Rule Extraction from Random Forests via Covariance Matrix Eigen Decomposition

William Jardee willjardee@gmail.com
Department of Physics
Montana State University
Bozeman, MT 59715, USA

Lin Shi linshi1768@gmail.com
Giantforte School of Computing
Montana State University
Bozeman, MT 59715, USA

Editor: N/A

Abstract

Decision trees are a common approach to real-world data because their underlying structure is naturally interpretable. The ensemble method of decision trees, random forests, is often used to increase generalizability at the sacrifice of global interpretability. Some work has been done in recent years to represent forests by either extracting rules from individual trees (Bénard et al., 2021; Mashayekhi and Gras, 2015) or a tree-like structure to best represent the forest (Boruah et al., 2022; Vidal and Schiffer, 2020). These methods search for already built structures that best represent the forest. This paper presents a new method for extracting rules from a random forest similar to PCA. By taking an eigendecomposition of the covariance matrix of node literals in the forest, general patterns can be derived and explained in the form of logical rules. A new performance measure is defined to account for this new data representation. By testing the proposed algorithm on three data sets, the application is shown to be flawed, either from the derivation of the covariance matrix or from the deduction of rules from the eigenspace. During the development of this project and the performance analysis, emphasis is put on the fact that this is a learning project, meaning a perfected algorithm comes secondary to personal growth and understanding.

1. Introduction

In real-world applications of machine learning models, interpretability often rivals performance. For this reason, there has been much work in developing white-box models, such as decision trees, and the representation of black-box models with simpler, understandable models. This problem has proven more difficult as machine learning algorithms become more complicated and outgrow the scope easily understood by individuals who are not already experts in data science-related fields. Some methods attempt to visualize the data in as simple a space as possible, often presenting the first-order representation in one, two, or three dimensions, such as Principle Component Analysis (PCA). Others attempt to describe the logical basis of the space with hierarchical rule structures, such as decision trees, or rule sets, as Inductive Logic Programming (ILP) aims to do. This paper outlines an approach to extracting logical rules from random forests via eigenvalue decomposition of

a covariance matrix of the possible literals that make up the decision trees. At the same time, the theoretical basis of both the pattern extraction and performance metric seems sound, poor performance points to either flawed implementation or, more likely, a weak relationship between the pattern extraction rule interpretation.

Beyond the project's primary focus, we aim to understand the model representation in the form of logical rule sets. By taking an original approach, an understanding of the models, mathematical formulation, and practical application of libraries must be generated to pursue this goal or rationalize why it was not reached. The fundamental purpose of this paper is to motivate the personal development of understanding in the realm of random forests, logical rule sets, and general machine learning practices, as opposed to defending the use of a perfected algorithm. On this objective, this project served as a great aid in developing the authors' literacy and understanding. The remainder of this paper is structured such that related works are in section 2, the theoretical basis of the procedure outlined in section 3, experimental design and results in section 4, and discussion of results and future work in section 5.

2. Related Works

First, we review the decision tree, random forest, some of the oblique methods, and the idea of interpretability vs. performance trade-off among the algorithms. Then we discuss existing methods to extract trees/rules from a random forest. Next, we discuss some rule representation methods used in other black-box models, such as the neural network. Lastly, we wrap up the section with the covariance dimensionality reduction since it is the primary motivation behind our method.

2.1 Decision Trees and Random Forests

A decision tree is a typical data structure used in classification and regression that provides an interpretable[1] representation of the model. Logical literals of the data are represented as nodes and arranged in a hierarchical structure, such that satisfying the condition progresses in one direction and not the other. The literal at each node is the learned model and deduced by information gain, i.e., Gini importance or entropy. After generating the tree, traversing through a path can be interpreted as crafting a clause, where the premises are the nodes visited (and decisions made), and the conclusion is the final classification. Returning a rule for a specific classification can be considered local interpretability, where understanding is only provided for the situation observed compared to observing and understanding the general structure of the whole model can be considered global interpretability. Arguably, decision trees have both.

