Estimation of Distribution Algorithms for Decision-Tree Induction

Henry E. L. Cagnini, Rodrigo C. Barros
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Rio Grande do Sul, Brazil
Email: henry.cagnini@acad.pucrs.br
Email: rodrigo.barros@pucrs.br

Márcio P. Basgalupp
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, São Paulo, Brazil
Email: basgalupp@unifesp.br

***Abstract*—Decision trees are one of the most widely employed classification models, mainly due to their capability of being properly interpreted and understood by the domain specialist. However, decision-tree induction algorithms have limitations due to the typical recursive top-down greedy search they implement. Such local search may often lead to quality loss while the partitioning process occurs, generating statistically insignificant rules. In order to avoid the typical greedy strategy and to prevent convergence to local optima, we present a novel Estimation of Distribution Algorithm (EDA) for decision-tree induction, namely Ardennes. For evaluating the proposed approach, we present results of an empirical analysis in 10 real-world classification datasets. We compare Ardennes with both a well-known traditional greedy algorithm for decision-tree induction and also with a more recent global population-based approach. Results show the feasibility of using EDAs as a means to avoid the previously-described problems. We report gains when using Ardennes in terms of accuracy and – equally important – tree comprehensibility.**

I. Introduction

Classification is the data mining task of predicting the class label of unseen objects based on previously labeled training data [1]. There are several algorithms for generating classification models. Decision-tree induction algorithms, in particular, are among the most popular classification approaches due to model comprehensibility, robustness to noise, speed regarding both training and inference phases, and ability to deal with redundant attributes [2]. A decision tree is a directed acyclic graph (DAG) with edges and nodes [3]. Inner nodes represent tests over the predictive attributes in order to divide the instance space into two or more subsets. The edges represent the possible outcomes of the test, allowing the objects to be distributed along mutually-exclusive paths within the tree. The inference stops when an object reaches a leaf node, which encodes one of the mutually-exclusive classes. Successful applications of decision trees include gene expression classification [4], [5], software fault prediction [6], [7], and molecular docking [8], just to name a few.

There are exponential many decision trees that can be built from the same data, with different levels of predictive quality and compactness. Generating the optimal decision tree regarding a given quality criterion is a combinatorial optimization problem. For instance, the problem of inducing an optimal decision-tree from decision tables is NP-hard, whereas evolving a minimum binary tree is NP-complete. In practice, current decision-tree induction algorithms are heuristic-driven, employing a greedy local-search for generating reasonably-accurate albeit suboptimal decision trees [3].

There are at least two major problems related to the greedy local-search employed by traditional decision-tree induction algorithms [9]: (i) the greedy strategy often produces locally (rather than globally) optimal solutions, (ii) recursive partitioning iteratively degrades the quality of the dataset for the purpose of statistical inference, contributing to the problem of data overfitting. To address these problems, distinct strategies have been proposed in the last decade or so, in particular focusing on meta-heuristics based on global search [10], [11].

In this paper, following the trend of inducing decision trees based on global-search population-based approaches, we propose a novel Estimation of Distribution Algorithm (EDA) [12], namely Ardennes. An EDA works by initially building a probability distribution encoded according to a Probabilistic Graphical Model (GM) [13]. From the initial distribution, S solutions (i.e individuals) are sampled. Then, each solution is evaluated based on a user-defined fitness function, similarly to traditional evolutionary algorithms (EAs) such as genetic algorithms (GAs) and genetic programming (GP). The $|\Phi|$ best-performing individuals are employed for updating the probability distribution, and the worst-performing individuals are replaced by resampling novel individuals based on the updated distribution. Since the resampled individuals have inherited features from the previous best-performing solutions, it is expected that this procedure leads to better and better solutions across time, until the probability distribution converges or a given time frame is reached. The main advantage of EDAs over other EAs is that it directly operates over a GM (rather than relying on operators such as crossover), providing a transparent evolutionary process where we can see how variables change during evolution. Ardennes performs decision-tree induction over real-valued predictive attributes and categorical class attributes. Based on experiments over public datasets, we show that Ardennes outperforms a GA-based approach while being competitive with the state-of-the-art greedy top-down approach. To the best of our knowledge,

978-1-5090-4601-0/17/$31.00

Ardennes is the first EDA proposed for inducing decision trees.

The rest of this paper is organized as follows. Section II introduces Ardennes in detail. Section III presents the algorithms, datasets, and evaluation criteria used for empirically analyzing the proposed method. Section IV reports the results obtained from the empirical evaluation, whereas Section V briefly describes related work. Finally, we present our conclusions and point to future work in Section VI.

II. Ardennes

Ardennes is our proposed EDA for decision-tree induction. It focus only on classification problems whose predictive attributes are numeric (ordinal) and with no missing values. Since there are many ways to deal with missing values [14], we assume it is part of a prior data preprocessing step. Extensions of Ardennes for categorical attributes will be considered in future work. In next sections, we present how solutions are encoded in Ardennes (Section II-A), as well as its splitting criterion (Section II-B), its GM updating process (Section II-C), and fitness function (Section II-D). We end this section with Ardennes' time complexity analysis (Section II-E).

A. Individual Encoding

In Ardennes, individuals are binary trees whose height is at most H, a predefined user parameter. They are sampled from a graphical model (GM) which resembles a complete binary tree with H levels, modelled as follows. Each variable within the GM is a probabilistic mass function (PMF) that indicates the probability of choosing a given attribute (either the class or a predictive attribute from A, the set of predictive attributes) to be selected for the corresponding node of the decision tree. The number of variables at a given level h is 2^{h-1}, and the total number of variables in the GM are $2^H - 1$. Since we assume there is no dependence between variables, i.e, the probability of selecting an attribute in a given node does not affect probabilities in other nodes, the GM is not a tree *per se* since there are no edges linking the variables.

