Jonathan Ho

CS 4375.003

SVM and Ensemble

Support Vector Machines (SVM) is an algorithm that can carry out binary and multiclass classification as well as regression on predictors. First, a separating hyperplane is decided which separates data between two classes. Classes are assigned a positive or negative 1 to represent one side of the separating hyperplane. SVMs are classified based on points closest to the separating hyperplane. Once the SVMs are determined, distances are measured between the separating hyperplane and the SVMs. This distance helps contribute to the margin. If the maximum margin (which is the greatest distance for SVMs of both sides) has been achieved, then the optimal separating hyperplane has been found. Since real data may not be separated neatly where a separating hyperplane could be easily found, a slack variable controlled by hyperparameter C is implemented. This slack variable essentially makes the margin wider allowing for more SVMs. The higher C is, the more slack is allowed and the bigger the margin is. A SVM Kernel is mapping a linear function to another function to linearly separate the data. Two common SVM Kernels are polynomial, and radial kernels. Polynomial kernels try to fit a polynomial function between data while the radial kernel roughly circles around one set of data.

I believe SVM is strongest when the target data can be seen as separatable to some degree. It also seems to be strong for small or medium data sets since it might be less of a chance for the target data to overlap too much. On the other hand, SVM seems weakest for large data sets which, in turn, can result in the target data overlapping with each other. Also, if SVM requires a large C, then even if accuracy is good, there would be a high bias and low variance.

Random forest utilizes bagging in which it repeatedly samples data then aggregates it. Specifically, it makes multiple independent trees on subsets of the data then aggregating them to try and mitigate the high variance that a simple decision tree would produce. While random forest can

prevent high variance and can be fast to execute, it seems like the predictors it chooses is restricted to the training data. There is also a chance for it to even over-fit the data.

XGBoost is a more robust version of Decision Trees and Random Forests. It is more robust because it can execute tree boosting significantly fast. This also is contributed to its use of multithreading utilizing all cores on a processor. XGBoost has three different types of parameters: general, tree boosting, and learning parameters. As mentioned before, it is very fast while being able to produce very accurate results. However, it potentially seems to be a very sensitive algorithm. This means if data has a lot of outliers, it may potentially mess with the accuracy produced by XGBoost.

Lastly is the AdaBoost ensemble method. A decision tree would use a subset of the data, but AdaBoost would use the whole data. However, unlike a decision tree building learning data parallel, AdaBoost builds data in sequence leading it to be a slower algorithm. In AdaBoost when it trains data, the first iteration has all observation weights equal. Every subsequent iteration, weights of observations with large errors are increased while those correct decreased. Each error is recorded after each iteration. Once all iterations are carried out, the weighted errors weigh the final learners to result in higher weights for more accurate learners. I think since AdaBoost trains learners sequentially, it is less likely to overfit the data. It also can help to classify any weaker classifiers within the data. However, like XGBoost, data with outliers can mess with the algorithm.