The random forest, introduced by Breiman (2001), is a classifier that consists of a collection of decision trees. Simple random forest algorithms build a predetermined number of trees, each with a different training set sampled from the dataspace with replacement. Using a different data set for each tree introduces model diversity. Other algorithms have been designed to train each tree on a separate subset of features, or a weighted linear

1. We do not wish to get caught in the semantics of interpretable vs. explainable. For this reason, we use the two terms interchangeably and only refer to interpretability.

combination of features, called oblique random forests (Breiman, 2001). When interpreting the random forest, the rules are not as straightforward. Since there will be a set of rules for every tree, interpreting the rules becomes problematic when the forest becomes populated. The random forest is considered a grey-box model, where the fundamental model is still interpretable but requires much work. This trade-off is familiar with machine learning algorithms, where uninterpretable models sometimes have better performance.

Other proposed extensions of random forest, such as Forest-RC and random rotation Random Forest will be even more challenging to interpret. Forest-RC constructs d univariate projections, where each projection is a linear combination of L randomly chosen dimensions (Breiman, 2001). Since the splits are a linear combination of L dimensions, creating a set of rules to interpret the Forest-RC is even more complicated. In random rotation Random Forest (RR-RF), the data is randomly rotated before inducing the trees (Blaser and Fryzlewicz, 2016). Then RR-RF splits on the feature axis; however, in the original data space, the split is a linear combination of the features. Once again, rule extraction for these abstract models is challenging.

2.2 Tree and Rule Extraction from Random Forest

Due to random forest's difficult interpretability, algorithms have been proposed to extract a subset or single tree that could best represent the forest. Boruah et al. (2022) proposed a method that takes the random forest and ranks the tree based on its accuracy performance, selecting the tree with the highest performance and extracting rules from it. Stable and Interpretable RUle Set (SIRUS) aims to address regression problems by extracting all rules from the forest and pruning the rule set via performance metrics (Bénard et al., 2021). Born-again tree ensembles implement a dynamic programming approach to extract a single tree from the forest (Vidal and Schiffer, 2020). Mashayekhi and Gras used the random forest clustering property to create a hill climb algorithm that extracts the best set of rules (Mashayekhi and Gras, 2015). To our best knowledge, all the algorithms either pool the decision characteristics from the whole forest and prune to get a simplified representation or build a decision tree to represent the forest as a whole. There does not seem to be any work in averaging over the rules directly to extract a whole new rule set not evident in any specific to any tree.

2.3 Covariance Dimensionality Reduction

Searching data for underlying patterns defines the field of unsupervised learning. Our approach to extracting rules from a random forest's covariance matrix can be considered an unsupervised learning problem. The most relevant example is Principal Component Analysis (PCA). PCA works by extracting the k most prominent eigenvectors from a problem's feature space and represents them as clusters in the larger data space (Schölkopf et al., 1997). Eigenvalue decomposition, sometimes referred to as Singular Value Decomposition when a full eigenspace is not present, is a general approach to representing spaces in a new basis that highlights the hierarchical importance of each basis vector (Strang, 2006; Griffiths and Schroeter, 2018). The general concepts in PCA are relevant to the motivation of our algorithm but not directly applied to it.

3. Algorithm

Since this algorithm is still in development, we will present the first two iterations developed. In section 4 we will test the performance of the algorithms, and then in sec 5 these results will be interpreted and future work proposed.[2]

3.1 Version 1

The first implemented version of this algorithm failed to account for logical inconsistencies when substituting to create an argument about the horn clauses. It is unclear if this is the primary theoretical flaw in the algorithm, but it was a key motivator in developing 3.2.

3.1.1 Covariance Matrix for Forest Clauses

Each path in a decision tree can be extracted as a logical rule. Consider a path in a binary decision tree that follows

$$\neg A \wedge \neg B \wedge C \rightarrow W_0\,,$$

using material implication and De Morgan's law, this rule can be written as

$$\neg(\neg A) \vee \neg(\neg B) \vee \neg(C) \vee W_0\,. \tag{1}$$

Notice that we have decided not to cancel out the negations; this will be useful for later as they will cancel out. Each path of each tree in the random forest can be extracted as one of these rules. We propose that if the problem can be explained with high order logical rules, these rules may be embedded in these tree paths and, consequently, in these extracted rules. If two literals often show up together, we propose that they probably come from the same logical rule.

