Peter Kress

9/27/17

Machine Learning

**Hyperparameter tuning**

The hyperparameters I tuned were the C values for the logistic regression classifier (lgr) and the support vector machine classifier (svm), and the max tree depth values for the decision tree classifier (tree). In order to tune for these, I checked each of the hyperparameter values and repeated the classification and testing with the same training and test sets for each one. The C values I checked for the lgr and svm were [.01, .1, 1, 10, 100, 1000, 10000]. The depth values I checked for the tree were [2, 3, 4, 5, 6, 8, 10, 50, 100].

From these, the hyperparameter with the best results (minimized max misclassification rate) was used in the final calibration of the classifiers. For lgr, the optimal value was C = 10 corresponding to 13.2% max misclassification and training time of 13.7 seconds. This C for lgr value was not on the extremes: very small C values under-penalized misclassification and large C values over-penalized it. For svm, the optimal value was C = .1 corresponding to 9.4% max misclassification and training time of 46.2 seconds, but this is actually misleading. The optimal C value for svm was more ambiguous since most C values provided a misclassification rate similar to the optimal rate from 9.4% to 10.6%. C = .1 and C = 1 were both tied as optimal. This indicates that, like lgr, a C value that is too small under-penalizes misclassification and a C value that is too large over-penalizes misclassification. For tree, the optimal value was depth = 50 corresponding to 0.3% max misclassification and training time of 33.8 seconds. The optimal tree depth was not one of the extreme values of the hyperparameter: depth = 10 did not provide enough specificity and depth = 100 provided too much, allowing for overfitting.

**Training time**

The times in tuning for lgr, svm, and tree were 13.7, 46.2, and 33.8 seconds respectively, but these do not reflect the time to train adjusted for the amount of data. lgr was using just 10.8% of the data, svm was using 22.5% of the data, and tree was using 90% of the data, defined by: tunefrac\_clf - testfrac X tunefrac\_clf. When comparing the same amount of data, tree was trained fastest, svm second fastest, and lgr slowest which is what lead to the chosen fractions of data used for tuning each of them. Changing C had little effect on the training time of svm, but did increase the training time for lgr. Increased tree depth also lead to longer training times, although still the fastest by far on large data.

**Misclassified classes**

I stored mistake percentages for each classifier and the classes that were most misclassified were different for each classifier. For lgr: 3, 4, 8, and 9 were the most often misclassified classes. For svm: 2, 3, 4, and 5 were the most often misclassified classes. For tree: 1, 3, 4, and 7 were the most often misclassified. Since many numbers can be skewed to look like others if lines get stretched/rounded/omitted (4s seen as 9s or 1s as 7s and vice versa), it makes sense that some letters can be confused as others, especially when they look similar. My intuition would guess that 3s, 4s, 5s, 8s, and 9s could most easily be confused as other numbers, and that is somewhat confirmed: 3 and 4 were issues for all three, but 5, 8, and 9 were less often confused. I suppose the people sampled tend to be sloppier with 3s and 4s than their other numbers.

**Recommendation**

I would recommend the decision tree classifier for this data since it is not only the most accurate, but it also maintains a fast training time even as the training data grows significantly, allowing the most scalability if more data were to be considered.