Consistent feature attribution for tree ensembles

Scott M. Lundberg SLUND1@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science, University of Washington, Seattle, WA 98105 USA

Su-In Lee SUINLEE@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science and Department of Genome Sciences, University of Washington, Seattle, WA 98105 USA

Abstract

It is critical in many applications to understand what features are important for a model, and why individual predictions were made. For tree ensemble methods these questions are usually answered by attributing importance values to input features, either globally or for a single prediction. Here we show that current feature attribution methods are inconsistent, which means changing the model to rely more on a given feature can actually decrease the importance assigned to that feature. To address this problem we develop fast exact solutions for SHAP (SHapley Additive exPlanation) values, which were recently shown to be the unique additive feature attribution method based on conditional expectations that is both consistent and locally accurate. We integrate these improvements into the latest version of XGBoost, demonstrate the inconsistencies of current methods, and show how using SHAP values results in significantly improved supervised clustering performance. Feature importance values are a key part of understanding widely used models such as gradient boosting trees and random forests. We believe our work improves on the state-of-the-art in important ways, and so impacts any current user of tree ensemble methods.

1. Introduction

Understanding why a model made a prediction is important for trust, actionability, accountability, debugging, and many other common tasks. To understand predictions from tree ensemble methods, such as gradient boosting trees or random forests, importance values are typically attributed to each input feature. These importance values can be computed either for a single prediction, or an entire dataset to explain a model's overall behavior.

Concerningly, current feature attribution methods for tree ensembles are *inconsistent*, meaning they can assign higher importance to features with a lower impact on the model's output. This inconsistency effects a very large number of users, since tree ensemble methods are widely applied in research and industry.

Here we show that by connecting tree ensemble feature attribution methods with the recently defined class of *additive feature attribution methods* (Lundberg & Lee, 2017) we can motivate the use of SHapley Additive exPlanation (SHAP) values as the only possible consistent feature attribution method with desirable properties.

SHAP values are theoretically optimal but can be challenging to compute. To address this we derive exact algorithms for tree ensemble methods that reduce the computational complexity from exponential to $O(TL^2)$ where T is the number of trees and L is the number of leaves in each tree. By integrating this new algorithm into XGBoost, a popular tree ensemble package, we demonstrate performance that enables predictions from models with thousands of trees, and hundreds of inputs, to be explained in a fraction of a second.

In what follows we first discuss the inconsistencies of current feature attribution methods as implemented in XGBoost (Chen & Guestrin, 2016), scikit-learn, and the gbm R package (Section 2). We then introduce SHAP values as the only possible consistent attributions (Section 3), and present Tree SHAP as a high speed algorithm for estimating SHAP values of tree ensembles (Section 4). Finally, we use a supervised clustering task to compare SHAP values with previous feature attribution methods (Section 5).

