Explaining Predictions from Tree-based Boosting Ensembles

Lucic, A.; Haned, H.; de Rijke, M.

Publication date
2019
Document Version
Author accepted manuscript
Published in
Proceedings of FACTS-IR 2019

Explaining Predictions from Tree-based Boosting Ensembles

Ana Lucic
University of Amsterdam
Amsterdam, The Netherlands
a.lucic@uva.nl

Hinda Haned
Ahold Delhaize
Zaandam, The Netherlands
hinda.haned@aholddelhaize.com

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Understanding how "black-box" models arrive at their predictions has sparked significant interest from both within and outside the AI community. Our work focuses on doing this by generating local explanations about individual predictions for tree-based ensembles, specifically Gradient Boosting Decision Trees (GBDTs). Given a correctly predicted instance in the training set, we wish to generate a counterfactual explanation for this instance, that is, the minimal perturbation of this instance such that the prediction flips to the opposite class. Most existing methods for counterfactual explanations are (1) model-agnostic, so they do not take into account the structure of the original model, and/or (2) involve building a surrogate model on top of the original model, which is not guaranteed to represent the original model accurately. There exists a method specifically for random forests; we wish to extend this method for GBDTs. This involves accounting for (1) the sequential dependency between trees and (2) training on the negative gradients instead of the original labels.

ACM Reference format:
Ana Lucic, Hinda Haned, and Maarten de Rijke. 2019. Explaining Predictions from Tree-based Boosting Ensembles. In *Proceedings of SIGIR '19: FACTS-IR Workshop, Paris, France, July 25, 2019 (SIGIR 2019),* 4 pages.
DOI: 10.475/123_4

1 INTRODUCTION

As information retrieval (IR) systems become more and more prevalent, it becomes increasingly important to understand how an IR system produces a particular prediction and what exactly drives it to do so. Understanding how "black-box" models arrive at their predictions has sparked significant interest from both within and outside the IR community. This can be in the context of rankings [11], recommendations [12], or digital assistants that engage in interactive question answering [8].

Explanations of an IR system can be provided for the system as a whole or for individual decisions produced by the system. Explanations based on interpreting the model in all regions of the input space are called *global explanations*, while those based on interpreting individual predictions are called *local explanations* [5]. The explainability problem is often cast in terms of supervised prediction models: IR systems usually involve a prediction at some point in the pipeline (i.e., predicting whether a document is relevant or not).

Given how often we use complex models to help us make difficult decisions, it is important to be able to understand what happens during the training phase of the model. We propose doing this by generating local explanations about individual predictions. Recent work on local explanations is usually conducted in either a model-agnostic or model-specific way [5]. Model-agnostic explanations typically involve approximating the original "black-box" model locally in the neighborhood of the instance in question [10], while model-specific explanations use the inner workings of the original "black-box" to explain the prediction of the given instance [13]. The obvious advantage of model-agnostic explanations is that they can be applied to any type of model [9], but since the explanation is based on a local approximation of the original model, there exists some inherent degree of error between the original model and the local approximation. Indeed, since the local model is an approximation, there is no guarantee that it is appropriately representative of the original model, especially in other parts of the input space [2]. In our work, we focus on generating model-specific explanations for boosting ensembles since they are widely used in industry and have demonstrated superior performance in a wide range of tasks.

This gives rise to our leading research question:

> *How can we automatically generate actionable explanations for individual predictions of tree-based boosting ensembles?*

2 AGENDA

In order to address our leading research question, we propose a three-part research agenda for using explanations to understand individual predictions.

(1) Generate explanations for individual "black-box" predictions in terms of (i) why a particular prediction was classified as a certain class, (ii) what it would have taken for the prediction to be classified as the alternative class, and (iii) how to perturb the model in order to change the prediction.
(2) Develop a mechanism that allows the user to change the prediction based on the explanation.
(3) Evaluate the effectiveness of such explanations on users' confidence in and trust of the original "black-box" model. This also involves determining appropriate baselines and metrics, and a sensible experimental environment in terms of the people involved and the questions asked.

In this work, we outline ideas along with a case study about items (1) and (2) above. The work of Tolomei et al. [13] has the potential to solve this problem but we argue that it (i) does not apply to boosting ensemble methods, and (ii) has scalability issues. In order

SIGIR 2019, Paris, France

DOI: 10.475/123_4

to come up with a satisfactory solution to our problem, we take the method from [13], explain it and articulate how it could be extended to accommodate tree-based boosting ensembles. In this extended abstract, we focus on adaptive boosting [6] first to disentangle the sequential training nature of boosting methods.

