Visualization of Personalized Pruning of Decision Tree

Yue Kang
Department of Electrical Engineering
Stanford, CA 94305
e-mail address yuekang@stanford.edu

Hua Feng
Department of Electrical Engineering
Stanford, CA 94305
e-mail address fengh15@stanford.edu

ABSTRACT
Pruning is an important post-processing step after a decision tree is trained. Though many existing machine learning packages includes automatic pruning as part of their function, most of them are static and don't help users in comprehension of the pruning process. For those allow manual pruning, the function is limited to the replacement of the whole subtree as a leaf node. In this paper we describe our work on making the pruning process more interactive as well as providing higher flexibility with manual pruning. The final result was an web application implemented with D3.

Author Keywords

visualization of machine-learning model, decision tree, decision tree pruning

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): User Interfaces.

INTRODUCTION
Machine learning techniques have been widely used in today's industrial world. Before applying any of the advanced learning algorithms such as neural network to a given dataset, the first step is generally plug in the data to a simpler model such as logistic regression or decision tree. This first step implementation would :

- provide a baseline for the prediction task.
- cast light upon the necessity of using the advanced methods(if baseline method already reached a satisfactory accuracy, then no there is no need for investing more effort into more complicated methods) .
- gain better understanding of the dataset .

The first two goals can be easily reached without much aid of visualization, since it mostly involves the comparison of a single metrics (accuracy or error for test set), where a table would suffice. However, for the understanding of the dataset, which involves comprehension of multidimensional data, visualization would help reduce the mental effort of (1) understanding the importance of the features by comparing their weights in logistic regression or position in the decision tree (2) discover error/bias of the dataset. It would also contribute to the tuning process of the baseline model if the variation of the model caused by tuning could be captured by visualization(though admittedly, baseline methods are supposed to have bad results to create reason for using advanced methods).

In this paper, we focus on the visualization of decision tree and present our work on the implementation which supports the following features:

- training of a decision tree online based with entropy as impurity measure
- direct manipulation on the decision tree via mouse operations with instantaneous response
- three choices of pruning include 2 manual methods and 1 automatic one allowing user to set the threshold
- animated decision tree pruning/expanding process
- a thumbnail of the previous step results for user to keep track of the training result achieved so far

The remaining part of the paper is organized as follows: section 2 presents state of art visualizations of decision tree follows by a discussion over the necessity of our work , section 3 introduces the training and pruning methods used in our work, section 4 presents the implementation results of our visualization and section 5 summarizes the implementation achieved and section 6 discuss the future work on this topic.

RELATED WORK
Various softwares supports the whole process of building, pruning and visualization of decision tree In this section, some of them are introduced with their pros and cons discussed.

Manual Construction of Decision Trees
Such kind of decision tree are built based on pure human experience or human understanding of the data instead of machine learning algorithms. Users are expected to insert the questions at each node and the edges(sometimes with cost and probability information [1][2]) that links it to downstream nodes. Leaf nodes would be the classification results or expected profit through a series of decisions.

For instance, iBoske is an online platform for creating/sharing decision tree, which is a good example of manually built tree visualization. Its interface is shown in figure 1. Decision tree builders can insert/delete nodes by clicking at plus/cross on the tree they are currently working on and leaf nodes can be represented in the form of text, external link or pictures. For users of the decision tree, they will go through a series of questions based on their choice of the previous question and get a result. The color encodes the nature of a node, with orange for the starting point, blue for questions .etc. Other softwares might use shape instead of color.