The initial probabilities for the predictive attributes in all nodes of the GM follow a uniform distribution. The probability of selecting the class attribute C, however, is zero for all variables V between the root and level $H-1$. At level H the probability of selecting the class is 100%. Note that the probability of selecting the class in intermediate levels may increase during evolution, since a homogeneous node (all objects belonging to the same class label) are automatically transformed into class nodes. Figure 1a depicts the initial state of a GM with $H = 4$.

The sampling of nodes is performed in a *on-demand* fashion following a depth-first search: the root node is sampled first, then its left child, the left child of that child and so on. If the class attribute is sampled, the current node is turned into a leaf and its label is set according to the most frequent class found in X_π, the subset of instances reaching that node. Otherwise if a predictive attribute is sampled, we perform a binary split on the instances reaching that node following the splitting criterion described in Section II-B.

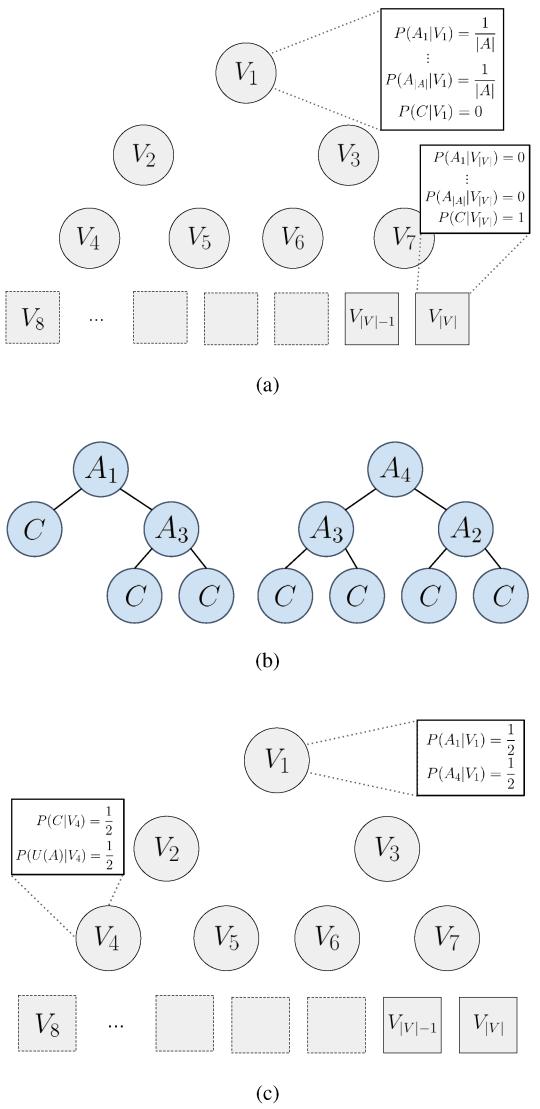


Fig. 1: (a) An initial GM with $H = 4$. Note that the probability of sampling class attribute C is 100% for the final level and 0% for all remaining levels. (b) The fittest population responsible for updating the GM. (c) The updated GM. Variables from the GM that are absent in the fittest population are updated by sampling predictive attributes from a uniform distribution over A.

The sampling process is repeated until one of the following leaf-turning conditions are met:

- The class attribute is sampled;
- None of the possible threshold values for the sampled attribute provides a gain in information, which occurs when all instances are identical regarding the sampled attribute;
- the current node is already homogeneous (all instances belong to the same class label).

Since the probability of sampling the class at level H is always 100%, we can assume that one of those conditions

will certainly be met.

B. Splitting criterion

For defining the threshold values for predictive attributes that are sampled within inner nodes, Ardennes make use of the information gain ratio, which is the same criterion used by C4.5 [15], the most well-known decision-tree induction algorithm. Information gain ratio is a formulation that considers the entropy [16] as a measure of impurity regarding the class distribution within a given node [2]:

$$H(\mathbf{X}_\pi) = -\sum_{c \in C} p(c|\mathbf{X}_\pi) \log_2 p(c|\mathbf{X}_\pi) \quad (1)$$

where C is the set of class labels and $\mathbf{X}_\pi$ is the set of instances at the current node. A set of homogeneous instances achieves minimum entropy, whereas a perfectly heterogeneous set yields maximum entropy (i.e., uniform class distribution).

Information gain [2] aims to measure the decrease in impurity (i.e., the gain in information) before and after splitting the data according to a given test:

$$IG(\mathbf{X}_\pi) = H(\mathbf{X}_\pi) - \sum_{\mathbf{X}_\varepsilon \in \mathbf{X}_\pi} \frac{|\mathbf{X}_\varepsilon|}{|\mathbf{X}_\pi|} H(\mathbf{X}_\varepsilon) \quad (2)$$

where $\mathbf{X}_\varepsilon$ is a subset of $\mathbf{X}_\pi$.