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

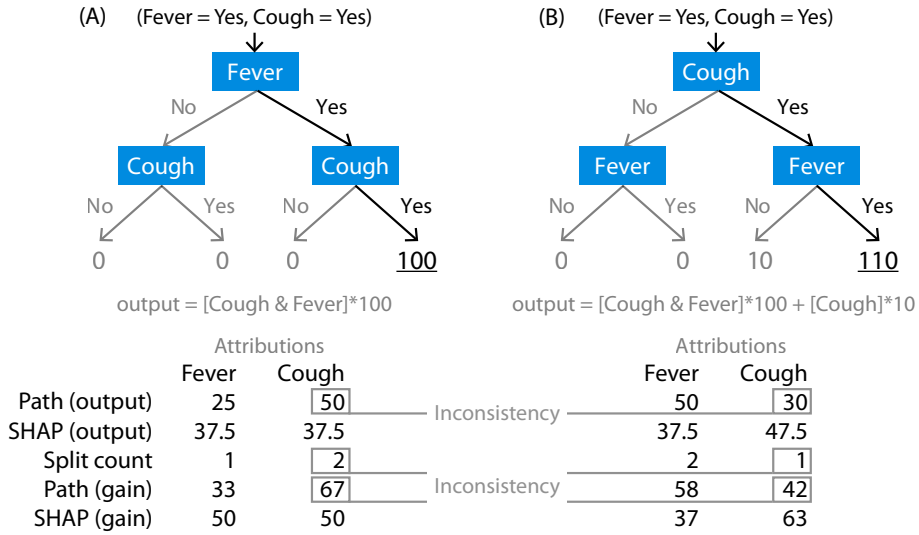

Figure 1. Two tree models meant to demonstrate the inconsistencies of current feature attribution methods. The Cough feature has a larger impact on tree B, but is assigned less importance by all three standard methods. The "output" attributions explain the difference between the expected value of the model output and the current output. The "gain" represents the change in the mean squared error over the whole dataset between when no features are used and all features are used. All calculations assume a dataset (typically a training dataset) perfectly matching the model and evenly spread among all leaves. Sections 3 describes the SHAP values and their interpretation.

2. Current feature attributions are inconsistent

Tree ensemble implementations in popular packages such as XGBoost, scikit-learn, or the gbm R package, allow a user compute a measure of feature importance. These values are meant to summarize a complicated ensemble model and provide insight into what features drive the model's prediction. Unfortunately the standard feature importance values provided by all of these packages are inconsistent, this means that a model can change such that it relies more on a given feature, yet the importance assigned to that feature decreases (Figure 1).

When feature importance values are calculated for an entire dataset they are classically based on the reduction of loss (termed "gain") contributed by each split in each tree of the ensemble. Feature importances are then defined as the sum of the gains of all splits for a given feature (Breiman et al., 1984; Friedman et al., 2001).

Methods computing feature importance values for a single prediction are less established, and of the above packages, only the most recent version of XGBoost supports these calculations natively. The method used by XGBoost (Saabas) is similar to the classical dataset level feature importance calculation, but instead of measuring the reduction of loss it measures the change in the model's output prediction.

Both current feature attribution methods described above only consider the effect of splits along the decision path, so we will term them *path* methods. Figure 1 shows the result of applying both these methods to two simple regression trees. For the gain calculations we assume equal coverage of each of the four tree leaves, and perfect regression accuracy. In other words, an equal number of dataset points fall in each leaf, and the label of those points is exactly equal to the prediction of the leaf. The tree in Figure 1A represents a simple AND function, while the tree in Figure 1B represents the same AND function but with an additional increase in predicted value when Cough is "Yes".

The point of Figure 1 is to compare feature attributions between A and B, where it is clear that Cough has a larger impact on the model in B than the model in A. As highlighted below each tree, we can see that current path methods (as well as the simple split count metric) are inconsistent because they allocate less importance to Cough in B, even though Cough has a larger impact on the output of the tree in B. The "output" task explains the change in model output from the expected value to the current predicted value given Fever and Cough. The "gain" explains the reduction in mean squared error contributed by each feature (assuming a dataset as described in the previous paragraph). In contrast to current approaches, the SHAP values (described below) are consistent, even when the order in which features appear in the tree changes.

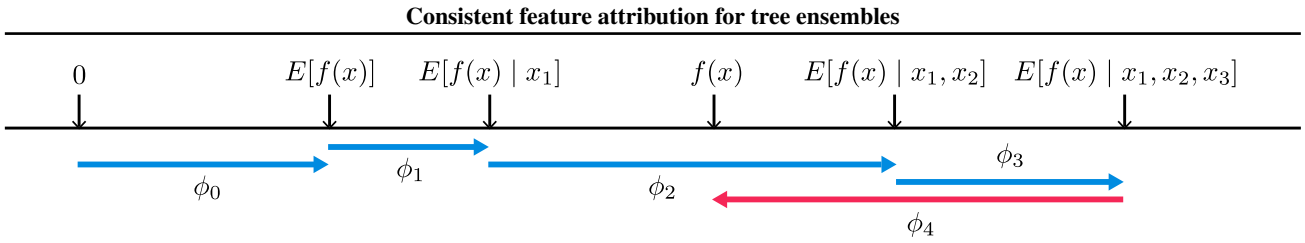

Figure 2. SHAP (SHapley Additive exPlanation) values explain the output of a function as a sum of the effects ϕ_i of each feature being introduced into a conditional expectation. Importantly, for non-linear functions the order in which features are introduced matters, so SHAP averages over all possible orderings. Proofs from game theory show this is the only possible consistent and locally accurate approach. In contrast, standard path methods for tree ensembles (Section 2) are similar to using a single ordering defined by a tree's decision path.