3 A CASE STUDY IN EXPLAINING INDIVIDUAL PREDICTIONS – WORK IN PROGRESS

We focus on explaining predictions from tree-based boosting ensemble methods (or simply boosting methods). Boosting methods are based on sequentially training (weak) models that, in each iteration, focus more on correcting the mistakes of the previous model. We train a boosting ensemble $\hat{f}$ using an input set X to predict a target variable $y \in \{-1, 1\}$, where $\{T_1, \ldots, T_K\}$ are the set of base classifiers for the ensemble and $\left\{\hat{h}_1, \ldots, \hat{h}_K\right\}$ are the corresponding predictions for each base classifier.

In adaptive boosting [6], each iteration $\hat{h}_k$ improves over $\hat{h}_{k-1}$ by upweighting misclassified instances (and downweighting correctly classified instances) by a factor of e^{α_k}, where α_k is the weight assigned to $\hat{h}_k$ in the ensemble and is defined as

$$\alpha_k = \log \frac{1 - err_k}{err_k} + \log(K - 1) \tag{1}$$

and err_k is the classification error of the k-th base classifier h_k.

3.1 Problem definition

Tolomei et al. [13] investigate the interpretability of random forests (RFs) by determining what drives a model to produce a certain output for a given instance in a binary classification task. They frame the problem in terms of actionable recommendations for transforming negatively labeled instances into positively labeled ones in a binary classification task. Our objective is to extend this method to work for boosting methods and later use these explanations to transform misclassified instances into correctly classified ones. This involves accounting for some components of boosting that do not apply to RFs: (i) the sequential dependency between trees, and (ii) training on the negative gradients instead of the original labels (in the case of gradient boosting decision trees (GBDTs)). We break the task up into two stages:

(1) We first extend [13] to work for adaptive boosting [6], which still trains on the original labels. This allows us to focus specifically on training trees in sequence and use this to narrow our search space.
(2) We extend our new method for adaptive boosting to gradient boosting [4], where we not only train in sequence but also train on the negative gradients of the previous tree.

This leads to the following research questions:

- **RQ1:** Given an instance, how can we perturb the instance such that the prediction for this instance flips from one class to another?
- **RQ2:** Given an instance, how can we perturb the model such that the prediction for this instance flips from one class to another?

3.2 Related work

The method in Tolomei et al. [13] is defined as follows: let x be an observation in the set X such that x is a true negative instance (i.e., $\hat{f}(x) = f(x) = -1$, where $\hat{f}$ is the overall prediction of the ensemble and f is the true label). The objective is to create a new instance, x', that is an ϵ-transformation from an existing positively predicted instance, x^+.

(1) The trees in the ensemble $\mathcal{T} = \{T_1, \ldots, T_K\}$ (an RF in this case) can be partitioned into two sets depending on whether the prediction resulting from each tree T_k is either positive or negative (the base classifier $\hat{h}_k$ corresponding to tree T_k is either +1 or −1). We are interested in the set of trees that result in negative predictions since we want to determine the criteria for turning these into positive predictions.
(2) Therefore, for every positive path $p^+_{k,j}$ (i.e., paths that result in a positive prediction, indexed by j) in every negative tree T_k (i.e., $\hat{h}_k(x) = -1$), we want to generate an instance $x^+_{j(\epsilon)}$ that satisfies this positive path (i.e., $\hat{h}_k(x^+_j) = +1$), based on our original instance x.
(3) We create $x^+_{j(\epsilon)}$ by examining the feature values of x and the corresponding splitting thresholds in $p^+_{k,j}$. For each feature in $p^+_{k,j}$, if x satisfies the splitting threshold for $p^+_{k,j}$, then we leave the value for this feature alone. If not, then we tweak the value for this feature such that it is ϵ-away from the splitting threshold and satisfies $p^+_{k,j}$.
(4) We construct an $x^+_{j(\epsilon)}$ based on x from every positive path j in every negative tree k and evaluate the output of the entire ensemble using this $x^+_{j(\epsilon)}$. If $\hat{f}(x_{j(\epsilon)})^+ = +1$ then $x^+_{j(\epsilon)}$ is a candidate transformation of x.
(5) We greedily choose the candidate transformation that is closest to the original instance and this is returned as the minimal perturbation of the original instance such that the prediction flips from negative to positive. We call this x'.
(6) Since this ϵ-perturbation allows us to discriminate between the two classes so it can be viewed as the contrastive explanation [7] for why $\hat{f}(x) = -1$ as opposed to +1.

