Report on Project 3

Tian Wu, wu000130@umn.edu

Implementation of decision tree classifier:

The decision tree is generated through a recursive method. At each recursive step, the program will find an attribute that has never been used before, by splitting which the data set would have highest gain of purity. Once the attribute is found, the training set will be split into 2 partitions based on the value of this attribute of each instance. Recursion will be stopped once one of the 3 stopping conditions is met. The decision tree is stored in the form of nodes, with both pre-order and in-order in the model file, so that the tree can be recreated when running another program for testing later.

Evaluation:

	1	5	10	20
Rep1	0.8767	0.8750	0.8775	0.8726
Rep2	0.8271	0.8268	0.8242	0.8200

The table above illustrates how the accuracy would change with setting different values to “minfreq” as well as using different data representations. The accuracy of prediction was expected to decrease as “minfreq” increased, and the truth is that it did except for the unexpected rebound when “minfreq” was set 10 for data in the form of “Rep1”. Besides, the accuracy dropped by 4% after switching the representation approach from “Rep1” to “Rep2” whose performance is still acceptable since the dimensions of data have been reduced from 785 to 20 and the runtime has been decreased substantially from 30 minutes to 1 minute.

Implementation of random forest classifier:

The approach of generating random forest is based on the method used for creating regular decision tree described above with the extra sampling process. When calculating the accuracy of classification, the program will read through all the 100 predictions and vote the majority as the final prediction. The command lines for “rf.cpp” and “rfmerge.cpp” are shown below:

./rf <path_of_training_file> <minfreq> <path_of_testing_file>

./rfmerge

Evaluation:

	1	5	10	20
Rep1	0.9536	0.9530	0.9528	0.9508
Rep2	0.9015	0.8996	0.8959	0.8870

The accuracy of prediction has increased by 7% after applying random forest classifier. However, the cost is that the runtime has also increased drastically. For data represented in the form of “Rep1”, the time it takes to run the code swell from 30 minutes to about 14 hours. For data represented in the form of “Rep2”, the runtime increases correspondingly from 1 minute to about 45 minutes. By weighing the time it takes and the improvement of accuracy it brings, from my perspective, a combination of random forest classifier and representing data in the form of “Rep2” can bring the most benefits, gaining higher accuracy of prediction while spending less time in training classification model.