Explainable Random Forests

James Garner, Hamish Huggard, Liam Rigby

July 2018

I. Introduction

Random Forests are widely regarded as a black-box classifier as, by default, they provide no insight into the reasons for the classifications or predictions that they make. This makes them incredibly difficult to debug, as well making it nearly impossible to know when the Random Forest can be trusted (as they usually are not perfectly accurate). This has resulted in less accurate but more explainable machine learning models being favoured in high risk domains. At a high level, our work seeks to open up this black-box classifier by generating explanations about the classifications from the Random Forest by using a decision tree; a machine learning model that has strong explainability. For each instance in the Random Forest, a decision tree is trained in such a way that it remains faithful to the classification of the original model, while being transparent in it's decisions.

II. LIME and LEMON

Our approach to explainable random forests is inspired by the Locally Interpretable Model-Agnostic Explanations (LIME) system by Ribeiro et al. [1]. LIME is a technique for finding an interpretable model which approximates the behaviour of an unexplainable black-box. By examining the decision procedure of the interpretable model, we can glean insights into the factors which contribute to the black-box's decisions. This can help us to decide whether the black box is trustworthy, and to debug if necessary. A global, interpretable explanation of the black-box is presumed intractable, so instead a local explanation is given in the neighbourhood of an instance whose classification we would like to explain. Because LIME assumes the model it tries to explain is a black-box, it is model agnostic and can be readily applied to a wide range of uninterpretable systems.

In particular, LIME selects an interpretable model $\xi(x)$ out of a class of interpretable models G, which emulates the local behaviour of a black-box model $f(x)$ around instance x. The neighbourhood of x is defined by a proximity function $\pi_x(z)$. The local fidelity of an interpretable model g to a black-box model f around the neighbourhood of x is specified by a loss function $\mathcal{L}(f, g, \pi_x)$. The interpretable model $\xi(x)$ is chosen such that

$$\xi(x) = argmin_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{1}$$

where Ω is a measure of the complexity of the model, such as the depth of a decision tree or the number of non-zero weights in a linear classifier. LIME is an attempt to express the intuition that within the domain of explainable models we must navigate a trade-off between expressivity and interpretability. If the interpretable model is simple, then it will not provide a good explanation of the black-box model. If the interpretable model is too complex, then it will not be fit for purpose.

However, we believe that equation 1 does not express this trade-off well. It is hard to imagine a situation in which the local fidelity of an interpretable model can be additively substituted for complexity. An interpretable model is useless if it is highly accurate and yet too complex to be understood, or trivial and yet highly inaccurate. With poor choice of functions $\mathcal{L}$ and Ω, either of these could be chosen by LIME. It seems much more natural to suppose that there is a maximum "complexity budget" beyond which a model becomes uninterpretable, and that the best interpretable model is that which achieves maximum local fidelity while remaining below the complexity budget. To express this idea, we propose LIME Except (with) More Organic Notation (LEMON):

$$\xi(x) = argmin_{g \in \{\gamma \in G : \Omega(\gamma) \leq \Omega_{max}\}} \mathcal{L}(f, g, \pi_x) \tag{2}$$

Where Ω_{max} is the maximum "complexity budget". As an illustration, suppose we have a class of models which requires keeping some arbitrary number of items in working memory to understand. If a model requires keeping more than seven items in working memory, then most humans will not be able to follow it [2]. Therefore we should let $\Omega(\gamma)$ be the number of items in working memory required to understand model γ, and take $\Omega_{max} = 7$.

Ribiero et al. seemed in practice to share our sensibilities about having a complexity budget, as when they actually implement LIME, they tend to choose Ω such that LIME is forced to act like LEMON. For instance, in a linear, bag-of-words text classification example, they use $\Omega(g) = \infty 1[\|w_g\|_0 > K]$, where w_g is the weight vector of the model. In other words, $\Omega(g) = 0$ if the model uses no more than K non-zero weights, and ∞ otherwise. It is much more natural to express the interpretable model selection criteria in LEMON using $\Omega(g) = \|w_g\|_0$ and $\Omega_{max} = K$.

