# Bayes Risk Pruning

Dylan Chou (dvchou)

January 2021

## 1 Bayes Risk Pruning Definition

For a given decision tree, we calculate the estimated bayes risk of each decision tree node $k$ to be $R_k^i(x) = \sum_{j=1, j\neq i}^{T_c} \lambda_{i,j} p(C_j|x)$ where $R_k^i(x)$ is the risk of node $k$ with associated attributes vector $x$ given the true class is the $i^{th}$ class $C_i$. $x$ is the vector of attributes of an example. $\lambda_{i,j}$ is the 0-1 loss of the example if we predicted (if not true class, then 1): if $i \neq j$, then $\lambda_{i,j} = 1$ otherwise 0. $p(C_j|x)$ is the probability that an example is classified as $C_j$ given its attributes $x$. This would be the proportion of class $C_j$ in the partitioned output at the node arrived to with attributes $x$. Then for each node, we would calculate its total estimated bayes risk to be $\sum_{i=1}^{N_k} R_k^i(x)$ where $N_k$ is the number of data examples that fall into node $k$. The pruning of the nodes occurs bottom-up, left to right. The node $k$ is pruned if its total risk over all relevant examples is less than the sum of risks of the leaves of the subtree under node $k$. Pruning the node $k$ would mean all nodes below it are removed and the node $k$ becomes a leaf node instead of a parent node.

## 2 C4.5 Pruning Importance

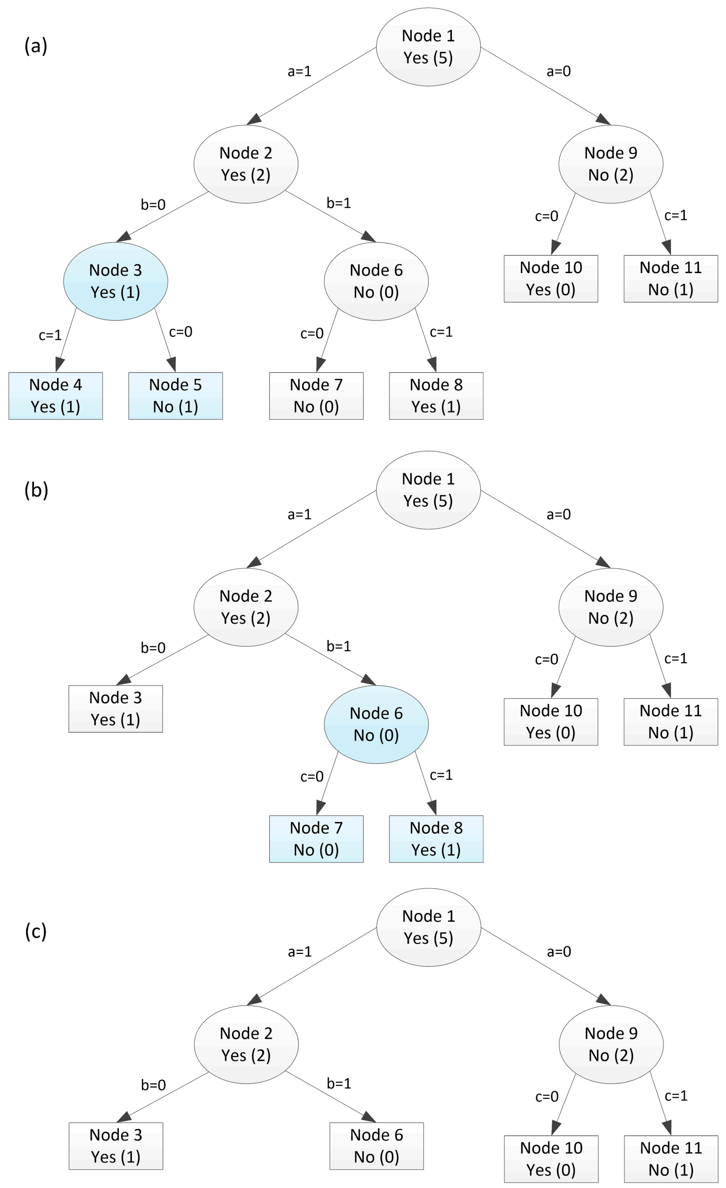
C4.5 is a modification of the ID3 decision tree algorithm in that it can split on both continuous and discrete features, missing values and post-prunes the final decision tree. As with pruning methods in general, the importance comes from reducing overfitting decision trees to training data so the decision tree may fit better on unseen validation data. C4.5 prunes decision trees by subtree raising that takes a subtree and replaces it with a child node that doesn't lower accuracy in the decision tree.