For simplicity, all of the features are assumed to be binary. Let us construct a $(2n + c) \times (2n + c)$ matrix of all zeros, where n is the number of features, and c is the number of classes. Each of the rows and columns will correspond to a possible logical decision. For the example in equation 1 the first $2n$ rows would correspond to $[\neg(A), \neg(B), \neg(C), A, B, C]$ and the last c would correspond to the possible classes. Each time a pair of literals show up together in a rule, 1 will be added to the corresponding cell. This concept describes a covariance matrix.

To illustrate this, see the rule matrix, Δ, for the running example:

$$\Delta = \begin{array}{c} \\ A \\ B \\ C \\ \bar{A} \\ \bar{B} \\ \bar{C} \\ W_0 \\ W_1 \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & \bar{A} & \bar{B} & \bar{C} & W_0 & W_1 \end{array} \\ \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \end{array},$$

2. This work is adapted from a larger piece that consisted of a fifteen-page report and hour presentation. Consequently, the length provided will not be able to go into complete detail about the nuances.

where $\bar{A} \equiv \neg A$ and zeros have been replaced with $\cdot$ for readability. Adding up the Δ for each path in the forest gives a complete covariance matrix. Similar to in PCA, dimensionality reduction can be done by using eigendecomposition.

3.1.2 Rule Set Extraction from Covariance Matrix

Given the eigendecomposition of the matrix, represented in the form of the set of eigenvalues, λ, and its corresponding eigenvector, $\vec{v}$, a reduced representation of the covariance matrix can be extracted. Eigenvectors with larger λ have more importance to the matrix, as the linear transformation favors that direction. Consequently, the k most important vectors can be chosen by picking the eigenvectors that correspond to the k largest eigenvalues. To pick out only the essential characteristics of the features and the classes independently the first $2n$ values of the eigenvector and the last c should be pruned individually, decreasing the input noise from the features and picking out the essential classes of the rule this vector should describe.

Because the distribution of the component values is not known, sophisticated models should not be used here; i.e., a one standard deviation cut. For this paper we take the 4th quartile of the features as that is distribution independent. Future work should look into how to include the weights into rule expression, but we choose to set all rule values to 1 for simplicity. The feature portion of the eigenvector can be explained by the function

$$f_f(x_i) = \begin{cases} 1 & \text{if } x_i > 75\% \text{ of vector} \\ 0 & \text{Otherwise} \end{cases} . \tag{2}$$

For the classes portion of the eigenvector, values are kept if their contribution is greater than random. For a normalized vector with equal weighting on all components, each has a value of $1/\sqrt{c}$. For all the values larger than random the value is kept, otherwise it is set to zero. This is explained with the function

$$f_c(x_i) = \begin{cases} 1 & \text{if } x_i > 1/\sqrt{c} \\ 0 & \text{Otherwise} \end{cases} . \tag{3}$$

The resulting vector is normalized, and each component is squared to provide a probability measure for each class. This quantitative value is not used in the project's current state, but it could be either reported with the rules to educate how likely each class outcome is or used in a scoring metric.

There is no guarantee that the resulting rules fully cover the class space, which means that the whole problem cannot be explained by the extracted rules so far. Therefore, the remaining eigenvectors should be searched for rules that cover missing classes to account for this. This is done with a greedy approach that adds the eigenvector with the largest eigenvalue to cover the missing classes. The preferred number of times a class is covered is unclear, as only one rule that describes the class may be insufficient. So, the number of times each class must show up, k^*, should be tuned. If k^* is zero, there is no enforcement that each class is covered.

The extraction of logical rules from the rule vectors will mirror how they were encoded. That is by a conjunction of OR statements, with the features gaining a negation. The

resulting form can be considered a horn statement, where the only positive literal is the union of classes. The eigenvalue information and ratio of classes can be extracted as well.

