Zhanghexuan Ji (zjv5649)

2017-04-11

EECS 349 Machine Learning

pROBLEM SET #2

1. The group numbers are:

Zhanghexuan Ji (zjv5649), Boyu Xu (bxp6650), Daowei Li (dlx4580)

2. I add three attributes in the Node data structure, which are *node.isleaf*, *node.parent*, *node.train\_num*. So the Node structure is:

*Node{label, children, parent, isleaf, train\_num}*

***isleaf*** is a Boolean bit (0 for non-leaf node, 1 for leaf node) used for marking whether a node is a tree node or leaf node (leaf node: label = Class value, isleaf = 1, children = {}), so we can identify a specific node is leaf node or not in prune();

***parent*** is a Node (root.parent = None) used for linking the child node to its parent node, so we can easily get the parent node of a specific node, this is used in prune(); any node can only have one parent;

***train\_num*** is a dictionary for recording the distribution of each class value at node n, where the dictionary keys are the class values and the dictionary values are the numbers of each class in the training examples at the current node n, which is used for evaluate() with missing attributes and prune(), so we can find the mode of the examples (sum of dictionary values) or class values for each node and its child nodes after training.

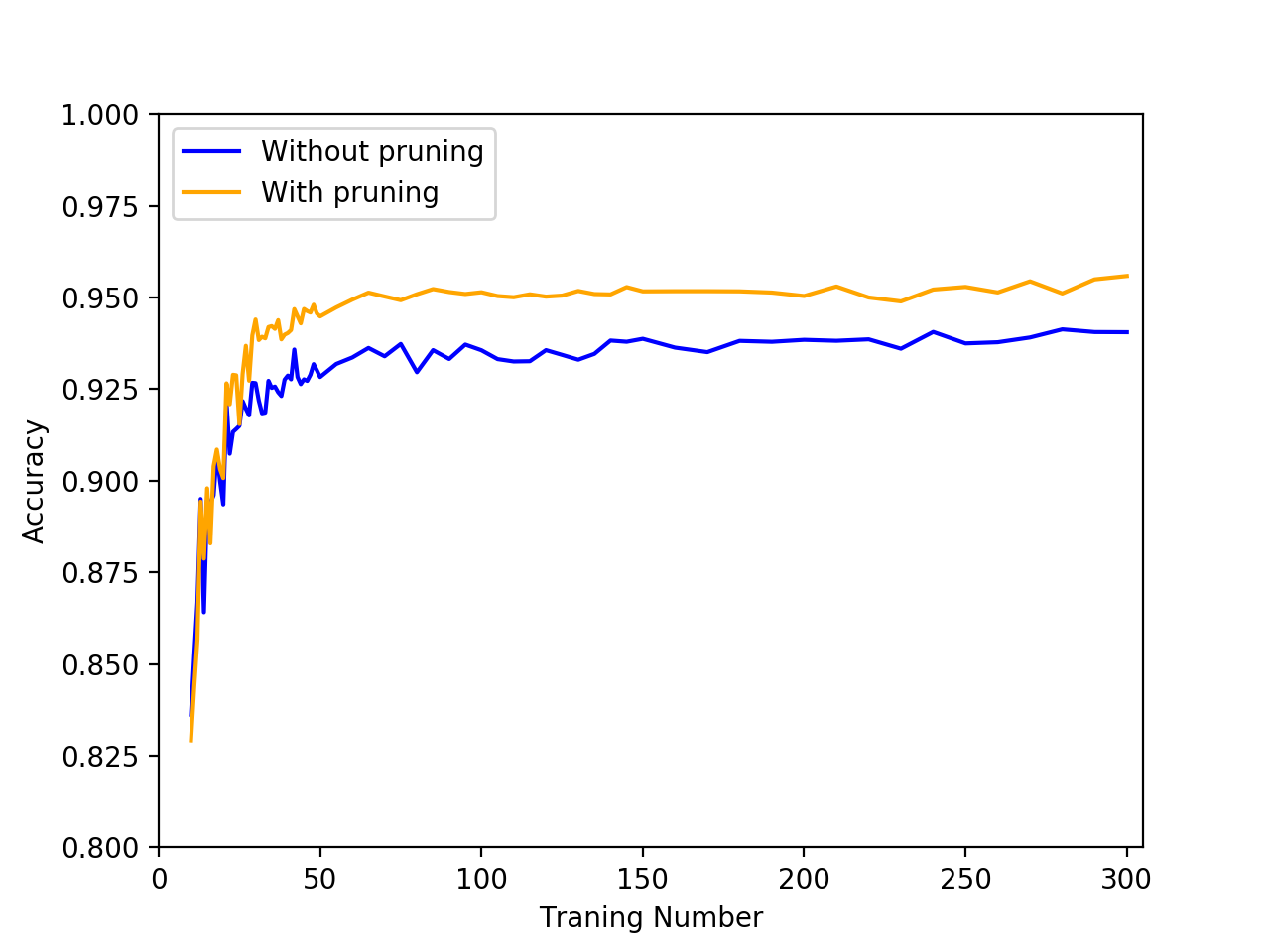
3. For missing attributes in the training process, we simply assign it the value that is most common among training examples at current node n. For missing attributes in the evaluating process, we simply assign it the value that has largest train\_num in the children nodes at current node n.