This work relies heavily on being able to enumerate the positive paths $p^+_{k,j}$ in each negative tree T_k, which is not possible when training on the negative gradients instead of the original labels. This is also very computationally intensive since we compute an ϵ-transformation for an x^+_j in each $p^+_{k,j}$. In our work, we want to use the sequential training nature of boosting methods to narrow the search space as early as possible.

3.3 Method outline

Given an instance x, we are interested in reducing the search space for x^+_j in order to make the method by Tolomei et al. [13] more scalable. To this end, we look for a subset of the original ensemble, $\mathcal{T} \subseteq \{T_1, \ldots, T_K\}$, such that the rest of the ensemble can safely be ignored. That is, we want to select the most important trees in the overall model without omitting trees that were particularly influential for this prediction.

We pursue two directions to determine whether such a $\mathcal{T}$ might be found. The first idea is to consider how much each tree contributes to the prediction by examining their corresponding weights $\{\alpha_1, \ldots, \alpha_K\}$. We want to determine whether or not they decrease for each iteration k in the training of the ensemble, and if so, how quickly this happens. The hypothesis is that if the weights drop quickly and to small quantities, then we can narrow the search space by only examining the trees in the beginning of the ensemble. We choose two binary classification datasets: *Adult* [1] and *home equity line of credit* (HELOC) [3] and train an adaptive boosting model with 100 iterations, each with maximum depth 4 on the two datasets. Figure 1 shows the weights $\{\alpha_1, \ldots, \alpha_K\}$ for each iteration in the model. Indeed, we see that the trees at the beginning of the ensemble seem to be more important to the overall prediction, as they have higher weights, than the trees towards the end. Therefore, if we want to reduce the search space, a sensible starting point would be to identify $K' < K$ based on the distribution of $\{\alpha_1, \ldots, \alpha_K\}$ and examine only the first K' trees in the ensemble. The potential error resulting from only considering the first K' trees is sufficiently small given that the weights of the remaining trees are small, and therefore their impact on the overall prediction is minimal in comparison to the first K' trees. In addition to giving us a way to reduce the search space, this K' can also provide some insight into how difficult it was for the model to classify this instance – the larger the K', the more difficult it was.

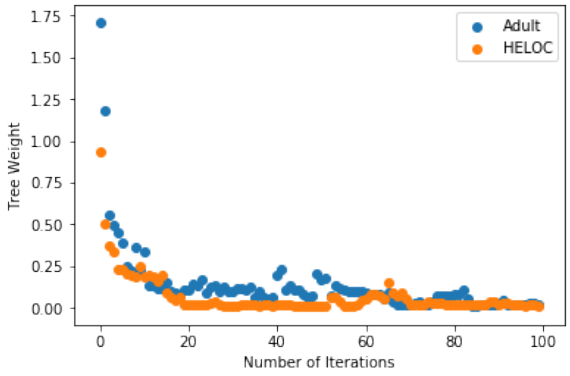

Figure 1: The distribution of weights α_k for each iteration (or tree) in the ensemble.

Another option for determining the subset of trees $\mathcal{T}$ that would allow us to reduce the search space is by looking for structure in how the sample weights $\{w_1(x), \ldots, w_K(x)\}$ change as an instance x goes through each iteration k of the model and identifying trees of interest based on this distribution. If the prediction of iteration k, $\hat{h}_k(x)$, is correct, then $w_k(x) < w_{k-1}(x)$; the opposite is true if $\hat{h}_k(x)$ is incorrect. Figure 2 shows the evolution of these sample weights for two random instances in each of the datasets, *Adult* and *HELOC*: one that is correctly classified (depicted in green) and one that is incorrectly classified (depicted in red). We see that in both datasets, the weights for the correct instances decrease substantially within the first 15 iterations, implying the model is continuously classifying the instances correctly. In contrast, the weights for the incorrect instances increase substantially within this same period, implying the model is continuously misclassifying these instances. When the weights flatten out (e.g., for the correct instance in the *Adult* dataset between $k = 18$ and $k = 40$), this implies $\hat{h}_k(x)$ is oscillating between $+1$ and -1, or analogously, oscillating between being correct and incorrect. The structure in the weight evolution of a particular instance gives us some insight into how the model learns to classify this point and how the the prediction fluctuates with each iteration. This can help us determine which trees should be included in the subset of the original ensemble we want to examine further and we outline some further ideas for this in Section 3.4.