III. Proposed Solution

A Random Forest is constructed using a collection of decision trees. However, in contrast to decision trees, Random Forests boast higher accuracy at the expense of explainability. We believe that part of this trade off can be eliminated by generating decision trees to provide explainability for a given instance. Our work entails the following:

1) Train a Random Forest on a random sample of half of the data set (this is the training data).
2) Randomly choose n instances to explain from the remaining test data set.
3) Perturb the variables of a given instance to generate a new training set composed of all instances within Hamming distance 1 of the given instance (or more than 1 provided the decision tree remains comprehensible). This is the neighbourhood of that instance. Labels for this data set are generated by classifying all instances with the Random Forest.
4) Train a decision tree based on the above training set (note that errors in the classification of the Random Forest flow through to the decision tree).
5) Use the decision tree to provide an explanation for the classification of the given instance.
6) Assess the explanation. This may involve manually verifying the explanation for accuracy. Inaccurate explanations may indicate inaccurate classifications and hence we have provided a means of understanding when a Random Forest can be trusted.

A. Algorithm

Our algorithm is detailed in Algorithm 1. The inputs are an instance to be explained x, a random forest to be explained RF and a maximum explainability complexity Ω_{max}. The neighbourhood of x within which the explanation will occur is specified by the distance d (as measured by metric $\pi_x(z)$), which is initialized to 1. For each value of d which is tried, the set of points within that neighbourhood and their classification by RF are used as training data to generate a classification tree via the $TREE$ function. The classification tree is the interpretable local model which minimizes $\mathcal{L}$. d is increased until the complexity of the tree exceeds Ω_{max}. If the tree generated by $d = 1$ already has complexity exceeding Ω_{max} then this will be returned (rather than $\emptyset$).

$\pi_x(z)$ is a generalized Hamming distance from x. That is, it gives the number of features along which x and z vary. We will only test domains with unordered, categorical features, but this metric could be extended to continuous features such as by equating a difference of one standard deviation with one categorical difference. Thus the "neighbourhood" of the instance to be explained is found by perturbing the features of x along up to d dimensions at a time.

B. Implementation

We will implement the above algorithm in R, using the package `randomForest` to generate random forests and `rpart` to generate decision trees. We will use the default settings for these functions.

C. Explanation extraction

Given the tree that is our locally faithful model of the RF, we can extract the following explanations for the RF's classification.

Algorithm 1 Random Forest Explanation via LEMON

```
Require: $x$, $RF$, $\Omega_{max}$
  $d = 1$
  $t_i \leftarrow \emptyset$
  $t_{i+1} \leftarrow \emptyset$
  while $\Omega(t_{i+1}) < \Omega_{max}$ or $d < 2$ do
    $t_i \leftarrow t_{i+1}$
    $d \leftarrow d + 1$
    $N = \emptyset$
    for $z \in X$ with $\pi_x(z) \leq d$ do
      $N \leftarrow N \cup \{(z, RF(z))\}$
    end for
    $t_{i+1} \leftarrow TREE(N)$
  end while
  return $t$
```

1) Local variable importance: The process of generating a decision tree involves deciding which features to split the data set on, using measurements of information gain. Thus the order of the splits gives a measure of local variable importance. Of course, this order may be different down different branches of the tree. That is, after the first split on some binary feature, the subsequent split on the 0 branch may be on a different feature to the subsequent split on the 1 branch. We can use this to compute both a local variable importance ranking by weighting features based on their appearance across all branches, and to get a feature ranking specific to the instance to be explained, by taking the features on the decision path corresponding to that instance.

2) Contrastive counterfactuals: The tree data structure also gives us the conditions necessary for alternative classification. This data can be presented in such a way as to answer two key questions:

1) "Why was the classification A rather than B?"
2) "If feature x had been different, how would this have changed the classification?"

To answer question 1, the different combinations of features required for the alternative classifications possible can be presented, while to answer question 2, the outcome of changing each feature in isolation can be presented. Clearly there is overlap between the information required to answer questions 1 and 2. Depending on the domain and instance, there might be too much information to present to the user. Instead of presenting the raw information, simple summaries can be generated, such as the smallest number of feature changes required to change the classification.