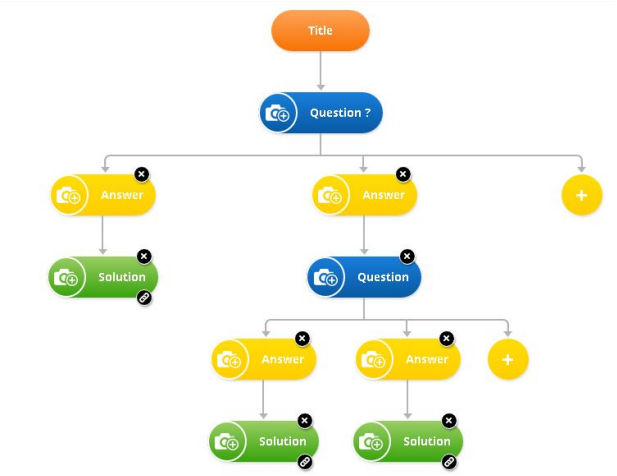

Figure 1 Decision Tree Creating Interface of iBoske

The building process of such kind of trees are simple and interactive, most of them involves clicking and typing of conditions. However inserting all the nodes by hand could be tedious and the application is limited to human biased ones like fun quizzes, making product recommendations(where [3] falls) or decisions in sequential orders like cash flow tracking (where [1][2] fits).

Automatic Construction of Decision Trees

On the other hand, decision trees built with machine learning algorithms are more well-founded in data, has much more applicable fields (can be applied to most of the prediction tasks) and therefore has more implemented softwares/platforms. The majority of machine learning softwares and platforms supports or has plug-ins that support visualization.

Graphviz is a python package that could visualizes the decision tree trained in the machine learning package scikit learn. Figure 2[4] is an example of its visualization using iris data. Each node is represented by a text box containing information about partition condition, the training metric, number of training data partitioned to the node, prediction class. Nodes are also colored by the prediction class and grayscale for impurity measure. With those visual encodings, viewer is able to make some preliminary judgement about the effectiveness of each decision node at making classification at first sight of the figure. However , size of the text is only dependent upon the text content of the text box, and the number of samples is not expressed with any visual variable. So even if some nodes have deep color, it might still be a non-decisive difference between 2 classes for the small amount of support from the training sample. Other environments/ programming language such as r and matlab also provides similar visualization[5][6], but with even fewer information shown in the plot.

Though pruning is supported in all those machine learning packages, the plots are static and viewer has to redraw the tree after pruning. Matlab's plot interface is more interactive by allowing user to choose pruning level through a drop-down menu, the outcome of the pruning is not clear on the graph. All the pruning are automatic and requires setting certain parameters via programming. So even if a user saw nodes s/he would like to prune in the figure, they might not able to prune it without pruning other nodes below the threshold for automatic pruning.