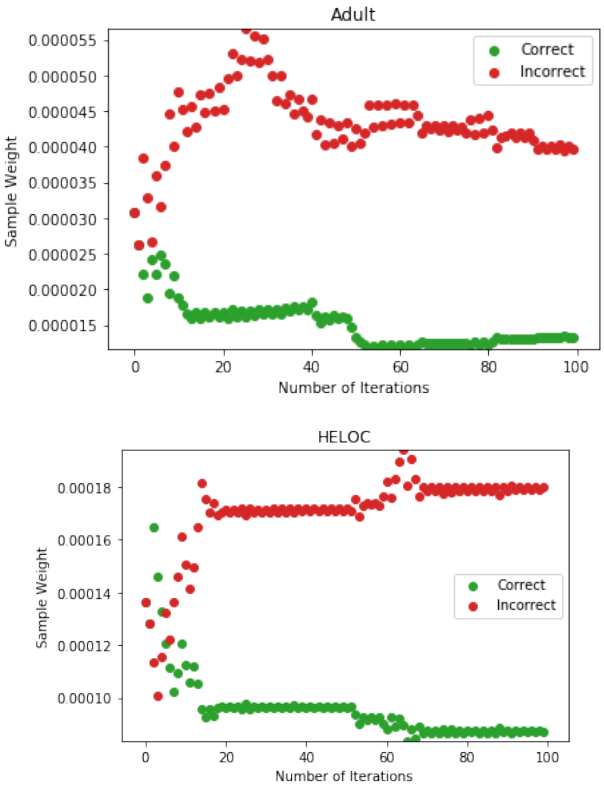

Figure 2: The evolution of sample weights $w_k(x)$ for one correctly classified instance and one incorrectly classified instance in the Adult (above) and HELOC (below) datasets.

3.4 Next steps

We have provided some initial ideas for generating explanations for tree-based boosting predictions. We plan to investigate learning K' for a given instance x, perhaps based on the distribution of training sample weights $\{w_k(x)\}_{k=1}^K$ along with the weights of each iteration $\{\alpha_k\}_{k=1}^K$. We also plan to investigate how this method could be extended to account for training on the negative gradients as is done in GBDTs.

Finally, we plan to evaluate our method, and, in particular, the degree to which our proposed explanations are actionable, through

a user study with participants from the MSc Data Science and MSc Artificial Intelligence programs at the University of Amsterdam.

4 CONCLUSION

We have sketched a research agenda for explaining predictions from boosting methods and sketched a case study to illustrate how to generate these explanations.

In our case study, we examined how we can use the sequential training nature of boosting methods to narrow the search space for alternative examples when generating explanations. We will also explore how training on the negative gradient can be used to generate explanations for GBDT predictions and will evaluate the impact these types of explanations have on users who interact with the system. Finally, we invite the community to join the discussion on how we can automatically and transparently fix algorithmic errors, in ways that are meaningful for IR system experts as well as those outside the community.

ACKNOWLEDGMENTS

This research was partially supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organisation for Scientific Research (NWO) under project nr 652.001.003. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] 1996. UCI Machine Learning Repository. (1996). https://archive.ics.uci.edu/ml/datasets/Adult

[2] David Alvarez-Melis and Tommi S. Jaakkola. 2018. On the Robustness of Interpretability Methods. *arXiv:1806.08049 [cs, stat]* (June 2018). arXiv: 1806.08049.

[3] FICO. [n. d.]. Explainable Machine Learning Challenge. ([n. d.]). https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2

[4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2000. Additive Logistic Regression: A Statistical View of Boosting. (2000), 71.

[5] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. 2018. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933* (2018).

[6] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class AdaBoost. *Statistics and Its Interface* 2, 3 (2009), 349–360.

[7] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artifical Intelligence* 267 (February 2019), 1–38.

[8] Chen Qu, Liu Yang, Bruce Croft, Falk Scholer, and Yongfeng Zhang. 2019. Answer Interaction in Non-factoid Question Answering Systems. *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval - CHIIR '19* (2019), 249–253. https://doi.org/10.1145/3295750.3298946 arXiv: 1901.03491.

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. *ICML Workshop on Human Interpretability in Machine Learning* (2016).

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *KDD*. ACM, 1135–1144.

[11] Maartje ter Hoeve, Mathieu Heruer, Daan Odijk, Anne Schuth, Martijn Spitters, and Maarten de Rijke. 2017. Do news consumers want explanations for personalized news rankings?. In *FATREC Workshop on Responsible Recommendation*.

[12] Nava Tintarev. 2007. Explaining Recommendations. In *User Modeling 2007*, Cristina Conati, Kathleen McCoy, and Georgios Paliouras (Eds.). Vol. 4511. Springer Berlin Heidelberg, Berlin, Heidelberg, 470–474.

[13] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17* (2017), 465–474.