3. SHAP values are the only consistent feature attributions

It was recently noted that many current methods for interpreting machine learning model predictions fall into the class of additive feature attribution methods (Lundberg & Lee, 2017). This class covers all methods that represent a model's output as a sum of real values attributed to each input feature.

Definition 1 Additive feature attribution methods *have an explanation model that is a linear function of binary variables:*

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{1}$$

where $z' \in \{0,1\}^M$, M *is the number of input features, and* $\phi_i \in \mathbb{R}$.

The z'_i variables typically represent a feature being observed ($z'_i = 1$) or unknown ($z'_i = 0$), and the ϕ_i's are the feature attribution values.

As previously described in Lundberg & Lee, an important attribute of the class of additive feature attribution methods is that there is a single unique solution in this class with three desirable properties: *local accuracy*, *missingness*, and *consistency*. Local accuracy states that the sum of the feature attributions is equal to the output of the function we are seeking to explain. Missingness states that features that are already missing (such that $z'_i = 0$) are attributed no importance. Consistency states that changing a model, such that observing a feature has a larger impact on the model, will never decrease the attribution assigned to that feature.

In order to evaluate the effect missing features have on a model f, it is necessary to define a mapping h_x that maps between the original function input space and the binary pattern of missing features represented by z'. Given such a mapping we can evaluate $f(h_x^{-1}(z'))$ and so calculate the effect of observing or not observing a feature (by setting $z'_i = 1$ or $z'_i = 0$).

SHAP values define $f_x(S) = f(h_x^{-1}(z')) = E[f(x) \mid x_S]$ where S is the set of non-zero indexes in z' (Figure 2), and then use the classic Shapley values from game theory to attribute ϕ_i values to each feature:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S|! - 1)}{M!} \left[f_x(S \cup \{i\}) - f_x(S) \right] \tag{2}$$

where N is the set of all input features.

The SHAP values are the only possible consistent, locally accurate method that obeys the missingness property and uses conditional dependence to measure missingness. This is strong motivation to use SHAP values for tree ensemble feature attribution, particularly since current tree ensemble feature attribution methods already obey all of these properties except consistency. This means that switching to SHAP values provides a strict theoretical improvement over existing approaches by eliminating the type of unintuitive consistency problems demonstrated in Figure 1.

4. Tree SHAP: Fast exact SHAP value computation

Despite the compelling theoretical advantages of SHAP values, their practical use is hindered by two problems:

1. The challenge of estimating $E[f(x) \mid x_S]$ efficiently.
2. The exponential complexity of Equation 2.

Here we focus on tree models and propose fast SHAP value estimation methods specific to trees and ensembles of trees.

4.1. Estimating $E[f(x) \mid x_S]$ efficiently

For a tree model $E[f(x) \mid x_S]$ can be estimated by starting at the root of the tree and either following all children if the variable that node splits on is not in S, or just the one child matching x_S if the variable is in S. Recursively follow this

approach, starting with a weight of 1.0 at the root and scaling this weight by the proportion of the cover (which represents how many training data points fall into that branch) of each child node with respect to its parent's cover, unless we are just following one child (because we are conditioning on that node's variable) in which case the weight remains unchanged. Collecting all the leaf nodes encountered, and summing the output values of each leaf weighted by their computed weights will estimate $E[f(x) \mid x_S]$.

4.2. Computing the Shapley equation in polynomial time

Recall that a subset S represents a binary vector z'. If we only take the expectation over a single point in the training data set (meaning for $z'_i = 0$ we follow only one path, not all paths) these binary vectors reflect paths to different leaf nodes. This means that each path is associated with exactly one leaf node, and so the leaves of the tree partition the space of binary vectors.

The SHAP values are computed as a weighted linear combination of function values for all possible binary vectors of length M. Since the function value is unchanging within a leaf we can aggregate the weights of all the binary vectors that associate with that leaf and apply that weight one time to the function value at that leaf. This results in $O(L)$ terms, significantly less than the 2^M terms in Equation 2.