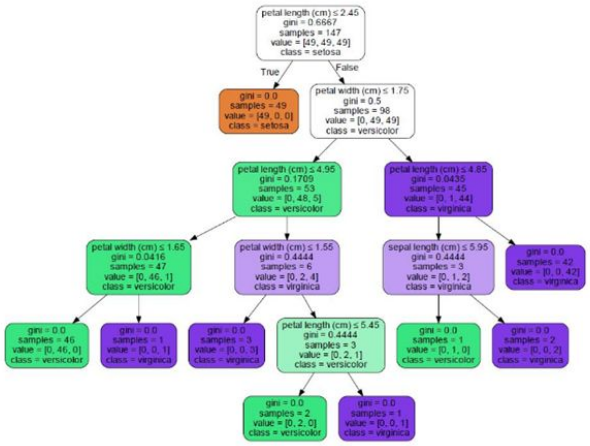

Figure 2 Decision Tree Visualization of GraphViz

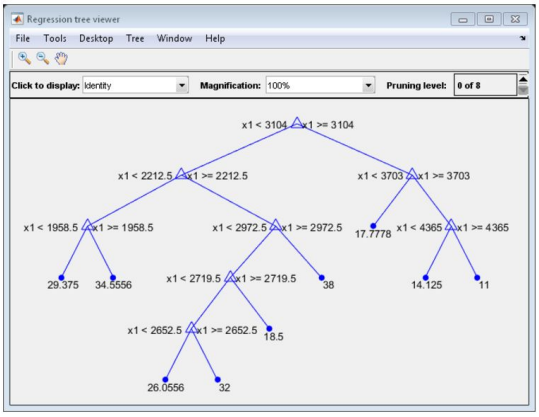

Figure 3 Decision Tree Visualization of Matlab

Except for the PC softwares, there is also a growing number of machine learning cloud platforms. BigML provides successful visualization of decision tree with highly interactive features using d3[7]. Node information is shown and the path is highlighted when mouse hovers over a certain node.Major visual variable used here are color of nodes and width of the branches. Width of the branch could provide users with some idea how many data points are being partitioned to the next layer. Color encodes the feature used for making partition at current node, this could be a way to see what are the features repetitively used in making classification, but it might not work so well when there are lots of features non-repeatedly used.The tree could be pruned by tuning the support/confidence parameters. A drop-down menu is provided to allow tree branch filter based on prediction class. All the modifications are visualized with animated transition of the tree, making the changes more obvious to users . However, left-click is an operation less intuitive, for it would collapse all the nodes other than the ones one selected node's path as leaf nodes instead of a more simpler modification like collapsing the nodes below and there is no clear clue about how to restore to its previous condition after left-click. Also, no manual pruning interaction is implemented.

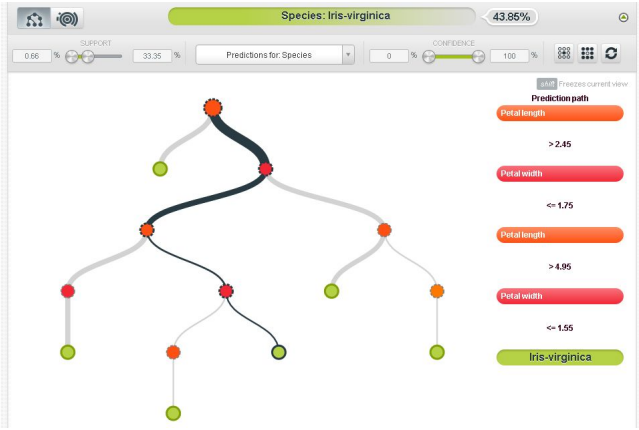

Figure 4 Decision Tree Visualization of BigML

Design Target of Our Work

Based on the existing work , we aim to create an interface that would combine the advantage of human judgement and machine learning for decision tree building. Starting from a decision tree trained from learning algorithm, we would like to offer user is the flexibility to delete nodes as wanted as in the manual decision tree while making the classification process as data based as possible like in an automatical decision tree construction process. We would also like to take advantage of D3 as the current state of art visualization does to animate the building/pruning process to improve comprehension. Detailed implementation would be covered in section 4.

DECISION TREE BUILDING AND PRUNING APPROACHES

This section introduces the behind-the-scene algorithms we used in our project, which provides update of the decision tree when users make interactions with the visualization.

Decision Tree Construction

A greedy algorithmis used to build the decision tree with entropy as the impurity measurement metrics.

The definition of the entropy is :

$$E = \sum_i p_i \log_2(2/p_i)$$

where i is the numbering of the classes, p_i is the percentage of samples belongs to class i among all the samples.

The decision tree is built in a top-down way, and can be represented with the following pseudo code[8] [9]:

```
buildTree(parent_node,data) {
    if (data only contains samples from one class){
            add a leaf node to parent_node;
            return;
    }
    bestGain = 0;
    bestCriteria = "";
    bestSets=[];
     for(i=0; i<#features in data; i++) {
             for(j=0; j<#samples in data; j++) {
            var sets = partition the dataset into 2 parts based on the ith feature, with jth sample's corresponding value as threshold;
            var p =sets[0].length /#samples in data;
             var gain = Entropy(data) - p*Entropy(sets[0]) - (1-p)*Entropy(sets[1]);
            if(gain > bestGain && sets[0].length > 0 && sets[1].length > 0) {
                    bestGain = gain;
                            bestCriteria = feature i >= data[j][i];
```

```
                    bestSets = sets;
            }
        }
    }
add current_node as child of parent_node;
    trueBranch = buildTree(current_node,sets[0]);
    falseBranch =buildTree(current_node,sets[1]);
}
```

Decision Tree Pruning

Automatic Pruning

Since the a decision tree would partition training data until all the leaf nodes are pure if no pre-pruning is involved in the learning process. Overfitting is expected from such a tree and post-pruning is required. There are various ways to do post-pruning such as reduced-error pruning[10], critical value pruning[11] and randomization pruning.