Finally, Information Gain Ratio [15] compensates the decrease in entropy that naturally occurs when multiple partitions are generated during splitting. It divides the information gain by the entropy of subset $\mathbf{X}_\pi$ [2]:

$$IGR(\mathbf{X}_\pi) = \frac{IG(\mathbf{X}_\pi)}{SI(\mathbf{X}_\pi)} \quad (3)$$

where $SI(\mathbf{X}_\pi)$ is the entropy of the subsets generated by splitting $\mathbf{X}_\pi$:

$$SI(\mathbf{X}_\pi) = -\sum_{\mathbf{X}_\varepsilon \in \mathbf{X}_\pi} \frac{|\mathbf{X}_\varepsilon|}{|\mathbf{X}_\pi|} \log_2 \frac{|\mathbf{X}_\varepsilon|}{|\mathbf{X}_\pi|} \quad (4)$$

C. GM updating process

We employ a lexicographic approach [17] for deciding which individuals will be used to update the GM. We sort the individuals based on three criteria: fitness (descending), tree height (ascending), and number of nodes (ascending). If individuals present the exact same fitness, then the tree height is used to break the tie; if a tie persists, then the individual with fewer nodes in its tree is prioritized. After sorting we select the first $|\Phi|$ individuals (with $|\Phi|$ provided by the user) for updating the GM's variables and we resample the $|S| - |\Phi|$ remaining individuals using the updated GM.

For each variable in the GM, we count how many times each attribute appeared in the corresponding node regarding the set of fittest individuals Φ. Variables in the GM that do not appear in Φ have their probabilities reset according to a uniform distribution U over A, $U(A)$. Figure 1 depicts the updating process.

By using the lexicographic approach plus the uniform component we can simulate a exploration-exploitation behavior. During the initial generations of evolution, ties in fitness are expected to be rare, and so the EDA performs exploration of solutions. When individuals start to present similar fitness, the exploratory procedure is replaced by exploitation, with a preference to smaller individuals. The EDA refines same-fitness trees to their more compact form, which may be viewed as an implicit pruning step. This process of exploration-exploitation-pruning may repeat itself several times throughout evolution, as will be further detailed in Section IV.

The evolutionary process stops when the GM has converged (all variables' probabilities have stabilized) or when a maximum number of $|G|$ generations is reached, with $|G|$ provided by the user. From Φ of the last generation, we then select as best individual the one that presents the best quality, as detailed next.

D. Fitness Function

Ardennes receives as input two sets: (i) a training set is used for setting the thresholds for the tests over the attributes according to the splitting criterion; and (ii) a validation set, which is used for verifying the performance of the tree on unseen data. The fitness of an individual is given by:

$$fitness(s_i) = acc_{s_i}(training) \quad (5)$$

where s_i is the i-th individual in a given generation g and $acc(training)$ is the accuracy regarding the training set, with $acc = (TP+TN)/(TP+TN+FP+FN)$[1]. Fitness ranges from 0 to 1, with higher values preferred. After evolution is completed, the best individual is selected from the last generation, but this time using the accuracy achieved over unseen data (the validation set). The quality of individuals in the last generation is given by:

$$quality(s_i) = \frac{1}{2}[acc_{s_i}(training) + acc_{s_i}(validation)] \quad (6)$$

Therefore, all S individuals from the last generation are employed to classify the objects of the validation set, and the best performing individual regarding this quality index is returned as the tree induced by Ardennes for the corresponding dataset. The pseudocode for Ardennes is presented in Figure 2.

```
1: function ARDENNES(H, |G|, |S|, |Φ|)
2:     initialize the graphical model GM using H
3:     sample population S from GM
4:     g ← 0
5:     while g < |G| and GM has not converged do
6:         select the |Φ| fittest individuals (based on Equation 5) from S to compose
       the fittest subsample Φ
7:         update GM based on Φ
8:         from the updated GM resample |S| − |Φ| individuals which are not in Φ
9:         g ← g + 1
10:    return best individual from S based on Equation 6
```

Fig. 2: Ardennes pseudocode for the training phase.

[1]TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

E. Complexity Analysis

Since all variables of the GM are updated independently of the fittest sub-population configuration, the cost of updating the GM is $\mathcal{O}(|G| \cdot (2^H - 1) \cdot |\Phi|)$. Sampling the rest of the population at each generation is $\mathcal{O}((|S| - |\Phi|) \cdot (2^H - 1))$, in the worst case of always generating complete trees. Let τ be the cost of calculating thresholds for each node and γ the cost of calculating the fitness function. Hence, the full cost complexity of Ardennes is given by:

$$\mathcal{O}\big(|G| \cdot \big[(\tau \cdot (2^H - 1) \cdot \gamma \cdot (|S| - |\Phi|)) + |\Phi| \cdot (2^H - 1)\big]\big) \quad (7)$$

III. Experimental Setup

In this section we describe the datasets, baseline algorithms, and criteria used to empirically evaluate the performance of Ardennes.

A. Datasets

We use 10 public datasets from the UCI Machine Learning Repository [18]. All datasets are restricted to numeric predictive attributes and a categorical class attribute. The selected datasets describe a wide variety of application domains: from flower classification to heart disease diagnosis. The characteristics of the datasets are presented in Table I.

TABLE I: Datasets from the UCI ML Repository [18].

name	# objects	# attributes	# classes
column-2c	310	6	2
column-3c	310	6	3
ionosphere	351	33	2
iris	150	4	3
liver-disorders	345	6	2
sonar	208	60	2
tep-fea	3572	7	3
transfusion	748	4	2
vehicle	846	18	2
wine	178	13	3

B. Hyper-Parameters and Baseline Algorithms

The hyper-parameters used in Ardennes are the following: $|S| = 200$, $|\Phi| = 100$ (i.e 50%), $|G| = 100$ and $H = 9$. We did not make any attempt at optimizing those parameters, a task left for future work. We perform a stratified 10-fold cross-validation procedure (10CV) on each of the datasets, using one fold for test, one for validation, and the remaining 8 for setting the thresholds according to the splitting criterion, per round of the 10CV. Since Ardennes is a stochastic algorithm, we run it 10 times for each fold. We compare Ardennes with J48 [19], the Java implementation of C4.5 [15]. Moreover, we also compare Ardennes with an evolutionary GA-based approach for inducing decision trees, namely LEGAL-Tree [9].

