Big Data Analytics Programming
Assignment 3
Efficient Decision Tree Learning

Johan Edstedt

I. Implementation

A. Datastructures

All data was mapped from a string representation to an integer representation. For categorical attributes the categories were simply enumerated, while for the numerical data a slightly more involved process was done where all observed values were stored in an ordered set and when all values had been observed an ordered mapping $[0, |V|)$ where 0 represented the largest seen number and $|V|-1$ the smallest. The training data samples was stored for each attribute in a vector. This made iterating through and copying data much faster.

For counting the label distribution of different values of an attribute, an array was for each split and attribute allocated on the stack where the size of the array was $|L||V|$. E.g. a sample with the attribute-value 5 and label-value 3 would end up at index $5|L|+3$. It was observed that for small numbers of samples for some attributes $|V|>>|S|$ which motivated a modified size of the array in which each time a split was made on a specific numeric attribute, a bound of its values was set and hence the array could be adaptively sized. A boolean array was also created which kept track of which values of the attribute had actually been used. This made sure that unused values were never counted. For datasets where almost all values are unique this approach would perform less well since the array being created would be prohibitively expensive. But since the target data had relatively fewer unique values than samples this approach worked very well, and no alternate approach was needed.

B. Splitting algorithm

First the label counts for the entire data was stored. For categorical values the samples were iterated through and all observed values stored in the previously mentioned array. Then for each value a split was made where the information gain was calculated as the gini-index for the observed values, and the total values - the observed ones.

For numerical values the structure of the mapped values was taken advantage of. Since lower mapped values corresponded to higher original values and these mapped values were used to index the aforementioned array the values of the attribute were implicitly sorted from largest to smallest. When the label distributions for the values had been done it was possible to simply iterate through array and continuously add to one side of the split while subtracting from the other.

To avoid calculating unecessary gains, e.g. when the best seen loss was much better than the recent ones a simple upper bound on the information gain from changing side of the split for n out of N samples. This bound was formulated as $IG(n) < \frac{2n}{N}$. The derivation of this bound can be found in the appendix. Tighter bounds could probably be found, but it was found that the performance gain from further optimization would be minor, since the conservative bound already prunes away much of the calculations.

C. Parallelization

Parallelization was done in a very simple fashion by running batches of trees in parallel. The batch was synched and stored, and then the next batch begun. The batch size was set to 5 to optimize performance for the 5 tree goal.

II. Results

The code was compiled using the compiler optimization flag -O3. For reproducability of results -O3 must be used since it gives major performance improvement. The time to construct 5 trees on the gent computer was estimated at around 1.95 seconds. A plot of the number of trees vs test accuracy can be seen in figure 1. It can be seen that the accuracy increases as the number of trees increases, this is because the variance of the prediction is reduced. Note that 2 trees does not improve the results since the classification is simply done by taking the most common prediction from the classifiers. Accuracy would probably improve by taking into account the number of examples covered as well as self estimated prediction accuracy.

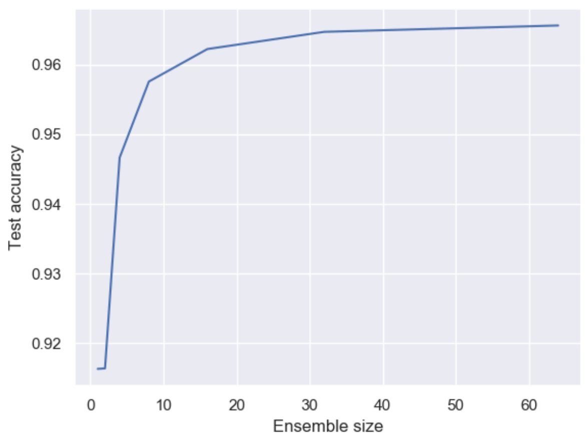

Fig. 1: Test accuracy as a function of ensemble size on the Covertype data set

Appendix

Proof. First the problem is reduced to finding a bound for the information gain of 1 sample. For simplicity we will refer to the different sides of the split as "left" and "right"-side. It is immediately obvious that the largest gain must come from changing side of the rarest class from the right side of the split, to the most common class on the left side. It is also known that for natural numbers the inequality in equation 1 holds.

$$(\sum_l n_l)^2 \geq \sum_l n_l^2 \tag{1}$$

This inequality implies that the highest information gain must come from when there are only 2 classes on the right side, since if there were more classes, changing side of the rarest class would give less of an effect. On the left side a similar argument gives that for maximal information gain, at most 1 class should be present. Following this reasoning it is also clear that the loss after this split will be 0. Since we want to maximize the gain and the loss on the left side is 0, we want to have as many samples as possible on the right side. All of the above give a maximum possible information gain as in equation 2

$$(1-(\frac{N-1}{N})^2-(\frac{1}{N})^2)-0=\frac{2}{N}-\frac{2}{N^2} \tag{2}$$

Since it is now known that $IG(1) < \frac{2}{N}$ then it is easy to see that $IG(n) < \frac{2n}{N}$ $\square$