In our implementation, we used critical value pruning, which is a bottom-up pruning algorithm. It can be represented with the following pseudo code[9]:

```
prune(node, threshold){
    if(node.children[0] is not leaf node){
        prune(node.children[0],threshold);
    }
    if(node.children[1] not leaf node){
        prune(node.children[0],threshold);
    }
     if(node.children[0] is leaf node and node.children[1] is leaf node){
        p = #samples at node.children[0]/#samples at node;
        entropyGain = node.Entropy-node.children[0].Entropy-node.Children[1].Entropy;
        if(entropyGain<Threshold){
            set node as a leaf node, remove its children
        }
    }
}
```

Manual Pruning

One straight forward way of manual pruning is to simply set the selected node to leaf node and remove the following subtree, which is our basic version of manual pruning. In case the pruned branches are restored, the deleted children are actually saved in a parameter of node to be toggled back.

Another possible situation is when the user don't agree with the priority assigned to a certain feature based on his/her personal judgement. This situation might occur when the training data is not randomly sampled and has some unexpected bias. To help user see the influence on accuracy by decreasing the priority of that feature, the pruning should cause as little modification to the tree structure as possible. Hence, we propose an algorithm for this kind pruning with constraint.

The general idea is to rebuild the subtree with the selected node as root while the rest of the decision tree remain unchanged. When rebuilding the tree, the selected node's feature is removed from the set of features available for root node. After the new subtree is built, it is compared with the original subtree. Only the nodes that have a corresponding node at the same position of the original subtree are kept to guarantee the newly generated tree won't grow deeper than the original tree and it is in similar shape.

IMPLEMENTATION AND RESULTS

The ideas are implemented with javascript. Node.js accomplishes the functionality of pruning decision tree based on the user interactions and d3.js is used to show the results.

Decision Tree Visualization

As shown in Figure 5, the interface of visualization is divided into two parts. The left part shows the tree for pruning and right part (with yellow background) records the history decision tree.

Color of the nodes represents the classification at current node based on majority class, with legend on the left side. As shown in Figure 6, when hovering over a node, a tooltip providing information about distribution of the training samples partitioned to current node will appear. A simplified stacked bar chart is shown simultaneously with the tool tip, width of each bar representing the number of samples belongs to the corresponding class. This stacked bar chart serves to provide user with an overall sense of the relative size of classified samples. Those statistics are not made static to avoid distracting users from observing the change of decision tree when making interactions.Filling of the circle is used to differentiate pure leaf nodes and impure leaf nodes. Hollow circles are pure leaf nodes with classification labels which is the end of a path and couldn't grow further to increase training accuracy. Solid circle

represents impure leaf node, which could become a decision node and be further expanded.

Because the structure of tree may change a lot when modificated, a thumbnail plot is provided on the right side to record the tree before making an interaction. With the unpruned tree in the background, the nodes before current modification are highlighted and previous training accuracy is displayed. User could decide whether to the modification is reasonable by comparing the two plots. As shown on the right side of Figure 5, the original tree only have 3 nodes (root and its 2 children, all are highlighted), with a training accuracy of 66.67%. After expanding the left child of root on the left side, now the training accuracy has become 96%, so we can tell the original model was underfitting and we should keep this modification.