J48 is the Java version of the C4.5 algorithm [15]. It performs deterministic top-down inference of decision-trees, and it is available in the Weka Toolkit[2]. We run it using its default parameters. Since J48 does not require a validation set, we provide the entire 9 folds per round of the 10CV for being used as training. In addition, since it is a *deterministic* algorithm, we run it only once for each fold.

LEGAL-Tree [9] is a genetic algorithm for performing lexicographic induction of decision trees. It can use more than one criterion for evaluating how good solutions are, prioritizing one over another when a tie is found. The authors use accuracy as fitness function, with tree height (smaller trees are preferred) for breaking ties. LEGAL-Tree also makes use of two distinct sets during its training process (a training set and a validation set), which are the same as those used by Ardennes. Since LEGAL-Tree is a stochastic algorithm, we use the same framework described before – 10 runs for each fold – with the parameters proposed by the authors in the original paper [9].

[2]Available at http://www.cs.waikato.ac.nz/ml/weka/downloading.html.

IV. Experimental Results

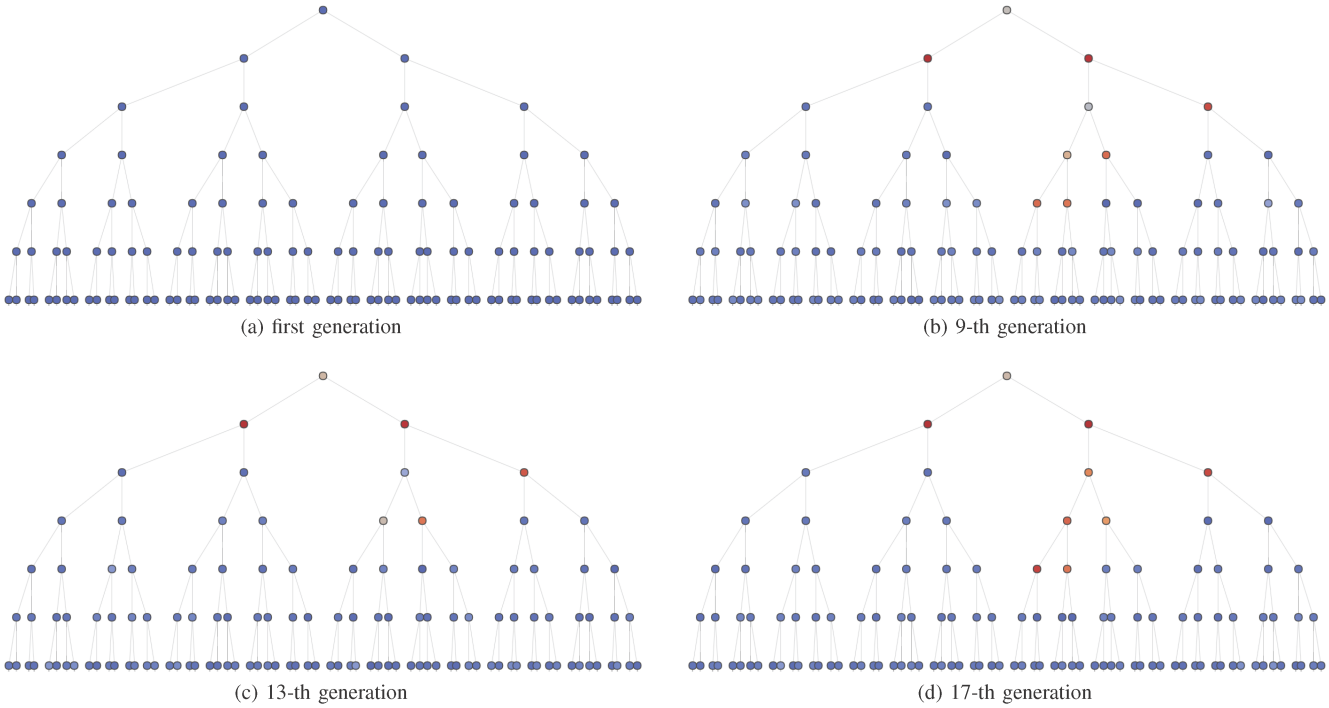
In this section, we present the thorough empirical analysis that was executed to validate the performance of Ardennes. First, in Section IV-A we carefully analyze the behavior of Ardennes throughout evolution in one of the 10 datasets from the UCI repository that were employed in this work. We show in Section IV-B the results obtained by Ardennes and the baseline approaches in terms of accuracy, tree size (number of nodes), and tree height (number of tree levels).

A. Evolution Analysis

In order to evaluate the assumption made in Section II-C regarding the exploration-exploitation-pruning capabilities of Ardennes, we carefully analyze the performance of Ardennes throughout a single evolutionary run over the well-known Iris dataset. As previously mentioned in Section I, we can understand the performance of EDAs throughout evolution since it explicitly modifies the probability distribution of the solutions. The evolution of probabilities through time within the GM is shown in Figure 3. Colder colors (blue-ish) indicate heterogeneous probabilities, i.e., closer to the uniform distribution, whereas hotter colors (red-ish) indicate homogeneous probabilities.