## 3 Weak Points of the Existing Pruning Method

The **PLOS** paper on using a novel estimation method of bayes risk to prune decision trees runs the current pruning algorithm sequentially. It's a bottom-up algorithm and enables propagation of error up the tree rather than estimating misclassification error. However, this can restrict the algorithm to function sequentially, which would not scale well when run over higher dimensional or big data.

## 4 Improving Bayes Risk Pruning

Bayes Risk Pruning, as described in the PLOS paper, is run sequentially on smaller datasets. This can be an efficiency problem for larger datasets. There is potential for this novel decision tree bayes risk pruning method to be parallelized by independently determining whether each node at a certain frontier will be pruned. "Frontiers" of the tree would start with its leaves. After each pruning round, every node in the frontier would be pruned or kept based on the risk values of the parent and leaves. Then we move upward until only the root node is in the frontier. We can think of each frontier as a layer of nodes that can be run in parallel to determine whether they can be pruned or not, then succeed to the next layer, reflecting the steps of breadth-first search (BFS).



For example, in the tree above, we start with the leftmost leaf Node 4, then go through Nodes 5,7,8,10,11. We obviously don't prune them because leaves don't have children. Then Nodes 3,6,9 are traversed. For Node 3, we check its leaves 4,5. For Node 6, we checks its leaves 7,8. For Node 9, we check its leaves 10,11. Then Node 2 and 1 are checked, where Node 2 compares itself to its leaves Node 3 and Node 6 after pruning. For Node 1, it compares itself to the leaves Node 3,6,10,11. Before discussing theoretical efficiencies, **the work** of an algorithm is the number of primitive operations that the available processors perform. **The span** of an algorithm is the longest sequence of operations performed in order due to data dependencies. This means parallelism of the bayes risk pruning algorithm can only occur by level at most. Theoretically, running the sequential bayes risk pruning algorithm would take $O(\sum_{i=1}^{M}\sum_{j=1}^{N_i} W_{n_{ij}})$ work and $O(\sum_{i=1}^{M}\sum_{j=1}^{N_i} S_{n_{ij}})$ span, where $M$ is the number of frontiers the pruning algorithm would iterate over before finishing, $N_i$ is the number of nodes in the frontier on the $i^{th}$ round of pruning, $W_{n_{ij}}$ is the work of running the pruning function for $j^{th}$ node $n_{ij}$ during the $i^{th}$ round, and $S_{n_{ij}}$ is the span of running the pruning function on node $n_{ij}$. However, pruning each frontier of nodes in parallel achieves a span of $O(\sum_{i=1}^{M} max_{j=1}^{N_i} S_{n_{ij}})$ as the most expensive span would make up the cost per frontier. In turn, there would be considerable speedup from $O(\sum_{i=1}^{M}\sum_{j=1}^{N_i} S_{n_{ij}})$ to $O(\sum_{i=1}^{M} max_{j=1}^{N_i} S_{n_{ij}})$ assuming an ideal machine with an unlimited number of processors by the definition of span.

## 5 Summary

The reasons to improve bayes risk are the potential speedups on larger data, clarification in the bayes risk algorithm that the original paper didn't have, and the easy-to-use nature that a parallel python implementation using joblib can offer.