The Shape of the Trees in Gradient Boosting Machines

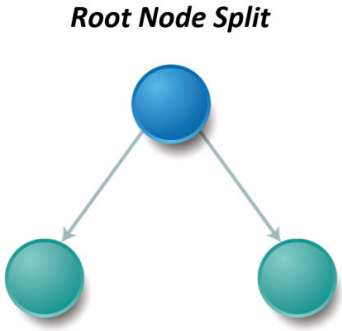
The gradient boosting machine has recently become one of the most popular learning machines in widespread use by data scientists at all levels of expertise. Much of the rise in popularity has been driven by the consistently good results Kaggle competitors have reported over several years of competition. Many users of gradient boosting machines remain a bit hazy regarding the specifics of how such machines are actually constructed and of where the core ideas for such machines come from. Here we want to discuss some details of the shape and size of the trees in gradient boosting machines.

The gradient boosting machine was originally developed by Stanford University Professor Jerome H. Friedman, a physicist by training who has been with the Stanford Statistics Department since 1982. Friedman is also known for his groundbreaking work on fast nearest neighbor computation (1974), recursive partitioning trees (CART®, 1984), MARS® Regression splines (1991), fast elastic net computation (PathSeeker, 2004, Generalized PathSeeker, 2008) among many others. His fundamental papers on gradient boosting appeared in 1999 and 2001, where he discusses his tree growing strategy. Friedman's software was released as a commercial product under the name TreeNet® by Salford Systems in 2002 and this software remains the only gradient boosting machine based on Friedman's proprietary code. Independently written open source gradient boosting machines have been published since 2002.

In Friedman's papers he outlines what he calls a "best-first" binary tree growing strategy which works as follows. First, of course, we must split the root node. We then must decide whether we will continue to grow the tree by splitting the left or the right child node. We make the decision by actually splitting both children but we commit only the split that most improves the performance of the tree on the training data. Having made this decision, we are free to stop the tree growth and operate with a 3-terminal tree. True, we have actually split both child nodes and know what the 4-terminal node tree could be. But at this stage we officially have only a 3-node tree. If the modeler has requested 3-node trees then we have completed the needed tree for this cycle of learning and we move on to the construction of the next tree. But if, for example, 5-node trees were requested, then we repeat the process of deciding which of the available terminal nodes in our tree we should split next. Again, to make that decision we proceed to split both of the most recently generated terminal nodes and commit to the split that yields the best improvement in performance on the training data. This means that as we grow the tree we never know which available terminal node will be split next and the trees can turn out to be highly unbalanced. For example, after splitting the root node, the tree may never end up being grown further on one side of the tree, and all further splits can appear on the other side. The "best-first" tree growing strategy is found only in Friedman's original gradient boosting software (TreeNet).

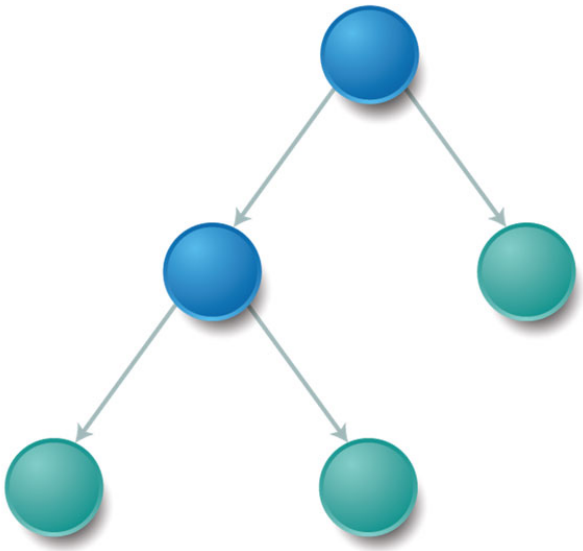
By contrast, the gbm and xgboost gradient boosting machines found in R and other places, and also the gradient boosted tree models being offered by some machine learning startups only grow (or at least try to grow) completely balanced trees. Users can only request trees described by their depth and since the trees are binary recursive partitioned trees they will contain a power of two terminal nodes, i.e. 2, 4, 8, 16, etc. terminal nodes. In other words, these gradient boosting machines are essentially confined to growing a very limited number of tree sizes, whereas Friedman's gradient boosting machine can technically grow trees of any integer size. In the latest version of TreeNet (SPM 8.0, 2016) users have the option of defining trees by their depth limit as well, so as to facilitate comparisons between the two styles of tree growing. In our preliminary experimentation we find the added flexibility of permitted by the "best-first" strategy to be an advantage. However, as always, it pays to experiment to learn which hyper-parameter settings appear to work best for a specific problem.

It might also be worthwhile experimenting with Friedman's idea of random tree size selection in which the intended size of each tree is drawn at random from a Poisson distribution with mean value selected by the modeler. Thus, we might specify that the average tree size we are looking for is one with eight terminal nodes, but the actual number of nodes we attempt to grow for a specific tree will determined by a random draw from a Poisson distribution with a mean of eight. Friedman's original rationale for this was to allow for varying degrees of interaction to be embedded in each tree with the intention of later post-processing the model for rule extraction. However, the idea may also be useful for normal model building.