In the first generation (Figure 3a) all variables are initialized uniformly. By the 9th generation (Figure 3b), note that most individuals within the fittest population present the same structure, as denoted by the dominance of hot colors in several variables. This can also be detected in Figure 4, which shows that at the 9th generation all individuals in Φ present a tree height of 6 levels. However, since the fitness of all $|\Phi|$ individuals converged to the same value at that point of evolution, trees with smaller sizes began to be prioritized.

Note that in the last generation (Figure 3d) possibly another set of predictive attributes were arranged in that same structure of the 9th generation, finally ending the evolutionary process. As expected, the fittest individual from the last generation, which is shown in Figure 7, presents the structure depicted in Figure 3d. It achieves an accuracy of 93.33% in the test set and 100% in both training and validation sets, confirming that



(a) first generation (b) 9-th generation

(c) 13-th generation (d) 17-th generation

Fig. 3: Evolution of the GM's probabilities throughout the first (a), 9th (b), 13th (c) and last (d) generations within an evolutionary cycle of Ardennes over the *iris* dataset. Hot colors indicate that an attribute is becoming more frequent than others. Best viewed in colors.

Ardennes is properly optimizing the data which is available for learning.

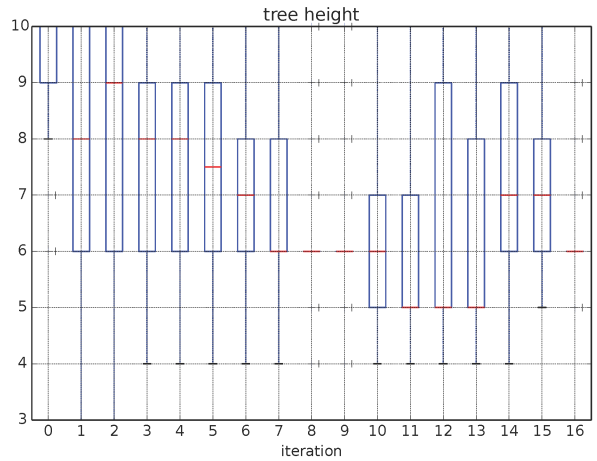


Fig. 4: Tree height across 17 iterations of Ardennes over the Iris dataset. Best viewed in colors.

By analyzing Figure 5, one can see that Ardennes is overfitting the training set. For this particular case, however, there is a significant correlation between training/validation and the test set, so this behavior is actually beneficial to the evolution of decision trees. Observe in Figure 6 that from the second to the last generation, the mean test accuracy (shown

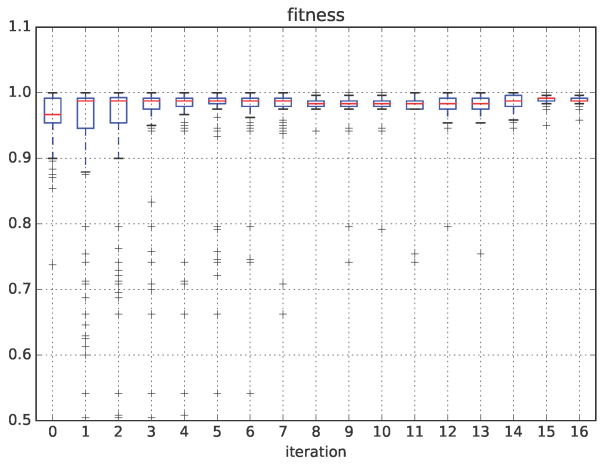


Fig. 5: Fitness across 17 iterations of Ardennes over the Iris dataset. Best viewed in colors.

here only for analysis purposes, since Ardennes does not use it during evolution) is 100%. This is confirmed when we verify the test accuracy of all individuals from the last generation: 196 individuals achieve a perfect score (100% accuracy) in the test set, whereas the remaining four achieve 93.33% of accuracy. Unfortunately, the individual selected by Ardennes (whose reported test accuracy is 93.33%) was one of the four individuals that do not achieve a perfect score. In order to

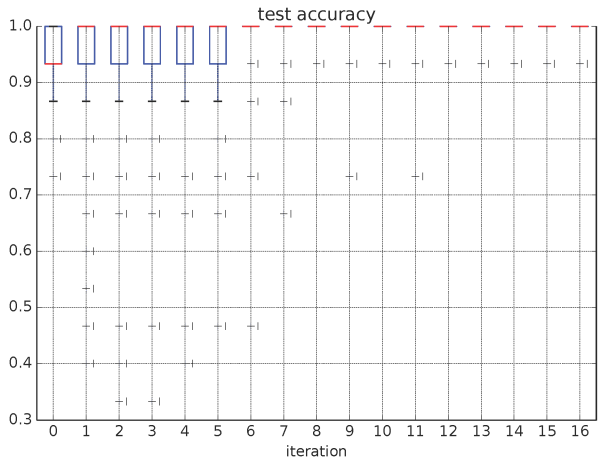


Fig. 6: Test accuracy across 17 iterations of Ardennes over the Iris dataset. Best viewed in colors.