This strategy is very simple to understand and easy for implement. For missing attributes, we need to assign it a value for further prediction, so the value that is most common should be our preference. And also, when training data is very large, this method can provide good performance due to the highest probability of the statistic distribution of attribute at each node.

4. We use reduced error post pruning for prune(). This method considers each node of the tree as candidates for pruning. When pruning a specific node, we replace it with a leaf node, and the classification is the most common classification value of the training examples at this node. Then we compare the accuracy of the tree with pruning the node and the accuracy of the tree without pruning this node, if the pruning makes the accuracy even worse, then we will keep this node. In order to make sure that before pruning a node n, all its child nodes have been pruning checked, nodes are pruned in the order of post order transverse of the original tree, so we can always check the highest child nodes first. After this transverse and pruning, we can only keep the nodes that perform more accurate on the validation set.

This method has the effect that any leaf node due to coincidence or bias in the training set is more likely to be pruned since these coincidences are unlikely to appear in the validation set. And this method is also easy to implement and with a quite good efficiency. It only requires one transverse of the tree and two accuracy calculation for each node.

5.



(a) The general trend of both lines is: as training set size increases, the learning curve will rise; and when training set is large enough, the learning curve will reach the plateau (with little fluctuation), which is usually the final accuracy of the learning model on the data set.

This makes sense because when training set size is small, the training data is very likely to have large bias and coincidence, which can always cause wrong classification in the decision tree; but when the training set size is large enough, the statistic distribution will make the data set nearly unbiased, therefore reduces the influence of wrong data or random noise, and the decision tree can make much more accurate classification using large training set.

(b) We can find from the learning curve that the curve with pruning has always a better accuracy than that without pruning. The advantage of pruning will firstly increase as the data set size increase, and then remain at a constant level (and even decrease a little bit) when the training set size is large enough.

This makes sense because the pruning can only handle overfitting and reduce some nodes that contain redundant attributes or the nodes that are generated due to random noise. But if the decision tree itself is not accurate due to the small size and bias of training dataset, then the reduce error pruning cannot generate correct nodes. So when the training set size is small, the decision tree is biased, small, and inaccurate (this is not overfitting), therefore pruning can hardly improve the performance of the model. When the training data size is getting larger, the tree is overfitting so pruning can improve the accuracy of the tree pretty well. However, the overfitting part for accuracy improvement via pruning is limited (this may relate to the noise rate of the data and upper limit of the model accuracy) and the accuracy without pruning is also increased a little, this is why the advantage of pruning remain constant (and even decrease a little) when the training set size is very large.