Is Custom Multilayer Perceptron Better Than Random Forest?

Kuang Jiang
Department of Electrical and Computer Engineering, Temple University
kuang.jiang@temple.edu

Complex data sets are usually hard to distinguish with given algorithms. In this project, a comparison between the performance of random forest from Scikit-Learn and a custom multilayer perceptron is performed in order to determine which algorithm has a between outcome in this specific case with 2d and 5d data that are used.

The random forest algorithm is an ensemble learning method for classification by constructing a combination of a variety of decision trees to mode classes of the training data. In order to obtain the best fit of a random forest for both 2D data and 5D data that were used in this project, parameters such as number of estimators, minimum number of samples that is required to split a decision tree, and the maximum depth of the random forest, are modified individually. By utilizing hyper parameter method that put all desired modification in a grid for a randomized search, it is able to identify that within the grid which combination of selected input has the best performance, the best fit, for the training data. In the case of the 2D data, the training data is the combination result of the provided training data and the development data while for the 5D data, only training data is used for corresponding training purposes.

The multilayer perceptron is a class of feedforward neural network with multiple layers of perceptron via activation functions. This function is built upon given sequential packaging provided by PyTorch the computing is sent to both the central processing unit and the graphic processing unit for comparison in the training error rate of each case. In this specific case, the multilayer perceptron is consisting of a layer of input, a hidden layer and an output layer with non-linear activation function of exponential linear unit and sigmoid. A more complexed version that consists of two hidden layers with a variety of nodes each layer and different activation function was also compiled in the search of a better performing neural network. During the training phase of both candidate network, different parameter values of learning rate, hidden nodes, and iteration cycles are used and output into a table for finding the best performing network with their corresponding optimum parameter settings in between of them. In the training of both of these networks, a combination of training and developing data was used for 2D data set and evaluation set was set up for performance validation while only training data set were used for 5D data set and its corresponding development data set was used for performance validation.

After obtain the optimum parameter for random forest with hyper parameter, the random forest performs with an error rate of 6.66% when evaluated on training data, 6.40% when evaluated on development data and 8.30% when evaluated on evaluation data for the 2D data set. As for 5D data set, it scores an error rate of 13.72% on training data, 37.42% on development data and

37.65% on evolution data. After obtained series of training scores for the comparison of those two-candidate network, the performance different in between two of them were minimum with roughly 0.30% of different within runs. Since the difference was sufficiently significant, the approach that is utilizing less layer were selected in favor of both training time and training complexity with no obvious downside in terms of performance. For the case of multilayer perceptron, when it is trained on the central processing unit it was evaluated with an error rate of 7.83% on training data, 7.50% on development data, and 8.15% on evaluation data for 2D data while performing 36.43% for 5D training data, 36.19% for 5D development data and 37.11% on 5D evaluation data. The following table shows the result in terms of error rate with each algorithm for each data set. The table also shows the tested error rate when the multilayer perceptron is compiled on the graphic processing unit. Based on that result, the performance when compiled on the graphic card shows no major difference in between those that are compiled on the central processing unit and in certain cases, those results are even better with a rough gap of 0.20% in between runs. The evaluation for multilayer perceptron that were trained on the graphics processing unit was not set for evaluation since the difference in between those two generated devices were not adequately significant while those generated on the graphics card has a reputation of worse performance with training and

	2D Data Set			5D Data Set		
Algorithm	Train	Dev	Eval	Train	Dev	Eval
Random Forest	6.66%	6.40%	8.30%	13.72%	37.42%	37.65%
Multilayer Perceptron (CPU)	7.83%	7.50%	8.15%	36.43%	36.19%	37.11%
Multilayer Perceptron (GPU)	7.60%	7.78%	8.10%	37.37%	36.12%	N/A

Table 1.Error Rate In between Random Forest and Multilayer Perceptron trained on CPU and GPU for Both 2D Data and 5D Data

evaluation.

Based on the result that is generated during the project, the random forest algorithm performs worse than the customized one-hidden-layer multilayer perceptron with the customized multilayer perceptron leading with a 0.15% lower error rate for the 2D data set and a 0.5% lower error rate for the 5D data set. It is worth noticing that for when training on 2D data set, the random forest algorithm performs between when fitting the training data, a combination of training set and development set, while performs worse during evaluation. In general, in this project, the customized multilayer perceptron performs reasonably better in comparison random forest algorithm within given parameters.