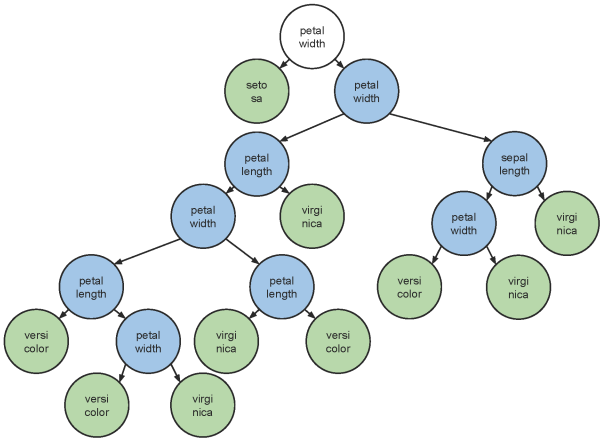


Fig. 7: Best individual found in the last generation of Ardennes for a given run over the Iris dataset. Note that its structure is in conformity with the one presented in Figure 3d.

further investigate that behavior, we selected all individuals that present maximum *quality* in the last generation and show them in Table II. One can see that, at that point of the evolutionary process, Ardennes did not have any other criterion to discern individuals from one another, so the selection of the best individual to be provided to the user became random, which is the reason why Ardennes did not achieve a perfect score for that particular data fold.

TABLE II: Fittest individuals (considering the *quality* criterion) from the last generation of Ardennes for the Iris dataset. Individual 6 was randomly selected among the 3 possibilities.

Individual	Training Accuracy	Tree Height	Number of Nodes	Validation Accuracy	Test Accuracy
6	1.0	7	19	1.0	0.933
17	1.0	7	19	1.0	1.000
44	1.0	7	19	1.0	1.000

B. Results

Next, we show the results obtained by Ardennes in the 10 UCI datasets, as well as the performance of the baseline algorithms for the same exact data partitions (Table III). We start our analysis by comparing Ardennes with the genetic algorithm for decision-tree induction, namely LEGAL-Tree. Regarding accuracy, note that Ardennes is capable of outperforming LEGAL-Tree in 8 out of 10 datasets, while presenting the same results in datasets *iris* and *tep_fea*. With regards to tree height, Ardennes produces trees with fewer levels in 9 of the 10 datasets, only presenting a deeper tree in the *iris* dataset. Recall that both *tree height* and *total number of nodes* can be used as proxies for model comprehensibility, which is a very important feature in several application domains (e.g., medical data). In terms of total number of nodes, Ardennes provides smaller trees for all datasets when compared to LEGAL-Tree. Hence, we argue that Ardennes generates trees that are not only more accurate trees but also more comprehensible trees than LEGAL-Tree. .

In Figure 8, we can also verify that Ardennes is much more stable in its evolutionary process than LEGAL-Tree, with much smaller standard deviations across runs. Moreover, Figures 9 and 10 demonstrate that Ardennes is better at exploiting the lexicographic approach than LEGAL-Tree, since it provides smaller trees without compromising accuracy.

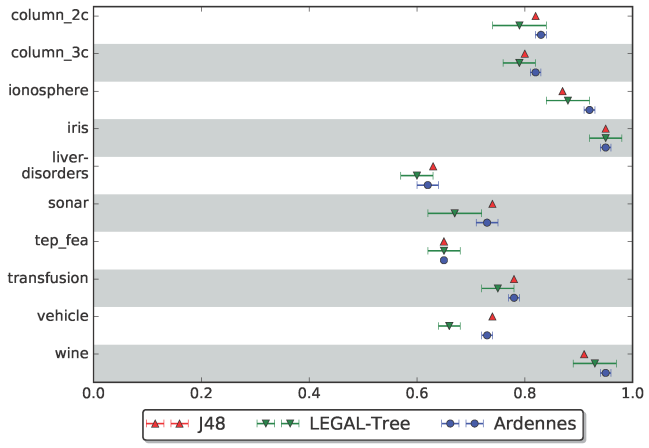


Fig. 8: Accuracy and standard deviation (the latter only for stochastic algorithms) in the evaluated datasets. Best viewed in colors.

Our last analysis is the comparison between Ardennes and J48. By looking at Table III it is clear that J48 is the strongest baseline. Ardennes is capable of outperforming it in 4 out of the 10 datasets, while presenting equivalent performance in 3 datasets and losing in the remaining 3. Note, however, that for the 3 datasets that Ardennes is outperformed by J48, the difference in accuracy is within a margin of 1%. On the other hand, in two datasets that Ardennes is superior (*ionosphere* and *wine*), the advantage in terms of accuracy is of $\approx 5\%$. Regarding tree height, Ardennes and J48 are tied in the first place, both presenting an average rank of 1.6 (the smaller the

TABLE III: J48, LEGAL-Tree and Ardennes results for all evaluated algorithms. Best accuracy results are shown underlined and in bold, whereas ties are shown only in bold.