3.2 Version 2

The feature space was separated along class slits to address one of the flaws in the previous approach. A similar flow happens, but each class has a covariance matrix. Each rule in the training set that aligns with the specific class is added to that class's matrix. Additionally, the matrix size was reduced from accounting for positive and negative examples individually to addressing positive examples by adding $+1$ to the matrix element and -1 for negative examples. This approach does remove some information from the matrix but could potentially give more insightful eigenvectors. Similar eigendecomposition can be done on these covariance matrices, following similar pruning procedures as in 3.1; however, the only applicable pruning will be the feature space modifications since each class will have unique eigenvectors and consequently be interpreted as the premises from the eigenvector implying the appropriate class. It is proposed that this rule structure, having only one possible value for each feature and one resulting class, creates a much more coherent ruleset.

4. Exeperimentation

For brevity, not all of the experimental results are displayed here. In their stead, an interpretation of the overall results is presented. Further, since the second version of the algorithm is still in development, reliable data has not been taken yet. Nevertheless, from some of the issues seen in the performance of Version 1, some of the ideas in Version 2 should be better motivated.

4.1 Implementation

To implement our proposed method, we used the SciKit-Learn library[3] for simplicity. We use the built-in OneHotEncoder package and the RandomForestClassfier package to encode the data and build the forests. The built-in algorithm uses the Gini impurity to optimize the forest. Lastly, we test different forest sizes but leave all other hyperparameters at their default settings.

4.2 Version 1: Experimental Design

Considering this project's scope, only three data sets were evaluated to see how well the algorithm performed under various challenges. All three of the data sets were collected from the University of California Irvine, Center for Machine Learning and Intelligent Systems' Machine Learning Repository[4]. The data sets consisted of categorical and ordinal features, but every feature was treated as categorical and dealt with via one-hot encoding to satisfy earlier constraints. Future work should expand the algorithm to handle both ordinal and continuous features, which will be brought up in the discussions section.

3. https://scikit-learn.org/stable/index.html
4. https://archive.ics.uci.edu/ml/index.php

The first data set used was the Tic-tac-toe data set. It had 958 instances, nine features with three values each, and two resulting classes. The set looked at the ending result of tic-tac-toe games, with either 'x' or 'o' winning. This dataset was chosen because of the smaller feature space and the underlying, rigorous rule set that defined tic-tac-toe. The second data set was the Car Evaluation data set, which ranks car quality by various metrics. This data set had 1728 data instances and six features with three to five ordinal data points. There were four possible output classes, which were very unbalanced (there were 70% in the most common class and 4% in the least common class). This data set supposedly has an understood, complex logic structure behind it. The final data set used was the Beast Cancer Wisconsin data set. This data set looked at 699 instances of cancerous tumors with nine ordinal features, each with ten possible quantities, and classified them as either benign or malignant. This set was chosen to see how far the algorithm performed in extreme conditions.

The algorithm was trained on various ruleset sizes, required coverage of classes and not, and various forest sizes (from 20 to 5000) to test the performance. There was some exploration into defining a measurement metric for the performance, but that has been discarded from this analysis because of an unfinished conversation about the theoretical premise. This metric added little to the conversation, and the overall points brought up are neither helped nor hindered by its omission.

4.3 Version 1: Qualitative Performance

The performance of the algorithm can be summarized by saying:

1. class coverage becomes more uniform on accurate forests,
2. with few classes, there was no difference in requiring class coverage when more rules than the number of classes were used,
3. rules sets on the order of 10 rules were the most understandable,
4. in general, the rules are abstract and difficult to understand without domain knowledge.