IV. Data Set

For this project we are proposing to use data sets that contain characteristics similar to that of the Connect-4 data set from the UCI Machine Learning Repository. The Connect-4 data set contains all legal board positions in the game of Connect-4 in which each player has placed 4 counters. In all board positions, neither player has won yet, and the next move is not forced. The outcome class is the game theoretical value for the first player. This data set has the following desirable characteristics:

- The proposed solution is most suited to classification tasks (for which this data set is appropriate).
- The proposed solution uses Hamming distances to perturb variables and generate decision trees to provide local explanations. Using Hamming distances requires special care when handling variables with continuous values (e.g. binning), which may lead to less consistent results. The Connect-4 data set contains only categorical data.
- The data set is large, and therefore we would expect noise or anomalies to have a limited impact.
- The explanations produced by our solution can be trivially verified. We know how to play Connect 4, and can therefore better interpret the explanations compared to a data set relating to a more specialized field (e.g. proteins, cancer, etc.).
- A Random Forest trained on this data set is not already perfectly (or almost perfectly) accurate. In a recent paper by Klambauer et. al.[3], the accuracy of the Random Forest was approximately 88% for this data set. Therefore, this provides opportunities to detect incorrect explanations as a way of debugging the black box classifier.

V. Evaluation

We have proposed that our method will generate locally accurate explanations for the classification of particular instances by a RF. Assessing the accuracy of explanations is difficult, and often relies on having a human domain expert in the loop. In lieu of this, we propose the following two tests.

Firstly, we can assess the extent to which the explainable model (the tree) actually models the RF. For our explanations to be good, it must at least be a locally faithful model. However, if the faithfulness of the model dropped to zero on instances outside the generated neighbourhood of the instance to be explained, this would not lead to confidence that the explanations actually reflect the RF decision criteria. Faithfulness at a given distance from the instance to be explained can be measured by generating instances at that distance (either exhaustively or a random sample), classifying them using both the RF and the tree, and computing the percent agreement. Repeating this process will allow us to plot the fidelity decrease over increasing distance.

Secondly, we can assess whether the explainable model actually detects features that are important to the RF, by adding a 'pathological' feature to the labelled training and test data. This feature will have perfect correlation with the true classification of the training data, but no correlation with the true classification of the test data. Our explainable model should detect this pathological feature as important to the RF's classification of instances from the test data, even though it has no correlation to the true classification.

Generating a neighbourhood in instance space within some distance of an instance to be explained may generate instances that do not make sense for the domain - illegal instances. The RF was trained on a training set of legal instances, and is intended to provide accurate classifications for legal instances. There is no expectation that the RF would be able to give sensible classifications of illegal instances. On the other hand, determining which instances are illegal requires domain specific information, and may be difficult to determine for some domains. Rational classification of illegal instances is also the type of task we might expect a human expert to be able to accomplish. It is difficult to say whether classifying these instances using the RF will help or hinder our understanding of the RF. Therefore, when assessing the fidelity of the explainable model, as described above, we will compare the fidelity for trees trained on the exhaustive neighbourhood and trees trained on the legal neighbourhood.

VI. Conclusion

The popularity of Random Forest classifiers combined with their "black-box" status motivated us to explore techniques for making Random Forests explainable. We drew inspiration from the LIME system, a model-agnostic approach to explainability which generates an interpretable model within the neighbourhood of some instance to be explained. Our modified system, LEMON, differs in that it regards there as being a fixed "complexity budget" which the interpretable model is not allowed to exceed, and our sampling strategy. We propose to apply this to Random Forests by generating a decision tree which can replicate the behaviour of the Forest within the neighbourhood of the instance we wish to explain. We will implement this in R and evaluate it firstly by exploring how closely the decision tree models the Random Forest, and secondly by whether it is able to accurately detect known features of importance to the Random Forest.

References

[1] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[2] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[3] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.