	J48			Legal-tree			Ardennes		
Dataset	Test Accuracy	Tree Height	Nodes	Test accuracy	Tree Height	Nodes	Test Accuracy	Tree Height	Nodes
column_2c	0.82	7.10 ± 1.73	17.60 ± 6.76	0.79 ± 0.05	10.66 ± 0.47	83.98 ± 4.81	**0.83 ± 0.01**	8.59 ± 0.17	28.10 ± 1.77
column_3c	0.80	7.70 ± 0.95	22.60 ± 5.64	0.79 ± 0.03	11.25 ± 0.54	86.18 ± 3.8	**0.82 ± 0.01**	8.66 ± 0.15	40.54 ± 3.28
ionosphere	0.87	9.70 ± 1.16	27.00 ± 3.69	0.88 ± 0.04	11.54 ± 0.79	55.36 ± 5.36	**0.92 ± 0.01**	7.93 ± 0.34	18.42 ± 1.06
iris	**0.95**	4.70 ± 0.48	8.40 ± 0.92	**0.95 ± 0.03**	4.71 ± 0.44	13.7 ± 1.84	**0.95 ± 0.01**	5.14 ± 0.32	10.92 ± 1.34
liver-disorders	**0.63**	9.60 ± 0.97	44.40 ± 8.58	0.6 ± 0.03	12.91 ± 0.47	190.74 ± 5.45	0.62 ± 0.02	8.98 ± 0.04	39.00 ± 1.61
sonar	**0.74**	8.00 ± 0.67	28.00 ± 3.00	0.67 ± 0.05	9.76 ± 0.28	82.78 ± 3.36	0.73 ± 0.02	7.79 ± 0.17	34.60 ± 1.64
tep_fea	**0.65**	4.40 ± 0.97	7.80 ± 1.83	**0.65 ± 0.03**	2.95 ± 0.48	7.04 ± 1.11	**0.65 ± 0.00**	2.00 ± 0.00	3.00 ± 0.00
transfusion	**0.78**	6.70 ± 2.16	12.80 ± 4.51	0.75 ± 0.03	12.32 ± 1.13	138.28 ± 26.72	**0.78 ± 0.01**	8.21 ± 0.32	20.64 ± 3.08
vehicle	**0.74**	14.60 ± 1.78	137.80 ± 26.35	0.66 ± 0.02	17.14 ± 0.67	422.8 ± 11.29	0.73 ± 0.01	8.85 ± 0.07	67.92 ± 2.64
wine	0.91	4.10 ± 0.32	9.80 ± 0.98	0.93 ± 0.04	5.48 ± 0.54	19.98 ± 2.61	**0.95 ± 0.01**	4.27 ± 0.24	8.96 ± 0.64
Average rank	1.85	**1.6**	1.6	2.6	2.8	2.9	**1.55**	**1.6**	**1.5**

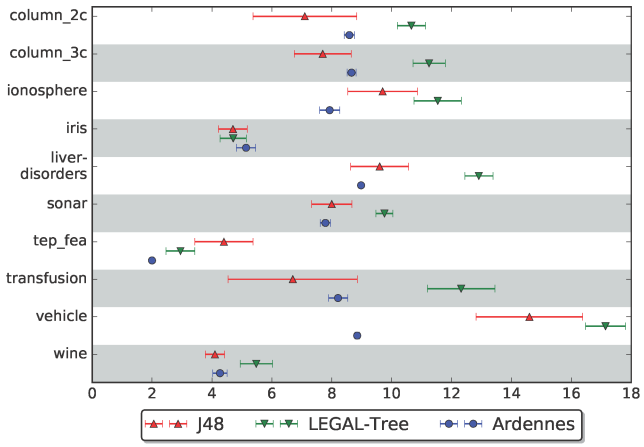


Fig. 9: Tree height and standard deviation (across folds for J48 and folds+runs for Ardennes and LEGAL-Tree) in the evaluated datasets. Best viewed in colors.

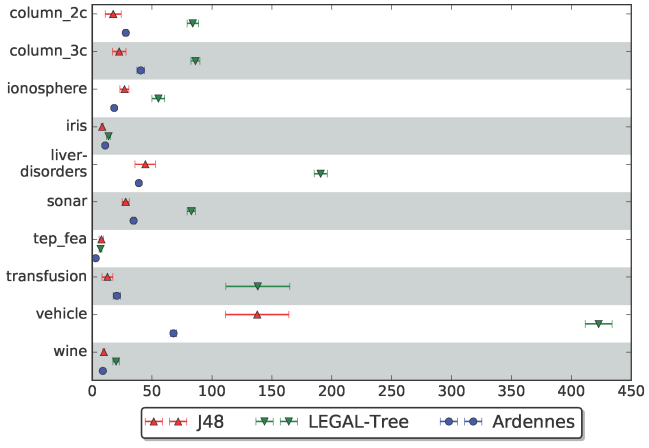


Fig. 10: Number of nodes and standard deviation (across folds for J48 and folds+runs for Ardennes and LEGAL-Tree) in evaluated datasets. Best viewed in colors.

better). However, we can see that Ardennes produces trees on average 5% smaller than J48. Indeed, Ardennes presents the best average rank in terms of tree size (1.5). For instance, the largest difference in tree size is in the *vehicle* dataset, where Ardennes has an accuracy 1% smaller than J48, though it manages to produce trees which are, on average, half the size of those provided by J48. In terms of stability, note that Ardennes seems to be the best choice among all baseline algorithms, be it either regarding accuracy, tree height, or number of nodes, as shown respectively in Figures 8, 9 and 10.

V. Related Work

Deterministic greedy top-down inference is by far one of the most popular approaches for generating decision trees [3]. Algorithms that follow the original Hunt's strategy [1] seek to recursively maximize the homogeneity of the generated nodes by splitting instances into purer subsets. Although maximizing homogeneity is a desirable feature of a decision tree, it presents two problems. First, partitioning towards purer subsets is a greedy heuristic, and as such performs local optimization rather than global optimization [9], which means it may not find good approximations for the global optimum. Second, it may lead to overfitting, since iteratively partitioning the training set reduces the significance of subsets with respect to the complete distribution. For instance, leaves with only one object achieve maximum purity for any measure. C4.5, a popular decision-tree induction algorithm proposed by Quinlan [15] uses error-based pruning to reduce complexity and avoid data overfitting.

An active field regarding decision-trees is to use evolutionary computation [11] for performing global induction, most notably through the use of EAs such as genetic algorithms (GAs) and genetic programming (GP). The behavior of a GA/GP may be summarized as follows: by randomly sampling S individuals in the first iteration, the GA will rely on operators such as mutation and crossover in order to generate novel individuals in the following iterations. Although EDAs and GAs/GP share a fair amount of similarities regarding their evolutionary processes, their differences lie in the way that novel individuals are generated. Whereas EDAs use explicit evolutionary operators [12], [20], which are easy to keep

track and less prone to yield irrelevant individuals [21], [22], GAs/GP use crossover and mutation. The latter need more iterations in order to overcome the random nature of its operators [20], and the evolution is not as easy to follow as in EDAs.