Further, the accuracy of the rules is difficult to deduce because of their un-interpretability. Considering the primary motivation of this project, that is a major oversight. Looking at the following example of a rule extracted from the Car Evaluation data set:

$$\text{“}\neg(\text{buying=low})\wedge\neg(\text{maint=vhigh})\wedge(\text{doors=5 more})\wedge\neg(\text{doors=2})\wedge(\text{persons=2})\wedge(\text{lug\_boot=med})\wedge\neg(\text{lug\_boot=big})\wedge(\text{safety=low})\longrightarrow\text{unacc or vgood .”}$$

It can be seen that the primary difficulties in understanding the rules are in the complex nature of the rules and their length. Further analysis needs to be done into how including factual, counterfactual, and multiple class literals hinder interpretability and how these concepts scale to larger sets.

For the Breast Cancer Wisconsin data set, the complexity of the underlying rules and size of the feature space meant that the majority of outputted rule sets consisted of verbose

sets of liberals that pointed toward an empty class set and the rule "$\emptyset \rightarrow$ benign or malignant." These ideas and the possible impact of the unbalanced data sets should be naturally addressed by Version 2.

5. Discussion

We discuss our contributions and possible future work. Furthermore, we talk about rules as an interpretation aid.

5.1 Contributions and Future Work

As can be seen from the experimental results, the general pattern of the algorithm was not optimistic. Nevertheless, the algorithm developed was able to pull out some general patterns, especially when patterns were searched for from multiple runs. Furthermore, there was an evidential difference in the rules deduced from better forests, meaning there is a relationship between the quality of the forest structure and the quality of the rules deduced. The algorithm would likely be improved if a couple of crucial facets were addressed:

1. better statistical analysis of how to construct the covariance matrix and how to weight rules, rule elements, and classes according to eigendecomposition characteristics,
2. expansion of the algorithm to deal with intervals from ordinal and real-valued features,
3. more sophisticated analysis of the quantitative measures of rule set performance, and
4. expansion of the algorithm to consider relationships between rules from the same tree before adding them to the covariance matrix.

Since this algorithm did not successfully tackle any of these points, it serves as the starting point for a larger project that could build off the concept outlined here.

5.2 Rules as an Interpretation Aid

Many issues stemmed from not correctly understanding how to use logical rules as an interpretation aid. As discussed in the related works, preexisting methods use rules to understand structures like neural networks. In development, most of the focus was on forest-specific literature and other methods that address representing either trees or forests. Guidance should be taken from those related works to address better how to present the ruleset to improve this method. For example, in theory, giving the counterfactuals of ordinal data points seemed logical, but in testing, it added little understanding to the logical rule. The fundamental algorithm might perform better if the rules were displayed with a more rigorously backed measure function.

5.3 Conclusion

This paper presented a new approach to compiling information over a random forest and attempted to extract rule sets directly from the derived covariance matrix. These rules were implied from eigenvalues of the covariance matrix and would hypothetically result in rules characterized by positive literals, negative literals, and an implication of one or more classes.

In empirical testing, the added understandability from the logical rule sets seemed flawed. It motivated a new approach to representing these rule sets and extracting understandable patterns from the forests. It seemed that either insufficient rigor was spent deriving the rules, or the proposed metric was flawed. This work is a good motivator for future work based on the same theory, such as some of the beginning work presented in 3.2. The next steps include a complete survey of the current state of rule representation of ILP models and representative models of neural networks.

References

Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. Interpretable random forests via rule extraction. In *International Conference on Artificial Intelligence and Statistics*, pages 937–945. PMLR, 2021.

Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. *The Journal of Machine Learning Research*, 17(1):126–151, 2016.

Arpita Nath Boruah, Saroj Kumar Biswas, and Sivaji Bandyopadhyay. Transparent rule generator random forest (trg-rf): an interpretable random forest. *Evolving Systems*, pages 1–15, 2022.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018.

Morteza Mashayekhi and Robin Gras. Rule extraction from random forest: the rf+ hc methods. In *Canadian Conference on Artificial Intelligence*, pages 223–237. Springer, 2015.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

Gilbert Strang. *Linear algebra and its applications*. Acad. Press, 4th edition, 2006.

Thibaut Vidal and Maximilian Schiffer. Born-again tree ensembles. In *International Conference on Machine Learning*, pages 9743–9753. PMLR, 2020.