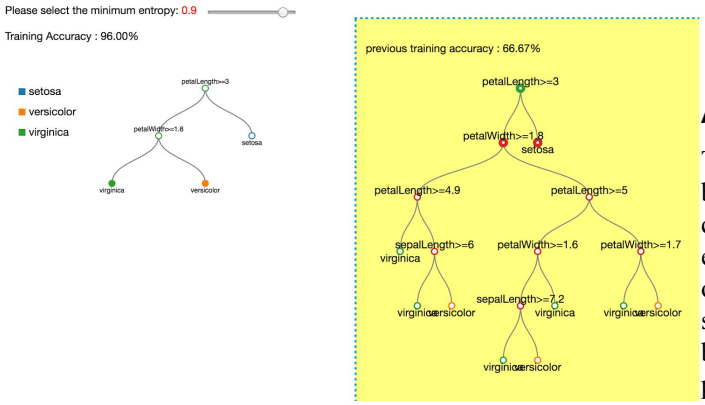

Figure 5 Visualization of decision tree

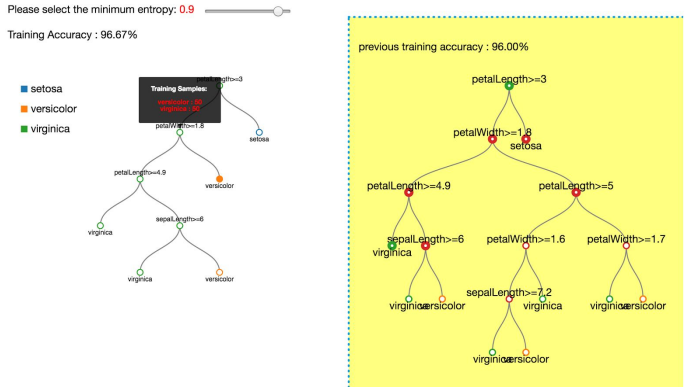

Figure 6 Visualization with mouse hovering

Basic Pruning Demo

The basic manual pruning is triggered when users left click on a decision node. The nodes below selected node will be pruned with the current node turned into a leaf node. Figure 7 shows an screenshot of the left child of root node being pruned in action. As can be seen, the node name has already been toggled to the majority class and the leaf nodes at the deepest layer are collapsing towards their parent nodes. The pruning is reversible by clicking on the node again.

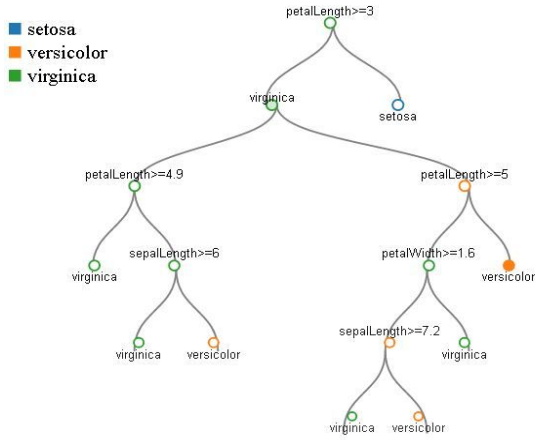

Figure 7 Basic pruning in process

Automatic Pruning Demo

The automatic pruning is triggered by dragging on a slider bar on the top left corner (shown in Figure 8). Slider bar is chosen because it allows continuous change of minimal entropy gain for pruning, which provides users with a sense of how entropy effects pruning results. By moving the slider leftwards, the threshold is lowered, so the tree might be expanded rather than pruned. After doing the manual pruning in Figure 7, minimum entropy is set to 0.7 for automatic pruning in Figure 8. As can be seen, the tree is extended back to its former size.

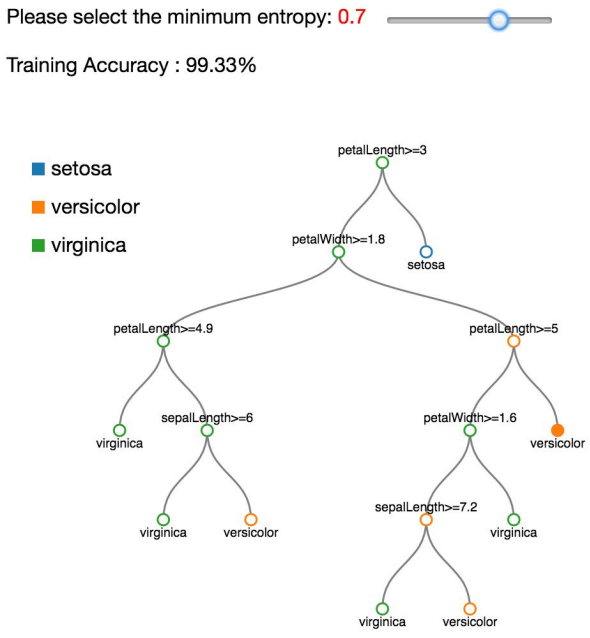

Figure 8 Automatic pruning with slider bar

Advanced Pruning Demo

When right click on a decision node, advanced pruning is triggered, the following subtree is recalculated subjecting to the constraints. Figure 9 shows the updated decision tree after right click on the left child of the root node when decision tree is in the form of Figure 8. As can be seen, the feature for that node was changed and it also influences the decision nodes and the tree structure of its descendants.

