<0.001	150000
KDD CUP	128	0.996	0.995	-0.001	<0.001	150000
KDD CUP	256	0.995	0.993	-0.002	<0.001	150000
KDD CUP	512	0.988	0.987	-0.001	0.036	50000
KDD CUP	1024	0.984	0.984	1E-04	0.477	10000

Table A2. CART testing accuracy.

Dataset	Number of slices	Ensemble CART accuracy	SySM CART accuracy	Difference	P-value	Testing size
Poker	32	0.563	0.549	-0.014	<0.001	100000
Poker	64	0.550	0.542	-0.007	<0.001	100000
Poker	128	0.536	0.533	-0.003	0.096	100000
Poker	256	0.521	0.504	-0.016	<0.001	100000
Poker	512	0.499	0.486	-0.013	<0.001	100000
Poker	1024	0.499	0.471	-0.027	<0.001	100000
SUSY	32	0.791	0.784	-0.006	<0.001	500000
SUSY	64	0.787	0.779	-0.008	<0.001	500000
SUSY	128	0.781	0.769	-0.012	<0.001	500000
SUSY	256	0.781	0.765	-0.016	<0.001	500000
SUSY	512	0.781	0.756	-0.025	<0.001	500000
SUSY	1024	0.781	0.751	-0.03	<0.001	500000
RLCP	32	0.999	0.999	-1E-05	0.44	100000
RLCP	64	0.999	0.999	-1E-05	0.44	100000
RLCP	128	0.999	0.999	-5E-05	0.222	100000
RLCP	256	0.999	0.999	-0.0007	<0.001	50000
RLCP	512	0.999	0.999	-1E-04	0.352	10000
RLCP	1024	0.999	0.999	0.0003	0.182	10000
KDD CUP	32	0.992	0.990	-0.002	<0.001	150000
KDD CUP	64	0.987	0.987	0	0.5	150000
KDD CUP	128	0.987	0.987	-8E-05	0.455	150000
KDD CUP	256	0.987	0.987	2E-05	0.488	150000
KDD CUP	512	0.984	0.984	0.0003	0.431	50000
KDD CUP	1024	0.984	0.998	0.0139	<0.001	10000

Received: 30 November 2016
Revised: 1 May 2017
Re-revised: 21 July 2017
Accepted: 8 August 2017