Random Forest Using Harp: Decision Tree + Ensemble

Rohan Ingale (ringale@indiana.edu), Anand Karandikar (askarand@indiana.edu)

ABSTRACT

Random Forest is a classifier which constructs multiple decision trees using the training data and predicts the class label of test data sample by taking majority vote of the predictions by the decision trees. We propose an approach to construct the Random Forest in distributed environment using Harp 3.0 so as to parallelize the task of constructing the decision trees and predicting class labels.
The Random Tree generation algorithm is run by multiple threads created by multiple map tasks. Each thread also performs the prediction task and then majority vote is taken by using harp reduce API.
We test the algorithm for different number of map tasks and trees and compare its accuracy and running time.

METHOD

We have referred to the existing sequential implementation of random forest.[2] and implemented a distributed version of the same. We referred to the existing research paper[1] for the data distribution and sampling technique.
The train and test files are read from the memory and written to HDFS.
The training data is split into chunks since the data is too large to fit on a single data node.
Each map task reads one training file. Each thread in the map task samples the data from training file and builds a decision tree. It also computes the Out of Box error based on the data from training set that was not sampled.
The test file is shared among all map tasks.
We have assumed that the test file will be small enough to be able to fit on a single node.
Each thread predicts the class labels for samples in test data and returns the result to map task along with the OOB error.

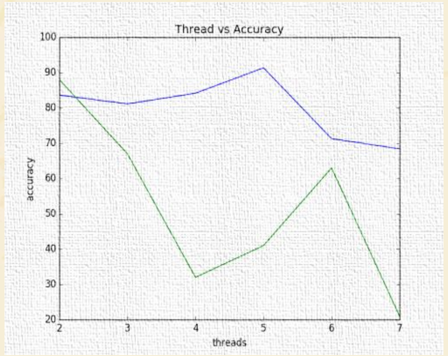
METHOD (Cont.)

The map task combines result from multiple threads.
The reduce API [3] which takes a combiner method is used to combine the results from multiple map tasks onto the master node. When combining results from multiple threads / map tasks, the OOB error is used to assign weight to label predicted by the tree.
The master node then compares the predicted label of each sample to actual label and plots the confusion matrix, determines the accuracy.

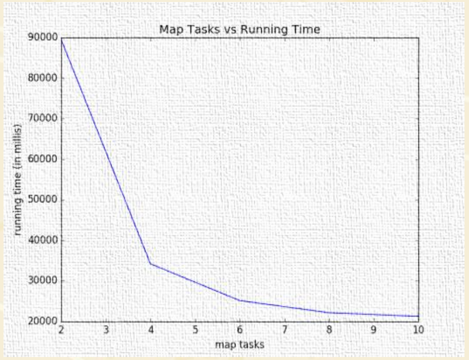
CONSTRAINTS

1. If the number of records at a node are < threshold then it is assigned as a leaf node.
2. The maximum depth of a decision tree cannot exceed 20.

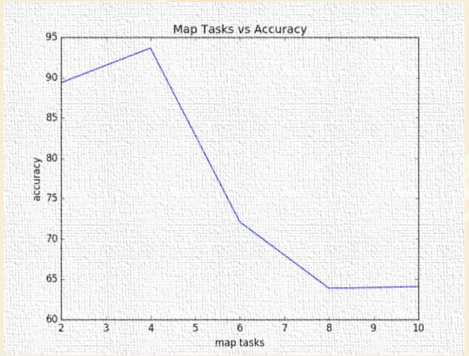
THREADS (TREES) VS ACCURACY



MAP TASKS VS RUNNING TIME



MAP TASKS VS ACCURACY



CONCLUSIONS

Parallel computation can help in faster computing of Random Forest and prediction of label, thus better utilizing the available resources.
It can be observed that the accuracy of predictions initially improves with increase in the number of trees but then decreases after a certain point.
The amount of time required for computation increases with increase in the number of trees since reduce function is the bottleneck.
The accuracy of predictions initially remains constant but later decreases with an increase in the number of map tasks since the training data is split into smaller chunks.
The amount of time required for computation decreases with an increase in number of map tasks.
Thus, there is a trade-off between the degree of parallelism and the accuracy of predictions which needs to be considered.

REFERENCE

[1] R., Poggi, J.-M., Tuleau-Malot, C., Villa-Vialaneix,,_N. 2015. Random Forests for Big Data. arXiv preprint arXiv:1511.08327. [https://arxiv.org/pdf/1511.08327.pdf]

[2] https://github.com/ironmanMA/Random-Forest

[3] Tutorial on Harp Map Collective

[4] https://github.iu.edu/IU-Big-Data-Lab/Harp3-Project/