Portfolio Component: Kernel and Ensemble Methods

Neo Zhao & Andrew Sen
CS 4375 - Introduction to Machine Learning

---

a. The runtime is quickest with XGBoost as after the decision tree is created, the result value is passed on and used to calculate the next result value. The Adaboost runtime is the second quickest as there is only a node and two children processed during calculation. Last is the Random Forest runtime, which almost took about 2 minutes. This is because several decision trees are created before combining all to get a result.
    i. Random Forest Runtime: 109.22s
    ii. XGBoost Runtime: 0.780000000000001s
    iii. Adaboost Runtime: 29.14s

b. SVM works by finding a decision boundary, called the hyperplane, between two classes in the data. SVM finds the hyperplane that maximizes the distance between the points in the two categories. The distance between the points closest to the hyperplane and the hyperplane itself is called the margin, and the points that lie on the edge of the margin are called support vectors. SVM can be performed with different kernels in order to change the shape of the resulting hyperlane. A linear kernel creates a linear hyperlane, a polynomial kernel creates a polynomial hyperlane, and a radial kernel creates an enclosed hyperlane.

   The primary strength of SVM is that by using different kernels, it is easier to apply SVM to different datasets. Not all classification datasets are linearly separable, so it is handy that SVM can be used with non-linear kernels in order to find other ways to classify the data. The main weakness of SVM is that it is computationally expensive. Training an SVM model is extremely slow, and because SVM takes various hyperparameters, one will likely have to tune those parameters by making many SVM models. This makes it difficult to tune an SVM model against a very large amount of data. One may be forced to subset their dataset in order to produce a model in a timely manner.

c. Random Forest is an algorithm in which multiple decision trees will combine into one whereas XGBoost calculates a score before processing. For AdaBoost, we use “decision lumps,” which have one node and two children. Some strengths we can look at include high accuracy as well as efficiency; however, some weaknesses include difficult interpretations.

d. Adaboost is a boosting algorithm, which means that it applies multiple machine learning algorithms in series to create a single, more accurate model. Each iteration of Adaboost extends the decision tree produced by previous iterations. In the first iteration of Adaboost, all examples in the training data are given equal weights. In subsequent iterations, observations with large errors get increased weights while correct observations get reduced weights. Over many iterations, the weights will become such that more accurate learners are given more weight than less accurate learners. Being a boosting algorithm, Adaboost can create a more accurate model than a simple decision tree. However, Adaboost is fairly slow, especially if it is implemented purely in R.

XGBoost is also a boosting algorithm, and it uses gradient boosting. Each iteration, a decision tree is produced where the tree is recursively split on features with the highest information gain until some stopping threshold. The tree is pruned to prevent overfitting, and the weights of the leaves are smoothed. XGBoost creates extremely accurate models and was easily the fastest algorithm utilized in this project. However, XGBoost comes with a large variety of hyperparameters and performance parameters, which greatly increases the complexity of using XGBoost in practice.