In practice we want to take the conditional expectation using more than just a single data point from the training set, we want to use the full data set. This means the same binary vector can now be associated with multiple leaves because $z'_i = 0$ (an unconditioned variable) means we can go down all branches. If we maintain the weights of these branch splits, just as above for computing the conditional expectation, then even though a single binary vector can land at many leaves, it will be weighted by the proportion of times it lands in each leaf. Using this weighting combined with the Shapley value weights for the vector allows us to compute the exact SHAP values in $O(L^2)$ time, or $O(L \log L)$ for a balanced tree. Since the SHAP values of a linear combination of functions is the same linear combination of those function's SHAP values, we can combine the values for each tree across an ensemble of trees in time $O(TL^2)$ (where T is the number of trees in the ensemble, and factorial functions are assumed to be constant-time).

We refer to this tree specific algorithm for SHAP values as Tree SHAP. The full algorithm is given by Algorithm 1, which uses the follow definitions:

$$\omega(m, i) = \frac{s!(|m_{\{0\}}|)!}{(s + |m_{\{0\}}| + 1)!} \tag{3}$$

$$s = \begin{cases} |m_{\{1\}}| - 1, & \text{if } m_i = 1 \\ |m_{\{1\}}|, & \text{if } m_i = 0 \end{cases} \tag{4}$$

In Algorithm 1 v is vector of node values, which takes the value $internal$ for internal nodes, a and b are vectors representing the left and right node indexes for each internal node, t is a vector of thresholds for each internal node, d is a vector of indexes of the features used for splitting in internal nodes, and r is vector representing the cover of each node (how many data samples fall in that subtree).

Algorithm 1 Tree SHAP

procedure TS(x, ϕ, tree $= \{v, a, b, t, r, d\}$)
  procedure RECURSE(j, m, w)
    if $v_j \neq internal$ **then**
      for $i \leftarrow 1$ to M **do**
        if $m_i = 1$ **then**
          $\phi_i \mathrel{+}= w \cdot \omega(m, i) \cdot v_j$
        else if $m_i = 0$ **then**
          $\phi_i \mathrel{-}= w \cdot \omega(m, i) \cdot v_j$
        end if
      end for
    else
      if $m_{d_j} \neq 1$ **then**
        $m' = copy(m)$
        $m'_{d_j} = 0$
        RECURSE(a_j, m', $w \cdot r_{a_j}/r_j$)
        RECURSE(b_j, m', $w \cdot r_{b_j}/r_j$)
      end if
      if $m_{d_j} \neq 0$ **then**
        $m' = copy(m)$
        $m'_{d_j} = 1$
        if $x_{d_j} \leq t_j$ **then**
          RECURSE(a_j, m', w)
        else
          RECURSE(b_j, m', w)
        end if
      end if
    end if
  end procedure
  $m =$ array of M values equal to -1
  RECURSE(1, m, 1)
end procedure

Figure 3. SHAP feature attributions produce better clusters than standard path attributions for supervised clustering of 518 participants in an Alzheimer's research study. An XGBoost model with 300 trees of max depth six was trained on 200 gene expression module features using a shrinkage factor of $\eta = 0.01$. This model was then used to predict the CERAD cognitive score of each participant. Each prediction was explained, and then clustered using hierarchical agglomerative clustering (imagine a dendrogram joining the samples above each plot). Red feature attributions push the score higher, while blue feature attributions push the score lower. A) The clusters formed with standard "path" explanations from XGBoost. B) Clusters using our Tree SHAP XGBoost implementation.

5. Supervised clustering experiments

One intriguing use for prediction level feature attributions is what we term "supervised clustering", where instead of using an unsupervised clustering method directly on the data features, you run clustering on the feature attributions (Lundberg & Lee, 2016).

Supervised clustering naturally handles one of the most challenging problems in unsupervised clustering: determining feature weightings (or equivalently, determining a distance metric). Many times we want to cluster data using features with very different units. Features may be in dollars, meters, unit-less scores, etc. but whenever we use them as dimensions in a single multidimensional space it forces any distance metric to compare the relative importance of a change in different units (such as dollars vs. meters). Even if all our inputs are in the same units, often some features are more important than others. Supervised clustering uses feature attributions to naturally convert all the input features into values with the same units as the model output. This means that a unit change in any of the feature attributions is comparable to a unit change in any other feature attribution. It also means that fluctuations in the feature values only effect the clustering if those fluctuations have an impact on the outcome of interest.