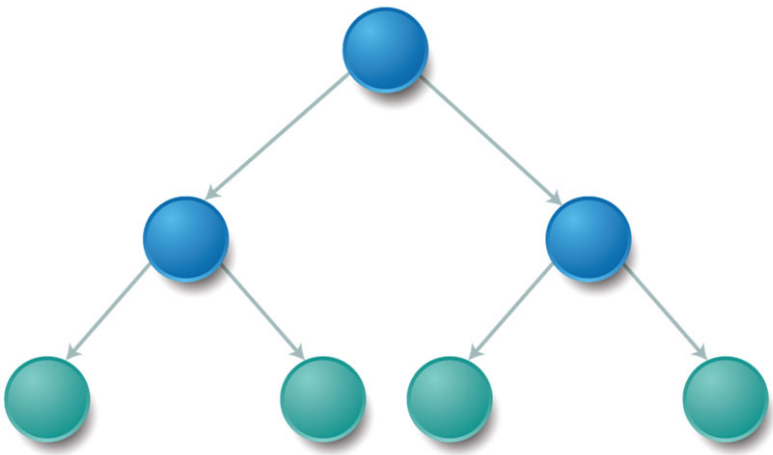
Above we start our tree by splitting the root node and of course all tree growing strategies start here. The question now is which node do we split next? The Friedman approach is to split both left and right child nodes but to keep only the better of the two. Thus the tree below is one that is possibly grown by TreeNet Gradient Boosting having only three terminal nodes.

Possible Best-First Tree



By contrast, the depth defined tree must complete all the splits at a given level and thus if we want a tree with more than two terminal nodes we have no choice but to go all the way to four nodes.

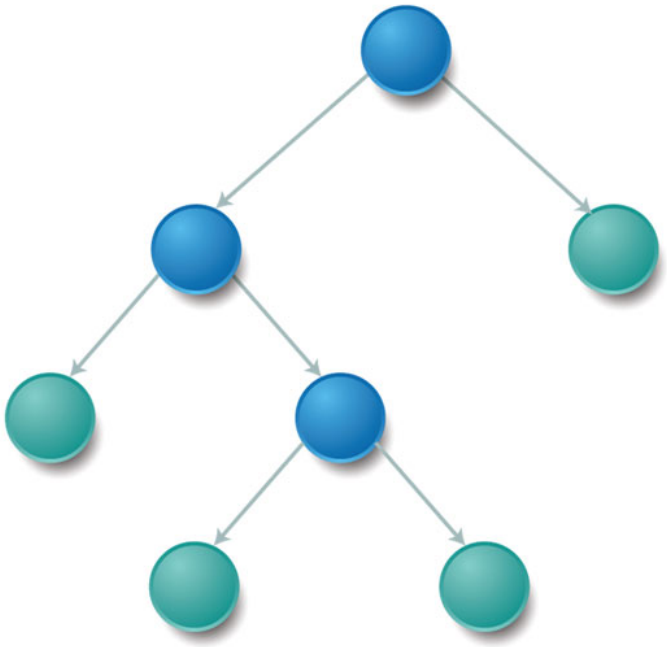
Depth Defined Tree



If you ask TreeNet for 4 node trees you might get trees like the balanced one above but you might also get trees like the one below. How did we come up with that shape? We actually grew out all of the branches leading to a balanced depth-2 tree but discovered that the unbalanced one below gave better results (on the training data). You might argue that the depth constrained tree might be less prone to overfitting because it lacks some of the flexibility of the best-first tree. In our experience, in general, the best-first trees perform

better than depth-defined trees. The best-first strategy allows us to experiment with 3,4,5,6 and 7 node trees while the depth defined tree can only choose between a balanced 4-node and a balanced 8-node tree in this range of sizes. Fortunately, TreeNet allows you to use either strategy when training gradient boosting machines and thus the data scientist has a choice and can tune trees as works best for the problem at hand.

Possible Best-First Tree

- 

The papers listed below were mentioned in this note and are well worth reading. CART, MARS, GPS, and Friedman's gradient boosting machine TreeNet are all embedded in Salford's predictive modeling software suite, SPM.

- Friedman, J. H., Bentley, J. and Finkel, R. A. (1977). "An Algorithm for Finding Best Matches in Logarithmic Expected Time" *ACM Trans. Math. Software* **3**, 209. https://www.salford-systems.com/resources/papers/an-algorithm-for-finding-best-matches-jerome-h-friedman

- Breiman, L., Friedman, J. H., Olshen, R. A. and Stone., C. J. (1983). *Classification and Regression Trees*. Wadsworth
- Friedman, J. H (1991). " Multivariate Adaptive Regression Splines" (with discussion). *Annals of Statistics* 19,1. http://www.slac.stanford.edu/pubs/slacpubs/4750/slac-pub-4960.pdf
- Friedman, J. H. (1999). "Greedy Function Approximation". http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

- Friedman, J. H. (1999) “Stochastic Gradient Boosting”. http://statweb.stanford.edu/~jhf/ftp/stobst.pdf
- Jerome H. Friedman. (2008). “Fast Sparse Regression and Classification” http://statweb.stanford.edu/~jhf/ftp/GPSpub.pdf