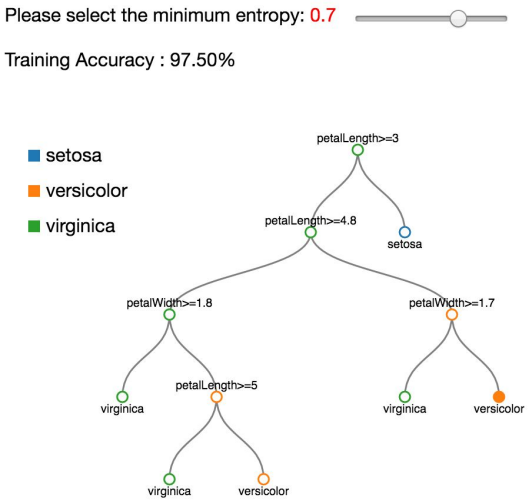

Figure 9 Advanced pruning

CONCLUSIONS

This project provides a tool for visualizing the process of pruning decision tree with simple interactions. By tracking the process of pruning decision nodes and corresponding results, users could easily understand how the decision node works, making modification through interaction with interface and the influence of modification. The project could be widely applied to the field of data science, for both educational or industrial purpose .

FUTURE WORK

There are several aspects of this project that are worth further exploring:

- The building process of the decision tree could be animated as well as the pruning process, to enhance understanding of the algorithm. This will also improve the animation for for advanced pruning when decision node's condition is changed rather than tree structure, for it is also a building process.
- The pruning algorithm should be optimized to deal with situations involving large amount of data. Current algorithms applied requires a full pruning pass through all the data partitioned to the nodes of interest and all the features and the computation are actually implemented at front end. Migrating the algorithm to a backend server and using some parallel computing algorithm would boost speed and allow instantaneous response of interaction even for large dataset.
- This project could also be extended to the visualization of other pruning methods, like reduced-error pruning, minimum error pruning and randomized pruning, so user can make comparison between different methods and choose the pruning process best suit their needs. On the other hand , allowing more choices of algorithm also require more consideration on how to allow user to trigger different pruning functionalities with simple interactions for the front end interface.
- Currently the focus of the project is on the influence of pruning on training set, it could be expanded to the visualization of influence on test set. However, by doing that, there is the risk of user trying to overfit test set through the interactions now they are given the power of manual pruning.
- Based on the use case, this project could be more personalized for education or research. Since there are different concerns for education and research of using decision tree. this visualization could be improved considering the potential users.

REFERENCES

1. Simple Decision Tree

https://sites.google.com/site/simpledecisiontree/

2.TreePlan

http://treeplan.com/company/

3. iBoske

http://www.iboske.com/

4. scikit learn 1.10 Decision Trees

http://scikit-learn.org/stable/modules/tree.html

5. R and Data Mining: Decision Trees

http://www.rdatamining.com/examples/decision-tree

6. MathWorks R2016b Documentation: Decision Trees

https://www.mathworks.com/help/stats/classification-trees-and-regression-trees.html

7. A New Way to Visualize Decision Trees

https://blog.bigml.com/2013/04/19/a-new-way-to-visualize-decision-trees/

8. Large Scale Machine Learning I

http://web.stanford.edu/class/cs246/handouts.html

9.Joon-Ku Kang's Github on machine learning

https://github.com/junku901/machine_learning/tree/master/lib

10. Quinlan, J. R. (1987). Simplifying decision trees. International journal of man-machine studies, 27(3), 221-234.

11. Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. Machine learning, 4(2), 227-243.

12.Oates T, Jensen D. Large Datasets Lead to Overly Complex Models: An Explanation and a Solution[C]//KDD. 1998: 294-298.