To the best of our knowledge, this is the first work to propose EDAs for decision-tree induction. Nevertheless, we take inspiration for using a GM which resembles a tree from the work of Salustowicz and Schmidhuber [23], which presents a similar implementation though with a totally different application. In their work, the authors sample programs from a univariate distribution. Each node at the GM encodes functions such as $\sin$, $\cos$, $\div$, $\times$, and other math operands. The probabilities at each node are initialized uniformly, according to the need of a given instruction at a given position in the individual program. Fitness is measured in terms of runtime (with lower values being better), since a valid program must necessarily solve a problem (i.e, approximate a function). The authors do not initialize a complete n-ary tree of H levels; it has its size dynamically modified throughout the evolutionary process. The authors affirm, however, that pruning the GM is required in order to reduce memory consumption.

VI. Conclusions and Future Work

This work presented Ardennes, a novel Estimation of Distribution Algorithm for performing evolutionary induction of decision trees. Ardennes is capable of substantially outperforming LEGAL-Tree, a genetic algorithm that also employs a lexicographic strategy for solution ranking and selection.

After an experimental analysis with 10 public datasets, we show that Ardennes provides results slightly superior than C4.5, the most traditional greedy approach for decision-tree induction, while providing more comprehensible trees. However, we believe Ardennes has a big margin for further improvements that can place it as the top-performing algorithm for inducing decision trees. By addressing the issue presented in Section IV-A, we believe Ardennes will be capable of substantially outperforming C4.5 and similar approaches. For motivating further research on decision-tree induction with EDAs, we make the Ardennes source code available[3].

Note that Ardennes, similarly to other EAs, is well-suited for parallelization since several steps are independent from one another. In fact, we already employ parallelism for verifying the quality of thresholds when splitting predictive attributes. However, several other aspects may be parallelized, such as (1) sampling individuals, (2) sampling nodes within an individual, (3) initialization and updating the GM, just to mention a few possibilities.

Acknowledgments

The authors would like to thank CAPES, CNPq, FAPERGS, and FAPESP (2016/02870-0) for funding this research.

[3]Available at https://github.com/henryzord/ardennes

References

[1] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison Wesley, 2005.

[2] R. C. Barros, "On the automatic design of decision-tree induction algorithms," Ph.D. dissertation, Ponthifical Catholic University of Rio Grande do Sul, 2014.

[3] R. C. Barros, A. C. de Carvalho, and A. A. Freitas, *Automatic design of decision-tree induction algorithms*. Springer, 2015.

[4] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "Automatic Design of Decision-Tree Algorithms with Evolutionary Algorithms," *Evolutionary Computation*, vol. 21, no. 4, 2013.

[5] R. C. Barros, M. P. Basgalupp, A. A. Freitas, and A. C. P. L. F. de Carvalho, "Evolutionary Design of Decision-Tree Algorithms Tailored to Microarray Gene Expression Data Sets," *IEEE Transactions on Evolutionary Computation*, pp. 873 – 892, 2014.

[6] M. P. Basgalupp, R. C. Barros, T. S. da Silva, and A. C. de Carvalho, "Software effort prediction: a hyper-heuristic decision-tree based approach," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 1109–1116.

[7] M. P. Basgalupp, R. C. Barros, and D. D. Ruiz, "Predicting software maintenance effort through evolutionary-based decision trees," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 1209–1214.

[8] R. C. Barros, A. T. Winck, K. S. Machado, M. P. Basgalupp, A. C. de Carvalho, D. D. Ruiz, and O. N. de Souza, "Automatic design of decision-tree induction algorithms tailored to flexible-receptor docking data," *BMC bioinformatics*, vol. 13, no. 1, p. 310, 2012.

[9] M. P. Basgalupp, R. C. Barros, A. C. de Carvalho, A. A. Freitas, and D. D. Ruiz, "Legal-tree: a lexicographic multi-objective genetic algorithm for decision tree induction," in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 1085–1090.

[10] M. P. Basgalupp, A. C. de Carvalho, R. C. Barros, D. D. Ruiz, and A. A. Freitas, "Lexicographic multi-objective evolutionary induction of decision trees," *International Journal of Bio-Inspired Computation*, vol. 1, no. 1-2, pp. 105–117, 2009.

[11] R. C. Barros, M. P. Basgalupp, A. C. De Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 291–312, 2012.

[12] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[13] C. Robert, "Machine Learning: a Probabilistic Perspective," *CHANCE*, vol. 27, no. 2, pp. 62–63, 2014.

[14] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data (second edition)*. Chichester: Wiley, 2002.

[15] J. R. Quinlan, "C4. 5: Programming for machine learning," *Morgan Kauffmann*, p. 38, 1993.

[16] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[17] A. A. Freitas, "A critical review of multi-objective optimization in data mining: a position paper," *SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 77–86, 2004.

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

[19] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[20] S. Baluja and S. Davies, "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space," DTIC Document, Tech. Rep., 1997.

[21] J. S. De Bonet, C. L. Isbell Jr, and P. Viola, "Mimic: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, vol. 9. MIT Press, 1997, p. 424.

[22] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing*. Springer, 1999, pp. 521–535.

[23] R. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evolutionary Computation*, vol. 5, no. 2, pp. 123–141, 1997.