Here we compare feature attribution methods by applying supervised clustering to disease sub-typing, an area where unsupervised clustering has contributed to important discoveries. The goal of disease sub-typing is to identify subgroups of patients that have similar mechanisms of disease (similar reasons they are sick). Here we consider Alzheimer's disease where the predicted outcome is the CERAD cognitive score (Mirra et al., 1991), and the features are gene expression modules (Celik et al., 2014).

By representing the positive feature attributions as red bars and the negative feature attributions as blue bars (as in Figure 2), we can stack them against each other to visually represent the model output as their sum. Figure 3 does this vertically for each participant. The explanations for each participant are then stacked horizontally according the leaf order of a hierarchical clustering. This groups participants with similar predicted outcomes and similar reasons for that predicted outcome together. The clearer structure in Figure 3B indicates the SHAP values are better feature attributions, not only theoretically, but also practically.

The improvement in clustering performance seen in Figure 3 can be quantified by examining how well each clustering explains the variance of the CERAD score outcome. Since hierarchical clusterings encode many possible groupings, we plot in Figure 4 the change in the R^2 value as the number of groups shrinks from one group per sample ($R^2 = 1$), to a single group ($R^2 = 0$).

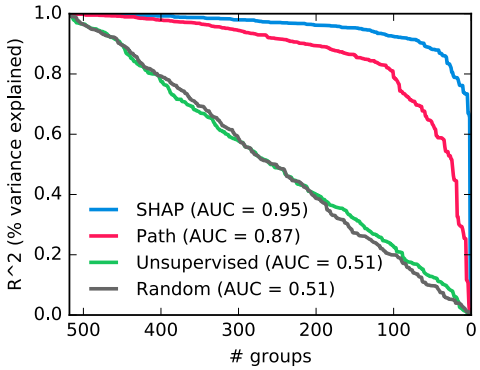

Figure 4. A quantitative performance measure of the clusterings shown in Figure 3. If all 518 samples are placed in their own group, and each group predicts the mean value of the group, then the R^2 value (the proportion of outcome variance explained) will be 1. If groups are then merged one-by-one the R^2 will decline until when there is only a single group it will be 0. Hierarchical clusterings that well separate the outcome value will retain a high R^2 longer during the merging process. Here unsupervised clustering did no better than random, supervised clustering with the XGBoost "path" method did significantly better, and SHAP values significantly better still.

6. Conclusion

Here we have shown that classic feature attribution methods for tree ensembles are inconsistent, meaning they can assign less importance to a feature when the true effect of that feature increases. In contrast, SHAP values were shown to be the unique way to consistently attribute feature importance. By deriving fast algorithms for SHAP values and integrating them with XGBoost, we make them a practical replacement for previous methods. Future directions include deriving fast dataset-level SHAP algorithms for gain (as opposed to the instance-level algorithm presented here), and integrating SHAP value algorithms into the released versions of common packages.

Acknowledgments

We would like to thank Gabriel Erion for suggestions that lead to a simplified algorithm, as well as Jacob Schreiber and Naozumi Hiranuma for providing helpful input.

References

Breiman, Leo, Friedman, Jerome, Stone, Charles J, and Olshen, Richard A. *Classification and regression trees*. CRC press, 1984.

Celik, Safiye, Logsdon, Benjamin, and Lee, Su-In. Efficient dimensionality reduction for high-dimensional network estimation. In *International Conference on Machine Learning*, pp. 1953–1961, 2014.

Chen, Tianqi and Guestrin, Carlos. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Lundberg, Scott and Lee, Su-In. An unexpected unity among methods for interpreting model predictions. *arXiv preprint arXiv:1611.07478*, 2016.

Lundberg, Scott and Lee, Su-In. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

Mirra, Suzanne S, Heyman, A, McKeel, D, Sumi, SM, Crain, Barbara J, Brownlee, LM, Vogel, FS, Hughes, JP, Van Belle, G, Berg, L, et al. The consortium to establish a registry for alzheimer's disease (cerad) part ii. standardization of the neuropathologic assessment of alzheimer's disease. *Neurology*, 41(4):479–479, 1991.

Saabas, Ando. Interpreting random forests. `http://blog.datadive.net/interpreting-random-forests/`. Accessed: 2017-06-15.