Current Status

Christian

March 4, 2021

Consistency in RF

Random forests are partitioning estimators and there two rough categories of them: Data-independent partitioning estimators and data-dependent partitioning estimators. They differ a bit when it comes to the consistency of the two estimators. The original CART algorithm by Breimann is a data-dependent partitioning estimator. However, almost all research focusing on consistency up to now has adapted the original algorithm such that it becomes a data-independent partitioning estimator.

Data-independent Partitions

Consistency can be shown for data independent partitions with Stone's Theorem which needs two conditions:

- The diameter of the underlying cells shrinks toward zero as $n \to \infty$
- The number of cells is small with respect to n. This ensures that each cells contains lots of data points

Once these conditions are given, consistency of the partitioning estimator follows. The crucial part here is the independence of the structure of the partition and the estimator per se. Examples for data-independent partitions in a RF setup include:

- Random Splits: The dimension and the exact location of the split are chosen randomly
- Median Forests: The split point is always exactly the median of a chose split dimension
- Honest Forests: One splits the dataset into two halves: "structure points" and "estimation points". The structure points are used to construct the tree and therefore determine the shape of the tree. When it comes to estimating the value inside the final leaf however, the average is taken over the Y-values of the estimation points. This way the same Y is not used for both construction of the tree and also estimation.

All of them are still random forests but they adapt their algorithm in such a way the the partition is independent of the data used for the estimation.

Data-dependent Partitions

Proving consistency for data-dependent partitioning estimators is more involved. That's because the same Y-values are used to construct a tree and then also for estimation. In addition to the requirements above for consistency the important new assumption is that the underlying function is "not too complex". This means that

- the maximum number of cells is small compared to n
- the maximum number of possible distinct partitions is small compared to n

One popular way of ensuring these requirements is to control the tree structure with a parameter α. This parameter determines the fraction of observations which have to be in a leaf at minimum for it to split it. If $\alpha = 0.2$ it means that as soon as there is less than 20% of the sample in a leaf, that particular leaf will not be split further.

The original algorithm, which did not have this tuning parameter, allows trees to be "fully grown" meaning that in the end there was only one observation inside a final leaf. For this type of approach only one paper by Biau et al. (2015) has shown consistency. However, they have the assumption, that the underlying function is additive. Their proof deviates from the "standard" way. Contrary to the requirement that the diameter of the cells shrinks to zero, they impose the assumption that the variation of Y inside a cell converges to 0.

Simulations

I looked at the behaviour of the original CART algorithm and the random-split point in both an additive and non-additive model. As far as we know up to know, the random-split point approach is consistent in both regimes whereas the optimizing CART algorithm is only known to be consistent with an underlying additive model. Specifically I looked at the average tree depth to see how "balanced" the tree is.
Interestingly, in the additive regime the CART algorithm chooses its split points always somewhat in the middle of the chosen split dimension. This makes it quite similar to the random-splitter and both approaches lead to pretty balanced tree structures. In contrast, the non-additive model leads the optimizing algorithm to split very often at the edges. The observed end-cut-preference is mentioned throughout the literature as a problem/disadvantage of random forests but nothing going beyond a couple of lines on this problem and how to mitigate it in practice. This behaviour makes the trees grown with the CART algorithm very deep whereas the random-splitter tree is still very balanced.

ToDos

For the next couple of weeks I'd like to see whether it can be shown mathematically why exactly the additive model leads to split points chosen by the CART to be in the middle whereas in the non-additive model it tends to go to the very edges.