Prof. Jyoti Deshmukh:Ensemble Learning- Bias Variance Tradeoff

The problem at hand here was to engineer an architecture that finds the optimal balance between training and testing performance of Machine Learning Models. In the domain of Machine Learning, the problem of Bias Variance Tradeoff is one of the biggest challenges. In simple terms, Bias is the error the model makes on the training data. Variance is the error the model makes on unseen test data. Ideally, we would want a model that performs very well on the training data as well as on unseen test data. This however is not possible as the test data may be from a different distribution and hence has dissimilar feature representation as compared to the training set. Our goal was to find an optimal architecture for increased performance of Machine Learning models with seen as well as unseen data for better generalization capability.

My approach involved reading about traditional approaches to tackle high bias (High errors on training set / Underfitting) and high variance (High errors on unseen data / overfitting). This led me to the ensemble techniques called Bagging and AdaBoosting. Bagging is said to help increase generalization of a machine learning model. It uses sampling with replacement to create ‘m’ different models. It then takes the mean of all these ‘m’ models to determine the final output. AdaBoosting on the other hand is a popular ensemble learning method that helps a weak learner make better decisions. It randomly subsamples data from the training set to create a model. It determines the samples that led to the errors and resamples them during the next iteration. After n such iterations, we have a model that has successfully learnt the distribution of data by retraining itself on data elements that led to errors initially.

At this stage, the architectural challenge was to integrate the bagging and boosting methods together. The first approach I used was taking the weighted average between the two. The second approach I used was to create an architecture that uses a voting method. For an input sample, my architecture would train two different models trained using Bagging and AdaBoost and the final decision would be of the model that had the better performance score of choice(Accuracy in my case but we can use precision/recall/f1 score etc).

I also experimented with the base classifiers. Adaboost performs best when used with a weak classifier such as a decision stub. Using a strong learner makes the model susceptible to overfitting and hence counter productive as to what our goal is. Bagging on the other hand does better when the base classifier is strong. The weak learners I tried were Decision stubs and Naive Bayes classifiers. The strong learners I used were Support Vector Machines and Decision Trees.

After running experiments on multiple real-world datasets (Iris Dataset, Pima Indian Diabetes dataset and Wisconsin Breast Cancer dataset), we noticed that the AdaBoost model generally does better in both cases. When using a weak learner, the Adaboost model tends to perform better than the bagged model on both seen and unseen data. Using a Strong learner makes the model achieve higher accuracies on the training set but does not generalize well. An alternative this would be to use a different metric such as F1 score rather than accuracy with the voting method. Another observation was the training time required to create both the models. For much larger and complex models such as neural networks, this technique might not be the most optimal considering how long it takes to train a large standalone model. The architecture I created could theoretically train the optimal model that not only performs well on seen data but does equally well on unseen data.